On Automatic Data Processing and Well-Test Analysis in Real-time Reservoir Management Applications

Stig Olsen

Dissertation for the degree philosophiae doctor (PhD)
at the University of Bergen

2011

Acknowledgement

The road leading towards the completion of this thesis has been long. However, finally the day has come to deliver this thesis. There have been many people along the road that I would like to thank.

First of all, I would like to express my gratitude to my advisor Dr. Jan Erik Nordtvedt for his support and encouragement. His contribution to this work has been invaluable.

Next, I would like to thank Epsis for allowing me to use their software in this thesis. I also had the great opportunity to work at Epsis, which was very motivating and educating. I would also like to thank all the great people I worked with at Epsis, and in particular Randi-Helen Halmøy, Ingve Fløysand, Frode Sekkingstad, Jørgen Tvedt and Bård-Henning Tvedt. A special thank to Dr. Ted Watson for good discussions and invaluable input regarding the wavelet theory in chapter 4 and the results from signal filtering in chapter 5.

Further, I would like to thank all my friend and fellow students. The years at the University of Bergen have been unforgettable. Both technically and socially the university has been terrific.

Also, I would like to thank my parents and my brother for always backing me up.

Last, but not at least, I would like to thank my wife Solveig, my daughter Isabell and my son Daniel for invaluable support and love.

I would like to thank the Royal Norwegian Research Council for providing funding for this work through their Oil and Gas Research Program and to Statoil for supporting my trip to ATCE in San Antonio, Texas, September 2006. I would also like to thank Statoil for providing data for this research.

Preface

This work considers improved and better use of real-time data measured from wells with permanently installed gauges. An increasing number of oil and gas wells are today equipped with such gauges. However, often the engineers have such a heavy workload that they do not have the time to analyse all measured signals from these gauges. Important information within the signal measurements might remain unnoticed and it is thus believed that additional information will be available if utilizing automatic analysis. In such a case the engineers will not spend time on routine analysis but will get a notification only if the automatic analysis gives unexpected results. We have in this work developed three automatic modules for use with continuously measured pressure and rate signals from oil and gas wells. The first module is automatic signal filtering of pressure and rate signals. The second module is process monitoring for detection of changes and events in the measured pressure and rate signals. The signal filtering and process monitoring modules can be used for automatic data preparation to analysis modules such as well-test analysis. The third module is hence automatic well-test analysis. The goal in the automatic analysis is to get the same information as from an analysis performed manually, but with only a fraction of the effort for the engineers.

This thesis is divided into two main parts, one theory part and one result part. ***In the first part*** of the thesis, the theory behind this work is explained. We start by introducing the real-time reservoir management concept in chapter 2. Improved and better use of data from wells with permanent installed gauges is an important part of the real-time reservoir management concept. In chapter 3, the theory for well-test analysis is presented. We provide the basic theory and develop the equations required for performing well-test analysis. The theory required for the signal filtering and process monitoring module is described in chapter 4. ***In the second part*** of the thesis, the results from this work are presented. The aim of this work has been to obtain modules and recommendations for automatic signal filtering, automatic process monitoring and automatic well-test analysis. In chapter 5, the proposed methodology and results from the signal filtering module is presented. Chapter 6 contains the proposed methodology and results from the process monitoring module while chapter 7 describes the methodology and the results from the proposed automatic well-test analysis. Finally, a summary is provided in chapter 8 while further work is discussed in chapter 9.

This work has been performed during my stay with Epsis AS and at the University of Bergen. Contributions to this work have been by Dr. Jan-Erik Nordtvedt. This work has been the basis for three different SPE papers presented at SPE conferences(1; 2; 3).

Contents

Purpose and Scope

1.1 Introduction

Improved and better use of real-time data from permanent downhole gauges has been a major concern in the oil and gas industry for some time now. As an increasing number of oil and gas wells are equipped with downhole and topside gauges for measurements of pressure, temperature and rate, information about wells and reservoirs can be retrieved more frequently and accurately than before. For engineers with limited time available, efficient tools for extraction of this information are critical. This will help the engineer spending time on analysis evaluation and decision making and not with data preparation and routine analysis.

To minimize the engineering time associated with data preparation, process monitoring and data analysis of a large numbers of measurements, an *automatic* approach is desirable. This is not an attempt to claim that automatic data preparation, process monitoring and data analysis will outperform a manual approach; it is simply an attempt to investigate if we can obtain some of the same information in an automatic approach as in a manual approach. By a *manual approach* we mean a sequence of operations performed on the measured signals where each step in the sequence is manually inspected by the user. In this context, an operation might be any data preparation or analysis performed with the measured signals, as for instance to remove some of the measurements, find the signal trend, estimate the average signal value and so on. The next step in the sequence of operations is performed if and only if the user has validated the results at the current step. By an *automatic approach*, we mean a sequence of operations performed without any user interaction. However, the user is required to go through a setup procedure where the specifications for the process considered will be chosen. In the setup procedure, the user might specify alarm settings for the results obtained, and will thus only need to inspect the results if alarms are triggered. The input for one operation might or might not depend upon the output from the previous operation. In *Figure 1.1,* a manual approach with three operations, compared to an automatic approach, is shown. Only three operations are illustrated, although any number of operations can be considered. In a manual approach, the user decides when a new loop of operations should be initiated. This might for instance be on a daily/weekly/monthly basis or after the occurrence of some particular events. In an automatic approach, the run frequency and the events triggering a new loop of operations is again specified in the setup. This means that the sequence of operations is performed repeatedly without user interaction and the user is only notified if alarms are triggered.

The goal in this work is to develop tools for improved utilization of continuously measured signals obtained from permanently installed gauges by the use of automatic approaches. To accomplish this, we need to determine which operations to perform and how to perform them. How to perform the operations will be specified in the user setup and is referred to as the *run specifications*. A set of operations required for one specific task, as for instance signal filtering, will be referred to as a *module*. In this work we will attempt to develop modules for automatic signal filtering, process monitoring and well-test analysis.

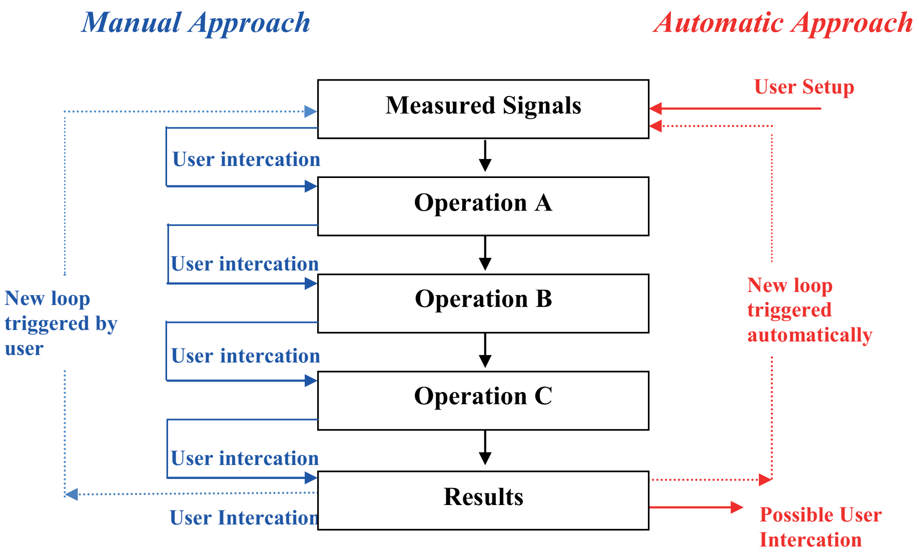


Figure 1.1: *Manual analysis compared to automatic analysis.*

Signal filtering will be the common phrase for three different sets of operations that can be performed on the signal measurements; namely outlier removal, de-noising and data reduction. *Date reduction* is referred to as a set of operations used to reduce the number of signal measurements where the aim of any good method for data reduction is to remove those measurements with little information content and keep the rest. *De-noising* means the removal of rapid and random fluctuations in the signal measurements caused by interference at the signal measurements and not from changes in the process state itself. These random fluctuations are normally assumed to have some known distribution and in most cases, also in this work, this is assumed to be a normal distribution. If the random noise fluctuations during some small limited time period are very large compared to the rest of the noise fluctuations, in fact that they are so large that they cannot be described adequately by the distribution imposed on the rest of the fluctuations, they are referred to as outliers and the removal of these measurements, single or multiple, is simply referred to as *outlier removal*. We will in this work develop a module for automatic signal filtering for use with typical signal encountered in the E&P industry as pressure and rate signals. The module will mainly be based upon wavelet analysis.

Wavelet analysis has proved valuable for de-noising and is in particular well suited for de-noising of signals from non-stationary processes. Such a process is here regarded as a process where the long term statistics (as for instance mean value) changes with time. Such signals are common in the E&P industry and some applications of wavelet analysis have already been published to deal with this. The use of wavelet analysis for analyzing data from permanent downhole gauges in the E&P industry was first introduced by Kikani et. al.(4). Athichanagorn et al. (5) further introduced a seven step procedure for processing data from downhole gauges including outlier removal, de-noising and data reduction. Several papers have in addition been published (6; 7; 8; 9; 10; 11; 12; 13; 14; 15) and a summary of wavelet analysis in the E&P industry has been published by Guan et al. (16). However, it is outside the E&P

industry that most applications of wavelet analysis are found and among these is an extensive investigation of the performance of wavelet de-noising on different kinds of signals by Antoniadis(17). It was shown that the performance of wavelet de-noising would depend upon the selections made (e.g. choosing different wavelets as is discussed later) Different selections are designed to take into account different signal characteristics and the performance of wavelet de-noising will depend critically upon the selections made. Little attention has been paid to this in the E&P industry. ***We will do a performance analysis to determine the "best" selections for use in wavelet de-noising for typical signals encountered in the E&P industry as pressure and rate signals***. By determination of the "best" selections for use in wavelet analysis for the signals considered, this holds the promises for automatic signal filtering. (What is meant by "best" will be discussed later.) In addition we will describe methods for automatic outlier removal and data reduction. ***We propose to remove outliers by a median filter while the data reduction is formulated as a minimization problem***. However, they will both rely upon elements from wavelet analysis. As for de-noising, the proposed methods for outlier removal and data reduction require no user input and hold the promise for automatic outlier removal and data reduction. Automatic signal filtering will be discussed in chapter 5 in this work, and the results are based upon the work by Olsen et al. (1)

Process monitoring refers to direct surveillance of signals to determine process and measurement characteristics that can be assessed directly from the signal measurements without explicit consideration of a process model. This includes continuous noise estimation, detection of abrupt signal changes and assessment of trends. In process monitoring, we need to consider a given number of the most recent measurements for computational efficiency. This *window size* will determine how fast and accurate changes in the signal measurements can be detected. A small window will result in fast detection of possible changes, but might in turn result in a higher frequency of "false" detections. On the other hand, a large window size will be better for an accurate detection of changes but might also result in a slower detection. Detection of abrupt signal changes, referred to as *transients* when caused by a change in the process state, will be a key element in process monitoring. If for instance an oil or gas well is shut-in or starts to flow, the pressure in the well will change as a result of this and a pressure transient will develop. Transient detection in pressure signals has been considered by several authors in the E&P industry(5; 6; 8; 11; 18; 19; 20). For reasons that will be discussed later, ***we propose a new method for automatic transient detection based upon pattern recognition. In addition we propose a module for automatic process monitoring.*** Automatic process monitoring and pressure transient detection is discussed in chapter 6 in this work, and the results are based upon work by Olsen et al.(2)

The goal of the proposed signal filtering and process monitoring modules is to process a large number of signal measurements in a computationally efficient manner, continuously and without user interactions. The module for automatic signal filtering and process monitoring can be used for data preparation in different kinds of analysis. If high resolution signals are used directly in these analyses, the analyses might be too computationally expensive, and tools to reduce the number of signal measurements will be critical. One such analysis is well-test analysis.

Well-testing is an important tool for determination of reservoir and well properties. In well-testing we measure the pressure response to rate changes. Based upon the pressure response, we obtain an estimate of important well and reservoir properties such as skin factor and permeability. These properties are important for determination of the fluid flow properties of the well and reservoir and will help us to describe the recovery of oil and gas from the reservoir considered. Traditionally, well-testing has been both time consuming and costly to perform. Tools for pressure and rate measurements had to be lowered into the well each time a well-test should be performed. This was an expensive operation and in addition resulted in lost production. The introduction of computers made the processing of the measured pressure and rate signals easier and well-test analysis was both improved and accelerated. In the recent years, the increased utilization of permanent gauges for continuously measurements has made the task of obtaining measurements for well-test analysis nearly straightforward. Well-tests can now be performed continuously with the measured pressure and rate signals. This gives us a unique opportunity to obtain additional information about the considered well and reservoir in a more cost efficient and frequent way than before.

The use of continuously measured signals for use in well-testing has previously been done using a time window approach by Athichanagorn et al.(5) and Khong(6). The aim is to estimate the unknown parameters in each time window to obtain time series of the unknown parameters. Hence possible changes in the unknown parameters with time can be identified. The main problem with this approach is that the estimated parameters will be highly dependent upon the window size and maybe more important, the estimated parameters will in general show unrealistically large oscillations in estimated values. It is frequently believed that the estimation of parameters like permeability and skin factor from flow periods should be avoided (21), and the results from Athichanagorn et al. and Khong in our opinion confirms that this also applies to the window approach. Lee (22) used a predefined function for the estimation of skin and permeability. Although this approach seems to be more promising and avoids large oscillations in the estimated parameters, it requires a function to be defined to describe the change in permeability and skin factor with time. This will in general require some prior knowledge about the parameters regarding which function to choose. Thus an approach only involving the analysis of the shut-in periods seems to be sounder. When the shut-ins originate from arbitrary shut-in periods in continuously measured signals, they are referred by some authors to as "free well tests" (23). In the paper by Airlie et al.(23), they describe a process for automatically loading data into PanSystem (24) (well-test software) for performing well-tests on arbitrary shut-ins. Automatic regression analysis is then performed on the detected shut-ins in order to determine permeability, skin factor and reservoir pressure, and thus time series of these parameters are obtained. Airlie et al. finally states that the results need to be validated by an engineer at the end of the analysis.

Coludrovich et al. (25) performed continuous estimation of skin factor and permeability from arbitrary shut-ins. They applied downhole flow meters for accurate and continuous rate measurements along with downhole gauges for pressure measurements. They typically detected 1-6 shut-ins for a well each month and based upon the well-tests analysis for each detected shut-in, they created time series of skin factors and permeabilities. The well-tests were performed manually on a monthly basis as a part of the overall production optimization strategy.

As described above, the use of continuous measured data for well-test analysis has to some extend already been published in the E&P industry. However, work still needs to be done for better and more efficient well-test analysis utilizing the large number of measurements available from wells with permanent downhole gauges. ***In particular, improvements are desired with respect to automatic data preparation and automatic result evaluation to minimize the engineering time associated with repeatedly performing well-test analysis***. Instead of spending time on such repeated "routine" tasks, the engineering effort should be directed towards the well-tests providing new information about the well and reservoir, i.e. the well-tests where the results differ from previous well-tests. Automatic well-test analysis will be discussed in chapter 7 in this work, and the results are based upon work by Olsen et al.[3].

1.2 Scope of Work

The scope of this thesis is to introduce methods for better and more efficient use of real-time data in the E&P industry. We will propose a methodology for automatic signal filtering and process monitoring of typical signals encountered in the E&P industry such as pressure and rate signals. Secondly, we will investigate whether well-test analysis can be performed automatically and without any user interaction to continuously obtain information from the measured pressure and rate signals. We will present a methodology for this as well.

More specifically we will design a method utilizing wavelet analysis to improve data analysis for typical signals encountered in the E&P industry. For applications of wavelets in signal analysis, particular signal characteristics need to be taken into consideration in the design of the methods utilized. A number of specifications must be made in the different methods and the efficiency of wavelet analysis will depend critically upon these choices. Little attention has been paid to this in the publications found in the E&P industry. It would thus be very important to do such a study to determine what are the “best” selections for use in wavelet analysis of typical signals encountered in the E&P industry. The determination of these best selections holds the promise for automatic de-noising. Further methods for automatic outlier removal and data reduction are proposed. These methods will not be based upon wavelet analysis, but both proposed methods require an accurate estimate of the noise level in the signal measurements. This noise estimate will be obtained from wavelet analysis. The method for outlier removal is based upon a median filter while the method for data reduction is formulated as a minimization problem.

A module for real-time signal filtering and process monitoring will be developed for continuously measured signals. For such signals, a window approach is required to handle a large number of signal measurements in a computationally efficient way. In process monitoring, the window size will in addition be important for how fast and accurate changes in the signal characteristics can be detected. Real-time process monitoring in this work includes real-time noise estimation, real-time trend estimation, real-time signal filtering and real-time transient detection. For transient detection, an approach based upon pattern recognition is proposed.

We will design a method for automatic well-testing utilizing continuously measured pressure and rate signals. The signal filtering and process monitoring module will be used for data preparation. As Airlie, we choose to only consider well-tests from arbitrary shut-in periods. We believe that these shut-ins will occur frequently enough to provide us with time-series of parameters typically found from well-test analysis such as skin factor and permeability. In well-test analysis, we propose a statistical evaluation of the results where the main objective is to detect changes in the estimated time series of parameters. If changes are detected in the estimated parameters, alarms are triggered and a notification will be sent to the engineer. On the other hand, if no changes are detected, this will support our initial parameter estimates. By the use of such a method, all arbitrary shut-ins in oil and gas wells will be used for well-test analysis, and additional information about the well and the reservoir might be obtained. Thus the engineering effort can be directed towards those well-tests where changes are detected.

We will provide several cases to illustrate the application of such a system. Detailed description and utilization of these modules will be given in this work as outlined in the next section.

1.3 Outline

In chapter 2, we give an introduction to the real-time reservoir management concept. An important element in real-time reservoir management is improved and better use of real-time data. Tools for better use of data continuously measured in oil and gas wells can for instance be automatic signal filtering, process monitoring and well-test analysis as will be described in this work.

In chapter 3 we introduce well-test analysis. We describe the mathematical model and provide the equation for the homogenous infinite reservoir. The equation is more easily solved in Laplace space, and we explain this method. The solution is inverted back to time space by a method for numerical inverse Laplace transforms referred to as the Stehfest routine (26). Then we introduce important concepts in well-test analysis such as wellbore storage and skin. The superposition principle is introduced to take into account a well with changing rates. Finally, we explain the concepts of modern well-test analysis including model identification by derivative plots, data preparation and non-linear regression analysis. In non-linear regression we seek to minimize the difference between the measured pressures and the pressures estimated from a mathematical model assumed to adequately describe the reservoir and well considered. We further explain how to obtain confidence intervals for the estimated parameters. Since we have chosen a gradient based regression method, function derivatives with respect to the unknown parameters are required. We have chosen to use analytical derivatives as these usually are more accurately estimated than numerical derivatives. We finally present the analytical derivatives for the parameters usually considered as unknowns for the homogenous infinite and homogenous finite reservoir model.

In chapter 4 we describe the basic theory behind wavelet analysis. We give an introduction to the mathematics and background for wavelet analysis. All important aspects of wavelet analysis such as scaling functions, wavelet functions, wavelet transform and inverse wavelet transform are explained. De-noising based on wavelets in the Waveshrink method is described next. In Waveshrink a number of selections need to be made. We need to determine which wavelet, shrinkage rule, threshold estimator, noise estimator and primary resolution level to apply. All the different selections considered in this work are described in this chapter.

In Chapter 5 we determine how to filter the kind of signals we consider in this work. This means to determine how to accomplish automatic outlier removal, de-noising and data reduction. However, we will first describe how the synthetic signals considered in this work are generated. Next we will describe the proposed method for automatic outlier removal. We will further determine the “best” de-noising method for the signals we have considered starting with the definition of “best” in this context. We will also discuss what to do in wavelet analysis in the case of unequally spaced signal measurements. Finally, a method for automatic data reduction is presented.

Chapter 6 describes real-time signal filtering and process monitoring. The application of wavelet transform and wavelet filtering in real-time is discussed and a computational optimized method for signal de-noising in real-time is suggested. A new method for automatic transient detection based upon pattern recognition is described next. In the context of real-time process monitoring we further discuss; continuous noise estimation, continuous signal filtering, continuous transient detection and continuous trend detection. When signal filtering and process monitoring is applied with signal measured in real-time, a window approach is required, i.e. we need to consider a given number of the most recent signal measurements. Recommendations for which window size to apply will be discussed for each of the elements in process monitoring; noise estimation, signal filtering, transient detection and trend detection.

Chapter 7 describes the proposed method for automatic well-test analysis. The proposed method consists of four main steps. The first step is the setup used for initialization of the proposed automatic analyses. The second step is signal filtering and process monitoring, while the third step is the non-linear well-test regression analysis. The third step also includes automatic evaluation of the results from the non-linear regression analysis by the use of hypothesis testing. The forth and final step is possible user interaction if alarms are triggered. Next, four cases illustrating the application of the proposed method for automatic well-test analysis are described. The four cases considered include two cases with synthetic data and two cases with data from a North Sea gas/condensate reservoir.

In Chapter 8 the conclusions from this work is given while further work is discussed in chapter 9.

In appendix A, we describe the Bessel functions. All mathematical equations for well-test analysis considered in this work contain Bessel functions. A fast and accurate numerical calculation of Bessel functions is critical in order to have fast and accurate solutions of the equations considered in well-test analysis. The accuracy and performance of the numerical implementations for Bessel functions found in the literature is discussed and improvements are suggested. Some of the actual numerical implementation in C++ is given as well.

Appendix B contains some of the wavelet implementation in C++. This includes wavelet and scaling coefficients for all wavelets considered in this work.

The reader familiar with well-test analysis and wavelet analysis should read chapter 2 and continue from chapter 5. The reader, who would like to get some additional information on well-test analysis and/or wavelet analysis, should read chapter 3 and/or chapter 4 as well.

Part 1: Theory and Background

2 Real-time Reservoir Management

A lot of effort is put into the development of real-time reservoir management. Some oil companies again refer to this as the digital oil field, the Field of the Future (BP), SmartFields (Shell), i-field (Chevron), e-field and integrated operation (Statoil) only to mention a few. Real-time reservoir management is gaining acceptance in the E&P industry and most E&P companies have the real-time reservoir management development as one of their main R&D activities and significant effort is put into this development. As seismic interpretation, wireline logging, offshore drilling, directional drilling and intelligent well completion in the last decades significantly contributed to the increase in recovery and reduction in costs, it is by many oil companies believed that real-time reservoir management will be the next technology contributing to this.

The motivation for companies to implement real-time reservoir management is to make better and faster decisions, and to be able to act upon these decisions. Making faster decisions with higher quality is believed to result in increased production, increased recovery, reduced capital expenses (capex), reduced operational expenses (opex) and improved HSE (health, safety, environment).[27] It is all about reducing the costs while increasing the output from the asset leading to an overall better and more optimized asset management. This might in particular be important for mature fields, development of marginal assets and when working in harsh environments as in the arctic. In these cases, the implementation of real-time reservoir management might be critical to whether or not an asset will actually be developed at all.

Real-time reservoir management has been enabled by new digitally technologies. This includes new sensors providing high quality data, the ability to transfer (fiber optics) and store large amounts of data and finally high quality video conference equipment. The initial step in the development of real-time reservoir management was mainly technology driven. The next step in real-time reservoir management was to create computer software to analyze and handle these large amounts of measured data. However, it was soon realized that real-time reservoir management should include a lot more than some specific technology or IT-system. People would not use the new technology simply because it was available. Thus real-time reservoir management is about implementing a continuously improved set of new technologies and work processes for continuously improved and optimized asset management. This is sometime referred to as *"changing the way we operate the asset."*[28] This also includes changing the way we share information and the way we are working. It has also been realized that proper training of staff is important to realize the full potential of real-time reservoir management. ***In summary, real-time reservoir management is about proper trained multidisciplinary teams working together in a common environment in real-time utilizing innovative software solutions to continuously improve and optimize asset management.***

A common environment is a place where people from various discipline can come together to do the analysis, identify problems and discuss solutions. In this common environment, also referred to as collaboration centres or decision environments, the latest measurements and analyses will be available to the asset team in real-time. The collaboration centre might be a physical operation room, a virtual environment or a combination of these.

Most E&P companies have built numerous such environments. Chevron, through their *i-field* program, has by 2008 build 20 collaboration centres. [29] Chevron refers to collaboration centres as Asset Decision Environments and was in 2008 planning to build 30 more of these. An illustration of a collaboration centre as envisioned by Chevron is shown in *Figure 2.1.*

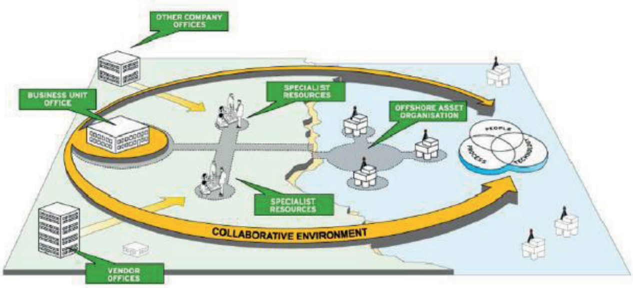


Figure 2.1: *Illustration of a collaboration centre as envisioned by Chevron*[29]

When we talk about real-time, it should be clarified that this does not necessarily refer to a second by second optimization of the asset, but to perform the optimization with as little time delay as possible. Instead we can use the concept of *relevant time* or *right time* [28]. The operator optimization loop will focus on increasing safety and minimizing downtime. This time loop range from minutes to hours. The production optimization time loop includes well work-over and reservoir surveillance and has a time loop of hours or weeks. Field optimization involves optimization of drainage area and injection wells and has a time loop from week to month. The slowest time loop involves optimization as secondary recovery and has a time loop of years. This is illustrated in *Figure 2.2.*

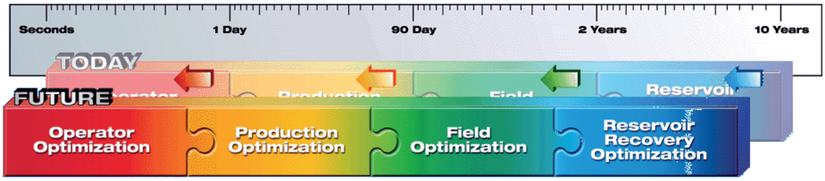


Figure 2.2: *Improved monitoring by optimizing the time loop from Unneland et al.* [28]

To perform the optimization with as little time delay as possible is referred to as *closed loop control* and is illustrated in *Figure 2.3*. The first step in the closed loop control is to increase the flow of information being available to the asset teams by ***measurements of data***. Traditionally the gathering of information from the wells and reservoirs has been difficult, time consuming and costly. During the last decade there has been an increase in the availability of sensors for permanent monitoring. These sensors installed downhole or topside are providing the asset team with continuous high quality and high resolution data. As well as providing the asset team with useful information, the cost of these sensor are continuously dropping in price and they have a continuously increased reliability. The second step in the closed loop control will be to ***extract useful information*** from the measured data. As an increasing number of measurements is available, efficient tools are required in order to gather the data, subsequently prepare the data for the analysis and finally to perform the analysis itself.

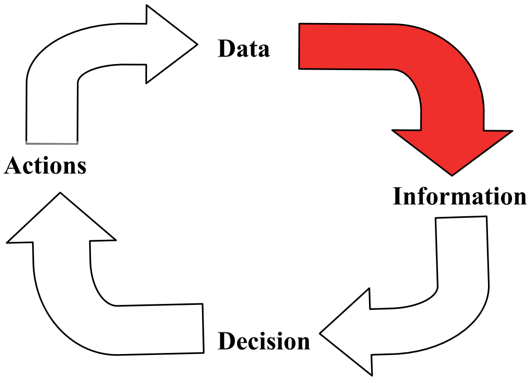


Figure 2.3: *Closed loop control.*

Once the result from the analysis is available, ***decisions*** can be made to improve the recovery and/or reduce the cost. This is the third step for the closed loop control. The final step in closing the loop is to implement the decisions made trough required or recommended ***actions***. Once the actions have been completed, new measured data will be available and ready for analysis. Thus the effects of the implemented actions can be evaluated and new decisions made, illustrating the concept of the closed control loop. Typical actions for a well can be to adjust the choke size in order to optimize the production of oil and gas. For instance, this might be important in a gas well to avoid liquid drop-out, while in other wells the motivation can be to minimize sand production [(21)]. With zone controlling equipment or intelligent completion systems it is possible to close zones with high water cut in order to optimize the production of oil and gas.

The first step in the closed loop control (*Figure 2.3*) is to measure data and obtain information about the asset. Even though a lot of data is being measured, non-integrated tools make the information flow around the asset slow and inefficient. The traditional way of sharing information has been by e-mails, phone calls and meetings. Efficient tools to improve this situation will be to use distributed software for sharing information, data and analysis results. Thus all relevant information and experience can be synchronized and shared among the assets teams.

The efficiency of implementing improved systems for data analysis and decision making can easily be identified by looking at a traditional way of performing an analysis. It is often stated that in order to perform a given analysis, the engineer spends 60-80% of the time looking for and preparing data.[(28)] Sometimes the analysis itself is even performed using a simple spreadsheet. If specialized analysis software were linked into a distributed system where all measurements were stored in a common database, the engineer could spend most of the time on the analysis itself. This would in addition remove sources of error as to whether the measured data were exported in the correct format or in the correct units. In addition, the final results from the analysis would be stored in the same distributed system making the result available to everyone, everywhere, anytime. By using integrated software solutions, faster and better solutions in real-time is believed to be the result. Faster decisions will reduce the cost by being more proactive, while better decisions will result from a better understanding of the reservoir and recovery process.

The task of designing a system for improved data analysis and decision making is usually in the hands of IT professionals. However, the petroleum engineer is the end user and she/he must play a key role in designing such a system. This requires a petroleum engineer with significant IT knowledge to communicate with the IT professionals. Special training of petroleum engineers in IT related subjects and decision making is required to make the *digital petroleum engineer*[30] a reality. This includes both training students at universities and professionals in the industry. For instance, an experimental program has since 2004 been available at The University of Southern California for training of the digital petroleum engineer[30].

From being purely a vision a decade ago, several assets have now implemented real-time reservoir management projects with measurable value. This includes e.g. Troll [31], Huldra [32], Brage [33], San Ardo [33] Agbami[34] Nelson[35], Ekofisk[36], Statfjord [37], Valhall and Ula[38]. The real-time reservoir management implementation in these fields includes data validation, data filtering, advanced monitoring, smart wells, optimization and visualization techniques. Several companies have reported that substantial values have been created from their implementation of real-time reservoir management. Shell has for the last decade developed a real-time reservoir management program (SmartFields) and has gained great value from this. The estimated value is approximately $5 bln. based upon the assessment of 50 assets for the period 2002-2009 [27]. This includes both new fields and retrofits in existing fields. Initially Shell screens the asset and then applies the correct level of smartness. The greatest contribution is from increased production and reduced capital expenses (capex). The technology that has contributed the most is advanced real-time monitoring, 4D seismic, visualization and integrated technology. Statoil, the largest operator at Norwegian Continental Shelf (NCS), reports to have gained $1.5 bln. in 2006 from real-time reservoir management [39].

Even though great values have been reported from real-time reservoir management, there are many hurdles along the road from the vision to the actual implementation, and additional work is still required to extend the full potential of real-time reservoir management. A study performed by OLF in 2007 valued the real-time reservoir management potential on the NCS to 295 bln. NOK.[40] Innovative software solution converting the large amount of measured data into useful information to optimize the overall asset management is a key component in the real-time reservoir management concept. We will in this work design a system for automatic signal filtering and process monitoring which subsequently will be used for automatic well-test analysis. Such a system will be an important component in the real-time reservoir management concept. It will allow the engineers to faster and more efficiently transform the measurements into information. When information is made available earlier, faster and better decisions can be made and actions can be taken based upon these decisions. Based upon the implemented actions, new measurements will be available and new information will be obtained. Hence, the system described here will optimize the time loop, believed to result in a better and more optimized asset management. As a consequence new information will be available to the asset team with little time delay.

The work described here will mainly be based upon wavelet analysis and well-test analysis. We will thus start by describing the theory required to understand well-test analysis followed by the theory required to understand signal de-noising based upon wavelet analysis. This will be described in chapter 3 and 4. The results from the implemented software modules will be discussed in chapter 5, 6 and 7.

3 Well-Test Analysis

Historically well-test analysis has been an important tool for reservoir and well evaluation. Well-tests have provided the engineers with valuable information about well and reservoir properties based upon analysis of measured pressures and rates (oil, gas and water). Typical information required for description and understanding of the flow properties of the reservoir (e.g. permeability) and the flow properties of the well (e.g. skin factor) has been obtained from well-tests. In addition, it is also possible to obtain an estimate of the drainage area size along with the average pressure of this drainage area. For heterogeneous reservoirs as layered reservoirs, fractures reservoirs and dual porosity reservoirs, to mention a few, additional well and reservoir properties might be obtained. All this information is valuable for the engineers for increased understanding of the oil and gas recovery process, and will ultimately help the engineers to develop an optimized production strategy for how much and how fast, oil and gas that can be produced from the well and reservoir considered.

In well-testing we measure the pressure response to rate changes. We will only consider the case where the pressure measurements and the rate changes occur in one specific well. Different part of the measured pressure response will be dominated by different wellbore, reservoir and boundary properties, i.e. the initial flow will be dominated by well properties. After a while, the pressure response will be dominated by reservoir properties, and finally, the pressure response is dominated by different boundary properties. This is because the oil and gas produced initially will origin from the well or close to the well, while the latter oil and gas produced, will have its origin further away from the well. A mathematical model, assumed to adequately describe the pressure response in the well to rate changes, needs to be selected. Given the input (rate changes) and the output (measured pressures) from the model together with the model assumptions, characteristic reservoir properties might be calculated explicitly from the mathematical model. This is illustrated in *Figure 3.1.* This is often referred to as an *inverse problem*. A typical difficulty is that several mathematical models might explain the pressure response from the rate changes. However, by combining the results from the well-test analysis with information from logs and geology, it is possible to remove some of the ambiguity.

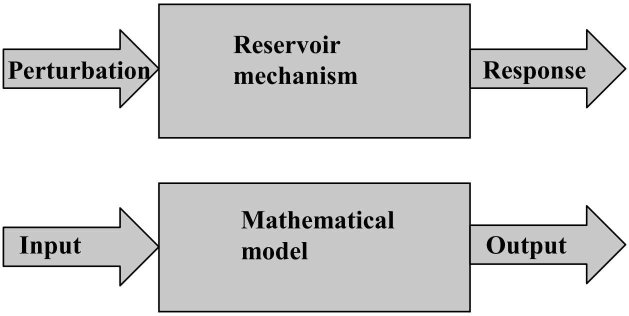


Figure 3.1: *A mathematical model is used to describe the reservoir mechanics.*

Parameters typically estimated in well-test analysis are;

- Permeability.

- Reservoir heterogeneities like dual porosity, multiple layers, composite reservoir etc.
- Well properties like skin factor.
- Reservoir geometry and reservoir size.
- Initial pressure and average drainage area pressure.
- (Porosity)

Porosity is placed in parenthesis since it is usually assumed known from logs and core samples. All these parameters will provide us with valuable information about the wellbore, reservoir and boundary conditions in the well and reservoir we consider. It should however be noticed that in many cases, as will be discussed later, only a limited number of the above parameters might be estimated from an actual well-test.

In the case where many wells are present within the boundaries of one reservoir, each well will have its own *drainage area*, i.e. the area of the reservoir where the oil and gas flows towards one distinct well. In these cases, the results from the well-test will provide information about the drainage area of this specific well and not the entire reservoir (which might contain numerous wells).

The first well-tests were performed by graphical methods referred to as Horner plot, MDH plot etc.[21; 41]. (Horner plot is a plot of $\mathrm{p} = \log\left(\frac{t + \Delta \mathrm{t}}{\Delta \mathrm{t}}\right)$ and MDH plot is a plot of $\mathrm{p} = \log(\Delta \mathrm{t})$. *P* is the pressure while *Δt* is the duration of the well-test). In the graphical methods (initially made by hand), it was possible to obtain estimates of different wellbore, reservoir or boundary properties [21]. A problem by applying these graphical techniques is that they are normally quite time consuming. The graphical techniques will also limit the number of properties that can be estimated for each well, i.e. it is limited how many model properties that can be identified from the different specialized plots[42]. The estimation will in addition be user dependent because of the requirement for the user to make a number of choices in the estimation of parameters from the specialized plots. Hence, improvements in well-testing techniques were desired, and a computer-based approach was introduced.

The introduction of computers first resulted in the opportunity to process a larger number of data and the graphical techniques were now accomplished with the help of computers. However, this only included a very small potential of what computers could offer, and soon multi-rate tests, non-linear regression analysis and derivative plots combined with a large number of available mathematical models resulted in faster and more accurate interpretation of well-tests (multi-rate tests, non-linear regression analysis and derivative plots will all be explained in more details later). The use of non-linear regression analysis in addition had the huge advantage of providing the estimated model parameters with a confidence interval, i.e. an estimate of the uncertainty in the estimated. These computer assisted methods are often referred to as *modern well-test analysis*.

In this chapter we will start by developing the mathematical model used to describe the radial flow towards a well in a homogenous reservoir. We next introduce dimensionless variables and rewrite the model equation using dimensionless variables. The equation we obtain is more easily solved in Laplace space than in time space, and next we show how to solve the equation for a homogenous infinite reservoir in Laplace space. Two important properties, the skin factor and the wellbore storage constant, used to accurately describe the early time fluid flow in the well, are explained next. The model equation is solved with the assumption that the flow rate into the well is constant. However, this is seldom the case, and superposition is introduced to take into account a well with variable flow rates. Next, we describe the concepts of modern well-test analysis and we introduce data preparation, model identification, non-linear regression analysis and confidence interval estimation. In this chapter, only a homogenous reservoir is described. However, numerous models are proposed in the literature to take into account different reservoir heterogeneities, wellbore conditions and different kinds of boundaries but that will not be in the scope of this work as only the homogenous reservoir model is applied.

3.1 Mathematical Model

There are several assumptions associated with well-testing in order to keep the equations simple enough to be solved easily. These assumptions usually are homogenous porosity and permeability, ideal horizontal flow, no gravitational effect, constant reservoir height, well perforated along the entire reservoir, constant viscosity and production at small pressure drop ($c\Delta P$ << 1 where c is the compressibility and ΔP is the pressure drop). Even though these assumptions will put limitations on the number of reservoirs adequately described by the mathematical model, some important pieces of information can usually be extracted. Nevertheless, before we start explaining the actual model itself, we need to define some basic reservoir properties. The porosity is defined as;

$$\phi = \frac{V_p}{V_t} = 1 - \frac{V_b}{V_t} \quad . \tag{3.1}$$

The porosity describes the relationship between the volume that might be occupied by fluids and the total volume element considered. V_p is the pore volume, V_b is the bulk volume and V_t is the total volume. We continue by defining the density as;

$$\rho = \frac{m}{V} \quad . \tag{3.2}$$

Here m is the mass and V is the volume. Further, we have the isothermal compressibility given as;

$$c = -\frac{1}{V}\frac{\partial V}{\partial p} = \frac{1}{\rho}\frac{\partial \rho}{\partial p} \quad . \tag{3.3}$$

To find the basic flow equations describing the flow through a porous medium, we start by considering the fluid flow through a volume element as shown below in *Figure 3.2*. Only fluid flow in the x-direction is shown, but the same applies to the y and z direction too.

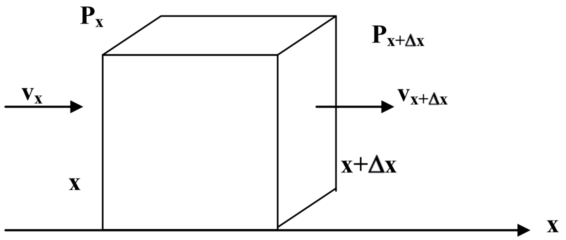


Figure 3.2: *Volume flow through a volume element.*

The mass conservation principle states that;

mass flow in – mass flow out = rate of change of mass in volume element

or by using mathematical terms for a compressible fluid through the volume element as;

(3.4) $$\nabla(\rho v) = -\frac{\partial(\phi\rho)}{\partial t} \quad .$$

From Darcy's law for a single fluid flow we have that;

(3.5) $$v = -\frac{k}{\mu}\nabla p \quad .$$

Here v is the Darcy velocity, k is the permeability, μ is the fluid viscosity and ∇p is the pressure gradient. Combining Darcy's law with the mass conservation principle will provide us with the following equation for fluid flow in a porous medium;

(3.6) $$\nabla\left(\frac{\rho k \nabla p}{\mu}\right) = \phi\frac{\partial(\rho)}{\partial t} \quad .$$

In well-testing, it is more practical to consider the flow in the radial direction towards the well and thus it is common to rewrite[(21)] the equation in radial coordinates as;

(3.7) $$\frac{1}{r}\frac{\partial}{\partial r}\left(\frac{\rho k r}{\mu}\frac{\partial p}{\partial r}\right) = \phi\rho c\frac{\partial p}{\partial t} \quad .$$

This equation is a second order non-linear differential equation since μ, ρ, ϕ, k and c all are functions of the pressure. However, the coefficients are frequently considered constant with the exception of ρ which will be a function of pressure in the case of compressible fluids. As a final step, the equation above can be simplified if ∂p/∂r is considered small and can then be rewritten as;

(3.8) $$\frac{1}{r}\frac{\partial}{\partial r}\frac{r\partial p}{\partial r} = \frac{\phi\rho c}{k}\frac{\partial p}{\partial t} \quad .$$

This equation is known as the *radial diffusion equation*. This equation can further be solved for various reservoir, well, boundary and initial conditions. Before we illustrate how this equation can be solved, we will introduce another important concept in well-testing referred to as dimensionless variables.

3.2 Dimensionless Variables

The use of dimensionless variables in well-testing provides us with easy and general equations. They simplify the equations by embodying well and reservoir parameters. The solutions obtained by using dimensionless variables will be independent of any particular system and provide is with "model" solutions. Note that while all equations until this point have been given in SI units, they will from now on be given in field units only. The variables used for a homogenous reservoir (as described above) in well-testing are in *field units* given as;

$$r_D = \frac{r}{r_w} \quad , \tag{3.9}$$

$$t_D = \frac{0.000264\ kt}{\phi c \mu r_w^2} \quad , \tag{3.10}$$

$$P_D = \frac{hk}{141.2 qB\mu}(p_i - p_{wf}) \quad . \tag{3.11}$$

In these equations, r is the radius at where the pressure is measured *(ft)*, r_w is the well radius *(ft.)*, k is the permeability *(md)*, t is the time *(hours)*, μ is the viscosity *(cp)*, ϕ is the porosity (*fraction), c* is the total compressibility *(1/psi), q* is the fluid flow rate *(std/d), B* is the B-factor *(rtb/stb)*, p_i is the initial reservoir pressure *(psi)*, p_{wf} is the well flow pressure *(psi)*, *h* is the *r*eservoir height *(ft)*. By the use of dimensionless variables, *eq. (3.8)* can be rewritten as;

$$\frac{\partial P_D^2}{\partial^2 r_D} + \frac{1}{r_D}\frac{\partial P_D}{\partial r_D} = \frac{\partial P_D}{\partial t_D} \quad . \tag{3.12}$$

We will now illustrate how this equation can be solved in Laplace space in the case of a homogenous infinite reservoir.

3.3 Homogenous Infinite Reservoir

By a homogenous reservoir is meant a reservoir where the reservoir properties, such as permeability and porosity, are assumed uniform throughout the entire reservoir. The reservoir is in addition considered infinite, i.e. the well is assumed to produce from an infinite large reservoir. We need to define initial and boundary conditions for such a reservoir to solve the differential equation given by *eq. (3.12).*

The initial pressure at *t=0* is assumed equal the initial pressure p_i throughout the reservoir. This can be mathematically stated in dimensionless variables as;

1. $P_D = 0 \quad for \quad t_D = 0 \quad at \quad all \quad r_D$.

We choose to have a constant rate solution, i.e. we choose the rate in the well to be constant. This can be mathematically expressed by Darcy's law (we assume that Darcy's law is valid in the wellbore) in dimensionless variables as;

2. $\left(\frac{\partial P_D}{\partial r_D}\right)_{r_D=1} = -1 \quad for \quad t_D > 0$.

The pressure at large distances away from the well, i.e. when $r \to \infty$, is assumed equal the initial pressure at any time and can be expressed in dimensionless variables as;

3. $(P_D)_{r_D \to \infty} = 0 \quad for \quad all \quad t_D$.

We would like to solve *eq. (3.12)* by the use of the Laplace transform and with the initial and boundary conditions as given above. In the Laplace transform, the equations are transformed to the Laplace space and solved more easily than in the time space. After solving the equations in the Laplace space, the solutions are inverted back to the time space by an inverse Laplace transform. The inverse Laplace transform needs to be done numerically since the solutions obtained in Laplace space will be too complicated to solve analytically. They can however efficiently and accurately be inverted numerically to the time domain through the Stehfest routine**(26)** described later in this chapter. This solution is preferred by most authors (us included) due to the combination of speed and accuracy offered by the Stehfest algorithm. If the equations instead were solved in the time domain, they would either be approximations or given as integral solutions. The use of approximations would result in inaccurate solutions while the integral solutions would result in computationally expensive (and thus slow) solutions. However, by solving the equations in the Laplace space and inverting the solutions to the time domain, we obtain a more accurate solution than from the time domain approximations and we obtain a faster solution than from the integral solutions in the time domain.

Hence we seek to solve the partial differential equation given by *eq. (3.12)* and the initial and boundary conditions above by a Laplace transform. Let

$$P_D(z) = P_D(r_D, z) = L[P_D(r_D, t_D)] \quad , \tag{3.13}$$

where L is the Laplace transform and z the Laplace variable. By Laplace transforming *eq. (3.12),* it can be rewritten as;

$$\frac{d^2 P_D(z)}{d{r_D}^2} + \frac{1}{r_D}\frac{dP_D(z)}{dr_D} = zP_D(z) - P_D(0) \quad . \tag{3.14}$$

Conditions 1 to 3 can also be rewritten in the Laplace domain as;

1. $P_D(z) = 0 \quad for \quad t_D = 0 \quad at \quad all \quad r_D$,

2. $$\left(\frac{\partial P_D(z)}{\partial r_D}\right)_{r_D=1} = -\frac{1}{z} \quad for \quad t_D > 0 \quad ,$$

3. $$(P_D(z))_{r_D \to \infty} = 0 \quad for \quad all \quad t_D \quad .$$

Applying condition 1 with *eq. (3.14)* allows us the rewrite the equation as;

(3.15) $$\frac{d^2 P_D(z)}{dr_D{}^2} + \frac{1}{r_D}\frac{dP_D(z)}{dr_D} = zP_D(z) \quad .$$

The solution of this differential equation is available in many calculus books**(43)** and given as;

(3.16) $$P_D(z) = AI_0(r_D\sqrt{z}) + BK_0(r_D\sqrt{z}) \quad .$$

where I_0 and K_0 are zero order Bessel functions of first and second kind. We further utilize conditions 2 and 3 to determine *A* and *B* in the above equation. If we start with condition 2, we first need to determine the derivatives of the two Bessel functions;

(3.17) $$\frac{d}{dr_D}[I_0(r_D\sqrt{z})] = \sqrt{z}I_1(r_D\sqrt{z})$$

and

(3.18) $$\frac{d}{dr_D}[K_0(r_D\sqrt{z})] = -\sqrt{z}K_1(r_D\sqrt{z}) \quad .$$

By inserting the initial and boundary conditions, given by conditions 2 and 3, into *eq. (3.16),* we obtain;

(3.19) $$A\sqrt{z}I_1(\sqrt{z}) - B\sqrt{z}K_1(\sqrt{z}) = -\frac{1}{z}$$

and

(3.20) $$AI_o(r_D\sqrt{z})_{r_D \to \infty} - BK_0(r_D\sqrt{z})_{r_D \to \infty} = 0 \quad .$$

These two equations are solved simultaneous to obtain the expressions for *A* and *B*. For large function arguments, K_0 approaches zero ($r_D \to \infty$) while I_0 approaches infinite for large arguments ($r_D \to \infty$). Hence, the only way to solve *eq. (3.20)* is by letting *A=0.* We can next determine *B* from *eq. (3.19)* (since *A=0*) as;

(3.21) $$B = -\frac{1}{z^{3/2}K_1(\sqrt{z})} \quad .$$

Substituting B into *eq. (3.16)* gives us the solution in dimensionless variables for a homogenous infinite acting reservoir with radial flow towards the well as;

$$(3.22) \qquad P_D(z) = \frac{K_0(r_D\sqrt{z})}{z^{3/2}K_1(\sqrt{z})} \quad .$$

In practice, the pressure response in any reservoir, sooner or later will be influenced by the reservoir boundaries. In this case we need to change the above boundary conditions to take into account the effect of boundaries at the pressure response. However as the model with finite boundaries is not utilized in the later well-test analysis, further description will not be required in the scope of this work.

The equation for the infinite reservoir was given in Laplace space. This means that we need to convert the equations back to the time domain to have the pressure as a function of time. This is commonly accomplished by the use of the Stehfest algorithm explained next.

3.4 Laplace Inversion

To get the pressure response in the time domain from Laplace equations such as *eq. (3.22),* an inversion scheme is required to convert the pressure response in the Laplace domain to the pressure response in the time domain. Converting these equations by analytical methods will in most cases not be possible due to the complex nature of the model equations. Because of this, it is common to apply a numerical Laplace inversion scheme.

There are many methods available for numerical Laplace inversion, but perhaps the most common and popular algorithm for inversion of equations in Laplace space, is the *Stehfest* algorithm **(26)**. The *Stehfest* algorithm applies a linear convergence scheme for a series approaching the inverse. This scheme is given as follow;

$$(3.23) \qquad P_D(t_D) = \frac{\log(2)}{t_D}\sum_{i=1}^{N} V_i P_D(z_i)$$

where z_i and V_i are given as;

$$(3.24) \qquad z_i = \frac{\log(2)}{t_D} i$$

and

$$(3.25) \qquad V_i = (-1)^{N/2+i} \sum_{K=\frac{i+1}{2}}^{\min\left(\frac{N}{2},i\right)} \frac{K^{\frac{N}{2}+1}(2K)!}{\left(\frac{N}{2}-K\right)!(K)!(K-1)!(i-K)!(2K-i)!} \quad .$$

N is the number of terms used in the inversion scheme and is chosen based upon the required precision in the inversion. It is in addition required to be chosen as an even number. There are many possible choices for N. A good choice suggested by Ozcan et. al. **(44)** is N equal 16. The choice of N equal 10 is also commonly found in the literature. If N is chosen large, the accuracy of the inversion will improve, however the magnitude of *N* will in most cases be limited by computer precision.

In the previous sections, the general equation for description of a homogenous reservoir was given. However, we further need to include skin factor and wellbore storage in this equation. This is well or near well effects that will dominate the pressure response for early time fluid flow in the well considered.

3.5 Skin Factor

The invasion of mud filtrate during drilling can cause the reservoir to be damaged with a smaller permeability in the near well area. It is also possible that the formation has been fractured during drilling, causing an increase in the permeability close to the well. This extra pressure drop or extra pressure increase can be modelled by a skin factor given as;

$$S = 2\frac{141.2kh}{\mu q B}\Delta p_{skin} \quad . \tag{3.26}$$

This can be illustrated from *Figure 3.3* below, where the pressure as a function of the radius is shown. The extra pressure drop/increase close to the well is shown with the red lines. Usually the skin factor will be in the range from -5 for a fractured well and up to 20 for a heavily damaged well.

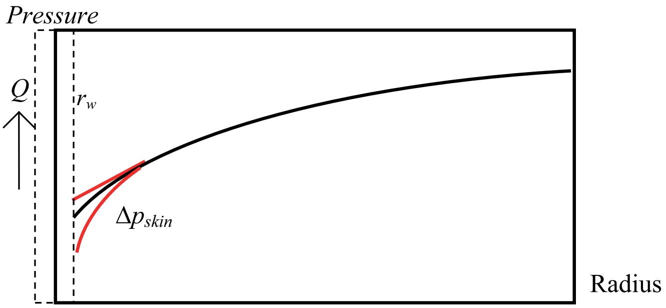


Figure 3.3: *Extra pressure drop/increase due to skin factor.*

3.6 Wellbore storage

Wellbore storage is also referred to as after flow, after production, after injection, wellbore loading or wellbore unloading. The wellbore storage effect is caused by opening and closing the well at the top. While closing the well at the top, fluid will for a short period continue to flow from the reservoir and into the wellbore, hence compressing the fluid in the wellbore. While opening the well, the initial fluid flow will then not be from the reservoir itself, but from the expansion of the fluid stored in the wellbore. If the well is completed without downhole packers between the tubing and the casing, one might in addition have a rising or falling fluid level in the wellbore when closing or opening the well. This is shown in *Figure 3.4.*

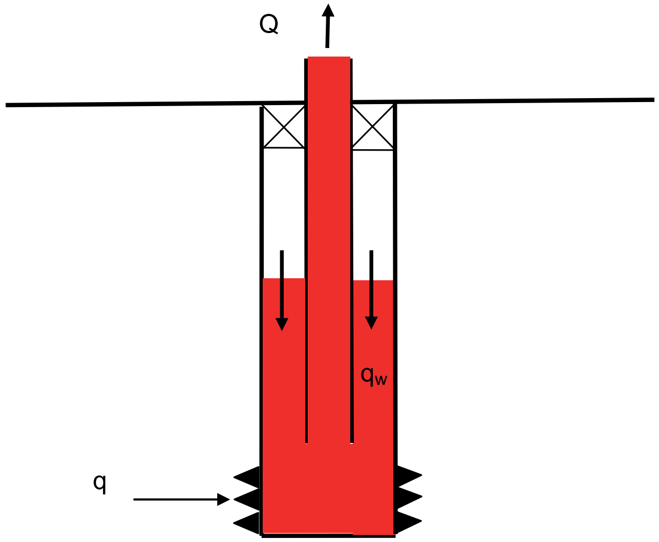


Figure 3.4: *Wellbore storage effects.*

The wellbore storage constant *C (std/psi)*, is used to describe volume changes of fluid in the wellbore caused by a pressure increase/decrease. The dimensionless wellbore storage constant is given as;

$$C_D = \frac{0.8936C}{\phi \mu c r_w^2} \quad . \tag{3.27}$$

One of the huge advantages of providing the equations in Laplace space is that this allows us to easily add wellbore storage and skin effects to the model solutions as[45];

$$P_{wD}(z) = \frac{zP_D(z)+S}{z(1+C_D z^2 P_D(z))} \quad . \tag{3.28}$$

In this equation, P_D can be the solution for the homogenous infinite reservoir given by *eq. (3.22)* or the solution for other kinds of wells and reservoirs considered, such as layered reservoirs or fractured wells.

3.7 Superposition Principle

The superposition principle is a powerful and useful tool that helps us to model the pressure response in cases with changing well rates. Assume that a well is producing at a constant rate. After a while, the rate is increased and/or decreased in a series of actions. A way to look at this is by imagining that a new well starts producing the increased/reduced amount *(q_2-q_1)* at a time t_1 at the exact same location as the initial well. Then a third well starts producing the increased/reduced amount *(q_3-q_2)* at time t_2, again in the exact same location as the previous two wells. This is done for each new rate change. In this way, *n* different rates can be modelled by *n* wells producing with rates equal the rate changes at times in the exact same location, starting at times $t_1,\ldots,t_n$. This can be mathematically stated as;

$$\frac{kh}{141.2B\mu}(p_i - p_{wfn}) = \sum_{i=1}^{n}(q_i - q_{i-1})P_{wD}(t_{D,n} - t_{D,i-1}) \quad , \tag{3.29}$$

and where P_{wD} in the time domain is found from the inverse Laplace transformation of *eq. (3.28)*. The above equation describes the pressure response p_{wfn} from the *n'th* rate change in the well considered.

In theory, a well-test can be performed with any changes of rate that takes place in the well by the use of *eq. (3.29)*, and we analyze the pressure response during this *n'th* rate change in a well-test analysis. The well might be closed and then starts to produce or the well might have been producing and is being closed. This is referred to as *drawdown* tests and *build-up* tests respectively and is illustrated in *Figure 3.5a* and *Figure 3.5b*. The actual well-test is performed with the pressure measurements taken after the last rate change. However it is important to notice that even if the pressure is measured only after the last rate change, ***all rate changes must be taken into account in eq.(3.29)***.

If the pressure is measured during several rate changes, we can analyze the pressure response during this period in what is referred to as a multi-rate well-test. A multi-rate test is illustrated in *Figure 3.5c*. Due to operational difficulties it is very difficult (if not impossible) to keep the rate constant during production (such as a drawdown test) and this is what makes multi-rate well-test analysis a valuable tool to obtain longer well-tests in time. The advantage of performing long well-tests is that the results from the well-tests will provide us with information from a larger part of the reservoir, i.e. the pressure response will reflect the reservoir properties of a larger area and possible reservoir boundaries.

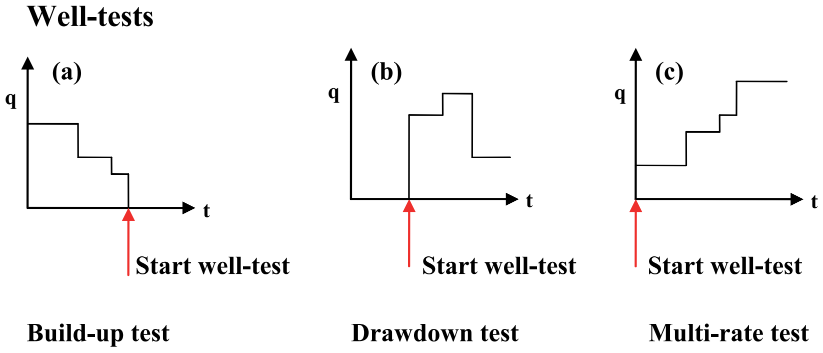
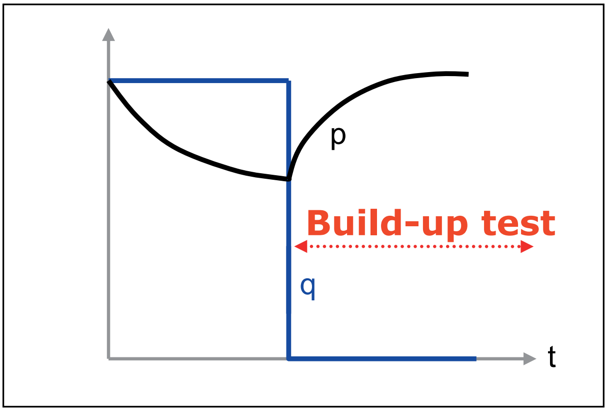


Figure 3.5: *Different types of well-tests illustrated.*

Even if one in theory can consider any series of rate changes in well-testing, build-up tests are normally the well-tests considered, also in this work. Inaccurate rate measurements make it very difficult to analyse the pressure response from wells where the rate is different from zero, as in drawdown and multi-rate well-tests. In general, the results from those well-tests will be poor. If the well is closed as in a build-up test, the rate is easily kept constant (equal zero). The amount of lost production and consequently the amount of money lost, often limits the time the well can be closed for well-test purposes. A build-up test is illustrated in *Figure 3.6.*



Typical Output

- **Permeability**
- **Skin Factor**
- **Average Reservoir Pressure**
- **Reservoir Boundaries**

Figure 3.6: *Pressure response during a build-up test.*

We have in the previous sections described some of the basic concepts in well-testing and we will continue by describing data preparation, model identification, non-linear regression analysis and confidence intervals, all important elements in modern well-tests analysis.

3.8 Modern Well-Test Analysis

In modern well-test analysis, a sequence of step is required. We will first provide a brief overview of all the steps in this section, and then provide a more detailed description of the individual steps in the subsequent sections. The first step is;

- Data preparation

With high resolution gauges the number of measurements available for well-testing can be tremendous. In addition outliers and noise might be present in the measurements. This step is a challenging step and is often more time consuming than the analysis itself.

- Model identification

Model identification is required to determine which mathematical model to apply with the observed pressure response, i.e. is the reservoir we consider homogenous, composite, layered etc. Model identification involves identification of characteristics in the measured pressure response. These characteristics are more easily identified by the use of *derivative plots,* where the pressure derivative is plotted along with the pressure.

- Non-linear regression

In non-linear regression the model parameters are tuned in order to minimize the difference between the estimated pressures, described by some chosen mathematical model, and the measured pressures. For this to work properly, it is critical that the correct mathematical model has been chosen in the previous step.

- Confidence interval estimation

In order to evaluate the goodness of the fit, confidence intervals are a valuable tool. If the confidence intervals are larger than some given predefined limits, little confidence can be given to the results obtained from the well-test analysis.

In the next sections, each of the above mentioned steps will be described in more details.

3.8.1 Data Preparation

In well-testing, a mathematical model is used to obtain an estimate of well and reservoir properties based upon the measured pressure response caused by rate changes. However, the measurements might be influenced by noise and outliers which makes the analysis more troublesome and in the worst case, gives erroneous results. In particular, the results from non-linear regression analysis might be influenced by outliers, while model identification (described in the next section) is more difficult to accomplish from noisy measurements.

Ideally, we would like to use all measurements directly together with the assumed mathematical model to estimate well and reservoir properties. However, even if modern well-testing is accomplished by the use of powerful computers, often the large number of measurements available would result in a too long computational time. Thus we usually chose to reduce the number of measurement to reduce the computational time of the analysis by data reduction. Signal de-noising, outlier removal and data reduction will be referred to as signal filtering and is described in chapter 5.

Next, if pressure and rate measurements taken at the same time, for some reasons should get different time stamps, this needs to be corrected for. There are many reasons why this can happen. The pressure transducer might need some time to stabilize, thus causing a delay in the pressure measurements, or it might take some time to change the flow rate (by changing the choke size), causing a delay in the rate measurements. This is referred to as *datum shifting effects* and might be necessary to correct for in a well-test if too large, more specifically, if it is in order of minutes. In addition, the pressure might not have been measured exactly at the beginning of a well-test. Thus the true start of the well-test is somewhere between the last pressure measurement before the well-test and the first measurement after the well-test. This is usually corrected for along with datum shifting effects.

After data preparation is completed, model identification is the next step. This is commonly accomplished by the use of derivative plots.

3.8.2 Model Identification

In well-testing, numerous models have been proposed to describe the pressure response in a well caused by different wellbore, reservoir and boundary properties. Many commercial well-test software have implemented between 10 and 30 different models (24; 46), while significantly more models are available in the literature. The obvious question is, how do we determine the model that adequately describes the pressure response in the well we consider? This is facilitated by the use of *derivative plots*, today a standard tool in well-test analysis.

Derivative plots were first introduced by Bourdet(47) and works in the way that they emphasize the characteristic model pressure response by a simultaneous presentation of *log Δp* vs. *log Δt* and *log (t dp/dt)* vs. *log Δt.* The pressure derivatives need to be estimated numerically from the pressure measurements as;

$$(3.30)\qquad t\left(\frac{\partial p}{\partial t}\right)_i=\left(\frac{\partial p}{\partial \ln t}\right)_i=\begin{bmatrix}\dfrac{\ln(t_i/t_{i-1})\Delta p_{i+1}}{\ln(t_{i+1}/t_i)\ln(t_{i+1}/t_{i-1})}+\dfrac{\ln(t_{i+1}t_{i-1}/t_i^{\,2})\Delta p_i}{\ln(t_{i+1}/t_i)\ln(t_i/t_{i-1})}\\ -\dfrac{\ln(t_{i+1}/t_i)\Delta p_{i-1}}{\ln(t_i/t_{i-1})\ln(t_{i+1}/t_{i-1})}\end{bmatrix}\,.$$

Calculations of numerical derivatives from the equation above will be heavily influenced by noise and outliers. The noise can be suppressed by the introduction of the *differentiation interval*(46; 42). Instead of choosing measurements next to each other (e.q. t_1 and t_2), the derivatives are calculated using measurements apart from each other (e.q. t_1 and t_5). We will later show that de-noising and outlier removal, as will be described in chapter 4 and 5, further can improve the calculation of derivative plots.

A typical derivative plot is shown in the *Figure 3.7*. If we consider a build-up test, the *Δp* is the pressure change since the time of the shut-in while *Δt* is the elapsed time since the shut-in. Typically the initial flow is dominated by wellbore effects as wellbore storage and skin. If present, wellbore effects can be identified as a unit slope line at early times (and illustrated as the red line to the left in *Figure 3.7*). Next there will be a transition period when the fluid flow starts to originate from the reservoir instead of the wellbore. If the reservoir is homogenous and the flow towards the well is radial, the derivative plot (lower line in *Figure 3.7*) will have a zero slope line (and illustrated as the red line to the right in *Figure 3.7*). During this period, the reservoir behaves as it is infinite in the sense that the pressure response is not influenced by boundaries.

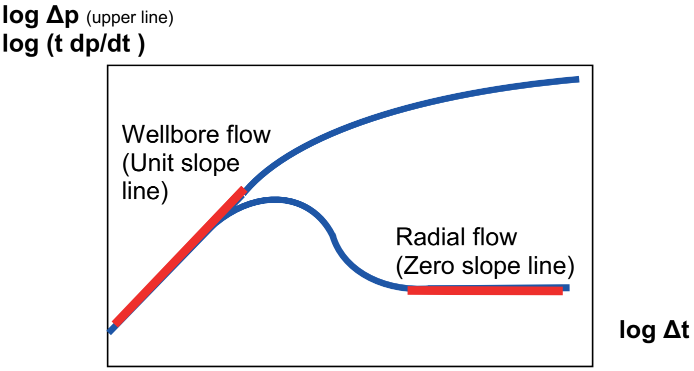


Figure 3.7: *Pressure derivative plot for a homogenous infinite reservoir.*

If the pressure response is influenced by boundary effects, the pressure derivative will increase at late time. In the case where the reservoir is closed i.e. the well produces from a defined and limited reservoir, a unit slope line at late time is seen (and illustrated as the red line to the right in *Figure 3.8*)

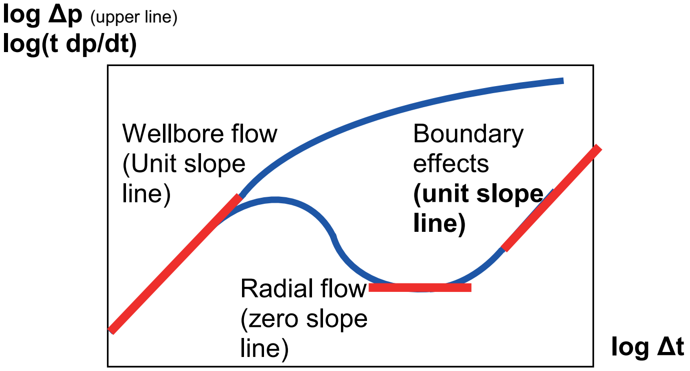


Figure 3.8: *Pressure derivative plot for a homogenous closed reservoir.*

Many of the model characteristics visible from the pressure derivative, is difficult (if not impossible) to identify from plots of the pressure itself. Thus in order to select the correct model for use in well-test analysis, derivative plots is a valuable tool. In most books and well-tests software, extensive libraries of pressure derivative plots are available to help facilitate model identification.

It should be noted that the model pressure response shown in these libraries are considered as typical derivative plots for a given model. Noise, long transition periods, large wellbore effect, fractures, boundaries close to the well etc. might mask some of the model characteristics. This needs to be taken into consideration when performing model identification.

Once we have identified a model believed to adequately describe the pressure response, model parameters such as permeability and skin factor, can be tuned in order to obtain the best possible match between the measured pressures and the pressures estimated from the mathematical model. This is accomplished by non-linear regression analysis and will be described next.

3.8.3 Non-Linear Regression Analysis

An important and valuable tool in the interpretation of well-test analysis is non-linear regression analysis. In non-linear regression analysis we seek to minimize the difference between the measured pressure and the estimated pressure, where the estimated pressure is obtained from a mathematical model assumed to adequately describe the well and reservoir considered. The minimization is performed by adjusting some selected model parameters, assumed to be unknowns, such as the permeability and/or the skin factor. Many different methods are available for performing this minimization and we have in this work chosen to use a method called Levenberg- Marquardt [(48)].

The Levenberg- Marquardt method is a gradient based optimization routine, i.e. it uses the function derivatives with respect to the unknown parameters for fast convergence towards the parameter values minimizing the square difference between estimated pressures and measured pressures. Numerical derivatives could be used, but this would be slower and less accurate compared to the use of analytical derivatives. Hence we will in this work prefer to apply analytical derivatives.

Frequently encountered in non-linear regression analysis is convergence failure, i.e. we do not find the parameters that minimize the difference between the model pressure response and the estimated pressure response. If we encounter convergence failure, we usually converge toward a local minimum and not the global minimum. A good initial estimate of the unknown parameters will in general improve the convergence. In well-testing, a good initial estimate of the parameters might be obtained from previous well-tests or from graphical analyses like Horner plot etc.

One of the main advantages of applying non-linear regression analysis in well-testing, is to obtain confidence intervals for the estimated parameters. This will provide us with an estimate of the uncertainty in the analysis. It should be noted that even if the visual match in some cases can appear to be good, the uncertainty in the analysis can be large. A well proven way to obtain an estimate of the uncertainties in the analysis is by the estimation of confidence intervals [(46)].

In non-linear regression analysis we seek to adjust the model parameters in order to minimize the difference between measured data from a process and estimated data from an assumed known process model where the process state is described by the parameter vector *v*. In our case, we will be working with discrete measured signals and the problem can thus be written as;

$$\min_{v\in\Re} J(v) = (f(t,v) - Y\,)^2 \quad , \tag{3.31}$$

where *Y* is the measurements of some process state, and *v* is a vector of model parameters of size *m*. *f(t,v)* is the estimated values from the process model at the same times as the process measurements. *J(v)* is the objective function to minimize.

Many different methods exist to accomplish this minimization. Mainly one can distinguish between those methods that make use of function derivatives or those that do not. The methods not using derivatives can be useful for problems that are highly non-linear and/or with several discontinuities. The methods that use derivatives are advantageous if the first order derivative is continuous. They are referred to as gradient based-methods. We have considered a gradient-based method in this work.

An approximation *f(v,t)** of the function *f(v,t)* can be written as a truncated Taylor series up to the quadratic term as;

$$f(t,v)^* = f(t,v_0) + \delta v^T g + \tfrac{1}{2}\delta v^T \mathrm{H}\delta x \quad , \tag{3.32}$$

and where

$$v = v_0 + \delta v \quad , \tag{3.33}$$

$$g = \left[\frac{\partial f}{\partial v}\right]_{v_0} \tag{3.34}$$

and

$$H = \left[\frac{\partial^2 f}{\partial v^2}\right]_{v_0} \quad . \tag{3.35}$$

The condition for the parameter vector *v* being at a minimum is given by;

$$\frac{\partial f}{\partial \delta v} = 0 \quad . \tag{3.36}$$

The method described above, is called the Newton method, and requires the calculation of the second order derivative matrix. This can be avoided in a modified method ignoring the second order derivative which is referred to as the Gauss-Newton method. A difficulty in the Gauss-Newton method might be ill-conditioning in the first order derivative matrix *g* in *eq. (3.34),* and modification was proposed by Levenberg and Marquardt**(48)**. The modified Gauss-Newton method is referred to as the Levenberg-Marquardt method. This method works by adding a constant to the diagonal elements of *g* to increase the magnitude of the eigenvalues. One of the most popular implementation of this method can be found in Netlib [(49)] by Burton et. al. [(50)]

The global minimum, i.e. the parameter vector giving the minimum value of the objective function *J(v),* might often be difficult to find. Often only a parameter vector corresponding to a local minimum of the *J(v)* is found if the initial parameter estimates are chosen away from the parameter vector that will result in the global minimum of *J(v)*. This is illustrated in *Figure 3.9* for the case where the parameter vector contains one element. With several elements in the parameter vector, high correlation between the different elements of the parameter vector will make this even harder. The magnitude of the correlation between the different parameters can be found from the covariance matrix. The covariance matrix will be introduced in the next section when we address confidence intervals.

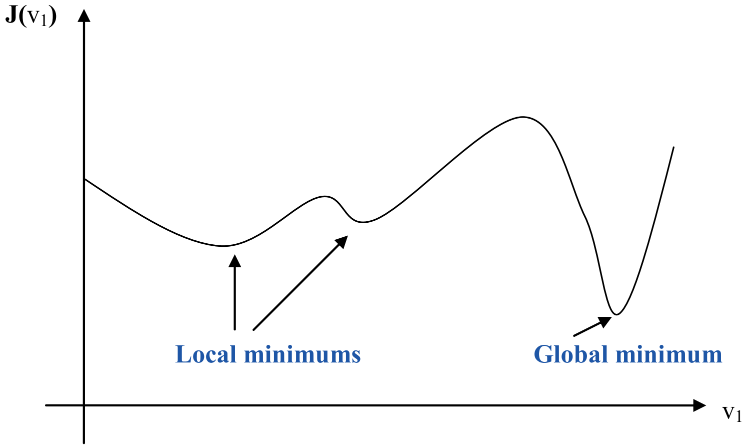


Figure 3.9: *Global minimum and local minimums for a parameter vector with one element, v_1.*

3.8.4 Confidence Intervals

To get an idea of the uncertainty in the estimated parameters, confidence intervals can be included. In order to obtain confidence intervals from regression analysis, we need to make the following assumptions; To start with, the mathematical model used in the well-test analysis must be correct. Secondly, the model is assumed to be linear in the region close to where the global minimum is found, and finally, the error in the variance is a result of measurement errors. If we rewrite *eq. (3.31)* using matrix notation, the objective function to minimize is given as;

$$(3.37) \qquad \min_{v\in\Re^m} J(v) = [f\ (t,v) - Y]^T [f\ (t,v) - Y] \quad ,$$

where *v* is the parameter vector with *m* elements and *n* is the number of measurements. If we assume that we have chosen the correct model for *f(t,v),* the measurements *y* from the process model can be given as;

$$Y = f(t,v) + \varepsilon \quad , \tag{3.38}$$

where the measurement error *ε* is assumed to have a normal distribution with zero mean and variance σ^2. The expected error is given as;

$$E[f(t,v) - Y]^T [f(t,v) - Y] = \begin{bmatrix} \sigma_1^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_m^2 \end{bmatrix} = \Sigma \quad , \tag{3.39}$$

where *Σ* is the covariance matrix. If the model is assumed linear around the global minimum, a Taylor expansion with the first two terms of the objective function around the true parameter vector *v* is given as;

$$J(v + \Delta v) \approx J(v_0) + \frac{\partial J(v_0)}{\partial v_0}(v - v_0) \quad . \tag{3.40}$$

We will try to minimize the objective function around the true parameters and differentiate with respect to the parameter vector *v* as;

$$\frac{\partial J(v)}{\partial v} \approx \frac{\partial J(v_0)}{\partial v_0} + \frac{\partial J^2(v)}{\partial v_0^2}(v - v_0) = 0 \quad . \tag{3.41}$$

By differentiating the original objective function and by performing some calculations, we can obtain the covariance matrix of the parameters estimate **(51)** as;

$$C_{m \times m} = (A^T \Sigma^{-1} A)^{-1} \quad , \tag{3.42}$$

where the sensitivity matrix A is given as;

$$A_{n \times m} = \frac{\partial f(t,v)}{\partial v} \quad . \tag{3.43}$$

Further, we can introduce the confidence level, where an estimate of the confidence interval for each parameter v_i is given as;

$$v_i \pm t_{n-m}(1 - \theta/2)\sqrt{c_{ii}} \quad , \tag{3.44}$$

c_{ii} is the diagonal elements of the covariance matrix C_{mxm} and the t_{n-m} is a *t-distribution* found from look-up tables with a given risk level *θ*. This is also referred to as a *Student t-test* **(51; 52)**. The ± in the expression above is used to obtain the upper and lower limit of the confidence interval. If for instance the assumption of model linearity close to the minimum is wrong, the estimate of the confidence intervals will be to optimistic.

As mentioned in the previous section, the covariance matrix can be used to investigate the correlation between the different parameters. If the only non-zero elements in the matrix are along the diagonal, we do not have any correlation between the parameters. On the other hand, many non-zero and large elements outside the matrix diagonal will indicate high correlation between the parameters.

We have in the previous section provided the general framework required for understanding non-linear regression analysis. We have chosen to apply a non-linear gradient based method. Hence we need to provide the derivatives for the unknown parameters for the models we consider. We will start by providing the general derivatives related to the superposition equation given by *eq. (3.29)*.

3.9 Model Derivatives

We will start by considering the superposition equation given in chapter 3.7 as;

$$(3.29)\quad p_{wfn} = p_i - 141.2\frac{\mu B}{kh}\left(\sum_{j=1}^{n}\Delta q P_{wD}(t_{Dn} - t_{D(j-.1)})\right) \ ,$$

where p_{wfn} is the pressure in the well at time t_n influenced by the *(n-1)* previously rate changes. P_{wD} can in this equation be any model equation describing any reservoir, well or boundary conditions. t_D and C_D were given as;

$$(3.10)\quad t_D = \frac{0.000264\ tk}{\phi\mu c r_w^2}$$

and

$$(3.27)\quad C_D = \frac{0.8936C}{\phi\mu c r_w^2} \ .$$

The superposition equation is common for all reservoir models considered since it only takes into consideration rate changes in a single well. All model specific properties reside inside the P_{wD} in *eq. (3.29)*. Hence we will start by providing the pressure derivatives of the above equations with respect to the parameters commonly considered as unknowns before we continue with model specific pressure derivatives. In general, we considered the following parameters as unknowns for all models considered in well-testing;

- Skin factor
- Permeability
- Wellbore storage
- Initial pressure
- (Porosity)

Recall that the porosity is placed in parentheses, since the porosity usually is considered known from either logs or core samples. Hence we need to find the pressure derivatives with respect to the parameters listed above for the superposition equation.

The analytical derivative of *eq. (3.29)* with respect to some parameter *u* (such as permeability *k*, skin factor *S* etc.) is given as;

$$(3.45) \qquad \frac{\partial p_{wfn}(t\ )}{\partial u} = 141.2\frac{\mu B}{kh}\left(\sum_{j=1}^{n}\Delta q\left(\frac{\partial P_{wD}(t_{Dn}-t_{D(j-.1)})}{u_D}\frac{\partial u_D}{\partial u}\right)\right) \ ,$$

where u_D is the dimensionless expression of the parameter *u*. The pressure derivative with respect to permeability, porosity, skin, wellbore storage, and initial pressure in *eq. (3.29)* are with analytical derivatives respectively given for the pressure at time t_n as;

$$(3.46) \qquad \frac{\partial p_{wfn}(t)}{\partial k} = 141.2\frac{\mu B}{kh}\left(\sum_{j=1}^{n}\Delta q\begin{pmatrix}\dfrac{P_{wD}(t_{Dn}-t_{D(j-.1)})}{k} - \\ \dfrac{\partial P_{wD}(t_{Dn}-t_{D(j-.1)})}{\partial(t_{Dn}-t_{D(j-.1)})}\dfrac{\partial(t_{Dn}-t_{D(j-.1)})}{\partial k}\end{pmatrix}\right) \ ,$$

$$(3.47) \qquad \frac{\partial p_{wfn}(t\ )}{\partial \phi} = 141.2\frac{\mu B}{kh}\left(\sum_{j=1}^{n}\Delta q\left(\frac{\partial P_{wD}(t_{Dn}-t_{D(j-.1)})}{\partial(t_{Dn}-t_{D(j-.1)})}\frac{\partial(t_{Dn}-t_{D(j-.1)})}{\partial \phi}\right)\right) \ ,$$

$$(3.48) \qquad \frac{\partial p_{wfn}(t)}{\partial S} = 141.2\frac{\mu B}{kh}\left(\sum_{j=1}^{n}\Delta q\left(\frac{\partial P_{wD}(t_{Dn}-t_{D(j-.1)})}{\partial S}\right)\right) \ ,$$

$$(3.49) \qquad \frac{\partial p_{wfn}(t)}{\partial C} = 141.2\frac{\mu B}{kh}\left(\sum_{j=1}^{n}\Delta q\left(\frac{\partial P_{wD}(t_{Dn}-t_{D(j-.1)})}{\partial C_D}\frac{\partial C_D}{\partial C}\right)\right)$$

and

$$(3.50) \qquad \frac{\partial p_{wfn}(t\ )}{\partial p_i} = 1 \ .$$

The dimensionless time derivatives with respect to permeability and porosity, are respectively given as;

$$(3.51) \qquad \frac{\partial t_D}{\partial k} = \frac{0.000264\ \ t}{\phi \mu c r_w^2}$$

and

$$(3.52) \qquad \frac{\partial t_D}{\partial \phi} = -\frac{0.000264\ \ tk}{\phi^2 \mu c r_w^2} \ .$$

The dimensionless wellbore storage derivative with respect to wellbore storage is given as;

$$(3.53) \qquad \frac{\partial C_D}{\partial C} = \frac{0.8936}{\phi \mu c r_w^2} \quad .$$

As mentioned previously in chapter 3.3, there are several benefits of solving the equations in the Laplace space. We mentioned that the equations could be fast and accurately inverted to the time domain by the use of the Stehfest inversion scheme. Secondly, we showed in chapter 3.7 that skin factor and wellbore storage easily could be added to any model solution through *eq. (3.28)*. Now we will show that by providing the equations in Laplace space, this in addition gives us very simple derivatives for P_{wD} with respect to t_D. The relationship between the derivatives in time space and in Laplace space can be written as;

$$(3.54) \qquad \frac{\partial P_{wD}(t)}{\partial u_D} = L^{-1}\left\{ \frac{\partial P_{wD}(z)}{\partial u_D} \right\} \quad ,$$

where L^{-1} is the inverse Laplace transform and u_D is the parameter that the derivation is performed with respect to. Derivation in Laplace space with respect to the Laplace variable *z*, is straightforward and given as;

$$(3.55) \qquad P'_{wD}(z) = zP_{wD}(z) \quad .$$

Hence we can perform the derivation of the model specific P_{wD} function with respect to time t_D by multiplying P_{wD} with the Laplace variable *z* and subsequently perform the inverse Laplace transformation. This approach can be utilized for parameters such as permeability and porosity which reside inside the time variable t_D. The derivative of dimensionless pressure with respect to the permeability is given as;

$$(3.56) \qquad \frac{\partial P_{wD}(t\ )}{\partial k} = \frac{\partial P_D(t_D)}{\partial t_D}\frac{\partial t_D}{\partial k} = L^{-1}\left(\frac{\partial P_D(z)}{\partial z} \right)\frac{\partial t_D}{\partial k} = L^{-1}(zP_S)\frac{\partial t_D}{\partial k} \quad .$$

The expression is similar for the derivative of dimensionless pressure with respect to porosity and given as;

$$(3.57) \qquad \frac{\partial P_{wD}(t_D)}{\partial \phi} = \frac{\partial P_D(t_D)}{\partial t_D}\frac{\partial t_D}{\partial \phi} = L^{-1}\left(\frac{\partial P_D(z)}{\partial z} \right)\frac{\partial t_D}{\partial \phi} = L^{-1}(zP_S)\frac{\partial t_D}{\partial \phi} \quad .$$

These equations are then used with *eq. (3.46)* and *eq. (3.47)* to get the pressure derivative with respect to time for a well with changing rates. These results are very advantageous with respect to computational speed and easy implementation. We do not have to develop specific equations for calculations of pressure derivatives with respect to permeability and porosity for each model considered.

Now recall from chapter 3.7 that one might add skin and wellbore storage to obtain P_{wD} from P_D for any model considered by the equation;

$$(3.28) \qquad P_{wD}(z) = \frac{zP_D(z)+S}{z(1+C_D z^2 P_D(z))} \quad ,$$

where P_{wD} is the pressure response in the well taking into account wellbore storage and skin effects. We can further find the derivatives with respect to skin factor and wellbore storage from *eq. (3.28)* as;

$$(3.58) \qquad \frac{\partial P_{wD}(t_D)}{\partial S} = L^{-1}\left(\frac{\partial P_{wD}(z)}{\partial S}\right) = L^{-1}\left(\frac{1}{z(1+C_D z^2 P_D(z))}\right)$$

and

$$(3.59) \qquad \frac{\partial P_{wD}(t_D)}{\partial C_D} = L^{-1}\left(\frac{\partial P_{wD}(z)}{\partial C_D}\right) = L^{-1}\left(\frac{-z^3 P_D(z)*(zP_D(z)+S)}{(z+C_D z^3 P_D(z))^2}\right) \quad .$$

Again, the huge advantage is that the above equations are general. They can be applied with any well-test model specified by a *$P_D(z)$* function. The expression above are substituted back into *eq. (3.48)* and *eq. (3.49)* to take into account a well with changing rates.

The derivatives of the model equation have until now been general without assuming any particular reservoir, wellbore or boundary models in P_D. For further calculations of derivatives for application in non-linear regression analysis, we need to include specific well, reservoir and boundary models described by the *$P_D(z)$* function. In this work the following models have been implemented;

Well models:

- Partial perforated well
- Fractured well – Infinite conductivity fracture.
- Fractured well – Finite conductivity fracture.
- Fractured well – Uniform flux fracture.

Reservoir Models:

- Homogenous Reservoir
- Layered Reservoir
- Composite Reservoir
- Dual Porosity Reservoir

Boundary Models:

- Infinite Reservoir
- Finite Circular Reservoir

Although all models above have been implemented as a part of this work, only the homogenous model in an infinite reservoir is actually utilized. Thus any further description of the other implemented models is not required.

For the homogenous infinite model, usually the permeability, skin factor, wellbore storage and initial pressure are considered as unknowns. They are all given by the general equations in this section. Hence we do not need to provide any additional equations for the derivatives in this case. Thus by substituting the expression for the dimensionless pressure in a homogenous model given by;

$$(3.22) \qquad P_D(z) = \frac{K_0(\sqrt{z})}{z^{3/2}K_1(\sqrt{z})}$$

into *eq. (3.56)*, *eq. (3.57) eq. (3.58)* and *eq. (3.59)*, we obtain the derivatives of P_{wD} with respect to permeability, porosity, skin factor and wellbore storage respectively. These equations are then substituted into *eq. (3.46), eq. (3.47) eq. (3.48)* and *eq. (3.49)* to take into account a well with changing rates. The derivative with respect to the initial pressure is given by *eq. (3.50).*

3.10 Summary

We have in this chapter provided the equation required to describe a reservoir assumed to be homogenous with respect to reservoir properties such as permeability and porosity and where the reservoir has infinite extension. The equation was converted to Laplace space and solved by the Stehfest scheme. This provided us with both a fast and an accurate solution for the pressure response in the well and reservoir considered. Another benefit by solving the equation in Laplace space was that we easily could add the effects of wellbore storage and skin to the model equation. The solution assumed a constant flow rate into the well, however, the use of the superposition principle made us capable of taking into account a well with changing rates. The concepts of modern well-test analysis were described next including data preparation, model identification, non-linear regression analysis and estimation of confidence intervals. The non-linear regression method chosen in this work is called the Levenberg-Marquardt method. This method minimizes the square difference between the measured pressures and the pressures estimated from a mathematical model assumed to adequately describe the well and reservoir considered. In non-linear regression analysis, we obtain an estimate of the parameters considered as unknown along with their confidence intervals which gives us an estimate of the uncertainty in the estimated parameters. We have chosen to use a gradient based optimization routine which requires the pressure derivatives with respect to the parameters considered as unknown. For the homogenous infinite model, the derivatives for the parameters commonly considered as unknown were given. This included permeability, skin factor, wellbore storage constant and initial pressure. In some cases, also the porosity can be considered an unknown, and the derivative of the pressure with respect to porosity was given as well. We have in this chapter only provided the derivatives for the homogenous infinite reservoir. Additional models have been implemented, but they are not utilized in this work and are thus left out.

4 Wavelet Analysis

In this chapter, we will develop the ideas and theory behind wavelet analysis. Wavelet analysis has been shown to be in particular useful for signal de-noising(53; 54; 55). Consequently we have in this work chosen to use wavelet analysis as a building block in a system for automatic signal filtering. Recall that signal filtering includes outlier removal, de-noising and data reduction. To show why wavelet analysis is so useful in signal de-noising, we need to carefully develop the theory behind wavelet analysis and wavelet de-noising, which will be the scope of this chapter. The results from our applications of wavelet analysis and wavelet de-noising will be given in chapter 5.

In wavelet analysis we choose to approximate the signal by the use of basis functions. We would like to determine one or more states of process which may be represented as a function of time. We set out to measure those states, typically discretely. The measured signal represents the combination of the desired state and the noise. Essentially, we want to estimate the process state from the measured signal. To do so, we are going to represent the process state as a continuous function. In fact, we will represent the process state as a linear combination of basis functions. A basis function is an element of a particular basis in a function space and any function within this function space can be represented as a linear combination of basis functions. Our estimate of the function is provided by the specification of the basis functions and corresponding weighting coefficients.

We can translate and dilate these basis functions in time to approximate different part of the signal with different resolutions. Hence a sum of basis functions is required for the approximation. It can be envisioned that by choosing different sets of basis functions, either with small or with large support, we can approximate the signal with different resolutions. The support of a function determines where the function is non-zero. We can obtain a fine approximation of the signal by applying narrow basis functions, or we can obtain a coarse approximation of the signal by applying broad basis functions. Actually we will develop a framework whereby some parts of the signal are approximated with narrow basis functions while other parts of the signal are approximated with coarse basis functions. This is what makes the wavelet analysis so suitable for analyzing signals from non-stationary processes and it will in addition be shown to be particularly valuable for applications such as de-noising. In wavelet analysis we can use many different basis functions for signal approximation. This provides us with a great flexibility since different function approximations might adapt well to different kinds of signals. Finally, the applications of wavelet analysis are well suited to be performed by a computer. All together this makes the wavelet analysis particularly well suited for investigating and describing non-stationary signals that typically occur in the oil industry, such as pressure, production and temperature.

The early work on wavelet analysis was performed by Morlet(56), Grossman(57), Meyer(56) and Mallat(58). However, it was the work by Daubechies(59) that really caught the attention to wavelet analysis. Daubechies developed a type of wavelets with wide applications and these wavelets are perhaps the most widely used wavelets today. Donoho and Johnstone (53; 54; 55) developed new methods for signal de-noising and compression based upon wavelet analysis. The wavelets by Daubechies and the wavelet de-noising by Donoho and Johnstone will both be described later in this chapter.

In this chapter, we will first describe signals and signal approximations. A signal can be represented by an approximation made by a set of basis functions. In wavelet theory these are called scaling functions and are introduced in the first section. However, often it is more convenient to have another set of basis functions that represent the difference between two signal approximations. These are called wavelet functions and will be explained next. The wavelet and scaling functions can be utilized together to better describe or represent a signal where the scaling function will give the approximation at some chosen resolution level while the wavelet function will provide the differences between this approximation and all finer signal approximations. How to obtain this function approximation of wavelet and scaling functions will be explained next by the use of the wavelet transform. In some cases, it might be useful to not only apply one basis in the function representation but two bases. This is called the *bi-orthogonal wavelet system* and allows a larger class of signals to be well represented by wavelets. All theory about wavelets will be given on sampled signals, although we in practice can sample the signal an infinite number of times. However, in the case where the signals are sampled a finite number of time on a finite interval, what is referred to as *boundary problems* might occur and is described next. It has been shown that the function representation obtained by the wavelet transform is well suited for de-noising signals in a method referred to as *Waveshrink* and we will continue by describing this method. Finally, it will be explained how to improve the de-noising properties by the introduction of slightly modified wavelet transform called the *stationary wavelet transform*.

4.1 Signals

The observations of a process state as a function of time are referred to as a *signal*. State variables are the primary objects used in modelling to describe the state of processes, such as temperature, pressure, etc. The process state might change with time and we need to continuously measure the process to adequately model and describe it. To represent the signals by the use of a computer, the signals can be monitored with some time interval *T*. When *N* samples at an interval *T* are obtained from a signal *S(t)*, it is said that the signal is discretized or sampled, and this is referred to as s_i where *i* is an integer index. We will in this chapter assume that all samples will be equally spaced. However unequally spaced signal measurements, which in reality often might encountered, will be discussed in chapter 5.

Even if the signal is sampled, it is still possible to have a continuous representation of the signal in the computers by the use of a function approximation to better describe the underlying signal process. Thus we can use the discrete signal measurements s_i at intervals *T* to obtain a signal approximation *f(t)* that can describe the signal at any time $t \in \Re$. An important feature of these approximations when applied in wavelet analysis is that they are required to be a part of a function space denoted $f \in L^2(\Re)$. This function space consists of all functions with a bounded square integral as;

$$\int_{-\infty}^{\infty} | f(t) |^2 dt < \infty \quad . \tag{4.1}$$

Hence the functions applied in the signal approximation are all required to be in the function space as described above.

All the theory about wavelet analysis in the next sections will be developed for continuously sampled signals, i.e. in theory the signals will be sampled an infinite number of times at an infinite domain. In practice, however, when the wavelet analysis is applied with a computer, the number of samples will be finite and sampled on a finite domain. The only adjustments in the wavelet analysis required to apply it with finite sampled signals on finite domains are provided in chapter 4.6, where solutions to what is referred to as boundary problems of finite sampled signals will be discussed.

We will first start with describing how a signal approximation can be obtained using scaling functions.

4.2 Scaling Function

We represent a signal approximation *f(t)* by the use of building blocks of a function $\phi \in L^2(\Re)$ as;

$$f(t) = \sum_k c(k)\phi_{j,k}(t) \qquad k \in Z \qquad j \in Z \tag{4.2}$$

where *c(k)* is called the expansion coefficients, $\phi_{j,k}(t)$ is chosen to be a set of *basis* functions called the expansion set, *k* is an integer index and *Z* denotes the set of all integers. The index *j* is a resolution index describing how broad or narrow the support of the basis functions chosen in the current function approximation is. If the functions are supposed to be a basis, the set of functions will be required to be independent and to span the vector space V_j as;

$$V_j = \overline{\underset{k}{Span}\{\phi_{j,k}(t)\}} \ . \tag{4.3}$$

If the $\phi(t)_{j,k}$ is a basis and spans the vector space V_j, this means that the set of expansion coefficients in *eq. (4.3)* will be unique for any particular *f(t)* considered. In the wavelet terminology these basis function $\phi(t)$ are called *scaling functions.* The larger the expansion set is, the greater the flexibility available in the function approximation, and a larger class of signals can adequately be approximated. A two dimensional family of scaling functions can be generated from the basic scaling function by scaling and integer translation as;

$$\phi_{j,k}(t) = \frac{1}{\sqrt{2^j}}\phi\left(\frac{t - 2^j k}{2^j}\right) \ . \tag{4.4}$$

By increasing *j* the scaling function will be broadened and by increasing *k* the scaling function will be moved along the x-axis. This means that by decreasing *j*, a finer approximation will be obtained, while increasing *j* will give a coarser approximation. If the functions are made very broad, most signal information will be lost. On the other hand, as the scaling functions are made narrower and approach zero, the function approximation will approach the signal itself.

The simplest of these scaling functions is the Haar basis where the basic scaling function is given as;

$$\phi(t) = \begin{cases} \frac{1}{\sqrt{2}} & if \quad 0 < t < 1 \\ 0 & otherwise \end{cases} \ . \tag{4.5}$$

The basic scaling function is referred to as the scaling function where one chooses *k=0* and *j=0* in *eq. (4.4)*. Three dilated and translated versions of the Haar scaling function are shown in *Figure 4.1*, where the relation between the basic scaling function defined in *eq. (4.5)* and the dilated and translated scaling functions, are given by *eq. (4.4).*

The use of scaling functions to approximate a signal at different resolution levels can be more formally explained by the introduction of a *multiresolution framework*. In the multiresolution framework, the different scaling functions with different resolutions will span different subspaces by *eq. (4.3)*. It can further be shown**(60)** that all information about the signal inside one subspace V_{j+1} will reside inside the subspace spanned by the scaling functions in subspace V_j;

(4.6) $$V_{j+1} \subset V_j \quad .$$

The relationship between all various subspaces spanned by scaling functions at various function resolutions will be related as;

(4.7) $$.....V_2 \subset V_1 \subset V_0 \subset V_{-1} \subset V_{-2} \subset .... \subset L^2$$

As *j* approach infinity all information about the signal is lost, while as *j* approaches minus infinity, any function can be represented adequately within the vector space considered, $f \in L^2(\Re)$.

(4.8) $$V_\infty = \{0\}$$

(4.9) $$V_{-\infty} = L^2$$

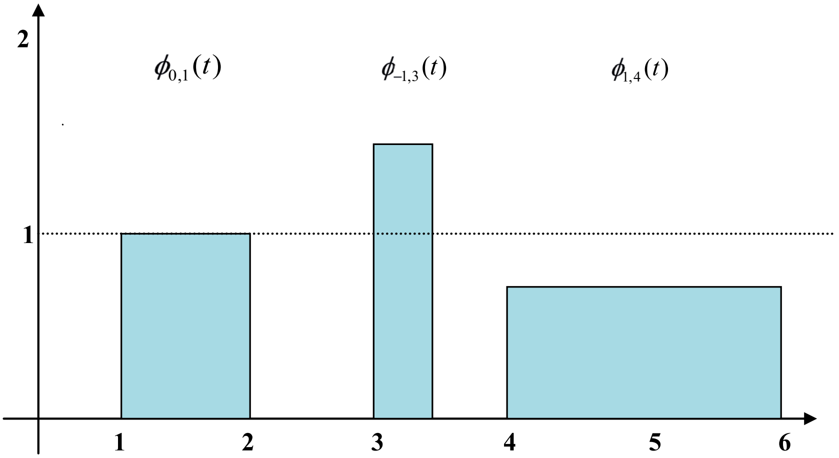


Figure 4.1: *Scaling function dilated and translated along the x-axis.*

Since each subspace spanned by the vectors $\phi(t)_{j+1,k}$ in V_{j+1} are contained within a subspace with higher resolution spanned by the vectors $\phi(t)_{j,k}$ in V_j, this allows us to express one scaling function as weighted sum of shifted scaling function as;

$$\phi(t) = \sum_n h(n)\sqrt{2}\phi(2t-n) \qquad n \in \mathrm{Z} \quad . \tag{4.10}$$

where h are called the *scaling coefficients* and relates one scaling function to another set of narrower scaling functions. n is an integer index. $\sqrt{2}$ is used to maintain the norm of the scaling function within the scale of two. This is a choice commonly made in wavelet theory, and other choices can be made as well **(60)**. *Eq. (4.10)* relates scaling functions at different resolutions to each other. Again as an example, we can relate the expression above to the simplest scaling function Haar, where we have that the scaling functions at two successive scales can be related as;

$$\phi(t) = \phi(2t) + \phi(2t-1) \quad . \tag{4.11}$$

This is also illustrated below in *Figure 4.2*. Thus by increasing the number of scaling functions, a higher resolution approximation can be obtained, and by *eq. (4.10)*, we see how this finer approximation is related to a coarser approximation by the use of scaling coefficients.

At this point we have shown how these scaling functions can give us different function approximations with different resolutions. However, it appears that it in many cases might be useful to introduce a second function, called a wavelet function, to represent the difference between two function approximations.

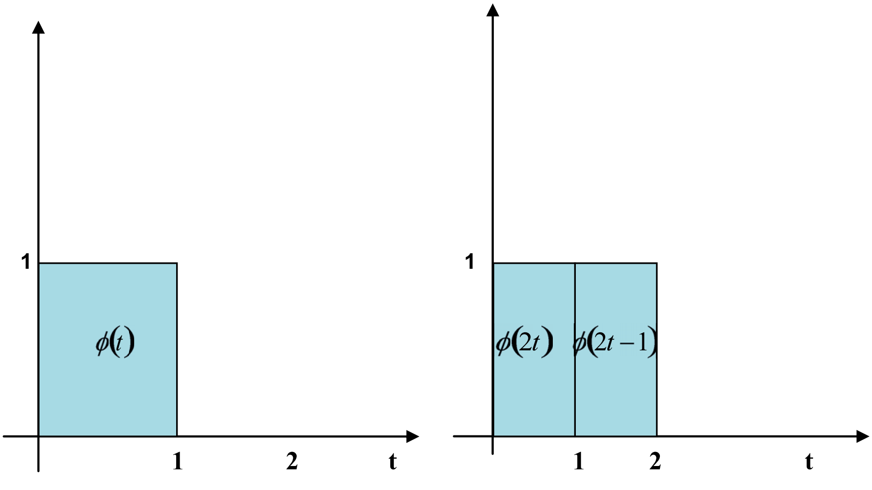


Figure 4.2: *The scaling function at one resolution level can be created as a sum of weighted scaling function at one finer resolution level.*

4.3 Wavelet Function

In the same manner as for the scaling function, translated and dilated wavelet functions can be created from translation and dilation of the basis wavelet function $\psi(t) \in L^2(\Re)$ as;

(4.12) $$\psi_{j,k}(t) = \frac{1}{\sqrt{2^j}}\psi\left(\frac{t-2^j k}{2^j}\right) .$$

By increasing *j* the wavelet can be broadened and by increasing *k* the wavelet can be moved along the x-axis. Again the simplest of the wavelet functions is the Haar function given as;

(4.13) $$\psi(t) = \begin{cases} \frac{1}{\sqrt{2}} & if \quad 0 < t < 0.5 \\ -\frac{1}{\sqrt{2}} & if \quad 0.5 < t < 1 \\ 0 & otherwise \end{cases} .$$

Translation and dilation of the Haar wavelet basis is shown in *Figure 4.3*.

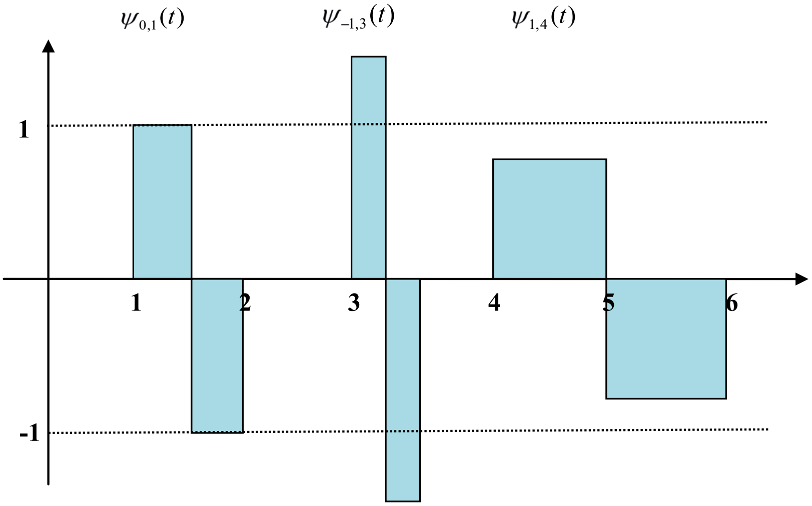


Figure 4.3: *Wavelet function translated and dilated along x-axis.*

We now require the wavelet and the scaling function to be orthogonal, i.e. their *inner-product*$^{(60)}$ is required to be equal zero as;

(4.14) $$\langle \phi_{j,k}(t), \psi_{j,l}(t) \rangle = 0 \qquad j,k,l \in \mathrm{Z} \qquad k \neq l \ .$$

By using to such closely related function we can now represent a signal, not by the scaling functions alone, but by both wavelet and scaling functions. To illustrate the use of the wavelet functions together with the scaling function, lets now consider two successive signal approximations at different resolutions f_j and f_{j+1}. What we would like to do is to express the difference between these two approximations. Any function approximation at resolution *j* can be written as the sum of the approximation at one coarser resolution level *j+1* and the difference between these two levels can be represented by the wavelet functions as;

(4.15) $$f_j(t) = f_{j+1}(t) + \Psi_{j+1}(t).$$

where Ψ_{j+1} is all the wavelet functions at resolution level *j+1*. Then successive approximations of the function, if we start at any arbitrarily chosen resolution level as, for instance resolution level *0*, can be written as;

(4.16) $$f_j(t) = f_0(t) + \Psi_0(t) + \Psi_{-1}(t) + \ldots.\Psi_{j-1}(t) \quad .$$

By requiring the wavelet and the scaling function to be orthogonal, this will in addition allow us to more formally explain the relationship between the different subspaces spanned by the scaling and the wavelet function by the multiresolution framework. Besides, as will be shown in the next section, this will allow a simple calculation of the expansion coefficients for use in the function approximation.

In the multiresolution framework, the differences between two successive subspaces will be spanned by the wavelet function and thus;

(4.17) $$V_j = V_{j+1} \oplus W_{j+1}.$$

In general by the use of the wavelet function (again if *j* is chosen equal to *0*), the signal can now be described by;

(4.18) $$L^2 = V_0 \oplus W_0 \oplus W_{-1} \oplus W_{-2} \oplus \ldots\ldots.$$

where ⊕ denotes an orthogonal sum of two subspaces. As with the scaling function, the wavelet function at one resolution level can be expressed as a weighted sum of shifted wavelet functions as;

(4.19) $$\psi(t) = \sum_n h_1(n)\sqrt{2}\psi(2t-n) \qquad n \in \mathrm{Z}$$

where h_1 is the *wavelet coefficients*. The wavelet coefficients are used to relate the wavelet functions at one resolution level to the next finer resolution level. Any signal can now be approximated, not by increasing the scale of the subspace spanned by the scaling function, but by using the wavelet function to span the difference between the spaces spanned by the scaling functions at different scales.

Since the wavelet functions and the scaling functions are chosen to be orthogonal it can then be shown(60) that the wavelet coefficients and the scaling coefficients are required to have the following relationship;

(4.20) $$h_1(n) = (-1)^n h(1-n) \quad .$$

In the literature, often only the scaling coefficients will be provided since the wavelet coefficients can be obtained from the scaling coefficients. It should however be noted that great confusion appears to exists and unfortunately these coefficients are often mixed.

By the use of *eq. (4.18)* from the multiresolution framework, a general expression for how to represent a function by series of scaling and wavelet functions can now be written as;

$$(4.21) \qquad f(t) = \sum_k c(k)\varphi_k(t) + \sum_{j=0}^{-\infty}\sum_k d_j(k)\psi_{j,k}(t)$$

where *k* is a finite or infinite integer index. This coarsest resolution level is often referred to as the primary resolution level and denoted as j_0, and the equations above, *eq. (4.18)* and *eq. (4.21)* thus becomes;

$$(4.22) \qquad L^2 = V_{j_0} \oplus W_{j_0} \oplus W_{j_0-1} \oplus W_{j_0-2} \oplus ...... \ ,$$

$$(4.23) \qquad f(t) = \sum_k c_{j_0}(k)\phi_{j_0,k}(t) + \sum_{j=j_0}^{-\infty}\sum_k d_j(k)\psi_{j,k}(t) \ .$$

Before we continue it might be enlightening to emphasize some of the advantages of representing a signal like this. The wavelet function was in *Figure 4.3* illustrated as an oscillating function that in general is well localized in time, i.e. it has local support meaning that the function is non-zero only on a finite interval. It can be shown [(60)] that all wavelets will have an oscillating behaviour where the square wave in *Figure 4.3* simply is one of the possible implementations. The wavelet can be shown to have many types of “wave shape” as long as it satisfies;

$$(4.24) \qquad \int_{-\infty}^{\infty}\psi(t) = 0 \ .$$

The wavelet being such an oscillating function will have importance for why we choose to use the signal approximation in *eq. (4.23)*. Recall that the difference between the finest signal approximations are represented with narrow wavelet function and this can be interpreted as representing the difference in the function approximations with fast oscillating or high frequency waves (wavelets). The difference between coarse function approximations will be given by broad wavelet functions and this can be interpreted as providing the difference between coarse function approximations by the use of slow oscillating of low frequency waves. In this context, since any signal can be represented by a large number of different wavelets where the different wavelets all are localized in time and in frequency, the representation of a signal like this is referred to as a *time-frequency representation*. The coefficients $d_j(k)$ in this representation *(eq. (4.23))* will provide the “strength” of the time-frequency localized wavelets in the different scales and locations.

We have argued above that the wavelet functions at different resolution levels might be envisioned to represent different frequencies in the signal since because of their oscillating behaviour. The strength of the coefficients in this representation can tell us where different frequencies are found within the signal approximation. Thus we do not only know if changes (in frequency) occur in the signal, but also when they occur. This is what makes the wavelet and scaling function representation of signals so useful when they are from non-stationary processes. In many cases, some or perhaps most of the wavelet coefficients in this signal approximation will be zero or close to zero. This is what makes this representation ideal for compression or de-noising of signals, i.e. we can represent the signal well by only a few large coefficients in *eq. (4.23).*

When representing a signal as explained above, it remains to explain how to obtain the coefficients for the scaling function and the wavelet function used in the signal approximation. This can be accomplished through the *wavelet transform*.

4.4 Wavelet Transform

The wavelet transform provides a process for obtaining the coefficients *(*$d_j(k)$ and $c_{j_0}(k)$*)* in the function representation in *eq. (4.23).* To obtain these coefficients, first recall that in section 4.2, we chose our function representation to be a set of basis functions. In the previous section we required these functions to be orthogonal. Consequently by choosing the scaling function and the wavelet function to be an orthogonal basis, we can (see Mallat [58]) use the inner product to calculate the *detail coefficients* in *eq. (4.23)* as;

$$d_j(k) = \langle S(t), \psi_{j,k}(t) \rangle = \int S(t) \bullet \psi_{j,k}(t) dt \tag{4.25}$$

and the *approximate coefficients* in *eq. (4.23)* from;

$$c_j(k) = \langle S(t), \phi_{j,k}(t) \rangle = \int S(t) \bullet \phi_{j,k}(t) dt \ . \tag{4.26}$$

where *S(t)* is the signal considered. The detail and approximate coefficients in this wavelet transform are called the wavelet expansion and the process of obtaining them is often referred to as *decomposition*.

However, often we do not have any explicit expression for the wavelet and scaling function as we did in the case of the Haar basis. This will make it difficult for us to obtain the wavelet and the scaling coefficient through *eq. (4.25)* and *eq. (4.26)*. However, if we do know the wavelet and the scaling coefficients, we can still obtain the detail and approximate coefficients by a slightly different approach. Accordingly it will be desirable to be able to calculate the detail and the approximate coefficients by utilizing the wavelet and scaling coefficients. It is possible to show[60] that by combining equation *eq. (4.19)* with *eq. (4.23)* we obtain;

$$d_{j+1}(k) = \sum_m h_1(m-2k) c_j(m) \qquad k,n \in \mathrm{Z} \tag{4.27}$$

where *m = 2k+n*, and where *n* and *k* is some finite or infinite integer indexes. Thus we can see that the detail coefficients at level resolution *j+1* can be calculated by convoluting the approximate coefficients at level *j*. Similar as for the detail coefficients we can for the approximate coefficients combine *eq. (4.10)* with *eq. (4.23)* to obtain;

$$c_{j+1}(k) = \sum_m h(m-2k) c_j(m) \qquad k,n \in \mathrm{Z} \tag{4.28}$$

where *m = 2k+n*, and where *n* and *k* is some finite or infinite integer indexes. The approximate coefficients at resolution level *j+1* are found in the same way as the detail coefficients by convoluting the approximate coefficient at level *j,* but this time with the scaling coefficients.

In addition, the number of approximate and detail coefficients at each coarser resolution level is halved. This is simply because we at the next coarser resolution level only applies half the number of scaling function in the signal approximation which can be represented by half the number of approximate coefficients. This removal of coefficients is referred to as down-sampling and denoted (↓), and in the case where every second sample is removed as in *eq. (4.27)* and *eq. (4.28),* the down sampling is denoted (↓2). This can be illustrated as shown below in *Figure 4.4*. The wavelet transform with up and down-sampling is referred to as the *dyadic wavelet transform*.

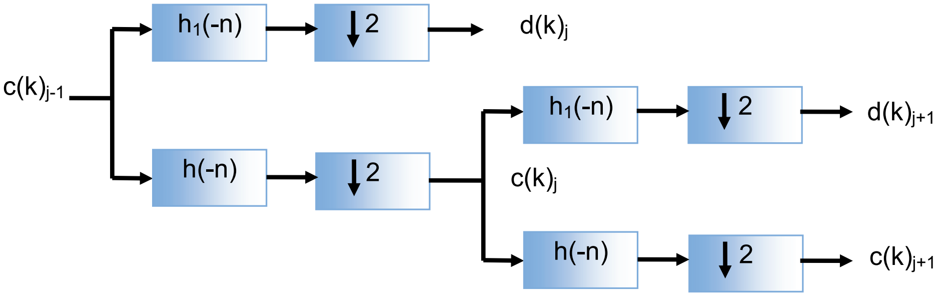


Figure 4.4: *Wavelet transform with down-sampling.*

When removing half of the approximate coefficients information about the signal will be lost in the signal approximation. However, all this information is stored within the wavelet coefficients. It can be shown [(60)] that it is possible to have what is referred to as a *perfect reconstruction* where the signal approximation at a finer level *j* can perfectly be reconstructed from the detail and approximate coefficients at level *j+1* as;

$$(4.29) \qquad c_j(k) = \sum_m c_{j+1}(m)h(k-2m) + \sum_m d_{j+1}(m)h_1(k-2m) \qquad k,n \in Z$$

where *m = 2k+n*, *n* and *k* is some finite or infinite integer index. Thus by convolution of the detail and approximate coefficients with the wavelet and scaling coefficients respectively, the approximation at one finer resolution level *j* with double the number of approximate coefficients is obtained. This doubling of the approximate coefficients at resolution level *j* compared to *j+1* is referred to as up-sampling and denoted (↑2). The scaling and the wavelet coefficients used in the reconstruction of the function approximations are the inverse of those used in the decomposition, and are thus often denoted *g* and $g_{1.}$ This to avoid mixing the wavelet and scaling coefficients utilized in the decomposition and reconstruction and we will from now on choose to denote *h(n)* as *g(n)* and $h_1(n)$ as $g_1(n)$ in the reconstruction in *eq.(4.29)*. (Note that *k-2m* was used as scaling coefficient index in *eq. (4.29)* in the reconstruction, while *m-2k* was used for the scaling coefficient index in *eq. (4.28)* in decomposition) *g(n)* and $g_1(n)$ are related to each other in the same manner as *h(n)* and $h_1(n)$ in *eq. (4.20)*. The reconstruction of the function approximation at level *j-1* is illustrated below in *Figure 4.5*.

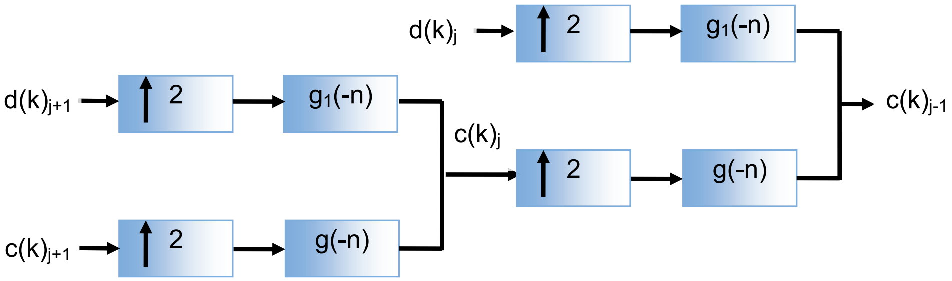


Figure 4.5: *Inverse wavelet transform with up-sampling.*

Although the advantage of using orthogonal basis in the wavelet analysis is clear, there are some cases where it might be preferred to not use an orthogonal system [(60)]. The next section will develop the bi-orthogonal wavelet system applying a non-orthogonal basis. This will provide us with a greater flexibility in the wavelet design and allows a greater class of signals to be well approximated by the wavelet and scaling functions.

4.5 Bi-orthogonal and Non-orthogonal Wavelets

Previously, we have stated that the relationship between the wavelet and the scaling coefficient (*h(n)* and *$h_1(n)$)* where given by *eq.(4.20)*. In this section we will develop the bi-orthogonal wavelet system where these restrictions can be relaxed. We now choose to use two different basis in the wavelet system, one for the wavelet function and one for the scaling function. However, to still have the perfect reconstruction as given by *eq.(4.29)*, it can be shown (60) that the scaling and the wavelet coefficient in the decomposition (*eq. (4.27)* and *eq.(4.28)*) and in the reconstruction *eq. (4.29)* of a bi-orthogonal wavelet system must satisfy the following relations;

$$\tilde{g}(n) = (-1)^n h(1-n) \tag{4.30}$$

and

$$g(n) = (-1)^n \tilde{h}(1-n) \quad . \tag{4.31}$$

We have chosen to use the notation $\tilde{h}(n)$ and $\tilde{g}(n)$ instead of *$h_1(n)$* and *$g_1(n)$* to avoid mixing the orthogonal and the bi-orthogonal wavelet coefficients since there for the bi-orthogonal wavelet system, as opposed to the orthogonal, are no relationship between $\tilde{h}(n)$ and *h(n)*. In the case of orthogonal wavelet, *h(n)* and *$h_1(n)$* were orthogonal to each other. In fact, it is possible to show the *h(n)* will be orthogonal even to translations of itself. Now we require a *dual basis* whose elements are not orthogonal to each other, but to the corresponding element in the expansion set, i.e. $\tilde{h}$ is orthogonal to *h*. Hence we have the name bi-orthogonal.

In the case of orthogonal wavelets, the number of scaling and wavelets coefficients in *h(n), $h_1(n)$, g(n)* and *$g_1(n)$* will be equal. Further it will be required that the number of coefficients is even. In the case of bi-orthogonal wavelet system, these restrictions are greatly relaxed. The number of scaling and wavelet coefficients might be either even or odd.

However, it is required that the difference in the number of scaling coefficient between $\tilde{h}$ and h is even.

Further we will simply repeat some of the theory described previously in this chapter in the case where bi-orthogonal wavelets are used instead of the orthogonal ones. In addition to *eq. (4.4)* relating the translation and the dilation of the scaling function, we will in addition for bi-orthogonal wavelet systems have that;

$$(4.32) \qquad \tilde{\phi}(t) = \sum_n \tilde{h}(n)\sqrt{2}\tilde{\phi}(2t-n) \qquad n \in Z$$

where $\tilde{\phi}$ in addition to ϕ from *eq. (4.4),* are the two dual basis used for the scaling functions in the signal approximation. Similar the translation and dilation for the wavelet function in the case of orthogonal case was given by *eq. (4.12).* For bi-orthogonal wavelet system we in addition have that;

$$(4.33) \qquad \tilde{\psi}(t) = \sum_n \tilde{g}(n)\sqrt{2}\tilde{\psi}(2t-n) \qquad n \in Z$$

where $\tilde{\psi}$ in addition to ψ from *eq. (4.12),* are the two dual basis used for the wavelet functions in the signal approximation. The multi-resolution framework described in section 5.2 and 5.3 can further be revised as;

$$(4.34) \qquad \ldots \subset V_2 \subset V_1 \subset V_0 \subset V_{-1} \subset V_{-2} \subset \ldots\ldots,$$

$$(4.35) \qquad \ldots \subset \tilde{V}_2 \subset \tilde{V}_1 \subset \tilde{V}_0 \subset \tilde{V}_{-1} \subset \tilde{V}_{-2} \subset \ldots\ldots,$$

and where

$$(4.36) \qquad V_j = \underset{k}{Span}\{\phi_{j,k}(t)\} \qquad \tilde{V}_j = \underset{k}{Span}\{\tilde{\phi}_{j,k}(t)\}$$

$$(4.37) \qquad V_j \perp \tilde{W}_j \qquad \tilde{V}_j \perp W_j$$

and where

$$(4.38) \qquad W_j = \underset{k}{Span}\{\psi_{j,k}(t)\} \qquad \tilde{W}_j = \underset{k}{Span}\{\tilde{\psi}_{j,k}(t)\}$$

For the orthogonal case we had that W_j where orthogonal to V_j, although the dual basis will in the case of bi-orthogonal wavelet system have much the same role. Hence we have four different basis that is used to form two hierarchies to span $L^2(\Re)$.

In general the benefits of bi-orthogonal wavelets are simpler wavelet design and that a larger class of signal adequately can be approximated by the scaling and wavelet functions. A more detailed explanation of why the bi-orthogonal wavelet system might be favourable for some type of signals is outside the scope of this work and details can be found within *Burrus et.al* **(60)**.

Finally note that it is possible to obtain even a third type of wavelet system where the restriction on the scaling and wavelet coefficients are even more relaxed, although they must satisfy the same dual basis as the bi-orthogonal wavelet system described above. One such wavelet systems are based upon the cubic spline;

$$(4.39) \qquad \psi(t) = \begin{array}{cc} 2(t+1)^2 & -1 \le t < -1/2 \\ -4t(t+1)-2t^2 & -1/2 \le t < 0 \\ -4t(t+1)+2t^2 & 0 \le t < 1/2 \\ -2(t-1)^2 & -1/2 \le t < 1 \end{array} .$$

For this wavelet system, $\tilde{\phi}$ is chosen equal ϕ, although the wavelet system is constructed in such a way that it satisfy *eq. (4.27)* and *eq. (4.28)*. However, this wavelet system does not satisfy *eq. (4.27)* and *eq.(4.28)* as the bi-orthogonal wavelet system does. These wavelets are referred to as non-orthogonal wavelets and the one we will consider later in this work is shown in *Figure 4.6.*

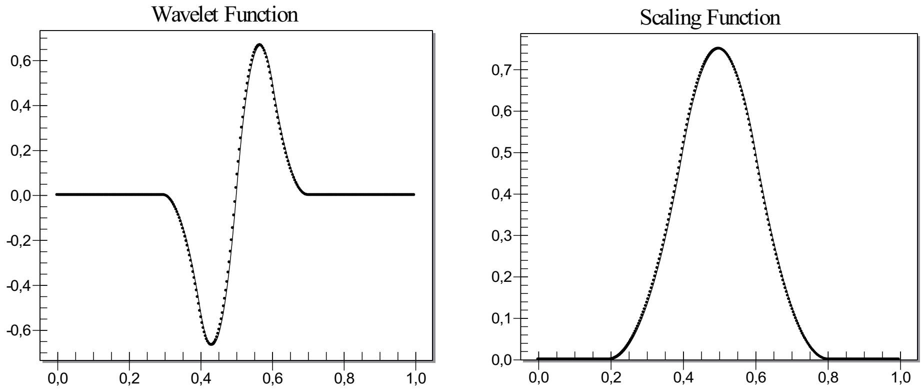


Figure 4.6: *Spline wavelet used in non-orthogonal wavelet system, wavelet function (left) and scaling function (right).*

The wavelet and the scaling coefficients required in the wavelet transform with *eq. (4.27)* and *eq. (4.28),* can be obtained through what is referred to as a *wavelet design*. By the wavelet and scaling coefficients obtained in this design process, we can calculate the detail and the approximate coefficients through *eq. (4.27)* and *eq. (4.28)*. Thus we will in fact perform the wavelet analysis without any specific knowledge of the scaling and wavelet function but only with the knowledge of the wavelet and the scaling coefficients. The next section will give a brief introduction on how these coefficients are obtained and some of the necessary constrains that must be put on *h(n)* and *$h_1(n)$* in the wavelet design process.

4.6 Wavelet and Scaling Coefficients

Until now we have stated that the scaling and wavelet coefficient, denoted *h(n)* and *$h_1(n)$* respectively, will depend upon the scaling and wavelet functions considered. Recall that the wavelet and scaling coefficients were required in order to relate the wavelet or scaling functions at one resolution level to the wavelet or scaling functions at the next coarser resolution level through *eq. (4.10)* and *eq. (4.19)*. These coefficients should not be confused with the detail and the approximate coefficients denoted $d_j(k)$ and $c_{j_0}(k)$ in *eq. (4.23)* used in the signal representation. How to determine these wavelet and scaling coefficients for a specific wavelet or scaling function has until now not been discussed. This section is hence intended to give a simple illustration on how the scaling and wavelet coefficients can be obtained in the case of the Haar and the Daubechies wavelets. Some ideas on how to obtain wavelet and scaling coefficients in general will also be discussed briefly, however for a detailed description on this, refer to Mallat(58) or Gao(61).

If we once more return to the Haar basis, we recall that the Haar basis was given by *eq. (4.5)* and *eq. (4.13)*. We further stated that it would be possible to obtain the scaling or wavelet functions at one resolution level as a sum of scaling functions from a finer resolution level as given by *eq. (4.10)* and illustrated by *eq. (4.11)*. However, for the recursion equation in *eq. (4.10)* to be satisfied, we are required to choose the scaling coefficients as $h(0) = 1/\sqrt{2}$ and $h(1) = 1/\sqrt{2}$. Thus the Haar scaling coefficient will simply be given as;

$$h = \left[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \right] \tag{4.40}$$

and the wavelet coefficients are obtained from *eq. (4.20)* as;

$$h_1 = \left[\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}} \right] . \tag{4.41}$$

Different wavelet and scaling functions can be specified directly by the wavelet and scaling functions $\phi(t)$ and $\psi(t)$ as long they are orthogonal and satisfy *eq. (4.4)* and *eq. (4.12)* (as in the case of the Haar wavelet above). Based on the wavelet and scaling functions, we then derive the wavelet and scaling coefficients required in the wavelet transform.

In cases where the wavelet and scaling function are not explicit given, which is usually the case, it is still possible to determine the wavelet and scaling coefficients through a *wavelet design.* It can be shown that all scaling coefficients must satisfy the following condition(60);

$$\sum_n h(n) = \sqrt{2} \tag{4.42}$$

and

$$\sum_n |h(n)|^2 = 1 \quad . \tag{4.43}$$

Thus the sums of the scaling coefficients are required to equal the square of two, while the norm of the scaling coefficients should equal 1. In the case of the Haar scaling coefficients we can easily see that these conditions are satisfied. In the case of more than two scaling coefficients, it can be shown(60) that in addition, the individual sum of even and odd elements must equal;

(4.44) $$\sum_n h(2n) = \sum_n h(2n+1) = \frac{1}{\sqrt{2}} \; .$$

In a wavelet design, the wavelet and scaling coefficients are determined from *eq. (4.42), eq. (4.43)* and *eq. (4.44).* First we choose the number of scaling coefficients to use and then we choose the individual scaling coefficients such that they satisfy *eq. (4.42) - eq. (4.44).* When utilizing four scaling coefficients or more to relate the scaling functions at one resolution level to the next coarser level, there might be some degrees of freedom left on how to choose the individual coefficients after the three conditions above are satisfied. In the case of the Haar basis, we have no degrees of freedom left, in other words, we do not have any flexibility on how to choose the coefficients since they are completely determined by the equations above. However, if we choose to use four scaling coefficients (refer to Burton et. al. [60]) we will have three equations *eq. (4.42)- eq. (4.44)* with four unknown scaling coefficients;

(4.45) $$\begin{gathered} h(-1)+h(0)+h(1)+h(2)=\sqrt{2} \\ h(-1)^2+h(0)^2+h(1)^2+h(2)^2=1 \; . \\ h(-1)^2h(1)^2+h(0)^2h(2)^2=0 \end{gathered}$$

n is chosen to range from -1 to 2. This is simply a selection made and could equally have been chosen to range from 0 to 3. The range of *n* will only affect the actual numerical implementation of the wavelet transform.

The remaining scaling coefficient can then be chosen based upon some desired design criteria [60]. This allows for the design of different scaling and wavelet coefficients, and consequently different wavelet and scaling functions, taking into consideration different signal properties. Daubechies designed a family of wavelets simply referred to as Daubechies wavelets based upon a design criteria that the function approximation should work well with polynomials. How this design criterion was specified and how it is related to polynomials, will be to extensive to cover in here, but the resulting scaling and wavelet coefficients for a Daubechies wavelet with four scaling and wavelet coefficients are given as;

(4.46) $$h=\left[\frac{1+\sqrt{3}}{4\sqrt{2}},\frac{3+\sqrt{3}}{4\sqrt{2}},\frac{3-\sqrt{3}}{4\sqrt{2}},\frac{1-\sqrt{3}}{4\sqrt{2}}\right]$$

and

(4.47) $$h_1=\left[-\frac{1-\sqrt{3}}{4\sqrt{2}},\frac{3-\sqrt{3}}{4\sqrt{2}},-\frac{3+\sqrt{3}}{4\sqrt{2}},\frac{1+\sqrt{3}}{4\sqrt{2}}\right] \; .$$

The wavelet and the scaling function can in most cases not be represented by any explicit function as in the case of the Haar function. This also applies to the Daubechies wavelet above. Nevertheless it is still possible to show what the wavelet and the scaling function looks like through what is referred to as the *cascade algorithm.* It is possible to perform an iteration of *eq. (4.10)* as;

$$(4.48) \qquad \phi^{(i+1)}(t)=\sum_{n=0}^{2^i-1} h(n)\sqrt{2}\phi^{(i)}(2t-n) \qquad .$$

$\phi^0(t)$ for the first iteration is the scaling coefficients given by *eq. (4.46)*. *2^i -1* is the number of points used in the *i'th* iteration for the scaling function representation. The operation above is repeated again and again on the output from the previous iteration. If the algorithm converges to a fixed point, this will be a solution to *eq. (4.10)*. Four different steps of the iteration with *eq. (4.48)* are shown in *Figure 4.7*. We clearly see that the scaling function converges towards the shape seen in the lower left figure in *Figure 4.7.* The wavelet function can be visualized in the exact same manner by utilizing the wavelet coefficients in a similar iteration process.

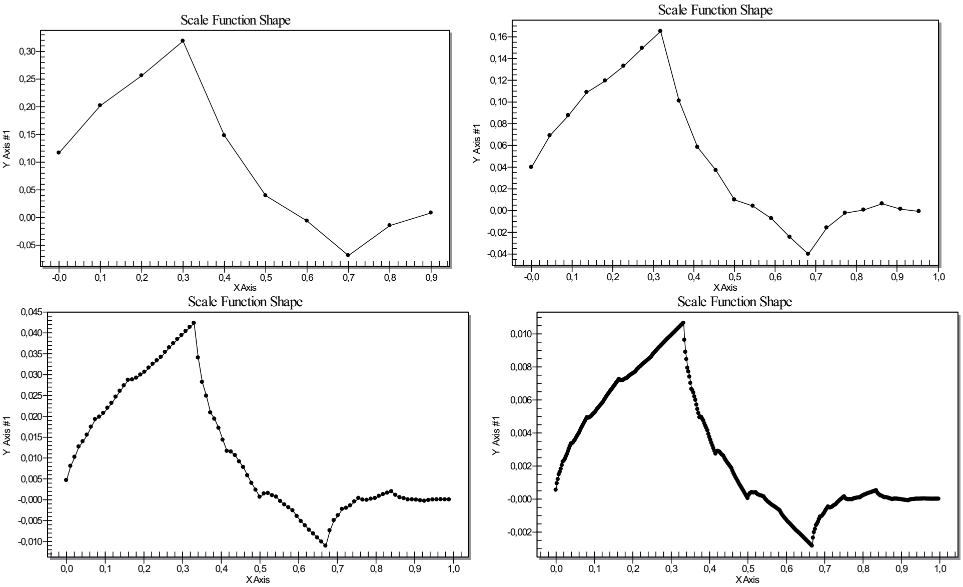


Figure 4.7: *Iteration of eq. (4.48) after iteration 1 (upper left), 2 (upper right) , 4(lower left) and 6 (lower right). Illustrated for Daubechies wavelet with four scaling coefficients.*

In theory, an infinite number of wavelet and scaling coefficients can be obtained both by utilizing a larger number of scaling coefficients and also by choosing the degrees of freedom available in the *eq. (4.42) - (4.44)* differently. By choosing the degrees of freedom available differently, other families of wavelets can be developed. These selections can be made in such a way that they approximate different types of signals well, as for instance, polynomials in the case of Daubechies. By a wavelet family means a collection of wavelets with similar properties but with a different number of scaling and wavelet coefficients. However, a more detailed description of various wavelet families will be outside the scope of this work. We will simply use the most common wavelets found in the literature to determine which wavelets to utilize with the signals we consider in this work. This will be one of the main topics in chapter 5.

Since we in practice will deal with a finite number of measurements, we will require that the signal representation in *eq. (4.23)* can be given as an approximation, where the finest resolution level will be given by the signal measurements. With a finite number of signal measurements, troubles might also occur in the wavelet transform in *eq. (4.27)* and *eq. (4.28)* when we are "running out of points". The next section will thus also give a brief introduction to practical aspect of the wavelet transform when dealing with finite sampled signals.

4.7 Discrete Wavelet Transform

When dealing with finite sampled signals, we can not have a signal representation as given by *eq. (4.23)* where the time summation over index *k* and the scale summation over index *j* are infinite. However, they can both be made finite with very little or no error. Therefore in the case of finite sampled signal only, we will have a signal approximation and not a signal representation as given by *eq. (4.23)*. The lower limit of *j eq. (4.23)* is made finite by simply choosing some desired resolution level, usually chosen as zero. The upper limit will be limited by the sampling rate. In general, this will provide us with a good approximation if the signal samples are dense enough.(58)

$$(4.49) \qquad f(t) \approx \sum_k c(k)\varphi_k(t) + \sum_{j=0}^{J-1}\sum_k d_j(k)\psi_{j,k}(t) \qquad k \in 0,1...,2^j - 1 \ .$$

The coarsest resolution level *J*, will be limited by the number of signal measurements available and given as;

$$(4.50) \qquad J = \log_2(N/\#WC) + 1 \ ,$$

where *N* is the number of signal measurements and *#WC* is the number of wavelet coefficients in the wavelet considered. This will allow us to calculate the detail and approximate coefficients from *eq. (4.27)* and *eq. (4.28)* with $k \in 0,1...,2^j - 1$ and where $j \in 0,1...,J-1$. The finest resolution *j=0* is approximated by the signal samples. Thus *eq. (4.27)* and *eq. (4.28)* can be rewritten with finite indices as;

$$(4.27) \qquad c_{j+1}(k) = \sum_{k=0}^{2^j-1} h(n)c_j(2k+n)$$

and

$$(4.28) \qquad d_{j+1}(k) = \sum_{k=0}^{2^j-1} h_1(n)d_j(2k+n) \qquad .$$

The range of *n* in the above equations will depend upon the wavelet considered. In the case of the Haar wavelet $n \in 0,1$, while in the case of the Daubechies wavelet with four scaling and wavelet coefficients $n \in 0,1,2,3$. In the same manner, the signal approximation is reconstructed from the detail and approximate coefficients applying *eq. (4.29)* with the same finite ranges of *k*, *n* and *j*.

The decomposition in the discrete wavelet transform can be best illustrated with an example. If a wavelet with two wavelet and two scaling coefficients is considered with a signal sampled 8 times, we could only obtain the signal approximation up to level *J=3* as given by *eq. (4.50)*. The signal will at this finest resolution level be approximated by the signal samples. At the first decomposition level, we have 4 approximate and 4 detail coefficients. At the next decomposition level we will have 2 approximate coefficients and 2 detail coefficients (+4 detail coefficients from the previous resolution level. At the third decomposition level we will have 1 approximate coefficient and 1 detail coefficient (+6 detail coefficients from the two previous resolution levels). This is illustrated below in *Figure 4.8*. It is obvious that at level *j=3* in the wavelet transform with only one approximate coefficient left, we cannot continue adding additional and coarser resolution levels. The signal will at the coarsest resolution level be represented by a single approximate coefficient and all the details are stored within 7 detail coefficients (4 at level *j=1*, 2 at level *j=2* and 1 at level *j=3*).

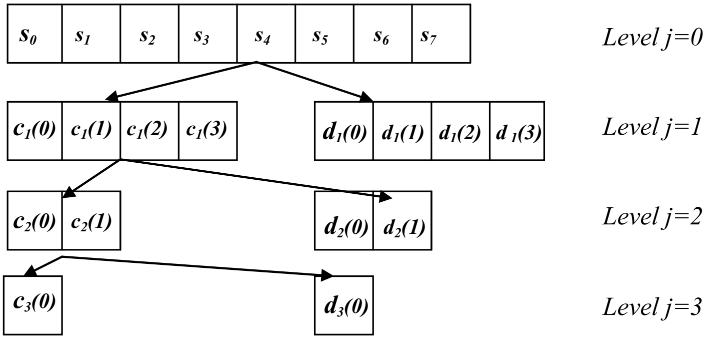

s_0 s_1 s_2 s_3 s_4 s_5 s_6 s_7
Level j=0
$c_1(0)$ $c_1(1)$ $c_1(2)$ $c_1(3)$
$d_1(0)$ $d_1(1)$ $d_1(2)$ $d_1(3)$
Level j=1
$c_2(0)$ $c_2(1)$
$d_2(0)$ $d_2(1)$
Level j=2
$c_3(0)$
$d_3(0)$
Level j=3

Figure 4.8: *The approximate and detail coefficients obtained in the wavelet transform at different decomposition levels.*

Often the calculation of the approximate and detail coefficients from *eq. (4.27)* and *eq. (4.28)* can be more easily envisioned if represented as matrix calculations. The wavelet transform at the first decomposition level is performed by convolution between the wavelet/scaling coefficients and the discrete signal measurements. The convolution of eight signal measurements with two wavelet and two scaling coefficients, can be calculated by the matrix calculation as illustrated below in *Figure 4.9*. *h(0)* and *h(1)* denotes the two scaling coefficients while $h_1(0)$ and $h_1(1)$ denotes the two wavelet coefficients. Recall that in the case of only two scaling and two wavelet coefficients; this is required to be the Haar scaling and wavelet coefficients as given by *eq. (4.40)* and *eq (4.41)*.

At the next coarser resolution level in the wavelet transform, the approximate coefficients obtained from the calculations above, are used instead of the signal measurements in the matrix calculations. The inverse wavelet transform by *eq. (4.29)* can be illustrated in a similar manner and is shown in *Figure 4.10*. We will have a perfect reconstruction and the original signal measurements are obtained.

$$\begin{bmatrix} c(0)_1 \\ d(0)_1 \\ c(1)_1 \\ d(1)_1 \\ c(2)_1 \\ d(2)_1 \\ c(3)_1 \\ d(3)_1 \end{bmatrix} = \begin{bmatrix} h(0) & h(1) & 0 & 0 & 0 & 0 & 0 & 0 \\ h_1(0) & h_1(1) & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & h(0) & h(1) & 0 & 0 & 0 & 0 \\ 0 & 0 & h_1(0) & h_1(1) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & h(0) & h(1) & 0 & 0 \\ 0 & 0 & 0 & 0 & h_1(0) & h_1(1) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & h(0) & h(1) \\ 0 & 0 & 0 & 0 & 0 & 0 & h_1(0) & h_1(1) \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix}$$

Figure 4.9: *Matrix illustrations of wavelet transform calculations.*

$$\begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix} = \begin{bmatrix} g(0) & g(1) & 0 & 0) & 0 & 0 & 0 & 0 \\ g_1(0) & g_1(1) & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & g(0) & g(1) & 0 & 0 & 0 & 0 \\ 0 & 0 & g_1(0) & g_1(1) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & g(0) & g(1) & 0 & 0 \\ 0 & 0 & 0 & 0 & g_1(0) & g_1(1) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & g(0) & g(1) \\ 0 & 0 & 0 & 0 & 0 & 0 & g_1(0) & g_1(1) \end{bmatrix} \begin{bmatrix} c(0)_1 \\ d(0)_1 \\ c(1)_1 \\ d(1)_1 \\ c(2)_1 \\ d(2)_1 \\ c(3)_1 \\ d(3)_1 \end{bmatrix}$$

Figure 4.10: *Matrix illustrations of inverse wavelet transform calculations.*

When calculating the signal approximation *f(t)* with a finite number of signal samples on a finite interval, little information of what happened before the first sample and after the last sample will be available. Yet, computing the signal approximation from *eq. (4.27)* and *eq. (4.28)* might require knowledge of what happens before the first signal sample and beyond the last signal sample.

The problem can easily be illustrated by matrix calculations at the first decomposition level in the wavelet transform. This time we chose to utilize the Daubechies wavelet with four wavelet and four scaling coefficients. Again consider a signal sampled 8 times and where the finest resolution level is approximated by the signal measurements. The matrix calculation required to compute the approximate coefficients at the first decomposition level is illustrated in *Figure 4.11*. *h(0) - h(3)* denotes the four scaling coefficients and $h_1(0)$ - $h_1(3)$ denotes the four wavelet coefficients. The index *n* in *eq. (4.27)* and *eq. (4.28)* is hence chosen to be integers from 0 to 3. Note that *n* might equally well be chosen from -1 to 2. This is simply a choice we need to make. To calculate the approximate coefficient with index 3 at the first decomposition level, the following matrix calculation is required; $c(3)_1 = h(0)s_6 + h(1)s_7 + h(2)s_8 + h(3)s_9$. However, the signal measurements s_8 or s_9 are not available. The same problem will occur at every subsequent decomposition level in the wavelet transform. Thus we need to make some assumption as to which values to utilize in the matrix calculation beyond the last available signal measurement.

$$\begin{bmatrix} c(0)_1 \\ d(0)_1 \\ c(1)_1 \\ d(1)_1 \\ c(2)_1 \\ d(2)_1 \\ c(3)_1 \\ d(3)_1 \end{bmatrix} = \begin{bmatrix} h(0) & h(1) & h(2) & h(3) & 0 & 0 & 0 & 0 \\ h_1(0) & h_1(1) & h_1(2) & h_1(3) & 0 & 0 & 0 & 0 \\ 0 & 0 & h(0) & h(1) & h(2) & h(3) & 0 & 0 \\ 0 & 0 & h_1(0) & h_1(1) & h_1(2) & h_1(3) & 0 & 0 \\ 0 & 0 & 0 & 0 & h(0) & h(1) & h(2) & h(3) \\ 0 & 0 & 0 & 0 & h_1(0) & h_1(1) & h_1(2) & h_1(3) \\ 0 & 0 & 0 & 0 & 0 & 0 & \boxed{h(0)} & \boxed{h(1)} \\ 0 & 0 & 0 & 0 & 0 & 0 & \boxed{h_1(0)} & \boxed{h_1(1)} \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix}$$

Boundary
Problems

Figure 4.11: *Matrix illustrations of wavelet transform calculations with boundary problems.*

The most common and popular method used to deal with boundary problems for finite sampled signals, is to assume periodic signals. However, this is often a poor assumption since we prefer wavelet analysis if dealing with non-stationary signal. Nevertheless, because of the simple numerical implementation this approach is frequently preferred. Consider the case where the signal is sampled *N* times on the interval [0,1]. Thus by assuming periodic signal at the boundaries of *t=0* and *t=1* we have that;

$$(4.51) \qquad f^{per}(t) = \sum_{p=-\infty}^{\infty} f(t+p) \qquad p \in Z \ .$$

This will normally result in large detail coefficients at the boundaries in the wavelet transform. A better approach is to assume that the signal can be folded (or mirrored) around the boundaries at *t = 0* and *t= 1.* Again if the signal is sampled *N* times on the interval [0,1], the folding of the signal around the boundaries at *t=0* and *t=1* implies that;

$$(4.52) \qquad f^{fold}(t) = \sum_{p=-\infty}^{\infty} f(t-2p) + \sum_{p=-\infty}^{\infty} f(2p-t) \qquad p \in Z \ .$$

This approach is far better than assuming the signal to be periodic, but folding the signal around the boundaries at *t=0* and *t=1* will cause discontinuities in the derivatives. This will again create big detail coefficients at the boundaries in the wavelet transform.

The matrix calculation required when utilizing the periodic boundary condition is shown in *Figure 4.12*. The coefficients $c(3)_1$ and $d(3)_1$ are calculated with the use of the signal measurements s_1 and s_2 instead of s_8 and s_9.

$$\begin{bmatrix} c(0)_1 \\ d(0)_1 \\ c(1)_1 \\ d(1)_1 \\ c(2)_1 \\ d(2)_1 \\ c(3)_1 \\ d(3)_1 \end{bmatrix} = \begin{bmatrix} h(0) & h(1) & h(2) & h(3) & 0 & 0 & 0 & 0 \\ h_1(0) & h_1(1) & h_1(2) & h_1(3) & 0 & 0 & 0 & 0 \\ 0 & 0 & h(0) & h(1) & h(2) & h(3) & 0 & 0 \\ 0 & 0 & h_1(0) & h_1(1) & h_1(2) & h_1(3) & 0 & 0 \\ 0 & 0 & 0 & 0 & h(0) & h(1) & h(2) & h(3) \\ 0 & 0 & 0 & 0 & h_1(0) & h_1(1) & h_1(2) & h_1(3) \\ h(2) & h(3) & 0 & 0 & 0 & 0 & h(0) & h(1) \\ h_1(2) & h_1(3) & 0 & 0 & 0 & 0 & h_1(0) & h_1(1) \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix}$$

Figure 4.12: *Matrix illustrations of wavelet transform calculations with periodic boundary adjustment.*

When performing the inverse wavelet transform by *eq. (4.29)*, we again need to adjust for boundary effects, and the matrix calculations with periodic boundary conditions thus becomes as shown in the *Figure 4.13*. *g(0) - g(3)* denotes the four scaling coefficients and *$g_1(0)$-$g_1(3)$* denotes the four wavelet coefficients used in the inverse wavelet transform**.**

$$\begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix} = \begin{bmatrix} g(0) & g(1) & g(2) & g(3) & 0 & 0 & 0 & 0 \\ g_1(0) & g_1(1) & g_1(2) & g_1(3) & 0 & 0 & 0 & 0 \\ 0 & 0 & g(0) & g(1) & g(2) & g(3) & 0 & 0 \\ 0 & 0 & g_1(0) & g_1(1) & g_1(2) & g_1(3) & 0 & 0 \\ 0 & 0 & 0 & 0 & g(0) & g(1) & g(2) & g(3) \\ 0 & 0 & 0 & 0 & g_1(0) & g_1(1) & g_1(2) & g_1(3) \\ g(2) & g(3) & 0 & 0 & 0 & 0 & g(0) & g(1) \\ g_1(2) & g_1(3) & 0 & 0 & 0 & 0 & g_1(0) & g_1(1) \end{bmatrix} \begin{bmatrix} c(0)_1 \\ d(0)_1 \\ c(1)_1 \\ d(1)_1 \\ c(2)_1 \\ d(2)_1 \\ c(3)_1 \\ d(3)_1 \end{bmatrix}$$

Figure 4.13: *Matrix illustrations of inverse wavelet transform calculations with periodic boundary adjustment.*

Periodic boundaries give a very simple numerical implementation and are often preferred for this reason. Although more sophisticated methods exist to further deal with this problem[58], the wavelet transform in this work appears to be handled well utilizing the two boundary methods described above. However, we will later in the framework of a real-time wavelet transform, discuss how to wavelet transform real-time signal measurements without the influence of boundary effects.

The next section will explain how wavelets can be used to remove noise from signal measurements in order to obtain a better signal approximation by the use of a method referred to as *Waveshrink.*

4.8 Wavelet Filtering

In the last decade the de-noising literature has been dominated by a technique called ***Waveshrink*** or ***wavelet filtering*** first developed by Donoho *et. al.*[53; 54] Wavelet filtering has proved to be an efficient way of finding structures in a signal contaminated with noise without assuming any particular parametric regression model on the signal measurements. In fact, it has been shown that the Waveshrink method have asymptotical properties within the ideal performance (55; 62; 63; 64) for applications in de-noising.

Recall from section 4.3 that we represented the signal measurements by a series of basis functions called wavelet and scaling functions as given by *eq.(4.49)*. The coefficients in this representation were referred to as detail and approximate coefficients. The magnitude of the detail and approximate coefficients would represent the "strength" of the different basis functions. As outlined in section 4.3, it might be envisioned that the detail coefficients at different resolution levels can be used to represent different frequencies in the signal approximation. The wavelets representing the difference between the finest signal approximations can be envisioned as representing the highest signal frequencies. Wavelets used to represent the difference between the coarsest approximations can be envisioned to represent the low frequency content of the signal approximation. Thus by removing different wavelet functions (or equally detail coefficients) from the signal approximation, this can be used to change or manipulate the signal approximation. In applications as de-noising, we would typically like to remove the high frequencies from the signal approximations assumed to origin from noise. This can easily be accomplished by removing the detail coefficients at the finest resolution levels in the signal approximation. However, large abrupt signal changes typically encountered in non-stationary signals would also be represented by the use of small wavelet functions. These changes will typically result in large detail coefficients in the signal approximation and will be important to maintain. As a result, a better approach would be to remove those wavelet functions at the finest resolutions levels with small detail coefficients. Hence we can have a detailed description of the signal for some time intervals while other time intervals can be described with little details in the signal approximation. This illustrates the idea behind wavelet filtering where only a few large detail coefficients are assumed to originate from the signal while the rest and small ones are assumed to originate from noise (58; 61; 53). Thus a better approximation of the underlying signal function can be obtained by removing the small detail coefficients caused by the noise and keeping the large ones. This will provide us with a better representation of the underlying signal function than the measurements themselves.

We seek to remove the noise from these measurements by utilizing the Waveshrink method. To de-noise signal samples by the use of the Waveshrink method, we need to wavelet transform the signal samples to obtain a signal approximation as given by *eq. (4.49)* and where the number of resolution levels is given by *eq.(4.50)*. The detail and approximate coefficients at each resolution level is calculated by *eq. (4.27)* and *eq. (4.28)*. The approximate coefficients at one resolution level are used as input to the next coarser resolution level, while the approximate coefficients at the finest resolution level are approximated by the signal samples themselves. For the moment, we will perform the wavelet transform with all available resolution levels, although we later might choose to only modify the detail coefficients at some of the finest resolution levels. We will seek to remove or modify detail coefficients assumed to origin from noise. How the detail coefficients are modified is determined by a *shrinkage rule* and which detail coefficients to modify is determined by the *threshold value.* If the detail coefficients have been modified in one way or another, the remaining detail and the approximate coefficients can be transformed back through the inverse wavelet transform in order to obtain a new signal approximation. The inverse transform is simply to reconstruct the signal approximation by the use of *eq. (4.29)* by the use of the modified detail coefficients and the (unchanged) approximate coefficients.

Let us consider N measurements taken of a signal $S = [s_0, s_1, ..., s_{N-1}]$. Each measurement will be assumed to be influenced by noise. This noise, $\varepsilon = [\varepsilon_0, \varepsilon_1, ..., \varepsilon_{N-1}]$, will be assumed normally distributed with zero mean and a given standard deviation, also commonly referred to as *white noise*. Thus

$$S = Y + \varepsilon \quad , \tag{4.53}$$

where S is the signal influenced by noise ε and where $Y = [y_0, y_1, ..., y_{N-1}]$ is the noise free signal. The Waveshrink method can be summarized in the following way;

$$\begin{aligned} X &= W(S) \\ Z &= \delta(X, \lambda) \quad , \\ F &= W^{-1}(Z) \end{aligned} \tag{4.54}$$

where W is the wavelet transform, δ is the shrinkage rule, λ is the threshold value and W^1 is the inverse wavelet transform. The new signal estimate, $F = [f_0, f_1, ..., f_{N-1}]$, is assumed to be a better approximation of the noise free signal Y then the original measurements S.

Since the wavelet transform is an orthogonal transformation, white noise in the signal samples will remain white after transformation. (In general, the Gaussian ideal distribution has been reported as a very good model for the detail coefficient distribution at each resolution level(17)). The noise will be present in the detail and approximate coefficients as;

$$\hat{c}_{j_0,k} = c_{j_0,k} + \sigma\varepsilon_{j_0,k}, k = 0,1,......,2^{j_0} - 1 \tag{4.55}$$

and

$$\hat{d}_{j,k} = d_{j,k} + \sigma\varepsilon_{j,k}, j = 0,.., j_0 - 1, k = 0,1,.....,2^j - 1. \tag{4.56}$$

However, even though the noise often is assumed to be white, the Waveshrink method is shown to work well even in cases where the noise is coloured (65).

Several choices need to be made to facilitate the removal of noise in Waveshrink. We need to select a wavelet and scaling function to utilize in the wavelet transform of the signal measurements. Further we have to determine a shrinkage rule for how to modify the wavelet coefficients, we need to determine a threshold to determine which wavelet coefficients to modify and we need to determine at which resolution level(s) to modify the wavelet coefficients. Some threshold estimators require an estimate of the noise level in the signal measurements. Hence we will need to determine how to estimate the noise level in the signal measurement as well. In the next sections we will give an overview of the most common wavelets, shrinkage rules, thresholds estimators and noise estimators found in the literature, at least in this author's opinion.

4.8.1 Wavelet

In order to apply the Waveshrink method as described above, we will need to select which wavelet to utilize in the signal approximation. Different wavelets will provide us with different approximate and detail coefficients in the signal approximation. As we in the Waveshrink method seek to modify the detail coefficients in the signal approximation, and where this modification will depend upon the magnitude of the detail coefficients, the performance of the Waveshrink method will ultimately depend upon the wavelet considered.

As already mentioned, when choosing a wavelet, we usually do not choose the wavelet function itself, but the scaling and wavelet coefficients which can be used to represent the wavelet function. Below is a list showing wavelets with the number of scaling and wavelet coefficients considered in this work.

• *Daubechies:*	*2 (Haar), 4, 6 , 8, 12, 20*
• *Coiflet:*	*6, 12, 18*
• *Symmlet:*	*4, 6, 8, 12, 20*
• *Bi-orthogonal splines:*	*26, 35, 39, 44, 48*
• *Inv. Bi-orhtogonal splines:*	*62, 55, 93, 44, 84*
• *Spline:*	*4246*

Daubechies, Coiflet and Symmlets are all orthogonal wavelets. The bi-orthogonal and the inverse bi-orthogonal splines are bi-orthogonal wavelets, while the Spline wavelet is a non-orthogonal wavelet. For the orthogonal wavelets, the number of scaling and wavelet coefficients will be equal, while the bi-orthogonal wavelets might have a different number of wavelet and scaling coefficients in $h(n)$ and $h_1(n)$. The orthogonal wavelets will be named after the number of wavelet coefficients (or equally scaling coefficients), as for instance, Daubechies4 or Symmlet8. This means a Daubechies wavelet with 4 wavelet coefficients or a Symmlet wavelet with 8 wavelet coefficients. The bi-orthogonal wavelets will be named both after the number of scaling coefficients and the number of wavelet coefficients, for example Spline39. This refers to a bi-orthogonal spline wavelet with 3 scaling coefficients and 9 wavelet coefficients. At the end of section 4.5 we introduced a third wavelet system, the non-orthogonal wavelets. For the non-orthogonal wavelets, there might be a different number of wavelet and scaling coefficients in both $h(n)$, $h_1(n)$, $g(n)$ and $g_1(n)$. Thus these wavelet will be referred to as, for example Spline4246. This refers to a non-orthogonal wavelet with 4 scaling coefficients and 2 wavelet coefficients in the wavelet transform, while there is 4 scaling coefficients and six wavelet coefficients in the inverse wavelet transform.

The scaling and wavelet coefficients for three wavelets, namely Haar, Spline39 and Spline4246 are shown in the *table 5.1 – table 5.3* below. The wavelet and scaling coefficients for the rest of the wavelets considered in this work are provided in appendix B.

***Table 4.1**: Scaling and wavelet coefficients for Haar wavelet.*

n	*h(n)*	*$h_1(n)$*
0	*0.5*	*0.5*
1	*0.5*	*-0.5*

Table 4.2: *Scaling and wavelet coefficients for Spline39 wavelet.*

n	***h(n)***	$\widetilde{h}(n)$
0		*2.34375 e-02*
1		*-4.6875 e-02*
2		*-1.25 e-01*
3	*0.25*	*2.96875 e-01*
4	*0.5*	*7.03125 e-01*
5	*0.25*	*2.96875 e-01*
6		*-1.25e-01*
7		*-4.6875e-02*
8		2.34375e-02

Table 4.3: *Scaling and wavelet coefficients for Spline4246.*

n	**h(n)**	$\widetilde{h}(n)$	**g(n)**	$\widetilde{g}(n)$
0				*-3.125e-2*
1	*1.25e-2*		*1.25e-2*	*-2.1875e-1*
2	*3.75e-1*	*-0.5*	*3.75e-1*	*-0.6875*
3	*3.75e-1*	*0.5*	*3.75e-1*	*0.6875*
4	*1.25e-2*		*1.25e-2*	*2.1875e-1*
5				*3.125e-2*

The Haar and Daubechies4 wavelets are perhaps the most common used wavelets for Waveshrink, while bi-orthogonal wavelets as the Spline39 and non-orthogonal wavelets like Spline4246 are more rarely used. This since people in general is more familiar with the orthogonal wavelets than the bi-orthogonal and non- orthogonal ones. Next we will describe the different shrinkage rules considered in this work.

4.8.2 Shrinkage Rule

A shrinkage rule determines which of the detail coefficients to keep, remove or modify in any desired manner. For instance, if all small detail coefficients were assumed to mostly origin from noise while the large wavelet coefficients were assumed to mostly origin from the signal itself, we might for instance choose to keep the large detail coefficients and remove the small ones. Hence the shrinkage rule would be to remove all "small" detail coefficients and to keep all "large" detail coefficients. By applying such an approach, we subsequently would need to determine what is meant by "small" or "large" detail coefficients, but for the moment, we will consider all detail coefficients above a specified threshold λ as large and all detail coefficients below this threshold λ as small. For simplicity, we will further consider the detail coefficients at all available resolution levels. The number of available resolution levels for a given sampled signal was previously discussed in section 4.7, and where the coarsest resolution level was limited by the number of measurements available and given by *eq. (4.50)*.

Hard Shrinkage rule

If we choose to apply a shrinkage rule which keeps the large detail coefficients and removes the small ones, this shrinkage rule is referred to as *Hard shrinkage* and was developed by Donoho and Johnson [(53)]. The Hard shrinkage rule can be mathematically stated as;

$$(4.57) \qquad \delta_\lambda^H = \begin{cases} 0 & |d_j(k)| \le \lambda \\ d_j(k) & |d_j(k)| > \lambda \end{cases} .$$

Here δ_λ^H denotes the shrinkage rule, $d_j(k)$ the detail coefficients (where j denotes the resolution level and k the index of the detail coefficients at the resolution level considered). The Hard shrinkage rule is often referred to as a "keep or kill" approach since we either keep or discards the detail coefficients.

As already discussed, one assumes that the small detail coefficients mostly origin from noise while the large detail coefficients mostly origin from the signal itself. However, also the large detail coefficients might be influenced by some noise as can be seen from *eq. (4.56)*. In the Hard shrinkage rule this is completely ignored. Hence the Hard shrinkage rule in general fails to remove the noise from the large detail coefficients and the filtered signal will typically contain noise spikes. To overcome this problem, the Soft shrinkage rule was developed.

Soft Shrinkage rule

Another popular shrinkage rule is referred to as the *Soft shrinkage* rule, also developed by Donoho and Johnson [(53)]. The Soft shrinkage rule can be mathematically stated as;

$$(4.58) \qquad \delta_\lambda^S = \begin{cases} 0 & |d_j(k)| \le \lambda \\ d_j(k) - \lambda & d_j(k) > \lambda \\ d_j(k) + \lambda & d_j(k) < -\lambda \end{cases} .$$

Here δ_λ^S denotes the Soft shrinkage rule. Again we remove all details coefficients below a given threshold λ. However, instead of keeping the detail coefficients above this threshold λ as we did in the case of the Hard shrinkage rule, we choose to shrink the detail coefficients above this threshold with a given value. In the Soft shrinkage rule, this value is chosen to be equal the threshold.

The Soft shrinkage rule shrinks all detail coefficients with a given value and generally this approach tends to remove more information from the signal approximation then just noise. Improvements have been proposed in new shrinkage rules which often are compromises between the Hard and the Soft shrinkage rule. Such shrinkage rules are the Garrote [(66)], SCAD [(63)] and Firm [(61)].

Scad Shrinkage rule

The SCAD shrinkage rule was proposed by Fan *et al.* **[(63)]** and given as;

$$(4.59) \qquad \delta_\lambda^{SC} = \begin{cases} \operatorname{sgn}(d_j(k)) * \max(0, |d_j(k)| - \lambda) & |d_j(k)| \le 2\lambda \\ \dfrac{(\alpha - 1)d_j(k) - \alpha\lambda \operatorname{sgn}(d_j(k))}{\lambda - 2} & 2\lambda < |d_j(k)| \le \alpha\lambda \\ d_j(k) & |d_{j(k)}| > \alpha\lambda_2 \end{cases} .$$

This is a keep, shrink or kill rule. Fan *et al.* suggested α equal to 3.7. In this shrinkage rule we keep the detail coefficients above λα, while the detail coefficients below are shrunken in two different ways depending upon their magnitude. This shrinkage rule is believed by Fan *et al.* to have some advantageous over the Hard and Soft shrinkage rules.

Non-negative Garrote

The non-negative Garrote, a compromise between the Hard and Soft shrinkage rules, was introduced by Breiman (66) and is defined as;

$$(4.60) \qquad \delta_{\lambda}^{G} = \begin{cases} 0 & |d_j(k)| \le \lambda \\ d_j(k) - \lambda^2 / d_j(k) & |d_j(k)| > \lambda \end{cases} .$$

The Garrote shrinkage function is continuous like the Soft shrinkage function. Gao (62) shows that the Garrote shrinkage rule might offer some advantages over the Hard and Soft shrinkage rule in terms of smaller bias than for the Soft shrinkage rule and smaller variance than for the Hard shrinkage rules.

Firm

Gao *et al.* **(61)** introduced the firm shrinkage rule as;

$$(4.61) \qquad \delta_{\lambda_1,\lambda_2}^{F} = \begin{cases} 0 & |d_j(k)| \le \lambda_1 \\ \operatorname{sgn}(d_j(k)) \dfrac{\lambda_2(|d_j(k)| - \lambda_1)}{\lambda_2 - \lambda_1} & \lambda_1 < |d_j(k)| \le \lambda_2 \\ d_j(k) & |d_j(k)| > \lambda_2 \end{cases} .$$

The firm shrinkage rule requires two threshold λ_1 and λ_2. Consequently, the drawback using the Firm shrinkage rule is the requirement of these two various thresholds. This makes threshold selection more difficult than for the other shrinkage rules. However, if appropriate thresholds are selected, it is believed by Gao *et al.* that the firm shrinkage rule should outperform both the Hard and Soft shrinkage rule.

Block shrinkage

In the shrinkage rules described until now, the detail coefficients are filtered term by term. If the detail coefficients are evaluated on a block basis rather than individually, this is referred to as block shrinkage. The idea is that a detail coefficient is more likely to contain important signal information if the neighbour coefficients do. Cai (67) proposed a block shrinkage method referred to as *BlockJS*. At each level *j,* the coefficients are grouped into blocks *B* of length *L* as;

$$(4.62) \qquad (jB) = \{(j,k) : (B-1)L + 1 \le k \le BL\} .$$

The detail coefficients in each block are modified by the following shrinkage rule;

(4.63) $$\delta_{\lambda}^{BS} = \max\left(0, \frac{S_{jB}^2 - \lambda L \sigma^2}{S_{jB}^2}\right) d_j(k)$$

where S_{jB}^2 is the signal energy of each block and is for a discrete signal given as;

(4.64) $$S_{jB}^2 = \sum\nolimits_{k=(B-1)L+1}^{BL} d_j(k)^2 \quad .$$

In addition, $L = log(N)$, $\lambda = 4.50524$ and σ is the noise level in the signal measurements. As opposed to the previous described shrinkage rules (Hard, Soft, SCAD and Garrote), the block shrinkage rule requires no separate estimate of a threshold to determine which detail coefficients to keep, shrink or remove. The relative shrinkage of the detail coefficients is small if the signal energy in the block is considered as high, while the relative shrinkage of the detail coefficients is large if the signal energy in the block is considered as low.

It is also possible to look at overlapping blocks as shown by Cai and Silverman(64). Thus, the treatments of the coefficients in the middle of the block depend on the entire block. At all resolution levels *j,* the coefficients are grouped into blocks of length L_0. The blocks are expanded to length $\mathrm{L}_1 = \max(1, \mathrm{L}_0/2)$ in each direction. At the start and end of each level, the blocks are only expanded in one direction. *L* is thus given as $L = L_0 + 2L_1$. If $L_0 = log(N)/2$ and $\lambda = 4.50524$, Cai and Silverman named the method *Neighblock.* If $L_1 = L_0 = 1$ (and thus $L = 3$) and $\lambda = 2/3\ log(N)$, they named the method *NeighCoeff.* Hence, the Neighcoeff chose a threshold for each coefficient with reference to the neighbour coefficients, while NeighBlock threshold coefficients simultaneous in blocks using the coefficients outside the block as a reference. The thresholding is accomplished in the same manner as for BlockJS in *eq. (4.63) eq. (4.64).*

Athichanagorn et al. (5) proposed a hybrid method where the Hard shrinkage rule was applied close to transients while the Soft shrinkage was applied elsewhere. This since Soft shrinkage rule might over-smooth the signal close to the transients while the Hard shrinkage rule might fail to suppress noise spikes elsewhere. This method lacks the simplicity of the other described shrinkage rule as it requires the detection of transients. We believe that SCAD, Garrote and Block shrinkage offers the same compromise as the hybrid shrinkage rule proposed by Athichanagorn et al., but without the requirement for transient detection. Therefore the hybrid shrinkage rule is not considered in this work.

In the Hard, Soft, SCAD and Garrote shrinkage rule shown above, a threshold is required to determine which detail coefficients to keep, shrink and/or remove. In order to apply the shrinkage rules considered, we need to determine this threshold. There are again many possible choices for how to select this threshold. The threshold is estimated by utilizing different *threshold estimators*. These will be discussed next.

4.8.3 Threshold Estimator

Numerous threshold estimators have been proposed in the literature in order to determine the value of the threshold required in the shrinkage rules described in the previous section. It is important to select the "correct" thresholds for use in Waveshrink. Selecting a too large threshold will over-smooth the signal while selecting a too small threshold will not remove the noise from the signal measurements. We have to main classes of threshold estimators, global or level dependent threshold estimators. A global threshold estimator utilizes the same threshold at all resolution levels while a level dependent threshold estimator applies different thresholds at different resolution levels. Below the most common threshold estimators found in the literature is described, at least in this author's opinion.

Universal Threshold

The most common and widely used threshold estimator is the *(global) universal threshold* proposed by Donoho and Johnson (53) and given as;

$$\lambda^{UG} = \sigma\sqrt{2\log(N)} \tag{4.65}$$

λ^{UG} denotes the (global) universal threshold estimator and N is the number of signal measurements. To apply this threshold estimator, we need an estimate of the signal noise. This will be discussed in a separate section next, since as in the case of which shrinkage rule and threshold estimator to select, we have different choices for how to obtain an estimate of the signal noise. However, it is important to notice that some threshold estimators might require an estimate of the signal noise while others do not.

The magnitude of the universal threshold will not depend upon the decomposition level considered. If we instead chose to use different threshold at every decomposition level, the threshold estimator is referred to as a level dependent *universal threshold* estimator and given as;

$$\lambda^{U} = \sigma\sqrt{2\log(N/2^{j-1})} \tag{4.66}$$

λ^{U} denotes the universal threshold estimator, N is the number of signal measurements, j is the resolution level considered and σ is the standard deviation of the signal noise.

Kovac *et. al.*29 experienced that the estimated universal threshold often was too large for the signal they considered and suggested a reduced universal threshold as;

$$\lambda^{UR} = \lambda^{UG}/3 \tag{4.67}$$

Minimax Threshold

Minimax thresholds are tabulated thresholds that will depend upon the shrinkage rule considered. Minimax thresholds have been derived by Donoho and Johnson (53) for Soft shrinkage, by Bruce *et al.* (68) for Hard shrinkage, by Gao (62) for Garrote shrinkage, by Gao *et al.* (61) for firm shrinkage and by Fan *et al.* (63) for SCAD. The tabulated thresholds for the different shrinkage rules are given in *Table 4.4.* We chose to apply linear extrapolation to calculate Minimax thresholds when the number of signal measurements is above those listed in *Table 4.4*.

Table 4.4: *Minimax threshold values for various shrinkage rules.*

	N=128	N=256	N=512	N=1024	N=2048
Garrote	*2.075*	*2.261*	*2.441*	*2.615*	*2.784*
SCAD	*1.691*	*1.881*	*2.061*	*2.241*	
Firm λ₁	*1.893*	*2.116*	*2.331*	*2.538*	*2.737*
Firm λ₂	*7.890*	*7.549*	*7.259*	*7.069*	*6.939*
Hard	*2.913*	*3.117*	*3.312*	*3.497*	*3.674*
Soft	*1.669*	*1.859*	*2.045*	*2.226*	*2.403*

The Minimax threshold is usually reduced at each decomposition level according to the values in *Table 4.4*. For example; when applying the Garrote shrinkage rule for a signal with 2048 measurements, the threshold at the first decomposition level is 2.784, at the second decomposition level 2.615 etc. On the other hand, if the threshold 2.784 is utilized at every decomposition level for a signal with 2048 measurements, the threshold is referred to as the *Global Minimax*. The final Minimax threshold is multiplied with the estimated noise level as;

$$\lambda^{M} = \sigma\lambda^{*} \tag{4.68}$$

where σ is the estimated noise level and λ^* is tabulated values from *Table 4.4* for the shrinkage rule considered.

Sure

Stein's Unbiased Risk Estimation [(69)] known as SURE, is for Soft shrinkage rule given as;

$$SURE(\lambda;X) = 2^{j+1} - 2\#\{i : |X_k| \leq \lambda\} + \sum_{k=0}^{2^j-1} \min(|X_k, \lambda|)^2 \tag{4.69}$$

and for Garrote shrinkage rule as;

$$SURE(\lambda;X) = \begin{Bmatrix} 2^{j+1} - 2\#\{k : |X_k| \leq \lambda\} + \#\{X_k^2 : |X_k| \leq \lambda\} - \\ (\lambda^4 + 2\lambda^2)\#\left\{\dfrac{1}{X_k^2} : |X_k| > \lambda\right\} \end{Bmatrix} \tag{4.70}$$

The goal is to find the threshold that minimizes *SURE(λ;X)*. If $d_j(k)/\sigma = X$ where *$d_j(k)$* is the detail coefficient with index *k* at resolution level *j,* one obtains;

$$\lambda^{SURE} = \arg\min_{0 \leq \lambda \leq \lambda^U} SURE(\lambda; \frac{d_j(k)}{\sigma}) \quad j = 0,.., j_0 - 1; k = 0,..,2^j - 1 \quad . \tag{4.71}$$

where j_0 is the primary resolution level. The above optimization is straightforward. The SURE threshold method requires the shrinkage function to be continuous and consequently it is not defined for Hard shrinkage rule. Mallat [(58)] suggest using Soft SURE threshold multiplied by two for Hard shrinkage rule. The Soft SURE threshold multiplied by two will be used as the Hard SURE threshold also in this work. We are not aware that any SURE estimate has been developed for SCAD. However, we chose to use the Garrote SURE threshold as the SCAD SURE threshold as well.

Hybrid Sure

SURE has a drawback in situation where there is extreme scarcity in the detail coefficients. A hybrid scheme is used in these cases. The estimator in the hybrid scheme works in the following way;

$$\lambda^{hybrid} = \begin{cases} \lambda^{u} & s_d^2 \le \gamma_d \\ \lambda^{sure} & s_d^2 > \gamma_d \end{cases} \tag{4.72}$$

where

$$s_d^2 = \sum_{k=0}^{2^j-1} d_j^2(k) \tag{4.73}$$

and

$$\gamma_d = \sigma^2 2^{j/2}(2^{j/2} + j^{3/2}) \,. \tag{4.74}$$

λ^{hybrid} denotes the hybrid threshold estimator, λ^{u} denotes the universal threshold estimator given by *eq. (4.66)* and λ^{SURE} denotes the SURE threshold estimator given by *eq. (4.71). $d_j(k)$* denotes the wavelet coefficient at resolution level *j* with index *k*.

Birge Massart

The method developed by Birge and Massart [(70)] includes a variable *a* that can be adjusted for high or low smoothing. It is common to call them penalized low, medium and high for *a* values of 1.5, 2.5 and 10 respectively. The parameter *a* describes the scarcity in the wavelet coefficients. The threshold is given as;

$$\lambda^{BM} = | d_j(\lambda^*) | \tag{4.75}$$

and where

$$\lambda^* = \underset{0<i<2^j-1}{\arg\min} \begin{pmatrix} -sum\{d_j^{\,2}(k); i<k\} + \\ 2\sigma^2 d_j^{\,2}(i)(a + \log\left(\dfrac{n}{d_j^{\,2}(i)}\right) \end{pmatrix} \qquad k = 0,....,2^j - 1 \qquad . \tag{4.76}$$

As with the SURE threshold, it involves an optimization for determination of the optimal threshold.

Multiple hypothesis testing

The filtering of wavelet coefficients can also be viewed as a multiple hypotheses testing problem to whether or not a given detail coefficient should be discarded;

$$(4.77) \qquad H_0 : d_j(k) = 0 \qquad H_1 : d_j(k) \neq 0 \qquad .$$

If H_0 is rejected, the coefficient is kept, otherwise it is discarded. The procedure described here to test the zero hypothesis, is the procedure described by Abramovich *et.al* (71). The idea behind the proposed procedure is to keep as many detail coefficients as possible, and only discard detail coefficients if there is a strong evidence for doing so. If the detail coefficients are assumed normal distributed with mean μ and standard deviation σ, one can test the zero hypothesis by calculating;

$$(4.78) \qquad p_j(k) = 2\left(1 - \Lambda\left(\frac{|d_j(k)|}{\sigma}\right)\right) \qquad ,$$

where Λ is the cumulative distribution function of a standard normal random variable and σ is an estimate of the noise level in the signal measurements. All estimated $p_j(k)$ are sorted according to their size and we need to determine the i index (all $p_j(k)$ values are indexed i) that maximize the expression;

$$(4.79) \qquad i_{\max} = \max(i : p_j(k) < (i/N)\alpha) \qquad ,$$

where N is the number of signal measurements and α is a selected significance level. This is usually chosen equal the traditional levels for significance in hypothesis testing, i.e. $\alpha = 0.01$ or $\alpha = 0.05$. From the i_{max} determined in the above expression, we can calculate the threshold for use Waveshrink as;

$$(4.80) \qquad \lambda^{FDR} = \sigma\Lambda^{-1}\left(1 - \frac{p(i_{\max})}{2}\right) \qquad ,$$

where Λ^{-1} is the inverse cumulative distribution function of a standard normal random variable. λ^{FDR} is a global threshold, i.e. the same threshold value is applied at all resolution levels. However, note that the detail coefficients at all resolution levels are used to calculate this threshold. While the multiple hypothesis testing procedure seeks to keep as many coefficients as possible, the recursive hypothesis procedure described next only keeps coefficients if there is strong evidence for keeping them.

Recursive hypothesis testing

Recursive hypothesis testing was developed by Ogden *et al.* (72). Instead of having a global threshold as produced in the multiple hypotheses testing procedure above, they developed a hypothesis testing procedure that creates level dependent thresholds. Again the hypothesis in *eq. (4.77)* is tested;

$$(4.77) \qquad H_0 : d_j(k) = 0 \qquad H_1 : d_j(k) \neq 0 \qquad .$$

The detail coefficients are again assumed normal distributed with mean μ and standard deviation σ. Ogden *et al.* proposed to test the above zero hypothesis by calculating what they refer to as the *critical point* of a distribution as;

$$x_n^{\alpha} = \left\{ \Phi^{-1}\left(\frac{(1-\alpha)^{1/b} + 1}{2} \right) \right\}^2 \tag{4.81}$$

where Φ is the cumulative distribution function of a standard normal random variable and α is the selected significance level usually chosen equal the traditional levels for significance in hypothesis testing, i.e. α = 0.01 or α =0.05. Initially, b in the above equation will equal the number of detail coefficients at the resolution level considered.

The square value of x_b^{α} is compared with the square value of the largest detail coefficient at the resolution level considered. As long as the square value of the largest detail coefficient is larger than the square value of x_b^{α}, the largest detail coefficient is eliminated. Each time a detail coefficient is eliminated, b is set equal the number of remaining detail coefficients and *eq. (4.81)* is recalculated. This means that one continuously throws out the largest detail coefficients until only pure noise is left. When the square value of the largest remaining detail coefficient is beyond the square value of x_b^{α} , the value of this largest remaining detail coefficient is chosen as the threshold at the resolution level considered. Hence the threshold is set equal the largest value of the "small" detail coefficients assumed to origin from noise only.

Threshold summary

Not every threshold estimators can be applied together with the shrinkage rules described in 4.8.2. The combinations of shrinkage rules and threshold estimators applied in this work are shown in *Table 4.5*.

As mentioned in section 4.8.3, the SURE threshold estimator is not defined for the Hard shrinkage rule. However, Mallat [(58)] suggested using the Soft SURE threshold estimator multiplied by two. We chose to do the same. We are not aware of any SURE threshold estimator developed for the SCAD shrinkage rule. In this case we chose to use the same threshold estimator as for the Garrote shrinkage rule. For the Firm shrinkage rule, only the Minimax thresholds are considered due to the complexity involved in evaluating two different thresholds. For the block shrinkage rule, no threshold estimator is required. However, we did describe three different block shrinkage methods namely; *BlockJS*, *NeighCoeff* and *Neighblock.* The methods differ in how the detail coefficients are shrunk and all three methods are considered in this work. To the right in *Table 4.5,* it is indicated whether the threshold estimator is a global or a level dependent threshold estimator, i.e. if the same threshold is applied at all resolution levels or if different thresholds are applied at different resolution levels. A global threshold is denoted *G* while a level dependent threshold is denotes *L*.

Table 4.5: *Shrinkage rules and threshold estimators considered in this work.*

	HARD	SOFT	SCAD	GARROTTE	FIRM	BLOCK	GLOBAL	LEVEL
Universal	x	x	x	x				L
Global Universal	x	x	x	x			G	
Reduced Universal	x	x	x	x			G	
SURE	x[1]	x	x[2]	x				L
Hybrid SURE	x[1]	x	x[2]	x				L
MINIMAX	x	x	x	x	x			L
Global MINIMAX	x	x	x	x	x		G	
Penalize low, med, high	x	x	x	x				L
Multiple hypothesis	x	x	x	x			G	
Recursive Hypothesis	x	x	x	x				L
No threshold required						x[3]		L

[1] Soft SURE threshold estimator multiplied by two as suggested by Mallat[18].
[2] The same threshold as for Garrote shrinkage. Choice made by this author.
[3] No threshold is required in the block shrinkage rule.

4.8.4 Noise Estimator

As mentioned in the previous section, we might need an estimate of the signal noise in order to utilize a particular threshold estimator. Once again, we have several different ways of obtaining such an estimate. However, common for the two noise estimators considered in this work, is that they will estimate the signal noise based upon the detail coefficients at the first decomposition level. Maybe the most popular noise estimator, proposed by Donoho and Johnson(54), is the *median noise estimator* given as;

$$\sigma = \frac{MAD}{0.6745} \quad . \tag{4.82}$$

MAD is the median value of the absolute values of the detail coefficients at the first decomposition level in the wavelet transform. The constant 0.6745 is obtained based upon the assumption that the noise is normal distributed with a given standard deviation and zero mean(54). Another method for estimation of noise is by standard deviation estimation;

$$\sigma = \sqrt{\frac{1}{N/2}\sum_{k=0}^{N/2-1}(d_1(k)-\bar{d}_1)^2} \quad . \tag{4.83}$$

N is the number of signal measurements and d_1 denotes the detail coefficients at the first decomposition level. Estimating noise by standard deviation is believed to give good results only if the underlying function is smooth. For a piecewise continuous function, the standard deviation noise estimation will give too large noise estimates.

4.8.5 Primary Resolution Level

Until this section, we have not discussed which resolution level(s) to include in Waveshrink. Should only the detail coefficients at some resolution levels be applied in Waveshrink or should the details coefficients from all possible resolution levels be applied in Waveshrink? (Recall that for finite sampled signals, the resolution levels range from the coarsest resolution level given by *eq. (4.50)* to the finest resolution level with index 0, where the finest resolution level is approximated by the signal samples.)

Donoho and Johnson states that for infinite samples all resolution levels should be filtered. However when the samples are finite, the coarsest resolution levels according to Donoho and Johnson (54), should not be filtered even if the detail coefficients fail to pass the threshold in the shrinkage rule considered. There have been several proposals for how many decomposition levels to filter. Donoho and Johnson (53) used a fixed level at $j_0 = 5$ and Nason (73) used a fixed level at $j_0 = 3$, i.e. they chose to filter the detail coefficients below the estimated threshold at the 5 or 3 finest resolution levels respectively. The selected filter level j_0 is referred to as the *primary resolution level*.

Another way, and maybe the most popular, is to estimate the primary resolution level from;

$$j_0 = \log_2(\log(N)) + 1 \quad . \tag{4.84}$$

The primary resolution level will in this case depend upon the number of signal samples N. This formula is based on asymptotical consideration by Härdle *et al.* (74).

When applying a fixed primary resolution level, we obviously would need to verify that this will not be limited by the number of resolution levels given by *eq. (4.50)*, i.e. we need to require that the number of available resolution levels in the signal approximation must be equal to or greater than the primary resolution level. In general this will only be a problem if the number of signal measurements is very limited and a fixed primary resolution level is considered (as for instance $j_0 = 5$).

Again, as with the shrinkage rule, threshold estimator, noise estimator and wavelet, we had several different choices for how to select the primary resolution level. Which one to choose for the signals we consider in this work will be the scope of chapter 5. In the next section we will introduce a slightly different wavelet transform called the stationary wavelet transform. This wavelet transform has been shown to have good properties when it comes to applications in de-noising.

4.9 Stationary Wavelet Transform

In the presence of discontinuities or sharp signal changes, the de-noising process in the previous section might introduce oscillations in the function approximation around these sharp changes or discontinuities (also referred to as Gibbs phenomena). This is nevertheless a much smaller problem in wavelet analysis than for instance in Fourier analysis. In addition, the problem can be further reduced in wavelet analysis by the introduction of the translation invariant wavelet transform or the wavelet stationary transform. This is also known as "*algorithme à trous*" and was introduced by Holdschneider et al. (75) The wavelet transform is made stationary by removal of the down-sampling step in the wavelet transform and the up-sampling step in the inverse wavelet transform. Recall from chapter 4.4 that down-sampling was applied to remove every second sample in the wavelet transform and denoted (↓2) while up-sampling was applied in the inverse wavelet transform and denoted (↑2). This is illustrated in the figures below, *Figure 4.14* and *Figure 4.15*, where the up-sampling and down-sampling is set equal 1, indicating that the same number of points will be kept in both the down-sampling and the up-sampling.

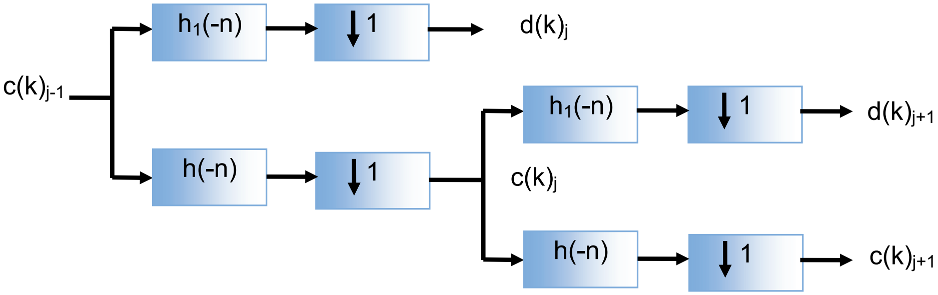


Figure 4.14: *Wavelet transform without down-sampling.*

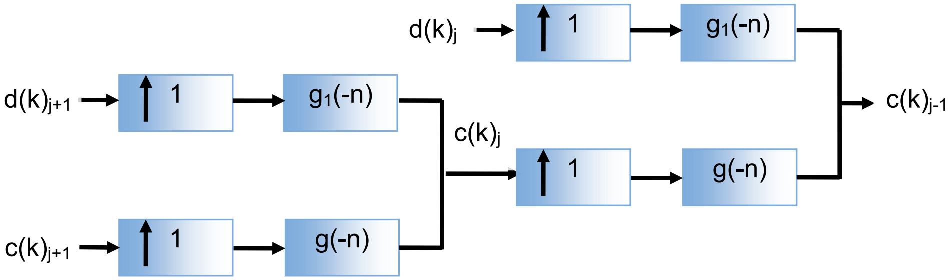


Figure 4.15: *Inverse wavelet transform without up-sampling.*

The stationary wavelet transform will in turn introduce redundancy in the signal approximation. As indicated earlier in chapter 4.4, by keeping half the number of approximate and half the number detail coefficients at level *j,* (compared to the number of approximate coefficients at level *j-1)* the signal approximation at the finer level *j-1* can be reconstructed without loss of information. This was referred to as perfect reconstruction. By keeping an equal number of approximate and detail coefficients at each resolution level, we keep more information than required in order to reconstruct the finer signal approximation, and hence we will have redundancy in the signal approximation. The stationary wavelet transform will however provide us with better properties when it comes to applications as de-noising.(76; 77)

The wavelet transform previously illustrated in *Figure 4.8,* can in a similar way be illustrated in the case of the stationary wavelet transform. This is shown in *Figure 4.16*. Even though the number of approximate and detail coefficients at each resolution level is constant, the number of levels that can be applied in the wavelet transform will still be limited by *eq.(4.50).*

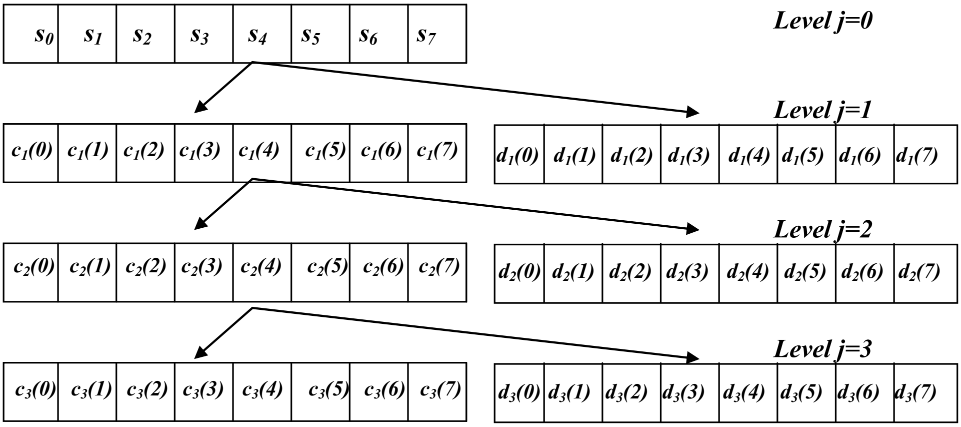

Level j=0
s_0 s_1 s_2 s_3 s_4 s_5 s_6 s_7
Level j=1
$c_1(0)$ $c_1(1)$ $c_1(2)$ $c_1(3)$ $c_1(4)$ $c_1(5)$ $c_1(6)$ $c_1(7)$
$d_1(0)$ $d_1(1)$ $d_1(2)$ $d_1(3)$ $d_1(4)$ $d_1(5)$ $d_1(6)$ $d_1(7)$
Level j=2
$c_2(0)$ $c_2(1)$ $c_2(2)$ $c_2(3)$ $c_2(4)$ $c_2(5)$ $c_2(6)$ $c_2(7)$
$d_2(0)$ $d_2(1)$ $d_2(2)$ $d_2(3)$ $d_2(4)$ $d_2(5)$ $d_2(6)$ $d_2(7)$
Level j=3
$c_3(0)$ $c_3(1)$ $c_3(2)$ $c_3(3)$ $c_3(4)$ $c_3(5)$ $c_3(6)$ $c_3(7)$
$d_3(0)$ $d_3(1)$ $d_3(2)$ $d_3(3)$ $d_3(4)$ $d_3(5)$ $d_3(6)$ $d_3(7)$

Figure 4.16: *The approximate and detail coefficients obtained in the stationary wavelet transform at different decomposition levels.*

4.10 Summary

We have in this chapter discussed the theory of wavelet analysis required in this work. First it was shown how a signal can be approximated by a sum of basis functions called scaling functions which was well-localized in time and space. These could be used to approximate a signal with different resolutions depending upon how broad or narrow these functions were chosen. We introduced a second set of basis functions called wavelet functions that were used to represent the differences between different approximations made by the scaling functions. The wavelets were small oscillating waves well localized in time and can thus be envisioned to represent different frequencies of the signal approximation. This will provide us with what we call a time-frequency representation of the signal. How to obtain this signal approximation by the use of wavelet and scaling functions were discussed next in the wavelet transform. In some cases, it might be useful to apply a bi-orthogonal wavelet system. Instead of only having one basis, we can choose to use two bases. This provides a greater flexibility in the design of wavelets and allows a larger class of signals to be well approximated by wavelets. In the case where we were dealing with finite sampled signals, solutions assuming periodic signals and mirrored signals at the boundaries were described. Next we said that wavelet analysis is well suited for de-noising signal measurements and we introduced the Waveshrink method. A number of selections need to be made in the Waveshrink method before it might be utilized for de-noising. This includes selecting which wavelet, shrinkage rule, threshold estimator, noise estimator, and primary resolution level to use. The most common wavelets, shrinkage rules, threshold estimators, noise estimators and primary resolution levels found in the literature, at least in the author's opinion, where described next. Finally, a wavelet transform without the sub-sampling step was introduced and referred to as the stationary wavelet transform and is preferred in applications as de-noising. Parts of the numerical implementation of the wavelet transform are shown in the appendix B.

The implementation of the wavelet transform and the wavelet filtering will be used to investigate if it is possible to obtain automatic signal de-noising. To achieve this, we need to make sure that the selected methods work well for the signals considered in this work. Recall that the signals considered in this work are signals commonly encountered in the E&P industry as pressure and rate signals. In other words we need to determine which wavelet, shrinkage rule, threshold estimator, noise estimator and primary resolution level to apply for the signals we have considered. This will be discussed in the next chapter.

Part 2: Results and Discussion

5 Automatic Signal Filtering

An increasing number of oil and gas wells are today equipped with high resolution gauges each capable of providing millions of measurements each year. Having a high resolution of the measured signals would be preferred in order to have a best possible description of the signals. However, it is computational costly to analyse such a large number of measurements and even simple visualization can be difficult. On the other hand, if we simply measured at a lower resolution, important signal information might be missed. Consequently we would like to reduce the number of measurements for analysis by removing those measurements with little information content and keep the rest. However, noise and outlier present in the signal measurements will make data reduction difficult. Hence we would like to remove outliers and noise from the signal measurements prior to data reduction. In addition, different types of analysis will become more difficult and in some cases even erroneous if noise and outliers are present in the signal measurements.

We will develop a module for automatic signal filtering. Thus all methods and selections required in signal filtering need to be predefined. We further need to make sure that the methods and selections proposed in automatic signal filtering work well with the signals considered. We believe it might be extremely difficult (or even impossible) to determine a single set of methods and selections that works well for any signals considered. However, we will only consider signal with particular characteristics, as for instance pressure and rate signals. Signal filtering includes outlier removal, de-noising and data reduction.

Wavelet analysis has become increasingly popular in signal analysis and in particular for de-noising. One of the reasons for this is due to the great flexibility found within wavelet analysis where numerous selections can be specified to adapt the wavelet analysis to signals with different characteristics. This will on the other hand leave us with the problem which selections to utilize for our kinds of signal. Few if any guidelines at all are in general available for this. To determine this, a performance study is required for the signal(s) considered, and will be completed as a part of this work.

The use of wavelet analysis for analyzing data from permanent downhole gauges in the E&P industry, was first introduced by Kikani et. al.[(4)]. They used wavelet analysis to de-noise pressure signals and to detect transients. Athichanagorn et al. [(5)] further introduced a seven step procedure for processing data from downhole gauges. The first four steps in the seven step procedure include signal filtering and event detection. These are; outlier removal, de-noising, transient detection and data reduction. The last three steps are well-test related and involve rate reconstruction, behavioural filtering and data interpretation. Athichanagorn et al. did not consider the performance of the selections made in Waveshrink (wavelet de-noising). Olsen et al.[(1)] did an extensive comparison study of the performance in Waveshrink for different selection. The performance of Waveshrink with different wavelets, shrinkage rules, threshold estimators and noise estimators for signals typically encountered in the E&P industry as pressure and rate signals was investigated. In addition, new methods for outlier removal and data reduction were proposed. The paper by Olsen et al.[(1)] is based upon this work. Recently Ortiz et al. [(12)] published a similar comparison study for Waveshrink. Several papers have in addition been published [6; 7; 8; 9; 10; 11; 13; 14; 15] on wavelet analysis, and a summary of wavelet analysis in the E&P industry has been published by Guan et al. [16]. It is however outside the E&P industry where most applications of wavelet analysis are found and among these is an extensive investigation of the performance of wavelet analysis on different kinds of signals by Antoniadis [17].

Automatic signal filtering will in this work be performed with the same three steps as by Athichanagorn et al. [5] First outliers will be removed. If outliers are present in the signal measurements, this can make de-noising and data reduction troublesome and they should therefore be removed first. Next, noise will be removed from the signal measurements. Finally, a large number of signal measurements might be reduced into a smaller subset that more easily can be handled by computers and through computer networks. Hence our automatic signal filtering approach is;

- Automatic outlier removal
- Automatic de-noising
- Automatic data reduction

The purpose of this work will be to improve the methods and selections utilized for outlier removal, de-noising and data reduction. A new method for outlier removal is presented. The method for outlier removal method presented by Athichanagorn et al. [5] could only remove single outlier and required a user specified threshold. We would like to have a method that can remove successive outliers. Secondly we would like to accomplish outlier removal without any user specified threshold(s) which is signal dependent. ***We will develop a method for outlier removal based upon a median filter for detection and removal of outliers, single as well as multiple, and without the need for a user specified threshold.***

Several selections need be made in Waveshrink (wavelet de-noising), and the performance of Waveshrink will depend critically upon the selection made, i.e. which wavelet, shrinkage rule, threshold estimator, primary resolution level and noise estimator to utilize. As already mentioned, Athichanagorn et al. did not consider the performance of the selections they made in Waveshrink. With the exceptions of the study by Olsen et al.[1] and the recent study by Ortiz et al. [12], few performance studies exists for typical signals in the E&P industry as pressure and rate signals. ***In this work, we will determine the "best" Waveshrink selections for the signals considered through a series of filtering exercises.*** What is meant by "best" will also be explained. It will be shown that the performance of Waveshrink will depend critically upon the selections made.

A new method for data reduction is presented as well. ***We will develop a simple method for data reduction by removing those measurements with small or none changes in magnitude since the previous measurement.*** The method requires no user selections or user defined thresholds. ***Thus all methods and selections in signal filtering will be predefined and this holds the promise for automatic signal filtering of the signals considered.***

Note that the signal filtering module is developed as a standalone module. This means that it can be used for data preparation to different analysis modules, not for well-test analysis only, as will be considered later in this work. *This means that even if we in this chapter present recommendation for how to filter rate signals, we will in chapter 7 in the scope of well-test analysis, chose to only filter pressure signals and a different approach will be* *chosen for rate signals.* However, rate signal filtering might be an important input in other kinds of analysis and is thus included in this work.

We will start by providing a detailed description of how the synthetic signals considered in this work were generated. Synthetic signals are required in this work as we need to know the true noise level in the signals we would like to de-noise. Otherwise it would be difficult to evaluate the performance of different selections and methods in Waveshrink. We will continue by describing the method preferred for outlier removal. Next the methods and selections preferred in Waveshrink for signal de-noising will be described. Waveshrink requires equally spaces signal measurements and we will next describe what to do in the case of unequally spaces signal measurements. Then we will describe how the performance analysis used to determine the "best" selections in Waveshrink was completed. Next the different methods and selections required in Waveshrink will be evaluated through a series of filtering exercises and Waveshrink will be used for signal de-noising with the selections and methods determined as "best". We will not provide any discussion for why some selections and methods in Waveshrink work well while other does not. We will simply investigate which Waveshrink selections and methods that work best for de-noising typical signals encountered in the E&P industry as pressure and rate signals**.** Finally, we will show how the number of signal measurements can be reduced by a simple iterative method when the signal measurements are assumed free for outliers and noise.

The results presented in this chapter are based upon the paper by Olsen et al. (1). However, the proposed noise estimator and threshold estimator from Olsen et al. is redefined in this work as will be described in chapter 5.3.

5.1 Signal Generation

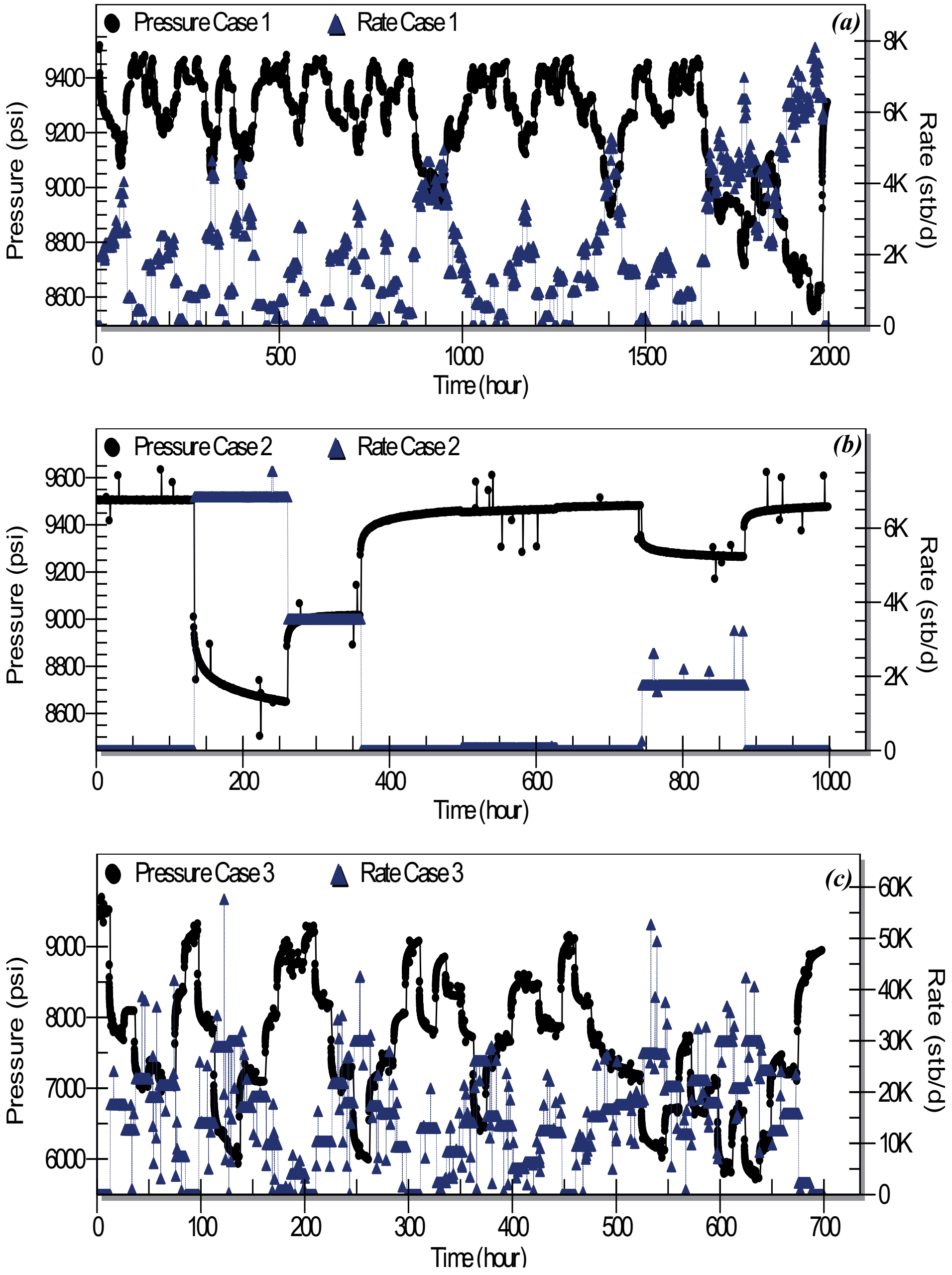
A synthetic generated signal is a signal created from a mathematical model and not from measurements of some real physical process. We will consider both synthetic pressure and rate signals assumed to be representative for measured pressure and rate signals typically encountered in the E&P industry. In our work, the process model is a model that describes the pressure response in a production well as oil, gas and/or water is withdrawn from a reservoir through this well applying a specified rate history. This model was described in details in chapter 3, and the pressure response was given be *eq. (3.29)*. However, to generate this synthetic pressure signal, we need to specify a synthetic rate history. Of course we could have utilized real rate signals measured in oil and gas wells, but this would leave us with two problems; first we did not have access to the number of rate signals desired in this study (>500) and secondly we would not know the true noise level in these signal measurements. Thus we choose to generate synthetic rate signals which are assumed to have the same magnitude and variance in rates as typically seen in real oil and gas wells. We choose to generate the rate signal as a piecewise constant function. To generate the rate history, we apply a random number generator. Parameters used to generate this rate history are drawn from uniform distributions within specified intervals (and with restriction). In order to have a piecewise constant function, we need to limit the number of rate changes to a certain fraction of the total number of measurements. Thus we choose to never have less than 10 measurements in each rate change. In addition, if the number of rate changes did approach the number of measurements, the rate history would be completely random which would make signal filtering impossible. This since there would be no underlying signal trend. A zero rate will in practice be the most common rate encountered and caused by well shut-ins. In addition to the random generated rates, we therefore chose to have a certain additional fraction of the rates equal zero. However, we limited the number of zero rates to never be more than a certain fraction of the total number of rates. If the rate is zero, the measurements are assumed to be noise-free. The rate history is hence generated based upon a random number generator drawing numbers from a uniform distribution within the limits shown below;

- Number of Measurements: 250 – 100.000.
- Time Interval: 3 – 90 seconds.
- Rate Changes: 10 – 5.000 *(Keep at least 10 measurements in each constant rate interval).*
- Measurements in Each Interval: 50% - 150 % *(of average number of measurements in each interval)*
- Zero Rates: 1 – 500 *(Keep it at least 10 times smaller than the number of rate changes. Zero rates are assumed noise-free).*
- Max Rate: 1.000 – 100.000 (stb/d).
- Max Rate Change: 2 – 100 % *(Allowed magnitude of change from one rate to next rate in term of max rate).*

This means that we might for instance have a rate history of 4600 measurements measured every 11 second. There might be 230 rate changes, i.e. we have 230 different time intervals with constant rate values. Thus the average number of rate measurements between each rate change is 20. The number of measurements for each constant rate is in addition allowed to vary with +/-50% of the average number of measurements in each time interval, i.e. the number of measurements for each constant rate will in this case be somewhere between 10 and 30. Next, 12 of the constant rate intervals might be zero. Both the number of zero rate values and their location in time will be selected on random. The maximum rate might for instance be 8900 std/d and the rate might be allowed to change with 10% of the maximum rate from one time interval to the next, i.e. if the rate is 1000 stb/d in one time interval, the next rate will be in the range from 110 stb/d to 1890 stb/d (+/- 10% of 8900 std/d). The maximum rate change limitation does not apply if the next rate is zero or if the previous rate was zero. This is simply an example of how a random rate history might be generated, and since the values are drawn from uniform distributions, any other outcome within the specified parameter intervals listed above would be equally likely.

Normal distributed noise with a given standard deviation and zero mean is added to the generated rate history and estimated pressure signal. *However, note that the synthetic pressure is generated based upon the “exact” and noise-free values of the rates*. The standard deviation for each noise distribution is chosen by random from a uniform distribution between 1 and 10. Outliers are added to the signals as well. Some small fraction of the total number of measurements is accordingly selected as outliers, where the fraction is chosen on random from a uniform distribution between 0.001 and 0.05 (0.1% to 5% of the total number of measurements). The magnitude of each outlier is chosen on random from a uniform distribution ranging from 5 to 100 times the noise level. Up to nine successive outliers are allowed.

The noisy synthetic signal measurements are then filtered to determine the best filter selections. Each time a filter selection or a filtering method is proposed, it is at least based upon 500 random generated rate signals, and 500 pressure signals which are calculated based upon the generated rate signals. Those filtered signals that have the smallest average root-mean-squared error compared with the original noise-free signal measurements (before noise and outliers was added) will be classified as the best filtered signal measurements and the selections utilized to obtain these best filtered signal measurements will be classified as the best filter selections. In *Figure 5.1* (*a*) - (*e*), typical rate and pressure signals considered in this work are shown.



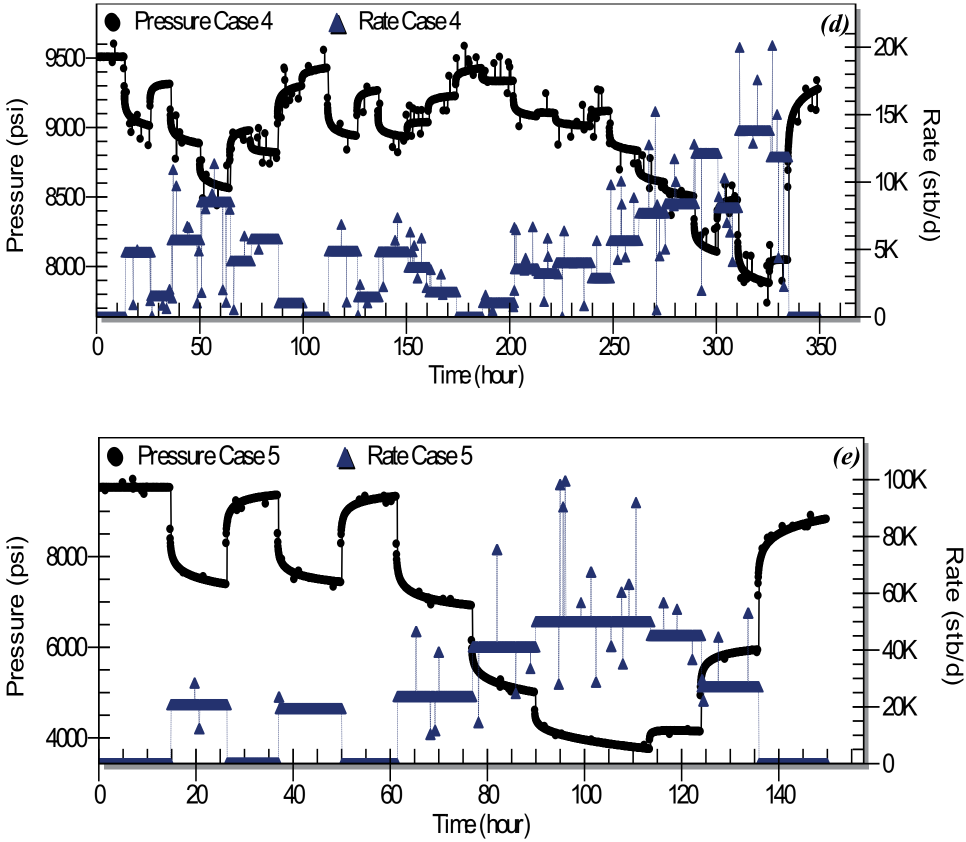


Figure 5.1: *(a)-(e) Typical pressure and rate signals generated for use in signal filtering.*

The pressure and rate signals generated are assumed to be representative for pressure and rate signals measured in real wells. ***Consequently we expect that the results obtained by utilizing these synthetic generated signals will be equally valid with measurements of real signals.***

The first step in signal filtering is outlier removal and this will be described next.

5.2 Outlier Removal

Athichanagorn et al. [(5)] proposed to use wavelet analysis to detect and remove outliers utilizing the non-orthogonal wavelet given by *eq. (4.39)*. They proposed to screen the detail coefficients at the first decomposition level in the wavelet transform for two successive large values in opposite directions. A transient, on the other hand, will result in only one large detail coefficient at the first decomposition level. Thus outliers and transients can be distinguished from each other. This is illustrated in *Figure 5.2* and *Figure 5.3*. Note that the index of the measurement is plotted along the x-axis. The original signal is shown in *Figure 5.2*. Outliers are present in measurements with indices *478, 507* and *513* while a transient is present in signal measurement with index *501*. The detail signal at the first decomposition level in the wavelet transform applying the non-orthogonal spline wavelet is shown in *Figure 5.3*. We clearly see that outliers and transients cause the response in the detail signal as described by Athichanagorn et al.

The removal of outliers requires the specification of a threshold to determine which detail coefficients to consider as large. Athichanagorn selects a fixed threshold of 5 psi to separate outliers from the rest of the signal measurements. Hence all signal measurements which result in two successive detail coefficients with magnitude above 5 psi and in opposite directions at the first decomposition level in the wavelet transform are removed.

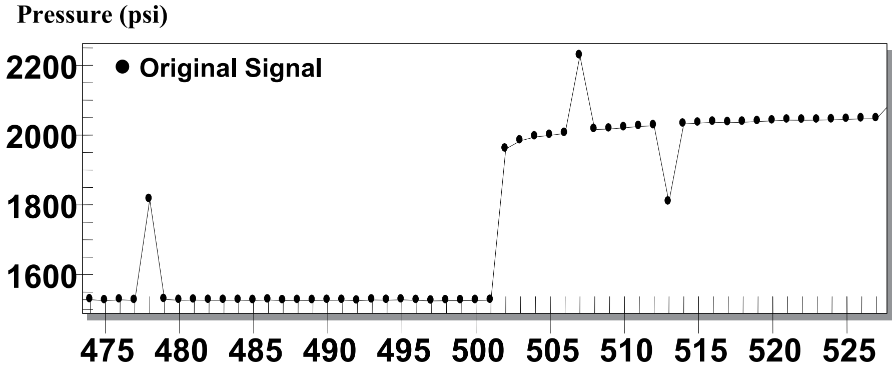


***Figure 5.2**: Original signal with outliers at indices 478, 507 and 513 and a transient with index 501.*

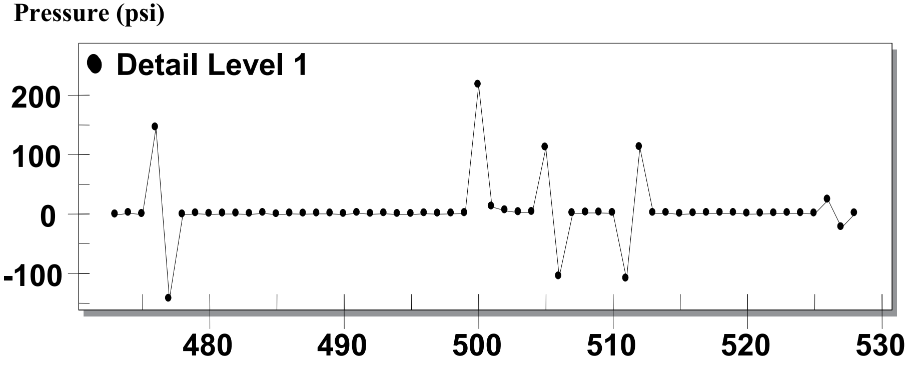


***Figure 5.3:** Detail signal at the first decomposition level in the wavelet transform.*

Khong [(6)] applied the method proposed by Athichanagorn to signal measurements with several successive outliers referred to by the author as step outliers. It was shown that the method proposed by Athichanagorn will not work if successive outliers are present. However, no improvements were proposed. The failure of the method proposed by Athichanagorn is illustrated in *Figure 5.4* and *Figure 5.5* for a signal with two successive outliers with indices

775 and 776. The two large detail coefficients indicating an outlier is now separated by a small detail coefficient and the detection method proposed by Athichanagorn will fail.

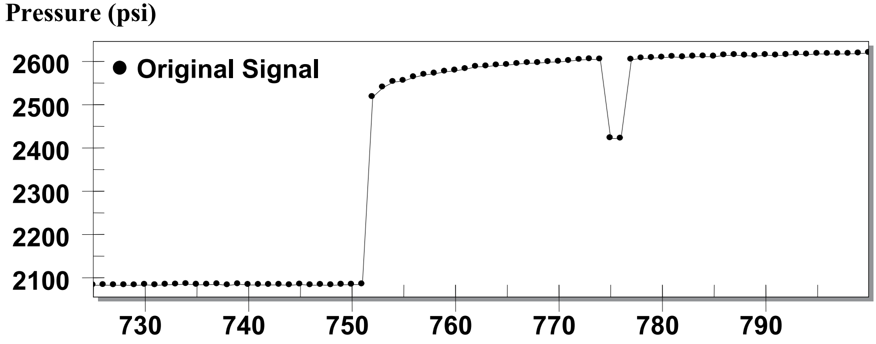


Figure 5.4: Original signal with two successive outliers at indices 775 and 776.

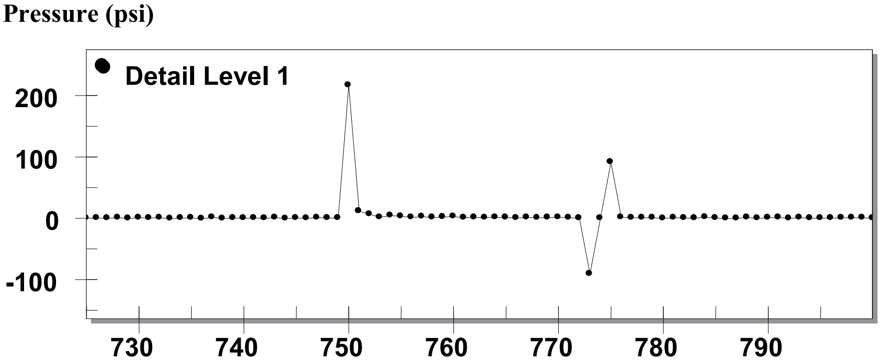


Figure 5.5: Detail signal at the first decomposition level in the wavelet transform for a signal with two successive outliers at indices 775 and 776.

Recently Viberti [(8)] proposed a similar method for outlier removal as proposed by Athichanagorn, although the method proposed by Viberti utilizes the Daubechies wavelet and requires two different user thresholds to be defined. Viberti states that the method proposed by Athichanagorn is not sufficient for outlier detection as transients also can create two large successive detail coefficients in opposite directions at the first decomposition level in the wavelet transform. In *Figure 5.6*, the Daubechies wavelet with four wavelet coefficients is used to compute the detail coefficients at the first decomposition level for the signal shown in *Figure 5.2.* We clearly see that the response from the transient with index *500* is difficult to distinguish from the response of the three outliers.

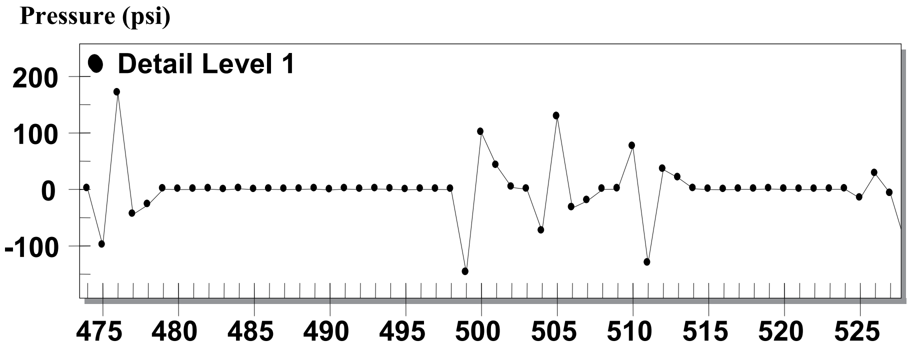


Figure 5.6*: Detail signal at the first decomposition level in the wavelet transform. Daubechies wavelet with four wavelet coefficients is applied.*

However, the detail coefficients and their characteristic will depend upon the wavelet considered. Thus the method proposed by Athichanagorn can separate single outliers from transients as long as the non-orthogonal wavelet given by *eq. (4.39)* is utilized. However, the method proposed by Athichanagorn will not work if utilized with a Daubechies wavelet with four wavelet coefficients.

Riberi et. al. [(10)] removes outliers by considering different wavelets and different decomposition levels in the wavelet transform. It is however unclear to us how the outliers actually are removed.

In our opinion, the outlier removal by wavelet analysis found in the P&E literature has two main drawbacks;

- Only single outlier can be removed.
- They require threshold(s) specified by the user depending upon the signal considered.

We would like to have a method that can remove successive outliers. Secondly we would like to accomplish outlier removal without any user specified threshold(s) which is signal dependent. We chose to use a median filter to detect and remove outliers.

5.2.1 **Median Filter**

Even thought outliers accurately can be detected using the wavelet transform [(5; 58)], we are not aware of any robust and accurate method for actually removing outliers, single or multiple, through a purely wavelet based approach. Our investigation shows that a median filter will work better. This is also supported by Kovac *et al.* [(78)]. In addition, we will combine the median filter with accurate noise estimation from the Waveshrink method for efficient removal of outliers. Finally, we will determine how many successive outliers to remove in the scope of well-test analysis.

A median filter is a moving window filter and we need to determine a window size *n* to utilize in the median filter. Next, we need to calculate the median value of the *n* signal measurements in the window considered. The number *n* is usually small compared to the total number of signal measurements available. If the value in the middle of the window considered has a value that deviates from the median value with more than a given threshold, it is considered an outlier and removed. This process is repeated for each signal measurement. This will allow us to remove single as well as multiple outliers in the signal measurements considered. The window size will determine the number of successive outlier that can be removed.

We need to determine two settings to utilize a median filter for outlier removal; namely how much outside the "normal" signal trend the measurements need to be before they are classified as an outlier and how many successive outliers to remove from the signal measurements. How much outside the "normal" signal trend the measurements should be in order to be classified as outliers, will be determined by the amount of noise estimated from the signal measurements. The amount of noise will depend on the signal measurements and there are different methods available to estimate the noise level. We will determine the noise level from Waveshrink as described in chapter *4.8.4.* Which of the noise estimators described in chapter *4.8.4* to use with the outlier filter will be determined in chapter *5.3.3*. This means that the median outlier filter does not require any user defined and signal dependent threshold to separate outliers from noise. However, for the moment, we will assume that we know the true noise level in the signal measurements.

To avoid removing any measurements solely affected by noise, three times the noise estimate is used as the limit for outlier removal. Thus, if the absolute difference between the measurement considered and the window median value is more than three times the estimated noise level, the measurement is removed. The outlier filter can be mathematically stated as;

$$s_i = \begin{cases} s_i & \left| median(m) - s_i \right| \leq 3\sigma \\ - & \left| median(m) - s_i \right| > 3\sigma \end{cases} \tag{5.1}$$

where s_i is the signal measurement, *m* is the number of signal measurements in the window considered and σ is an estimate of the noise level in the signal measurements. (-) in the above expression simply means that the signal measurement will be removed.

The window size allows us to specify the largest number of successive outliers that can be removed. How many successive outliers to remove will be important to determine. One single measurement far outside the "normal" signal trend is normally considered an outlier. Two or even five successive measurements far outside the "normal" signal trend might often be considered as successive outliers. Hundreds of successive measurements outside the "normal" signal trend are normally considered to be caused by an abrupt temporary change in some process variable that in turn temporarily has changed the measured process state. In the last case the measurements should *not* be detected as outliers and removed. ***In reality, we cannot distinguish outliers (not even single outlier) from measurements far outside the signal trend that is caused by some abrupt rapid temporary change in the measured process state.*** This since we often do not know the true process model and the true process state (Recall that outliers were assumed to be caused by large disturbance in the signal measurements at a limited time interval.) However, even if we do remove some signal measurements caused by the very smallest temporary changes in the process state, we do believe that the error caused by doing this will be small. We will determine how many successive outliers to remove in the scope of well-test analysis. We are aware that some of the removed outliers will be transients, i.e. we will remove some pressure measurements outside the “normal” signal trend which origins from a change in the process state. This might be the case if the rate for a very short period is temporarily increased/decreased. This is considered acceptable if we in the scope of well-test analysis do not lose any important information.

We choose to remove a maximum of nine successive outliers from pressure signals. If a maximum of nine successive outliers are removed, we believe that even if some of the smallest pressure transients are removed, no important information will be lost in the scope of well-test analysis. Note that for other applications than well-test analysis, it might be desirable to remove a different number of successive outliers. We will illustrate this choice with an example.

We generate a synthetic pressure signal based upon a known reservoir model and a specified rate history. The reservoir model considered is a homogenous infinite reservoir with no wellbore storage effects and was described in chapter 3. All known fluid and reservoir properties are applied in the well-test analysis, including our knowledge of which model to use. White noise with zero mean and 10 standard deviation is added to the pressure and rate signals, but the pressure signal is generated based upon the noise-free rate signal. The pressure and rate signal are shown in *Figure 5.7*.

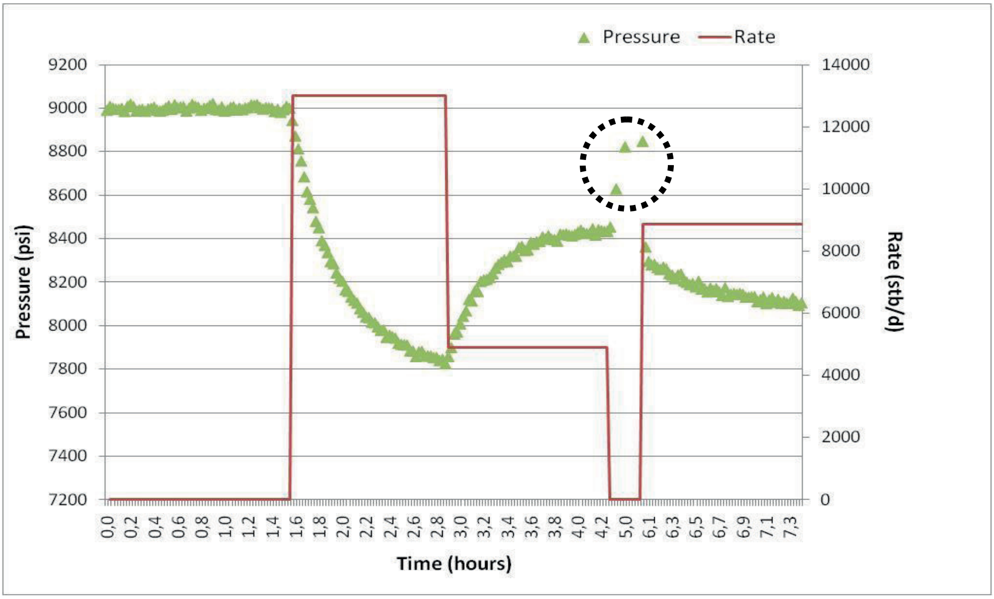


Figure 5.7: *Pressure and rate history of pressure build-up with only three pressure measurements.*

At time 4.3 hours, the well is shut-in and a pressure build-up occurs. During this pressure build-up, only 3 pressure measurements are available. Recall from chapter 3 that we in well-test analysis prefer to analyze the pressure during well shut-ins, i.e. pressure build-ups.

The permeability and skin factor are considered as unknowns and a non-linear regression analysis for the build-up at time 4.3 hours is performed. The permeability with confidence interval is estimated to 19.5 +/- 7.7 mD while the skin factor with confidence interval is estimated to 6.5 +/- 5.6. The limits for when the results from the non-linear regression analysis should be considered as reliable are for instance given by Horne[(46)]. Horne states that little confidence should be given to the results from well-test analysis when the confidence intervals for the permeability is more than 10% of the estimated permeability and more than +/- 1 for the skin factor, i.e. if the estimated permeability is 20 mD, the confidence interval should be less than +/- 2 mD However, note that in the scope of automatic well-test analysis in chapter 7, the limits given by Horne is considered as rather strict and we chose to double the confidence interval for when the results from the non-linear regression analysis should be considered reliable. The above estimated confidence intervals are clearly above the limits for when the results from the non-linear regression analysis should be considered reliable, (even when doubled) and little confidence should be given to the results. *Consequently, no important information is lost if this build-up is considered as successive outliers and removed.*

The non-linear regression analysis is repeated, but this time the number of measurements during the pressure build-up is increased. The results from non-linear regression analysis for the same pressure build-up with 3, 5, 7, 9 and 11 measurements are summarized in *Figure 5.8* and *Figure 5.9*. 11 measurements during the pressure build-up is required to estimate the permeability with confidence intervals within +/- 10% of the estimated permeability, while 7 measurements are required to estimate the permeability with confidence interval within +/- 20% of the estimated permeability (the doubled limit). More than 11 measurements are required to estimate the skin factor with confidence intervals within +/- 1, while 11 measurements are required to estimate the skin factor with confidence interval within +/- 2 (the doubled limit).

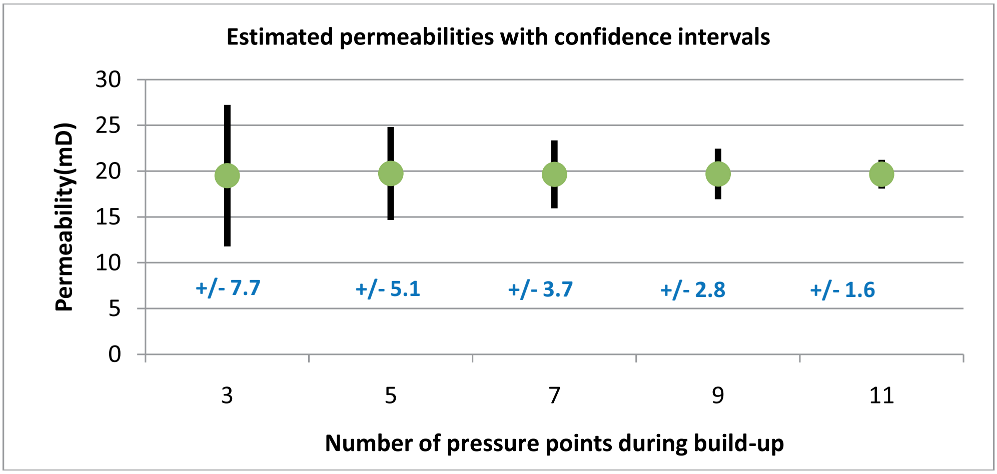


Figure 5.8: *Estimated permeabilities with confidence intervals for different numbers of pressure measurements during build-up test. Blue text is the estimated confidence intervals.*

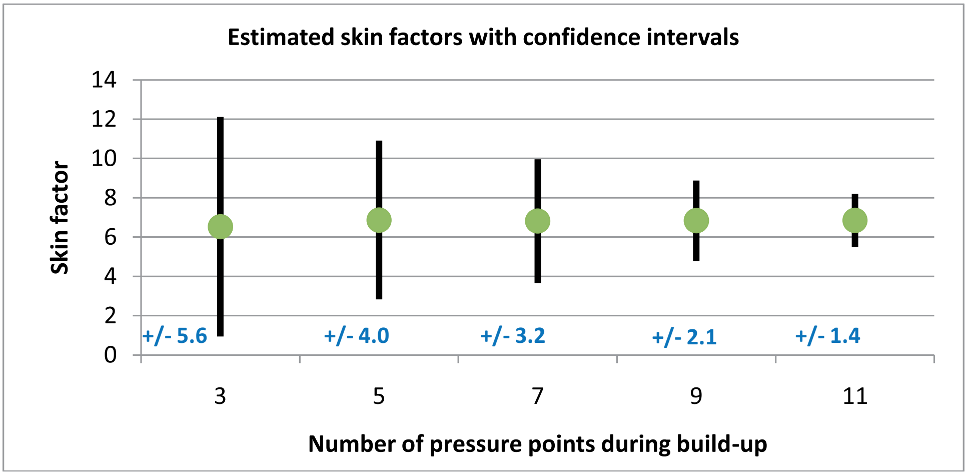


Figure 5.9: *Estimated skin factors with confidence intervals for different number of pressure measurements during build-up. Blue text is the estimated confidence intervals.*

The results here are only meant as an illustration and will clearly depend upon noise level, duration of the shut-in, well-test model and well-test parameters. However, we considered this example as a rather "uncomplicated" case and still 11 measurements during the pressure build-up is required to obtain reliable results if the recommendation by Horne is doubled. Wellbore storage effects, fractured reservoirs, layered reservoirs and dual-porosity reservoirs, just to mention a few, are considered as more "complicated" cases and usually require more measurements for an accurate well-test analysis. The case shown is simply meant to illustrate that it is very unlikely to lose important information even if the smallest pressure build-ups are removed by a median outlier filter. *Thus we conclude that the removal of up to nine successive outliers from pressure signals is a sound approach and will not result in loss of important information in the well-test analysis, even if the smallest pressure build-ups are considered as outliers and removed.*

If less then nine pressure measurements are available during a pressure build-up lasting for hours or days, the signal sampling rate is extremely poor and should be increased. On the other hand, if the signal sampling rate is high e.g. every second or every tenth second, this means that the duration of the pressure build-up is extremely short. Recall from chapter 3.6 that the first couple of minutes during a pressure build-up is dominated by wellbore storage effect and in general provides no information regarding the reservoir. Thus unless the signal sampling rate is extremely poor, our approach seems sound. In the case where the signal sampling rate is extremely poor, the signal sampling rate should be increased.

Rate signals are often stored as average or aggregated values, and it is therefore more difficult to determine how many successive outliers to remove from rate signals. ***As a general rule, only single outliers should be removed from rate signals.*** However, note that in the later automatic well-test analysis, outlier removal and de-noising is only performed with the pressure signals.

Let us next illustrate how the outlier median filter works. Consider a signal with 11 signal measurements. We choose to use a window size of five. This means that two successive outliers can be removed. Let us for the moment assume that we have a good estimate of the noise level, in this case $\sigma = 2$. Let us consider the outlier filtering of five signal measurements (s_3-s_7) as shown in *Figure 5.10.* The median value in the first window considered is 2. The signal measurement s_3 with a value of 1 is therefore required to have a value below -1 or above 3 to be considered as an outlier according to *eq. (5.1).* Thus s_3 cannot be classified as an outlier and is kept. The same applies to the measurement s_4. The signal measurement s_5 is required to have a value between 1 and 5, otherwise it will be classified as an outlier and removed. Since s_5 has a value of 8, it is classified as an outlier and removed. The same applies to s_6 which also is classified as an outlier and removed. Finally, s_7 is not classified as an outlier and is kept.

The window size will determine how many successive outlier we can remove. The maximum number of successive outliers that can be removed is given as;

$$outlier_{\max} = \frac{m-1}{2} \tag{5.2}$$

where *m* is the window size. Thus if a window size of 19 is used, the maximum number of successive outliers that can be removed is nine.

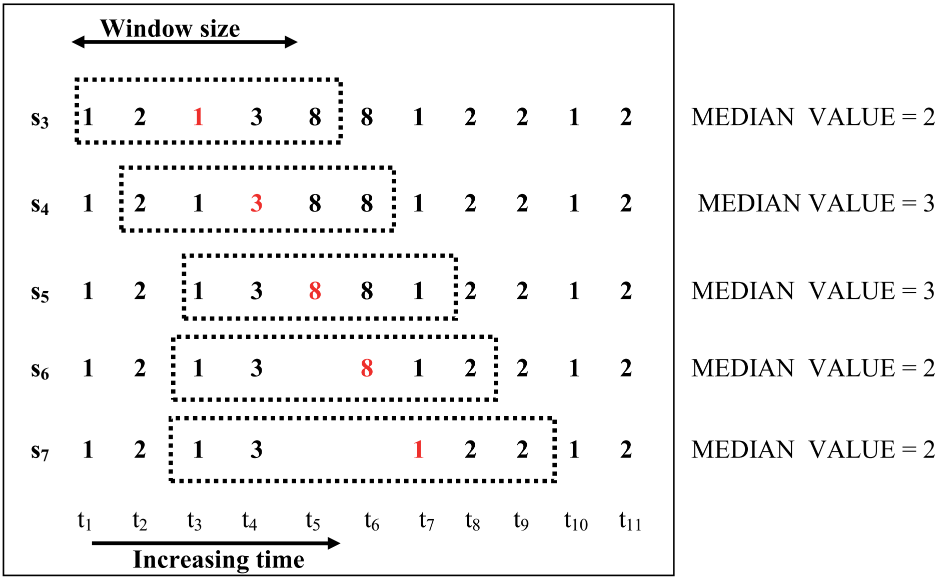
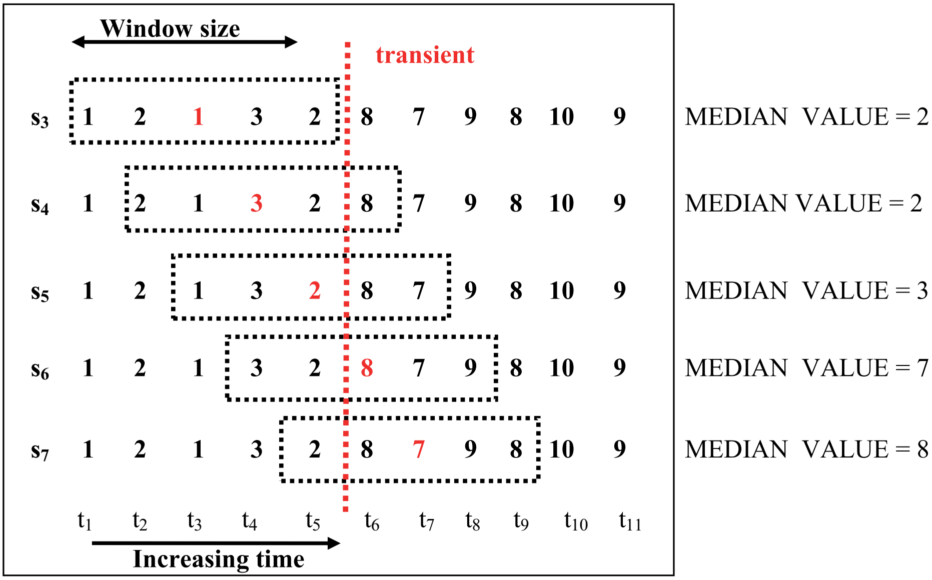


Figure 5.10: *Median filter used to filter the signal measurements s_3-s_7 for outliers. The signal measurements s_5 and s_6 are detected as outliers and removed.*

An important feature of the median filter is that it distinguishes outliers from transients. Consider a signal with 11 measurements, a median filter with window size five and estimated noise level $\sigma = 2$. Outlier removal for the signal measurements $s_3 - s_7$ is illustrated in *Figure 5.11*. A transient starts at time t_6 and none of the signal measurements considered is removed according to *eq. (5.1)*.



	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	
s_3	1	2	1	3	2	8	7	9	8	10	9	MEDIAN VALUE = 2
s_4	1	2	1	3	2	8	7	9	8	10	9	MEDIAN VALUE = 2
s_5	1	2	1	3	2	8	7	9	8	10	9	MEDIAN VALUE = 3
s_6	1	2	1	3	2	8	7	9	8	10	9	MEDIAN VALUE = 7
s_7	1	2	1	3	2	8	7	9	8	10	9	MEDIAN VALUE = 8

Figure 5.11: *Median filter used to filter the signal measurements s_3-s_7 for outliers.*

Our recommendation for outlier removal is hence based upon a median filter with noise estimation from the Waveshrink method. We suggest removing up to nine successive outliers from pressure signals while only single outliers are removed from rate signals. For the moment, we will assume that we can get an accurate estimate of the noise level in the signal measurements from Waveshrink. An example of the median filter used with signal measurements containing several successive outliers is shown in *Figure 5.12, Figure 5.13 and Figure 5.14*. The outliers present are successfully removed utilizing the median outlier filter, although no more than nine successive outliers are assumed.

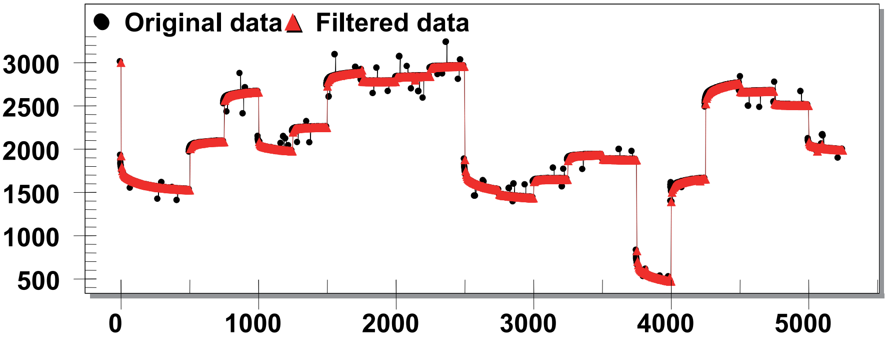


Figure 5.12: *Successful removal of single as well as successive outliers using a median filter.*

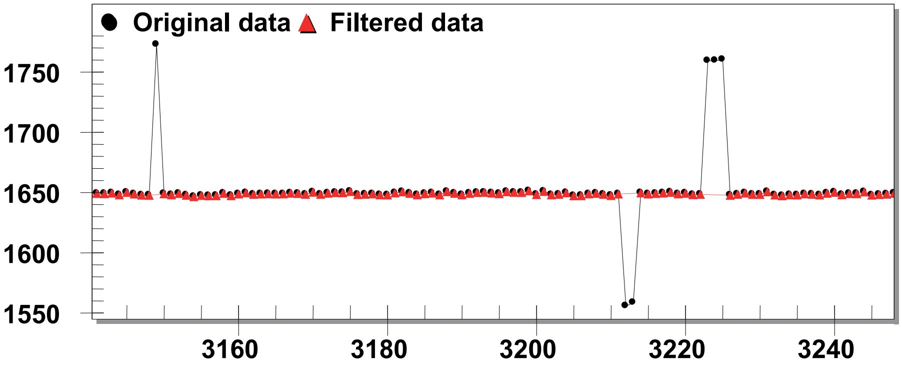


Figure 5.13: *Zoom in on Figure 5.12. Successful removal of single as well as successive outliers applying a median filter.*

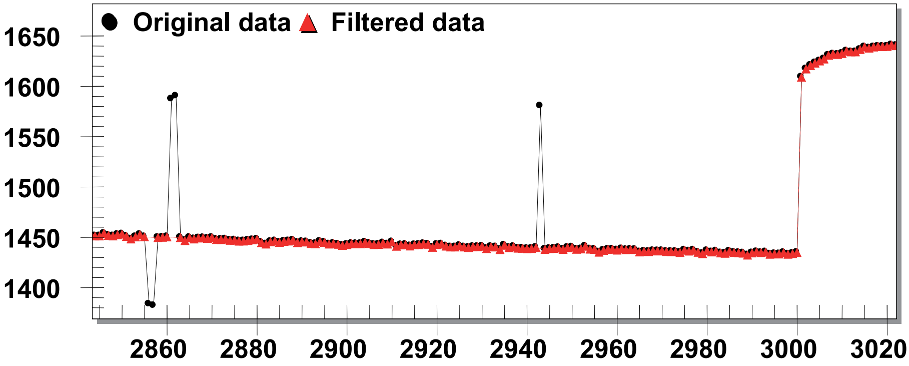


Figure 5.14: *Zoom in on Figure 5.12. Successful removal of single as well as successive outliers applying a median filter.*

5.3 De-noising

We choose to use the Waveshrink method for de-noising pressure and rate signals as previously done by Kikani et. al.[(4)] and Athichanagorn et al. [(5)] Kakani proposed to use the Haar wavelet (*eq. (4.13)*) for de-noising while Athichanagorn proposed to use the non-orthogonal spline wavelet (*eq. (4.39))*. They both applied the universal threshold estimator. In addition, they both utilized the Hard and Soft Shrinkage rules. Athichanagorn in addition proposed a hybrid approach; i.e. to use the Hard shrinkage rule around sharp signal discontinuities and the Soft shrinkage rule otherwise. However, few authors discuss the selection they make as to which wavelet, shrinkage rule, threshold estimator, noise estimator and primary resolution level to choose. Consequently they do not know the performance of Waveshrink with the selection they have made compared to other possible selections.

The exception is a recent study by Olsen et al.[1] and the recent study by Ortiz et al. [12]. This work is based upon the work by Olsen et al. However, we propose a new noise estimator and a new threshold estimator in this work as a numerical error later was discovered in the work by Olsen. Ortiz et al. applied different wavelets, shrinkage rules and threshold estimators for de-noising with Waveshrink. They compared the wavelets (with the exception of Spline4246) and the shrinkage rules as described in chapter 4.8. In addition they considered the SURE, Universal and Minimax threshold estimators. The noise estimator used in their study is MAD *eq. (4.82)* and the primary resolution level is the one given by Olsen et al. [1] as will be discussed later in this chapter. Ortiz et al. concluded that Daubechies8 was the best wavelet, Firm shrinkage rule was the best shrinkage rule while the best threshold estimator was Minimax. The best selections for use in Waveshrink were determined from visual inspection of the filtered signals. The results are different from the results that will be shown in this chapter. In particular, it is surprising for this author that the Daubechies8 wavelet is considered the best wavelet. However differences in both signal generation and performance evaluation is believed to explain the difference between the results from Ortiz et al. and the results presented in here.

Many different selections are available in Waveshrink, and the performance of Waveshrink will depend critically upon the selections made. By determination of the "best" selections for the signals we consider, this hold the promise for automatic signal de-noising. Before the actual performance analysis is done, we need to define how the performance of the different selections and methods in Waveshrink is evaluated. In addition, Waveshrink requires equally spaced signal measurements. Often this is not available and we next need to determine what to do in the case of unequally spaced signal measurements.

5.3.1 Best Filter Definition

First we need to define what we mean by "best" selections in Waveshrink. There are many ways to define this, but we choose to define it as described in this section. Let's consider the signal measurement s_i influenced by noise;

$$s_i = y_i - \varepsilon_i \tag{5.3}$$

The signal measurements from the true process state are called y_i and the noise with zero mean and a given standard deviation is denoted ε_i. Outliers are assumed removed from the signal measurements prior to de-noising. The signal approximations, where noise have been removed by some suitable method(s), are called $f_{i.}$. The error is estimated from;

$$RMSE = \sqrt{\frac{1}{N} \sum_{j=0}^{N-1} (y_i - f_i)^2} \quad . \tag{5.4}$$

This is the root-mean-squared error which is a very common way to estimate the error, and throughout this work, this is the error considered. An error of zero means a perfect match between the signal without noise and the filtered signal. A large error means that the filtered signal is still heavily influenced by noise. If the error after filtering is larger than the initial error, this might in addition indicate that the filtering has over-smoothed the signal, i.e. some of the underlying signal has been removed in the filtering process instead of or in addition to noise. Over-smoothing is something that is important to avoid. Thus, instead of obtaining a better estimate of the underlying signal by performing the signal de-noising, we can actually obtain a worse estimate. In this work we have chosen to define the best filter selections in the following way;

The best methods and selections used in Waveshrink are chosen as the selections and methods that on average results in the smallest error** (eq. (5.4))* ***between the signal without the influence of noise and the filtered signal, while filtering a large number of signals with similar characteristics.

This means that for one single measured signal, other selections and methods than what has been classified as "best" in this work might actually have a smaller error. However, while considering the average error for a large number of measured signals with the same characteristics, as for instance a large number of measured pressure signals, the selections and methods recommended in here on average should give the smallest error, and will for that reason be classified as best. In general, at least 500 different synthetic pressure or rate signals will be considered for each filtering exercise.

It should be emphasised that we by no means claim that Waveshrink should be considered as the "best" method for de-noising. The performance of Waveshrink is not compared to other available methods for de-noising. However, if Waveshrink is utilized for de-noising, the selections recommended in Waveshrink will be shown to be best in terms of root-mean-squared error for the signals considered. It should in addition be noted that it has been shown that Waveshrink have asymptotical properties within the ideal performance (55; 62; 63; 64) for applications in de-noising.

When calculating the average error for a large number of de-noised signals, we need to take into consideration that the noise level in the various signals might be different. Recall that we will only consider synthetic signals where the true noise level is known. To be able to compare the error between different de-noised signals with different noise levels, we need to divide the *RMSE* value by the true known noise level added to the signal measurements.

$$RMSE_{\sigma} = \frac{RMSE}{\sigma} \tag{5.5}$$

σ is the standard deviation of the normal distributed noise added to the signal measurements and *RMSE* is given by *eq. (5.4)*. Before the signal measurements are de-noised, $RMSE_{\sigma}$ will be close to one, while $RMSE_{\sigma}$ will close to zero if most of the noise successfully has been removed in the de-noising. Notice that the error considered in Olsen et al. (1) was the mean-squared error (MSE), while we in this thesis consider the root-mean-squared error (RMSE). We chose to present the results in this work as RMSE as this has the same unit at the dependent variable. In addition, doubling the RMSE means that the error is twice as big, but doubling the MSE does not mean that the error is twice as big.

When utilizing the error estimation in *eq. (5.5)*, we need to know the true underlying signal trend denoted $Y=[y_0,y_1,...y_{N-1}]$ where N is the number of signal measurements. In general, the only way to know the true underlying signal is to generate the signal synthetically, i.e. we need to generate the signal based upon a known underlying process model described by some known mathematical model. The creation of these synthetic signals was discussed in chapter 5.1.

Let us illustrate the difference in performance of Waveshrink with an example. One standard deviation white noise was added to a pressure signal with 2000 measurements. The pressure signal is generated as described in chapter 5.1 and is shown in *Figure 5.15.*

The pressure signal in *Figure 5.15* was de-noised using different wavelets in Waveshrink. If the Waveshrink selections work well, the error should be close to zero and if the Waveshrink selections work badly the error should be close to one. (The initial error by *eq. (5.5)* will be close to 1 if $F=[f_0,f_1,...f_{N-1}]$ equals $S=[s_0,s_1,...s_{N-1}]$. N is the number of signal measurements). Hard shrinkage rule *eq. (4.57)*, Universal threshold estimator *eq. (4.65)*, primary resolution level $j_0=5$ and the MAD noise estimator *eq. (4.82)* is applied in all cases. It is clearly demonstrated in *Figure 5.16* that the performance of Waveshrink will depend critically upon the selections made. The best wavelet for this signal in terms of root-mean-squared error is the Spline39. The worst wavelets in terms of root-mean-squared errors are the Haar, Coiflet18 and Spline4246. The root-mean-squared error for the worst wavelet is about 50% larger than for the best wavelet.

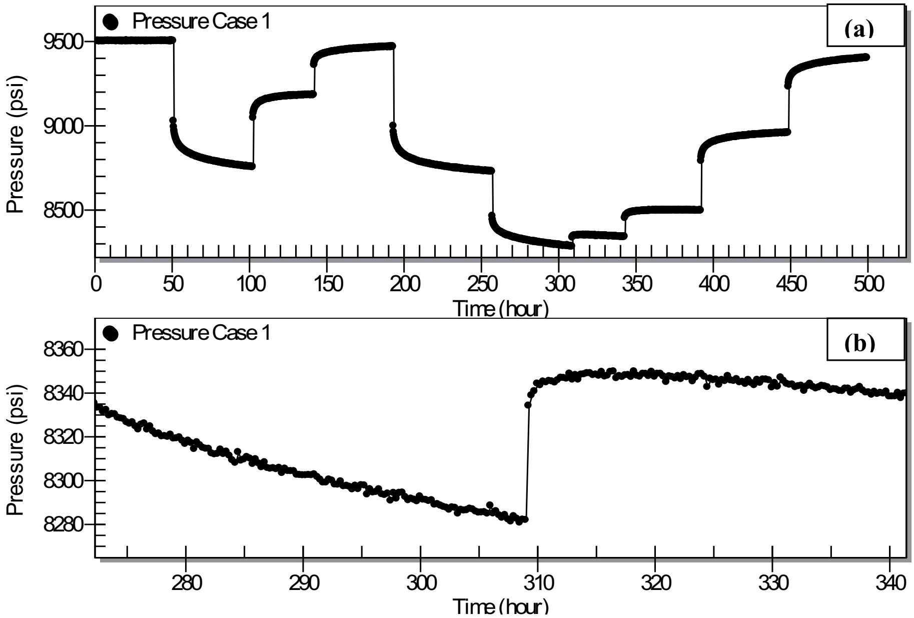


Figure 5.15: ***(a)*** *Pressure signal added one standard deviation of white noise.* ***(b)*** *Zoom in on pressure signal in Figure 5.15 (a).*

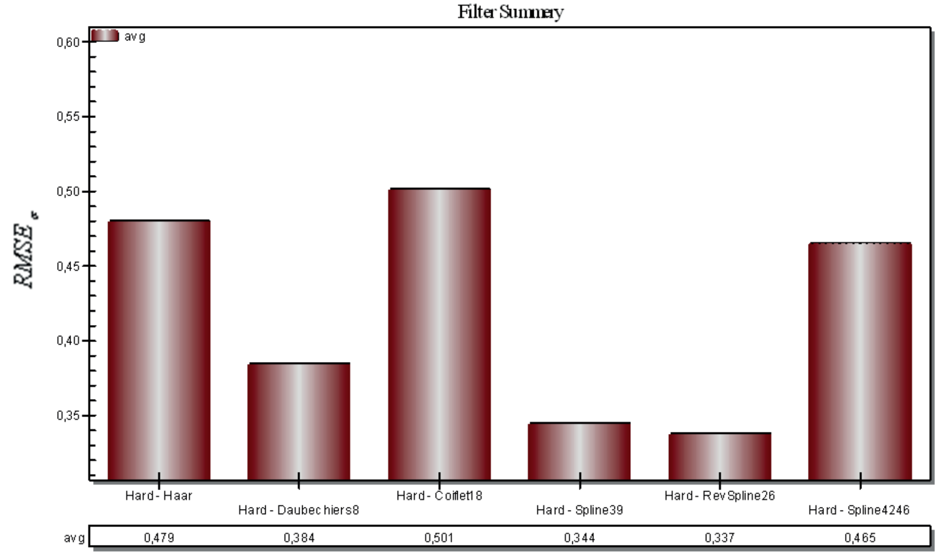


Figure 5.16: *Root-mean-squared error after noise filtering of pressure signal applying different wavelets.*

It is in particular easy to see the difference in performance close to transients. This is shown in *Figure 5.17* for wavelets Haar, Daubechies8, Coiflet18 and Spline39. Oscillations around the transient are observed for wavelets Daubechies8, Coiflet18 and Spline39. This is a well known problem from Fourier analysis. This is however a much smaller problem in wavelet analysis, in particular since we have the opportunity to chose among wavelets that minimize this problem. In this case, the Haar wavelet results in very limited oscillations close to the transient as shown in *Figure 5.17.*

Let us again consider filtering of the signal shown in *Figure 5.15.* This time we apply different shrinkage rules in Waveshrink and the results are shown in *Figure 5.18.* The performance of the Hard, Garrote and SCAD shrinkage rules for this signal is rather good, while the performance of the Soft shrinkage rule is poor. This since the Soft shrinkage rule tends to over-smooth the signal. Wavelet Daubechies8, Universal threshold estimator *eq. (4.65)*, primary resolution level $j_0=5$ and the MAD noise estimator *eq. (4.82)* is applied in all cases. Again we have demonstrated that the performance of Waveshrink will depend critically upon the selections made. Similar difference in performance can be shown for different noise estimators, primary resolution levels and threshold estimators.

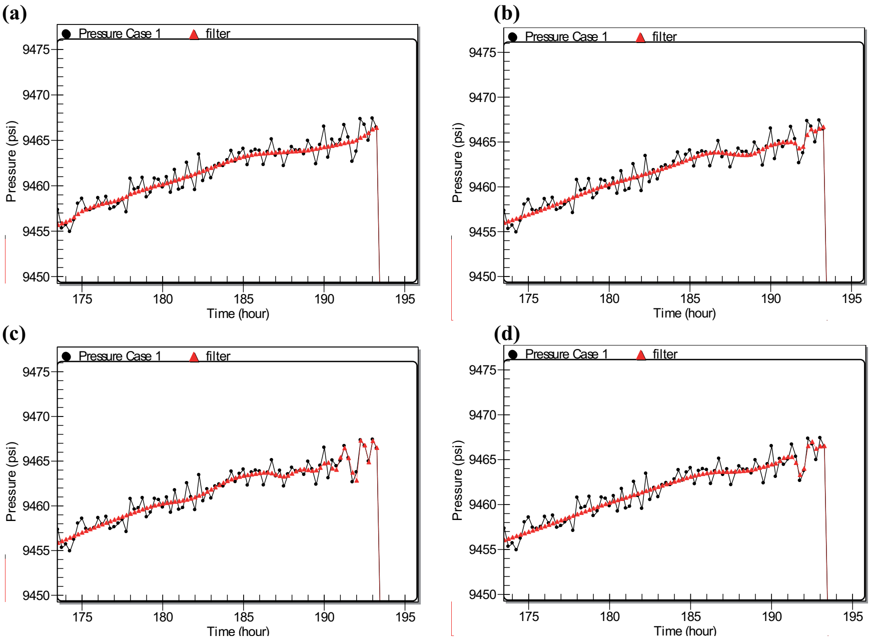


Figure 5.17: *Signal filtering around transient for wavelet **(a)** Haar, **(b)** Daubechies8 **(c)** Coiflet18 and **(d)** Spline39.*

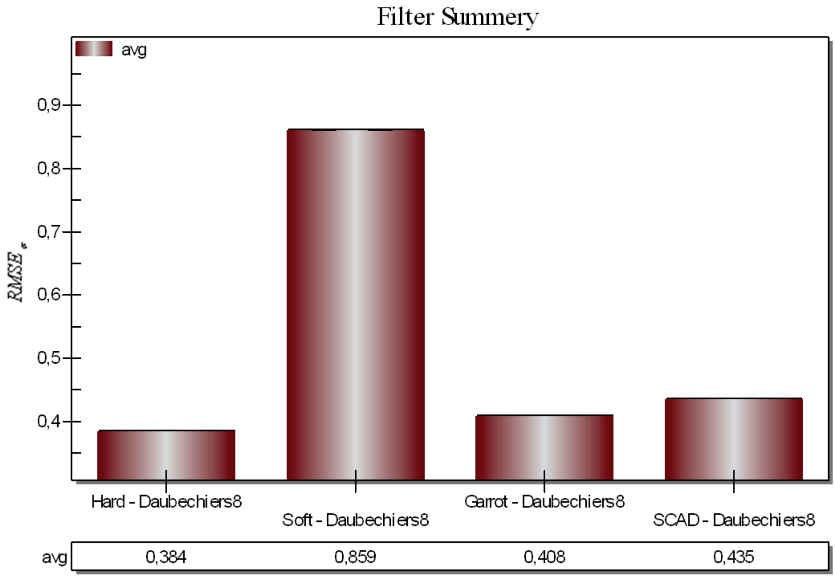


Figure 5.18: *Root-mean-squared error after noise filtering of pressure measurements applying different shrinkage rules.*

5.3.2 Unequally Spaced Signal Measurements

Often we are required to deal with unequally spaced signal measurements. However, in wavelet analysis the measurements are assumed equally spaced. Often this is corrected for by interpolating the signal measurements to equally spaced signal measurements. This is however a poor solution. ***Instead we recommend treating the signal measurements as equally spaced when used in wavelet analysis, even in the cases where they are unequally spaced.*** The reason for this is since interpolation tends to magnify noise structures within the signal measurements.

Let us consider the pressure signal as shown in *Figure 5.12* with 5251 measurements. We reduce the number of signal measurements by 25% (3938 measurements), 50% (2626 measurements) and 75% (1313 measurements). The reduction is done by removing those signal measurements with the smallest change since the previous measurement. The initial measurements *without noise and outliers* are used to create these three cases with unequally spaced measurements. There are few measurements in periods with small signal variations and many signal measurements in periods with large signal variations. Unequally spaced measurements are illustrated as the red curve in *Figure 5.19* where the number of measurements is reduced with 75% compared to the black curve. The measurements shown as the black curve is the initial and equally spaced measurements. Next, two standard deviations white noise is added to the three cases with unequally spaced measurements. The performance of wavelet filtering with unequally spaced measurements is then compared to the cases where linear interpolation is used to obtain equally spaced measurements.

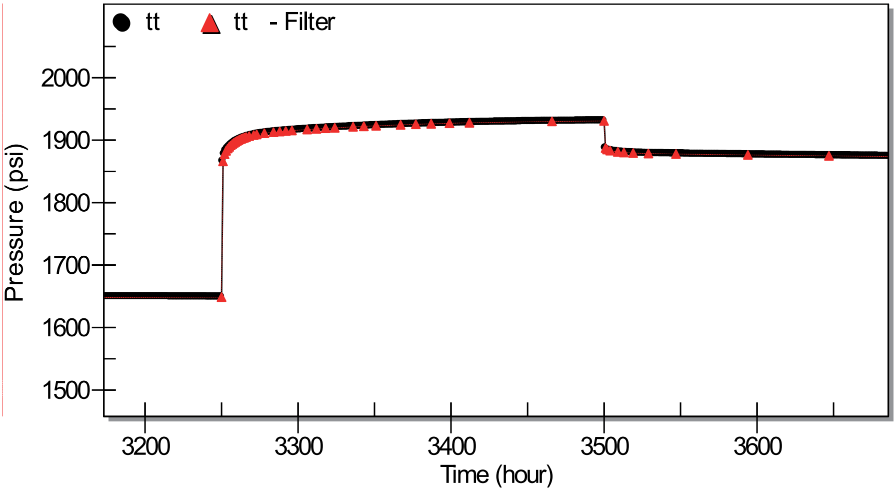


Figure 5.19: *The number of measurements is reduced by 75% in the red curve compared to the black curve.*

The three cases with unequally spaced noisy measurements are interpolated back to the initial spacing and will consequently have the same number of measurements as in the initial data set. We de-noise both the interpolated noisy data sets and the unequally spaced noisy data sets and compare their performance. In all cases, wavelet Spline39, SCAD shrinkage rule, Universal threshold estimator, MAD noise estimator and primary resolution level $j_0=5$ is applied. Root-mean-squared errors are calculated for all de-noised signals. The interpolated measurements are compared to the initial noise free pressure signal with 5251 measurements. The unequally spaced noisy signal measurements are compared to the unequally spaced noise free signal measurements. The estimated root-mean-squared errors are shown in *Figure 5.20*.

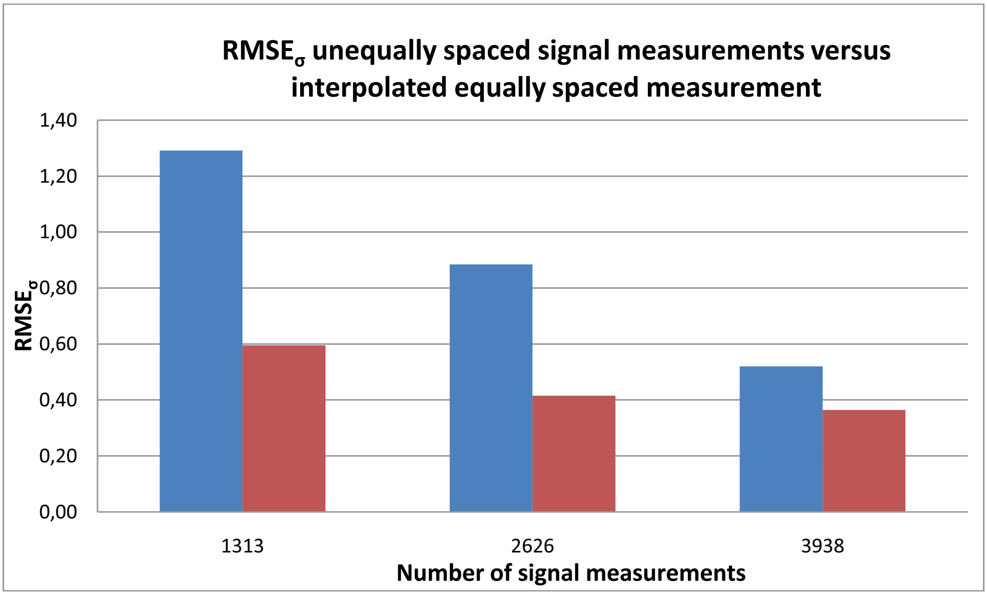


Figure 5.20: *RMSE$_\sigma$ in wavelet filtering for interpolated equally spaced measurements (blue) versus unequally spaced measurements (red).*

We clearly see that the RMSE$_\sigma$ increases considerably if interpolation is utilized with unequally spaced signal measurements. Consequently it is highly beneficial in wavelet filtering to treat the signal measurements as equally spaced even if they are unequally spaced. The reason for this is since interpolation tends to magnify the underlying noise structures in the signal measurements as illustrated in *Figure 5.21*. The red curve, initially containing unequally spaced measurements, is interpolated back to original spacing. The original and noise-free signal measurements are shown as the black curve. In particular between *t=1800* and *t=2000*, interpolation magnifies the underlying noise structures in the signal measurements.

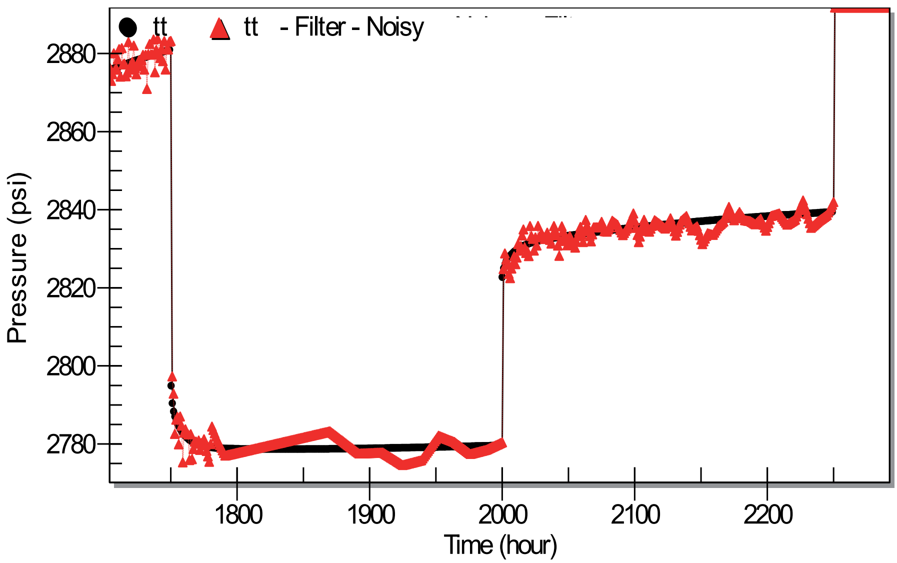


Figure 5.21: *Interpolation tends to magnify the underlying noise structures in the signal measurements as illustrated with the red curve. The black curve shows the initial noise free and equally spaced signal measurements.*

Next we will describe how the performance analysis used to determine the best selections in Waveshrink was performed.

5.3.3 Performance Analysis

Several selections need to be made in Waveshrink. Various threshold estimators might require an estimate of the noise level in the signal measurements, and thus we first need to determine how to estimate the best noise level for use with our signals. Different methods are available in order to determine the primary resolution, and we will need to determine which primary resolution level to apply for the signals we consider. The wavelet filtering can be accomplished with several different *wavelets* and next we need to consider which wavelet to apply to obtain the best results for the signals we consider. The *shrinkage rule* determines how the different detail coefficients should be treated, in other words, which ones to keep, remove or change. Thus we further need to determine the best shrinkage rule for the signals we consider. Recall that small detail coefficients in the wavelet transform was assumed to be originating from noise while the large ones were originating from the signal itself. We will consequently need to determine a *threshold* for how to separate “large” wavelet coefficients from “small” ones. This threshold will depend on the resolution level considered and the level of noise in the signal measurements. Different methods can again be used to estimate this threshold and we will need to select the one best suited for our signals. How many resolution levels to change or remove wavelet detail coefficients from in the wavelet transform will be determined by the *primary resolution level*. The different selections required in Waveshrink will be determined in the following sequence;

- Noise estimator
- Shrinkage rule and wavelet (simultaneously)
- Threshold estimator
- Primary resolution level

We will start the performance analysis by determination of which noise estimator to apply. Since we are utilizing synthetic pressure and rate signal added white noise, we know the "true" noise level. ***Hence the best noise estimator is the one that on average estimates a noise level with the smallest root-mean-squared error compared to the true known noise level while filtering a large number of signals with similar characteristics.***

Next, we will simultaneously determine which shrinkage rule and wavelet to utilize. The best wavelet and shrinkage rule can be determined separately from the best noise estimator, threshold estimator and primary resolution level. For now, all resolution levels available in the wavelet transform, and given by *eq. (4.50)* will be included in Waveshrink. Hence the primary resolution level is initially chosen equal the coarsest resolution level. The assumption is that we are able to choose the best possible threshold for use in Waveshrink at each resolution level. Note that the best possible threshold at the any of the considered resolution levels might be zero. To determine the best wavelet and shrinkage rule we will therefore utilize non-linear regression to determine the threshold at all available resolution levels that minimize the square difference between the filtered signal and the original noise free signal.

Let us considered a signal with 5250 pressure measurements were the finest resolution level is 12 as given by *eq. (4.50)*. Recall that the finest resolution level is approximated by the signal samples while the coarsest resolution level in the wavelet transform is given by *eq. (4.50)*. For the moment we choose to include all available resolution levels *(j=1, 2, ...,12)* in de-noising. In *Figure 5.22*, the upper blue line shows the thresholds estimated by the universal threshold estimator for the 12 resolution levels considered. However, imagine that we can adjust the thresholds at each resolution level by an optimization routine in order to obtain the smallest error by *eq. (5.5).* Then we would obtain the optimal threshold at each resolution level as shown by the red line. This simply means that no matter how well we estimate the threshold and no matter which threshold estimator we apply, we can not for this signal obtain a root-mean-squared error smaller than 0.285 with the wavelet and shrinkage rule considered. The root-mean-squared error for the Universal threshold and the optimal threshold selection is shown in ***Figure** 5.23*. To obtain a smaller error (if possible at all) we would have to change the wavelet and/or the shrinkage rule. Hence the purpose of any threshold estimator should be to estimate the thresholds as close as possible to this red lower line for the wavelet and shrinkage rule considered. Note that the best possible threshold at resolution level 6 and 8-12 is zero.

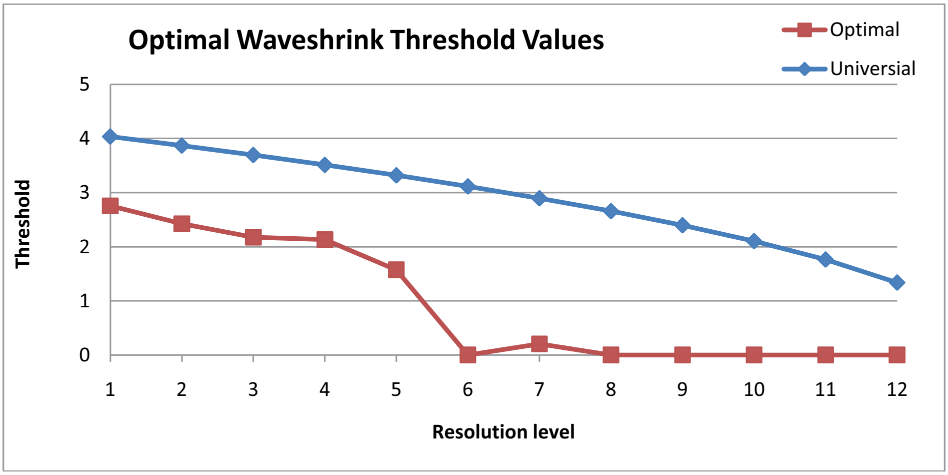


Figure 5.22: *Estimated thresholds at 12 decomposition level by Universal threshold and by optimization routine.*

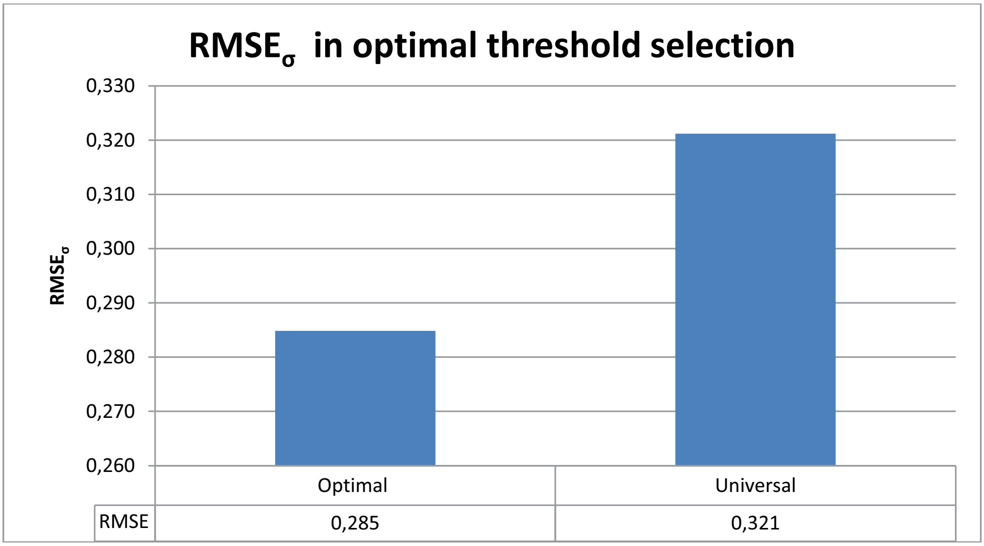


Figure 5.23: *Root-mean-squared error in optimal threshold selection versus Universal threshold.*

The optimization routine is similar to the non-linear regression analysis described for well-test analysis in chapter 3 and the optimization problem can be mathematically stated as;

$$\min_{\lambda \in \Re \geq 0} J(\lambda) = (F - Y)^2 \tag{5.6}$$

where $Y=[y_0,y_1,...,y_{N-1}]$ are the noise free measurements and the filtered signal measurements $F=[f_0,f_1,...,f_{N-1}]$ were defined in *eq. (4.54)* as;

$$\begin{aligned} X &= W(S) \\ Z &= \delta(X,\lambda) \\ F &= W^{-1}(Z) \end{aligned} \qquad (4.54)$$

where $S=[s_0,s_1,...,s_{N-1}]$ is the measurements contaminated with white noise $\varepsilon=[\varepsilon_0,\varepsilon_1,...,\varepsilon_{N-1}]$, *W* is the wavelet transform, W^1 is the inverse wavelet transform, *δ* is the shrinkage rule and *λ* is the thresholds. Again the Levenberg- Marquardt method is used in the optimization. This optimization routine requires function evaluation as well as function derivatives with respect to the parameters considered as unknowns. The function evaluation is the output from Waveshrink as given by *eq. (4.54)*. For the function derivatives we apply numerical derivatives with step length of 0.1 for SCAD, Garrote and Soft shrinkage rules. For the Hard shrinkage rule, a step length of 0.3 is used. This since instability is experienced with the Hard shrinkage rule for smaller step length. The Firm shrinkage rule is excluded in the optimization routine since two different thresholds are required, making threshold selection difficult. In addition, the Block shrinkage rule is excluded in the optimization routine since no threshold estimation is required with this shrinkage rule.

The best shrinkage rule and the best wavelet for use in Waveshrink will be determined as the shrinkage rule and wavelet that on average give the smallest root-mean-squared error (eq. (5.5)) from filtering at all available decomposition levels with thresholds determined from the optimization routine. The exception is for the Block shrinkage rule where no thresholds are required and for the Firm shrinkage rule where only the Minimax thresholds as shown in *Table 4.5* are considered.

Next we will determine the best threshold estimator. ***The best threshold estimator is the estimator that estimates the thresholds with the smallest root-mean-squared error compared to the threshold estimated by the optimization routine with the wavelet and shrinkage rule considered as best.***

Finally, we need to determine which primary resolution level to choose. Obviously the best would be to include all available resolution level given that the best possible threshold is chosen at each resolution level. However, the gain by including the coarsest resolution levels is usually very small or equal zero in terms of improved root-mean-squared error. A more practical approach in thus to stop filtering when there are small improvements by including additional resolution levels in Waveshrink. However, in the case of the primary resolution level, we will not classify the proposed primary resolution level as "best" since the root-mean-squared error for high primary resolution levels in most cases will be constant. ***Instead we choose to define the minimum recommended primary resolution level as the resolution level where the error in Waveshrink is reduced by less than 0.001 in eq. (5.5) when including an additional and coarser resolution level.***

All signals utilized in the performance analysis are generated as described in chapter 5.1. The performance of Waveshrink with different selections will be evaluated as described in section 5.3.1.

The next sections will provide a comparison of a large number of selections and methods for use in de-noising of typical signals encountered in the E&P industry. Recommendations for the selections and methods to use in Waveshrink will be given for the signals considered. In addition, a new noise and a new threshold estimator are suggested. We will start by determining which noise estimator to apply in Waveshrink.

5.3.4 Noise Estimation

In this section we will check the performance of the two most common noise estimators found in the literature; the MAD noise estimator *eq. (4.82)* and the standard deviation noise estimator *eq. (4.83)*. They where both previously described in section 4.8.4. In addition, we will propose a new estimator referred to as the RMAD noise estimator.

The standard deviation and MAD noise estimator both estimate the noise from the absolute value of the detail coefficients at the finest resolution level in the wavelet transform. Consequently they might depend upon the wavelet considered. Initially we hence need to consider whether the noise estimators considered depend upon the wavelet utilized. We chose to test this by considering the MAD noise estimator with different wavelets. 500 different pressure and rate signals were generated and three standard deviations white noise was added to the various signals. The average noise level is estimated from the MAD noise estimator for the wavelets considered. If the wavelet and the noise estimator considered works well, the noise estimate should be close to three (the amount of noise added to the original and noise-free signal measurements).

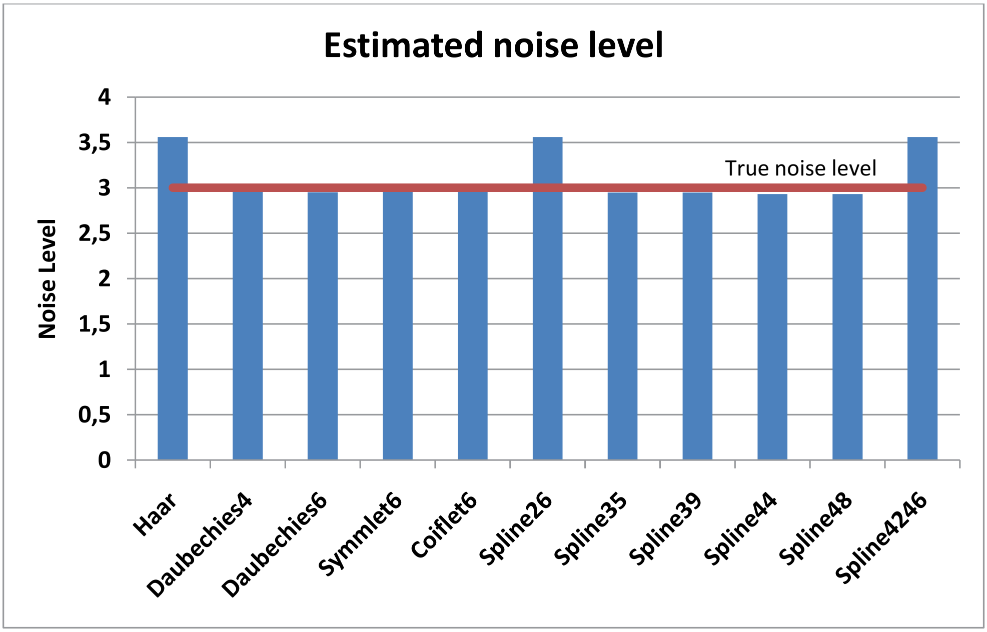


Figure 5.24: *Noise level estimated by MAD noise estimator utilizing various wavelets for pressure signals.*

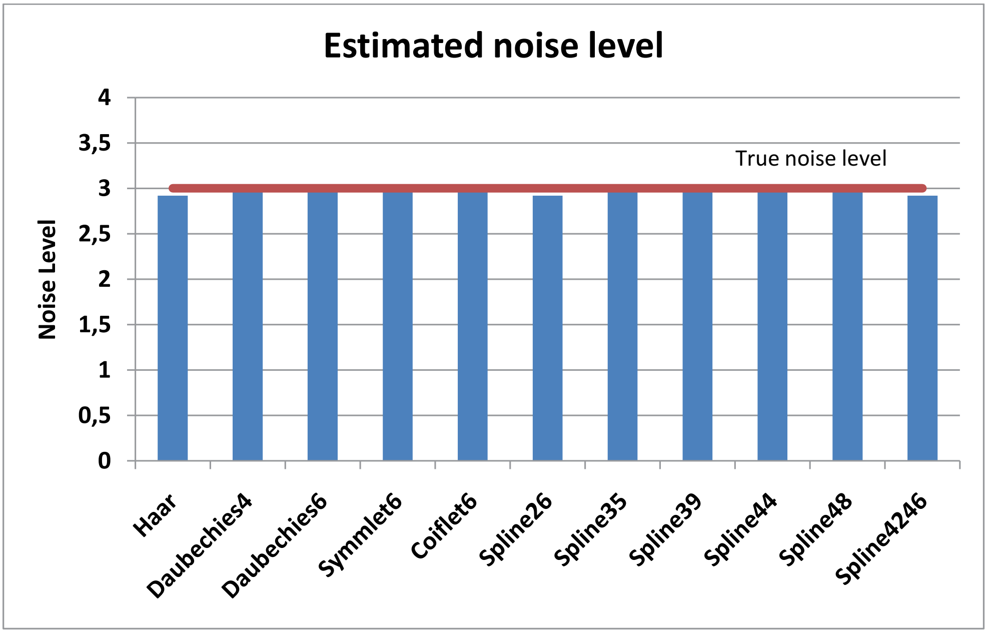


Figure 5.25: *Noise level estimated by MAD noise estimator utilizing various wavelets for rate signals.*

In *Figure 5.24* it is shown that the MAD noise estimate for pressure signals only is vaguely dependent upon the wavelets considered. The exceptions are the wavelets Haar, Spline26 and Spline4246. Thus, the importance of choosing a particular wavelet for noise estimation with the MAD for pressure signals appears to be small as long as the three wavelets mentioned above are avoided. In *Figure 5.25* it is shown that the MAD noise estimate only is vaguely dependent upon the wavelets considered for rate signals. As opposed to pressure signals, all wavelets considered in this work can be used for noise estimation with rate signals.

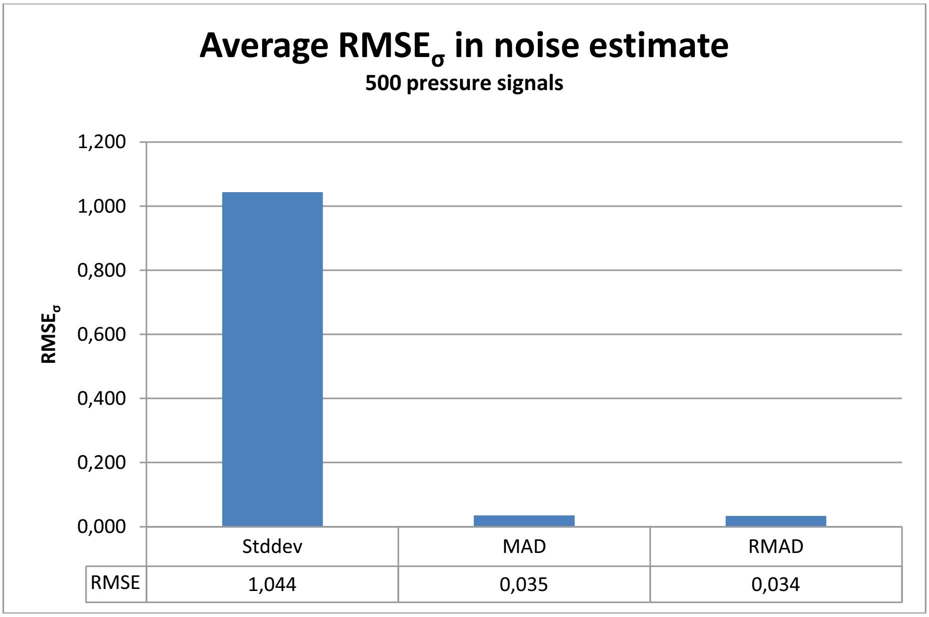
In the initial filtering exercises it was discovered that if a significant part of the signal was zero (and accordingly without any noise as we have defined it), as for instance will happen for rate signals during shut-in periods in oil and gas wells, this needs to be taken into account in the MAD noise estimator. Thus, zero wavelet coefficients are not included in a new proposed noise estimator referred to as the robust median noise estimator (RMAD) which is proposed as;

$$(5.7) \qquad \sigma = \frac{RMAD}{0.6745}$$

RMAD is the median value of the absolute value to the ***non-zero detail coefficients*** at the first decomposition level in the wavelet transform, as opposed to MAD noise estimation taking into account all detail coefficients available at the first decomposition level in the wavelet transform. Thus this modification of MAD assumes that there will be such a large number of zero values among the detail coefficients caused by well shut-ins that they cannot adequately be described by the normal distribution imposed on the detail coefficients and thus must be handled separately.

Next, we would like to compare the performance of the three noise estimators; the RMAD noise estimator *eq. (5.7)*, the MAD noise estimator *eq. (4.82)* and the standard deviation noise estimator *eq. (4.83)*. 500 pressure and rate signals were considered and white noise was added to the signal measurements. The signals are generated as described in chapter 5.1 and the error is calculated as described in chapter 5.3.1. The results from noise estimation with the pressure signals show that the MAD noise estimator is superior to the standard deviation noise estimator. Excellent agreement between the estimated noise level and the true noise level is obtained with the MAD noise estimator resulting in a small root-mean-squared square error as shown in *Figure 5.26*. The only major drawback is encountered if parts of the signal are zero as typically encountered in rate signals during well shut-ins. This is corrected for by the introduction of the RMAD noise estimator in *eq. (5.7)*. Thus by considering the RMAD noise estimator, excellent agreement is also obtained for the rate signals considered as shown in *Figure 5.27.*

It should finally be noticed that a median noise estimator (MAD or RMAD) might be slightly unstable if utilized on very small data sets. Then the assumption of normal distributed detail coefficients might be poor. Statistically we need a "large" number of measurements before any assumed distribution can be considered as a good model. However, as long as at least a few hundred measurements are considered with the MAD or RMAD noise estimator, excellent results are obtained. This is shown in *Figure 5.28*. In this figure, the estimated noise level is poor compared to the true noise level when the number of measurement is below 200.



	Stddev	MAD	RMAD
RMSE	1,044	0,035	0,034

Figure 5.26: *Average root-mean-squared error in Waveshrink noise estimation from 500 different synthetic pressure signals added white noise.*

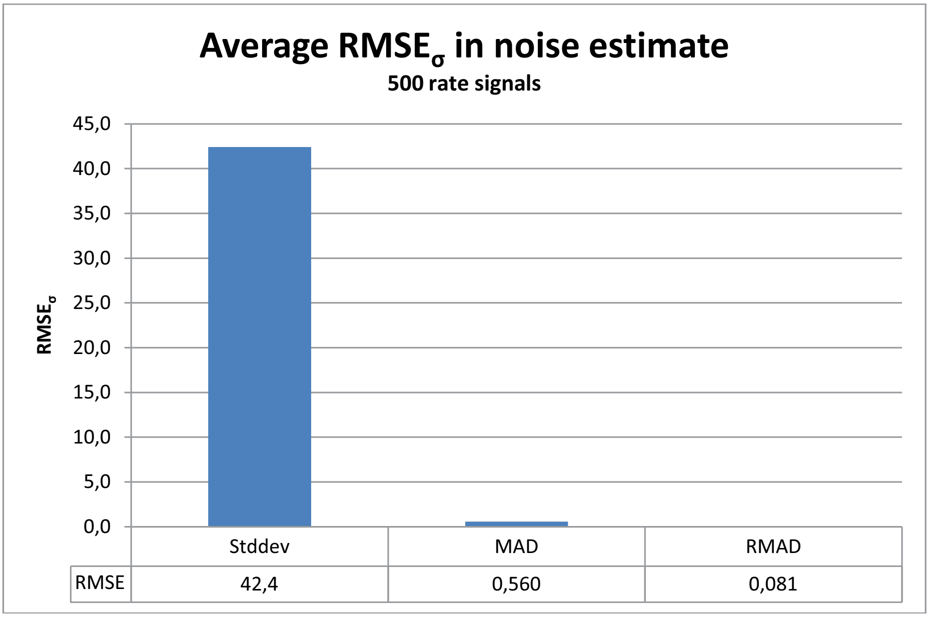


Figure 5.27*: Average root-mean-squared error in Waveshrink noise estimation from 500 different synthetic rate signals added white noise.*

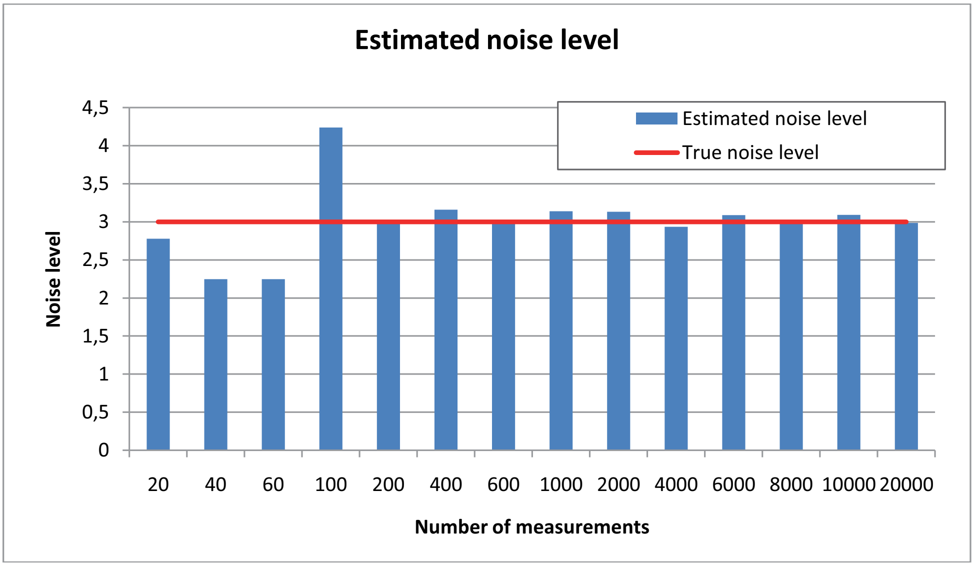


Figure 5.28: *Estimated noise level for 14 different data sets with different number of measurements.*

When an accurate estimate of the noise level is available, outliers can be removed by the use of a *median filter* as described in chapter 5.2.1.

Note that the noise estimator RMAD in *eq. (5.7)* is a redefinition of the RMAD noise estimators proposed by Olsen et al. [(1)] In that work, different RMAD noise estimators were proposed for pressure and rate signals. Unfortunately, an error was later discovered in the numerical implementation used by Olsen et al. [(1)]. Thus the RMAD noise estimator has been redefined in this work.

The best shrinkage rule and the best wavelet were determined simultaneously through a series of filtering exercises. We will address the shrinkage rule first.

5.3.5 Shrinkage Rules

The shrinkage rules considered are Hard, Soft, SCAD, Garrote, Block and Firm. The shrinkage rules tell how the various wavelet coefficients should by treated; which ones to keep, kill or shrink. The Soft shrinkage rule is supposed to perform well if signals are smooth and contain few transients but many noise spikes. The Hard shrinkage rule is supposed to work well in signals with many transients but few noise spikes. SCAD and Garrote provide a compromise between these two. The Block shrinkage rule is believed to work well in signals with transients since it considers coefficients in blocks rather than one by one. Our expectation to Firm shrinkage rule is small since it is only considered with the Minimax thresholds as given in *Table 4.4*.

The optimal thresholds for shrinkage rules Hard, Soft, SCAD, Garrote were determined by the optimization routine to obtain the best possible thresholds with each shrinkage rule. No separate determined threshold is required for Block shrinkage while only the Minimax threshold is considered for Firm shrinkage. The root-mean-squared error results from filtering of 500 pressure and rate signals are shown in *Figure 5.29* and *Figure 5.30* respectively. The results show that SCAD on average performs best in terms of the smallest root-mean-squared error for both rate and pressure signals. Hard shrinkage also performs well and in many cases achieves almost as good results as SCAD. In general, Soft shrinkage rule is the one that performs worst and should be avoided. Wavelets Spline39 and Spline4246 were applied with pressure and rate signals respectively.

At this stage we do not know which wavelet to use even though Spline39 and Spline4246 were utilized above. One might envision that other combinations of wavelets and shrinkage rule could provide better results. Consequently, the same filtering exercise as done in *Figure 5.29* and *Figure 5.30* had to be done for all wavelets considered. It was then established that the combination of wavelet Spline39 and shrinkage rule SCAD gave the smallest error for pressure signals, while the combination Spline4246 and shrinkage rule SCAD gave the smallest error for rate signals.

It might finally be noted that it is in the author's opinion that SCAD shrinkage yields visually more appealing results than the other shrinkage rules, since it does not tend to over-smooth the signals, as might be the case with Soft shrinkage, and it is also capable of reducing noise spikes as opposed to Hard shrinkage rule[(63)]. In terms of both error and more visual appealing signals, SCAD is the best choice.

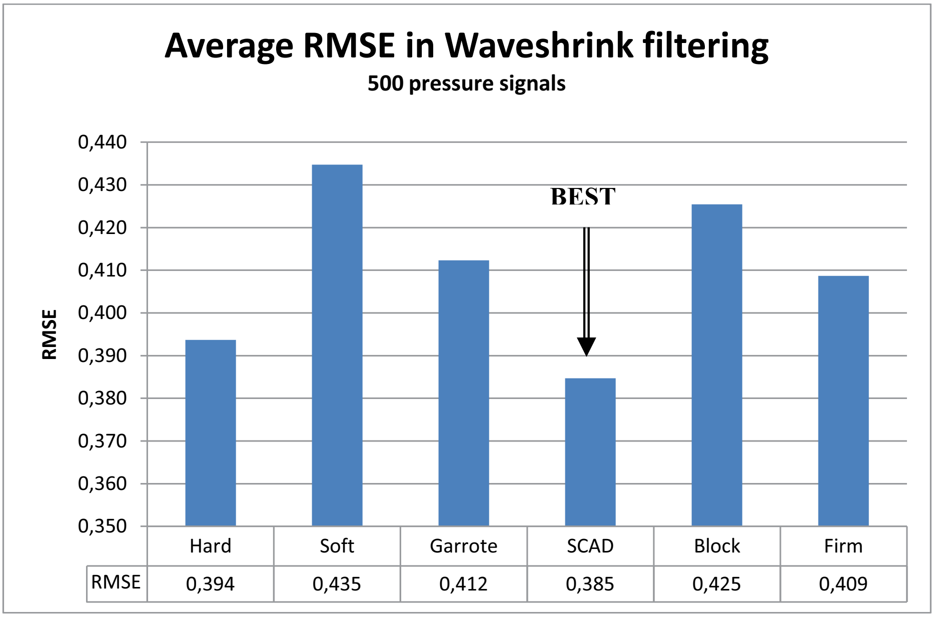


Figure 5.29: *Comparison of root-mean-squared errors using various shrinkage rules for 500 pressure signals. Wavelet Spline39 was used.*

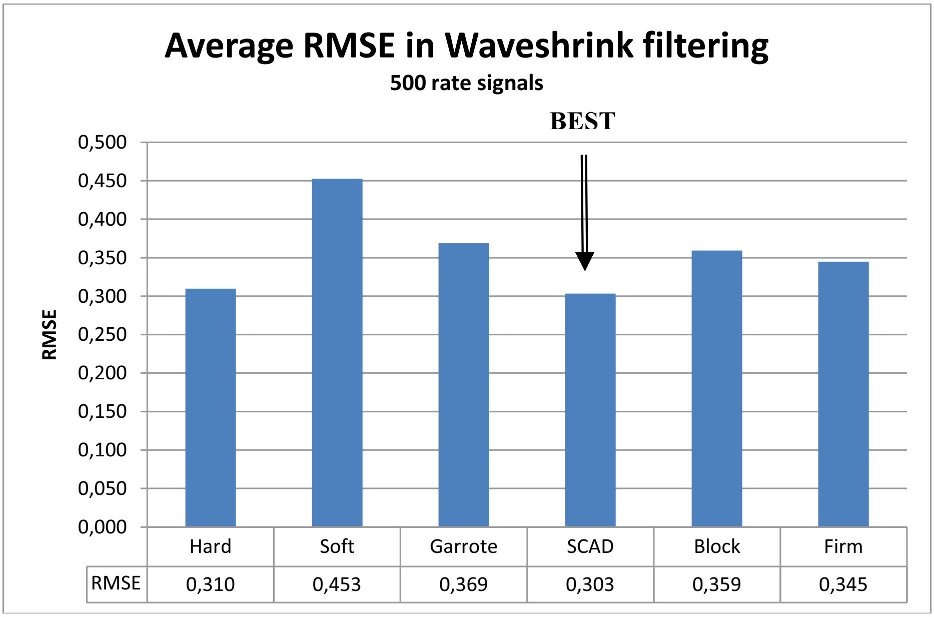
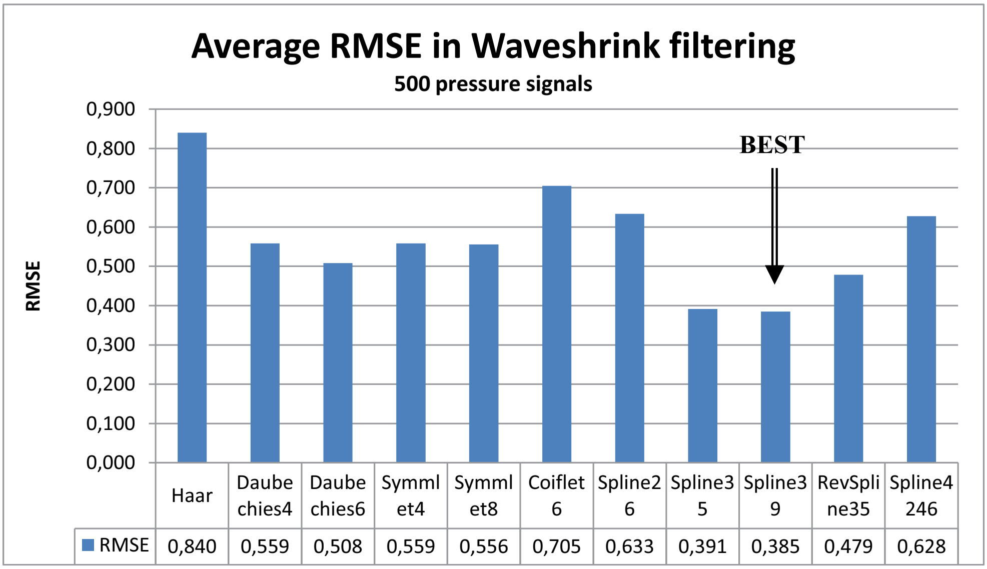


Figure 5.30: *Comparison of root-mean-squared errors using various shrinkage rules for 500 rate signals. Wavelet Spline4246 was used.*

5.3.6 Wavelets

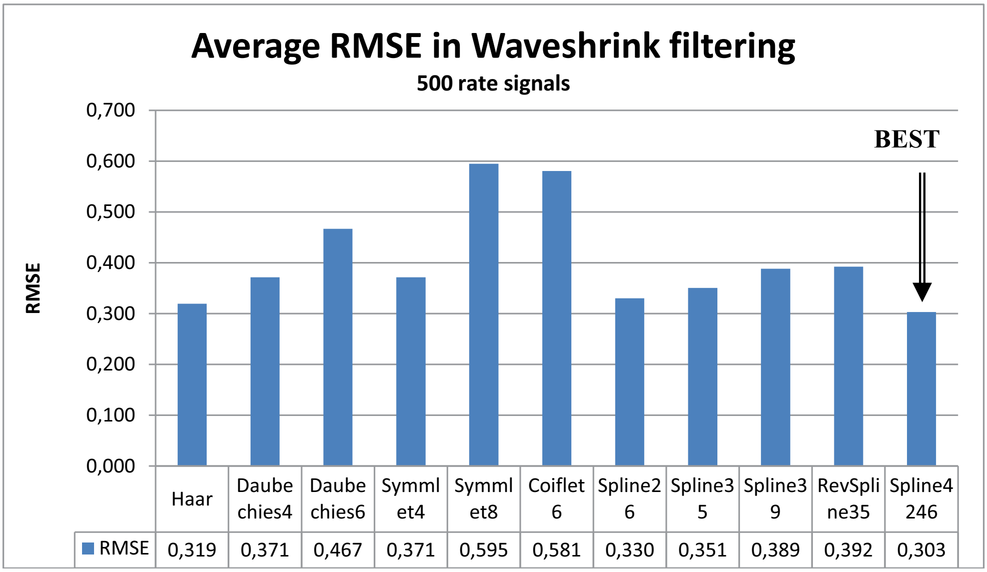
Different authors have designed numerous wavelets with the intension of approximating different kinds of signal well. We are however not aware of any particular wavelets specifically designed for the kind of signals we consider. Nevertheless, even if such wavelets did exist, there would be no guarantee ensuring that these wavelets would be the best ones to apply. Hence one usually will have to determine the best wavelet for the signal considered through a series of filtering exercises. No attempts will be made to design new wavelets. We will simply try to determine which of the existing wavelets, designed by other authors, that have the best properties for the kind of signals we consider.

In the previous section, the wavelet Spline39 and Spline4246 gave the best results along with the SCAD shrinkage rule for pressure and rate signals respectively. Thus it is actually already established that the wavelets Spline39 and Spline4246 should be used with pressure and rate signals respectively along with the SCAD shrinkage rule. The summary of the analysis for the different wavelets are shown in *Figure 5.31* and *Figure 5.32*. These figures show the performance of different wavelets when utilizing the SCAD shrinkage rule. The wavelet and scaling coefficients for Spline39 were previously given in *Table 4.2* and for Spline4246 in *Table 4.3*. Note that only 11 out of the 27 wavelets considered are shown in *Figure 5.31* and *Figure 5.32*. The wavelets shown are the 11 that performed best for either pressure or rate signals. In addition, the best wavelet from each wavelet family (e.g. Daubechies, Symmlet, Coiflet, etc.) is included.



	Haar	Daubechies4	Daubechies6	Symmlet4	Symmlet8	Coiflet6	Spline26	Spline35	Spline39	RevSpline35	Spline4246
■ RMSE	0,840	0,559	0,508	0,559	0,556	0,705	0,633	0,391	0,385	0,479	0,628

Figure 5.31: *Spline39 wavelet performs best for pressure signals with SCAD shrinkage rule. Average results from 500 synthetic pressure signals.*



	Haar	Daubechies4	Daubechies6	Symmlet4	Symmlet8	Coiflet6	Spline26	Spline35	Spline39	RevSpline35	Spline4246
■ RMSE	0,319	0,371	0,467	0,371	0,595	0,581	0,330	0,351	0,389	0,392	0,303

Figure 5.32: *Spline4246 wavelet performs best for rate signals with SCAD shrinkage rule. Average results from 500 synthetic rate signals.*

Until now we have determined the thresholds required in Waveshrink through an optimization routine. We could do this since we knew the true underlying signal. However, in general a threshold estimator needs to be selected and the best method to estimate the thresholds for the signals considered will be determined next.

5.3.7 Threshold Estimator

We determined the best wavelet and shrinkage rule by the use of an optimization routine selecting the optimal thresholds at each resolution level. We assumed that we later would be able to estimate these optimal thresholds for the wavelet and shrinkage rule determined as best. Hence we decided to determine the best threshold estimator as the one that gives the thresholds with the smallest root-mean-squared error compared to optimal thresholds for the wavelet and shrinkage rule considered, where the optimal thresholds are estimated as described in chapter 5.3.3. Wavelet Spline39 and SCAD shrinkage rule will be used for pressure signals and wavelet Spline4246 and SCAD shrinkage rule will be used for rate signals.

During the initial filtering exercises, it was discovered that the estimated thresholds from the threshold estimators described in section 4.8.3, usually deviated from the estimated optimal thresholds. Thus we will propose a new threshold estimator based on the results from our optimization routine. Through a series of filtering exercises it was established that the optimal thresholds will depend upon the noise level in the signal and the resolution level considered. Based on these observations, we chose to estimate the thresholds for use in the Waveshrink from an empirical threshold RED (Regression Estimated Decline) on the form;

$$RED = \sigma(-aj + b) \tag{5.8}$$

σ is an estimate of the noise level, *j* is the resolution level considered. The parameters ***a*** and ***b*** were then determined through the optimization routine for the signals considered. The results from 500 pressure and rate signals gave the following parameters for *eq. (5.8)*;

$$RED = \sigma(-0.51j + 3.25) \qquad j = 1,2,3,4,5,6 \tag{5.9}$$

The estimated thresholds will be negative for resolution levels above the 6 finest resolution levels available in the wavelet transform. Therefore we limit the above threshold estimator to the six finest resolution levels. The RED thresholds (red curve) compared to the optimal thresholds (black curve) and several other estimated thresholds are for three of the pressure signals considered shown in *Figure 5.33*, *Figure 5.34* and *Figure 5.35*. Note the parameters *a* and *b* in the RED threshold estimator is the average parameters determined from all 500 pressure and rate signals considered. Many of the threshold estimators require an estimate of the noise level. We have previously determined how to estimate the noise by RMAD given in *eq. (5.7)* and classified this noise estimator as the best. Thus the RMAD noise estimator is applied with any threshold estimator which requires an estimate of the noise level in *Figure 5.33*, *Figure 5.34* and *Figure 5.35*.

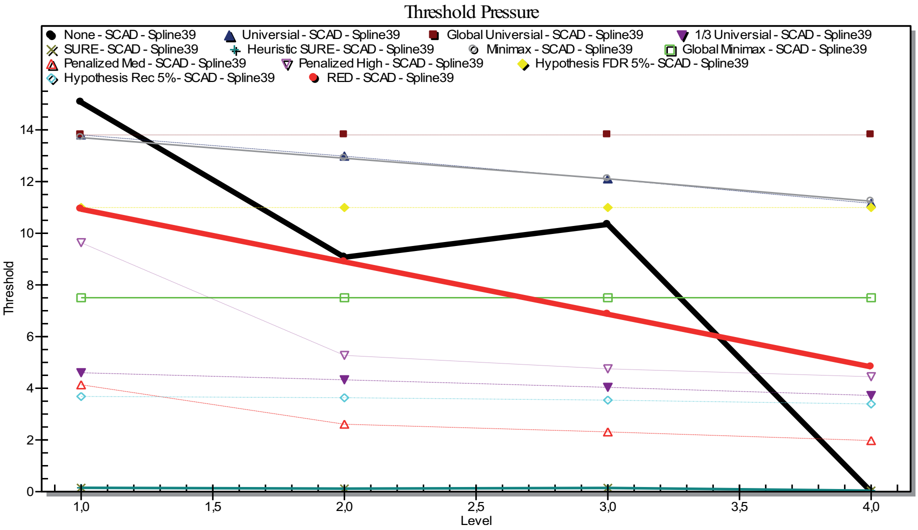


Figure 5.33: *Different estimated thresholds for a pressure signal with 4 decomposition levels. 4 standard deviation of white noise is added to the signal measurements. Optimal thresholds are shown as the black curve while thresholds from RED threshold estimator are shown in the red curve.*

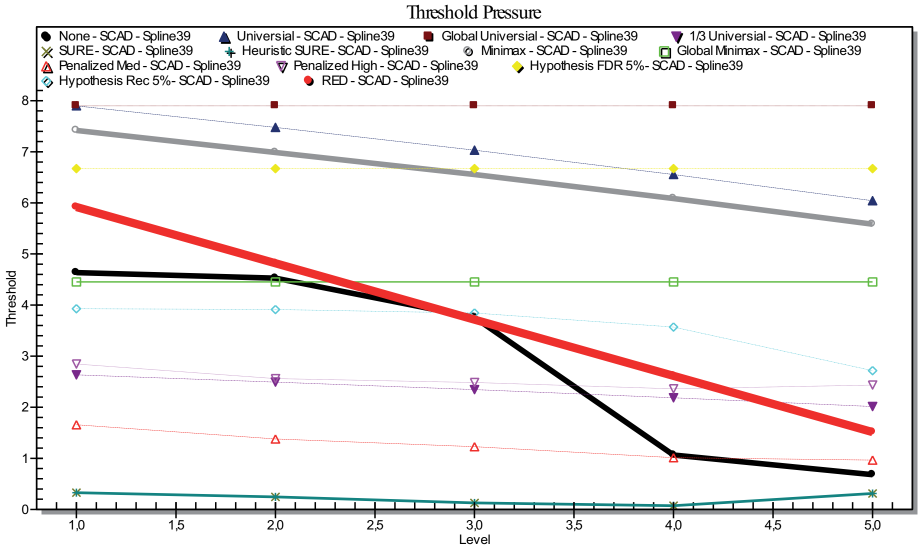


Figure 5.34: Different estimated thresholds for a pressure signal with 5 decomposition levels. 2 standard deviation of white noise is added to the signal measurements. Optimal thresholds are shown as the black curve while thresholds from RED threshold estimator are shown in the red curve.

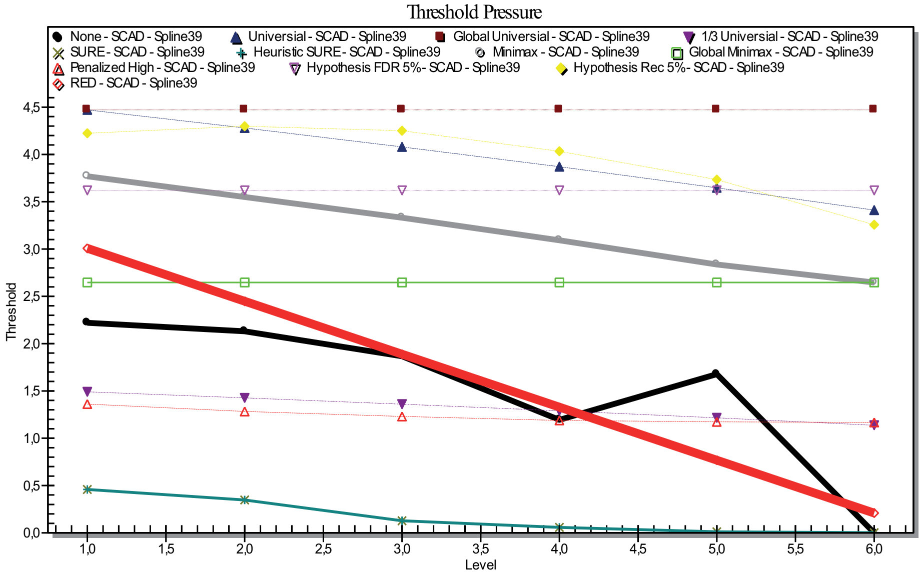


Figure 5.35: Different estimated thresholds for a pressure signal with 6 decomposition levels. 1 standard deviation of white noise is added to the signal measurements. Optimal thresholds are shown as the black curve while thresholds from RED threshold estimator are shown in the red curve.

The RED threshold estimator proposed above is a redefinition of the RED threshold estimators previously proposed in Olsen et al. [(1)] The proposed RED threshold estimator in

Olsen et al. [(1)] work was proposed on the form $RED = a \times \sigma \times (b \times \log(n))^{level}$ where that coefficients *a* and *b* were determined with the same approach as described above. We later discovered that equally good results could be obtained with a much simpler threshold estimator as proposed in eq. *(5.9)*. Thus we chose to redefine the RED threshold estimator as described above.

Based on our criteria for selection of the best threshold estimator, we will define the RED threshold estimator as the best. Obviously we finally need to confirm that the RED threshold estimator works well with the signals considered and better than other available threshold estimators. However, before we can confirm that our proposed threshold estimator works best, we need to determine which primary resolution level to apply.

5.3.8 Primary Resolution Level

The primary resolution level in wavelet filtering is used to describe the number of detail levels in Waveshrink where the wavelet coefficients are modified. We would like to determine how many resolution levels to include in Waveshrink in order to obtain the smallest error by *eq. (5.5)*. The common idea is to filter up to a fixed or estimated resolution level and ignore the coarsest resolution levels under the assumption that the detail coefficients at the finest resolution levels will be caused by noise, while the detail coefficients at the coarser resolution levels will originate from the signal itself.

Our investigation shows that in terms of minimizing the error in *eq. (5.5)*, all possible resolution level available in the wavelet transform should be used in the Waveshrink method. This assumes that we are able to choose the optimal threshold at each resolution level. Note that this threshold might be zero at some of the coarsest resolution levels as illustrated in *Figure 5.22*. Nevertheless, the gain by filtering the coarsest resolution levels is usually very small. In addition, if wrong thresholds are selected at the coarsest resolution levels, there might be a risk of over-smoothing the filtered signal. A more practical approach is therefore to stop filtering when there are small improvements in adding additional and coarser resolution levels in Waveshrink. The root-mean-squared error versus the number of decomposition levels is illustrated for three of the signals considered in *Figure 5.36*, *Figure 5.37* and *Figure 5.38*

$RMSE_\sigma$ vs. primary resolution level

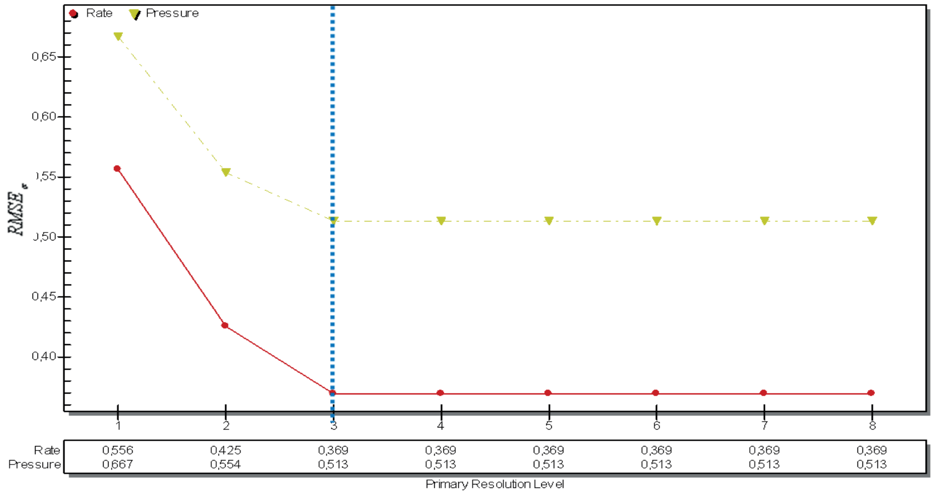


Figure 5.36: *Root-mean-squared error vs. primary resolution level for a pressure and a rate signal, both with 256 measurements. The blue stippled line indicates the minimum recommended primary resolution level.*

$RMSE_\sigma$ vs. primary resolution level

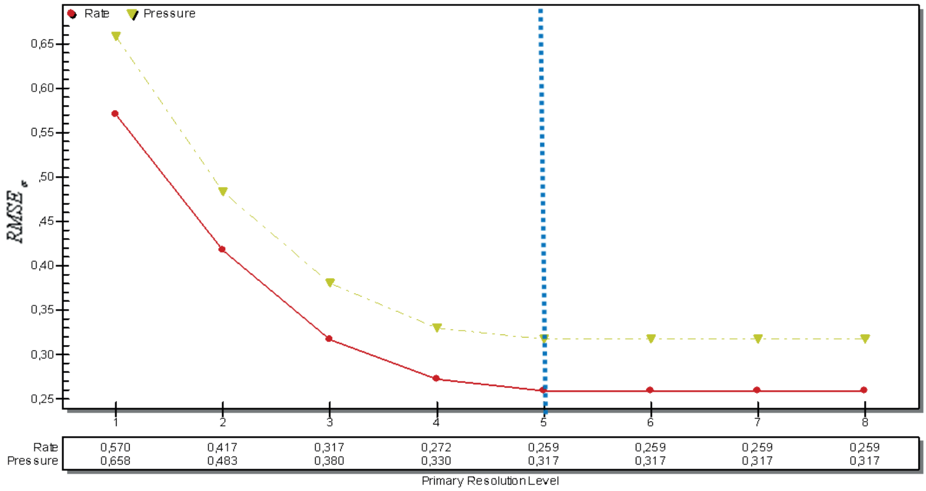


Figure 5.37: *Root-mean-squared error vs. primary resolution level for a pressure and a rate signal, both with 4096 measurements. The blue stippled line indicates the minimum recommended primary resolution level.*

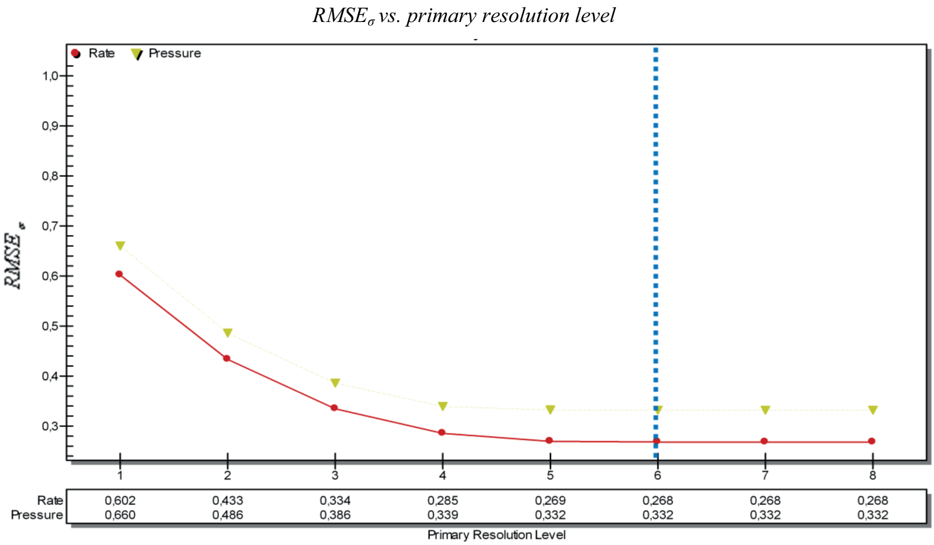


Figure 5.38: *Root-mean-squared error vs. primary resolution level for a pressure and a rate signal, both with 32768 measurements. The blue stippled line indicates the recommended primary resolution level.*

We choose to define the *minimum recommended* primary resolution level as the resolution level where the root-mean-squared error (*eq. (5.5))* is reduced by less than 0.001 when including an additional and coarser resolution level in Waveshrink. This minimum recommended primary resolution level is shown as a blue stippled line in *Figure 5.36*, *Figure 5.37* and *Figure 5.38.*

From 500 pressure and rate signals considered, the following formula gave excellent agreement with this definition of the minimum recommended primary resolution level;

$$j_0(N) = \log_8(N) - 1 \qquad (5.10)$$

and where *N* is the number of measurements. Consequently we do propose to estimate the minimum recommended primary resolution level in Waveshrink by *eq. (5.10).* This estimated minimum primary resolution level is slightly higher than the primary resolution level given by another common estimator found in the literature and given by *eq. (4.84).* This is illustrated in *Figure 5.39.* By utilizing the primary resolution level as given by *eq. (5.10)*, we will typically include some additional resolution levels in Waveshrink. Although the improvements in the root-mean-squared error is quite small by doing this, as can be seen from *Figure 5.36 - Figure 5.38*, we can get a slightly improved root-mean-squared error with little extra computational effort. For instance, in the case with 4096 pressure measurements, the root-mean-squared error would by improved from 0.330 to 0.317 if *eq. (5.10)* was used instead of *eq. (4.84).*

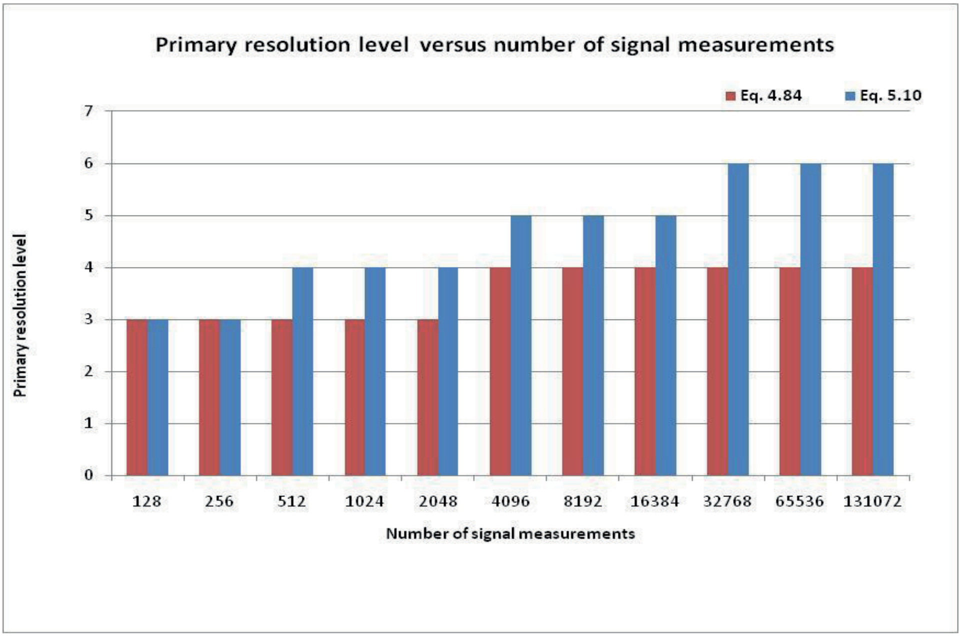


Figure 5.39: *Comparison of primary resolution level estimated from eq. (4.84) and eq. (5.10).*

Finally we need to confirm that the RED threshold estimator works well compared to other common threshold estimators. We could not do this in the previous section since we did not know how many resolution levels to include in Waveshrink. With the primary resolution level estimated from *eq. (5.10),* we can now compare the RED threshold estimator with other common threshold estimators described in chapter 4.8.3. This is shown in *Figure 5.40* and *Figure 5.41*.

The use of the new RED threshold estimator gave better results than any other threshold estimators. Among the other threshold estimators considered, the Penalized High, Universal and FDR threshold estimators gave good results. It is also interesting to notice that two commonly used threshold estimators, the SURE and Minimax, are among those that performed worst for the kind of signals we consider. We conclude that the RED threshold estimator on average provides us with the best results. As a result, RED threshold estimator is suggested for filtering of both rate and pressure signals in Waveshrink.

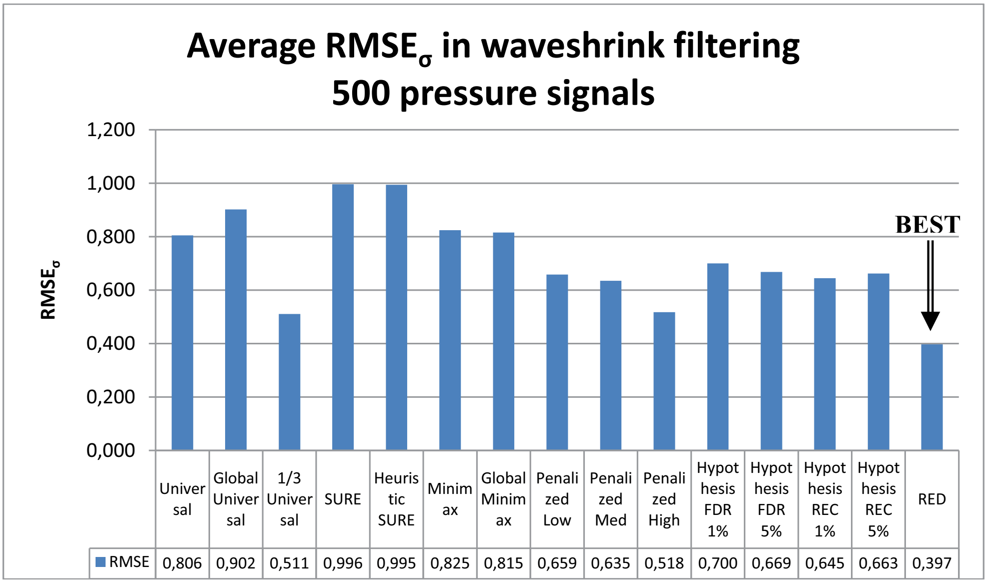


Figure 5.40: *New RED threshold estimator compared with several other threshold estimators at 500 random generated pressure signals. Wavelet Spline39, SCAD shrinkage rule, RMAD noise estimator eq. (5.7) and primary resolution level from eq. (5.10).*

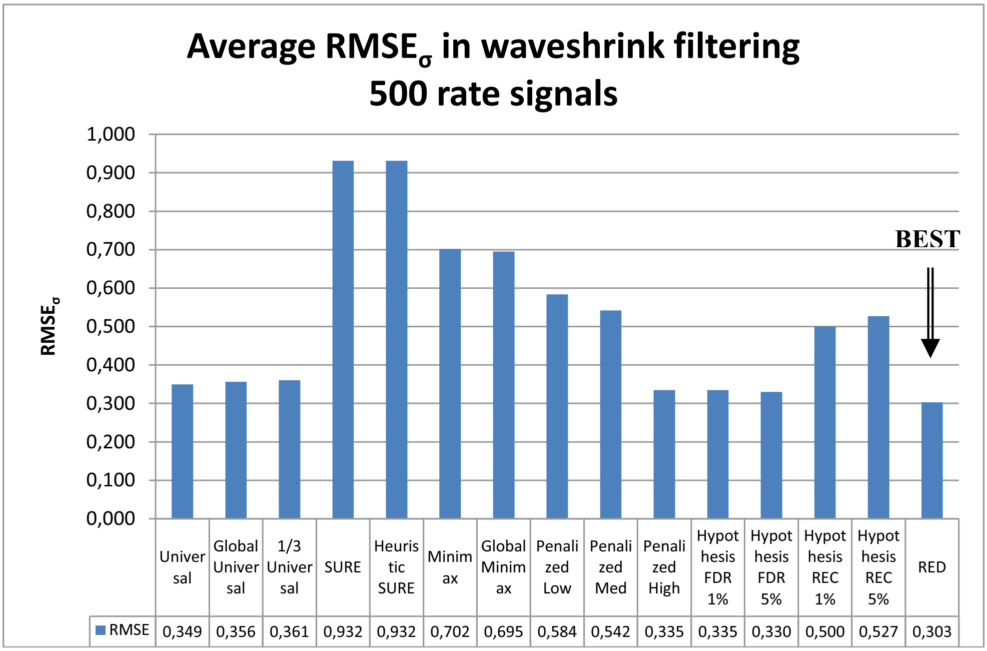


Figure 5.41: *New RED threshold estimator compared with several other threshold estimators at 500 random generated rate signals. Wavelet Spline4246, SCAD shrinkage rule, RMAD noise estimator eq. (5.7) and primary resolution level from eq. (5.10).*

Finally we will summarize the recommended Waveshrink setting for pressure and rate signals. For pressure signals we recommend;

- Spline39 wavelet given by table. *Table 4.2*
- SCAD shrinkage rule given by *eq. (4.59)*
- RED threshold estimator given by *eq. (5.9)*
- RMAD noise estimator given by *eq. (5.7)*
- Primary resolution level given by *eq. (5.10)*

For rate signals we recommend;

- Spline4246 wavelet given by *Table 4.3*
- SCAD shrinkage rule given by *eq. (4.59)*
- RED threshold estimator given by *eq. (5.9)*
- RMAD noise estimator given by *eq. (5.7)*
- Primary resolution level given by *eq. (5.10)*

The only difference in the recommendation for pressure and rate signals is the wavelet.

Note the primary resolution given by *eq. (5.10)* is the *minimum recommended* primary resolution level, i.e. small or none improvements is achieved in terms of root-mean-squared error by including coarser resolution levels in Waveshrink. The other recommended selections in Waveshrink are the selection corresponding to the minimum root-mean-squared error for the signals considered.

Next we propose a simple approach for data reduction if noise and outliers are already removed by the above suggested methods.

5.4 Data reduction

In data reduction, the aim is to reduce the number of signal measurements by removing the measurements with little information content and keeping the measurements with significant information content. This is often accomplished by removal of signal measurements from time intervals with small (and assumed insignificant) changes in the signal, and to keep the signal measurements from the time intervals with large (and assumed significant) signal changes. Thus we would like to reduce the number of signal measurements but without loosing significant signal information. A straight forward way of accomplishing this, e.g. as to only keep measurements that have changed with a given amount since the previous measurement, will be heavily influenced by noise and/or outlier. If measurements are not de-noised prior to this kind of data reduction, typically the largest noise peaks will be preserved, actually meaning that the noise in the signal is exaggerated. On the other hand, if signals simply were measured at a lower resolution, valuable information might be lost.

Athichanagorn et al. [(5)] performed data reduction by pressure thresholding, i.e. unless a measurement has changed with more than a user specified threshold from the previous measurement, it is removed. Athichanagorn et al. recommended data reduction after outlier removal and de-noising. This method requires a user specified threshold as the threshold is signal specific.

We also choose to perform data reduction with thresholding but without the requirement for a user defined and signal specific threshold. As Athichanagorn et al., we will remove outliers and noise from the signal measurements prior to data reduction. Outlier removal and de-noising will be accomplished as described in the two previous sections. The number of measurements to remove in data reduction will be determined by an iterative scheme.

We choose to define the loss of signal energy as the *square difference between the de-noised signal measurements and the original noisy signal measurements*. We would like to remove an amount of signal energy equal to the estimated squared noise level. This will be accomplished by considering the loss of signal energy from de-noising subtracted the estimated squared noise level given as;

$$\left(\frac{1}{N} \sum_{i=0}^{N-1} (f_i - s_i)^2 - \sigma^2 \right), \tag{5.11}$$

f_i is the de-noised signal measurements, s_i is the original noisy measurements while N is the number of signal measurements. The noise level σ, is estimated as for outlier removal and de-noising by the RMAD noise estimator given by *eq. (5.7)* and is calculated from the detail coefficients at the first decomposition level in the wavelet transform applying the original noisy signal measurements s_i. We seek to reduce the number of measurements in a sequence of step; *[N/2, N/4, N/8, N/16,.....,1]*. In the first step, half the number of signal measurements will be attempted removed while only one single measurement is attempted removed in the last step. To accomplish data reduction, the following iterative scheme is considered;

1. *Try to remove the $N/2^k$ signal measurements with the smallest change since the previous measurement from the de-noised signal measurements f_i and where k =[1,2,3,4,.....,log$_2$(N)] is an iteration counter.*

2. *Interpolate the de-noised and reduced number of signal measurements f_i back to the original signal measurement spacing s_i. Constant interpolation is applied for rate signals while linear interpolation is applied for pressure signals.*

3. *Calculate eq. (5.11) utilizing the interpolated and de-noised signal measurements f_i. If eq. (5.11) has a positive value, permanently remove $N/2^k$ measurements from the de-noised signal measurements f_i during the current iteration step. If eq. (5.11) has a negative value, do not remove any de-noised signal measurements f_i during the current iteration step.*

4. *Step 1-3 is repeated in a sequence of step. The counter k is increased by one for each iteration step as we attempt to remove [N/2, N/4, N/8, N/16, ..., 1] measurements with the smallest change since the previous measurement from the de-noised signal measurements f_i.*

If all noise has been removed in the de-noising, the ideal energy loss in data reduction should be close to the estimated noise level, of course under the assumption that the signal is not over-smoothed. If important signal information has been removed in the de-noising as well, the energy loss will be larger than the estimated noise level.

Let us illustrate the proposed data reduction method by an example. Rate measurements are considered in this example. We start by eliminating half the number of de-noised signal measurements as illustrated in *Figure 5.42* below. Green measurements are the ones kept while grey measurements are the ones removed. The signal measurements after the largest changes are always kept. After the initial removal of half the number of de-noised signal measurements in *Figure 5.42b*, the remaining signal measurements are interpolated back to the original spacing by constant interpolation. In *Figure 5.42b*, the expression in *eq. (5.11)* will be positive. Thus half the number of measurements is permanently removed from the de-noised signal measurements. Next in *Figure 5.42c,* half the number of remaining de-noised signal measurements from *Figure 5.42b* has been removed and once again the expression in *eq. (5.11)* is positive. Thus half the number of remaining measurements is permanently removed from the de-noised signal measurements.

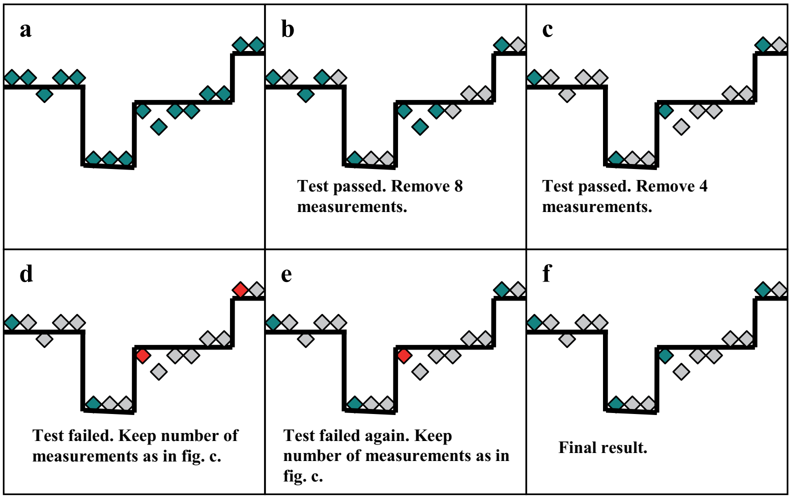

a
b
Test passed. Remove 8 measurements.
c
Test passed. Remove 4 measurements.
d
Test failed. Keep number of measurements as in fig. c.
e
Test failed again. Keep number of measurements as in fig. c.
f
Final result.

Figure 5.42: (a) *Original signal,* ***(b)-(c)*** *removal of measurements,* ***(d)-(e)*** *no further removal of measurement.* ***(f)*** *Final result.*

In *Figure 5.42d*, it is obvious that too many measurements are removed if half the number of remaining signal measurements once again is removed. Thus the next step will result in a negative value in *eq. (5.11)* and is consequently skipped. This means that the square difference between the noisy signal measurements and the interpolated de-noised signal measurements will be high, much higher than the estimated noise level. Then we will remove significant information from the de-noised signal measurements and should continue to the next step without removing any signal measurements. In *Figure 5.42d* we try to remove two de-noised signal measurements and the expression in *eq. (5.11)* will once again be negative. Next we will try to remove a single de-noised signal measurement as shown in *Figure 5.42e.* Once again the expression in *eq. (5.11)* will be negative and thus the final result in *Figure 5.42f* will be the same as in *Figure 5.42c.*

In *Figure 5.43*, we have illustrated the data reduction ratio where the expression in eq. *(5.11)* is zero. The number of remaining measurements in percent is given along the x-axis. The estimated noise level for this signal is four and the square value of the noise level is shown by the horizontal brown line. The vertical blue line indicates the data reduction ratio where the expression in eq. *(5.11)* is zero. The energy loss in the signal measurements is shown as the black curve.

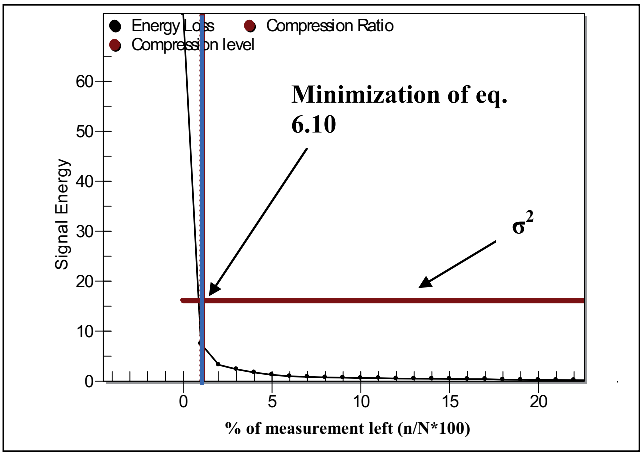


Figure 5.43: *Illustrates the data reduction ratio where the expression in eq. (5.11) is minimized for a signal added four standard deviation of white noise.*

In the author's opinion, this method works well. Since de-noised signals will be available in the cases we will consider, this makes the proposed data reduction method in particular simple to use. The proposed data reduction method requires no user input and holds the promise for automatic data reduction of pressure and rate signals.

It should finally be noted that wavelet analysis is referred to as well suited for ***signal compression***. The idea behind signal compression in wavelet analysis is to keep the "large" detail coefficients and discard the small detail coefficients. This since most of the detail coefficients will be zero or close to zero and most of the "important" signal information will be located within a small fraction of the "large" detail coefficients (and the remaining approximate coefficients). Thus the discrete wavelet transform is well suited if the objective is *data compression and efficient storage* of the signal measurements. However, we need to perform the inverse wavelet transform if we like to utilize the wavelet compressed signal measurements in various kinds of analyses, e.g. in well-test analysis. In the inverse wavelet transform, we will end up with the same number of signal measurements as before the wavelet transform (even if we only store the "large" detail coefficients and the remaining approximate coefficients). In different kinds of analysis, we would like to reduce the number of signal measurements to improve numerical performance, e.g. in well-test analysis. ***Consequently, signal compression by wavelet analysis is extremely useful if the objective is data compression to reduce the data storage requirement for the measured signals. However, if the objective is data reduction (reduce the number of signal measurements), as is the objective in this work, wavelet analysis is not well suited for this and a thresholding approach as described above is preferred.***

The measured signals can be efficiently stored by the use of wavelet analysis reducing the requirement for data storage capacity. By the use the discrete wavelet transform for the signals considered in this work, compression ratios of 8-25% is experienced for pressure signals while compression ratios of 3-15% is experienced for rate signals. The compression ratio is the number of remaining approximate and detail coefficients divided by the number of uncompressed signal measurements. Wavelet signal compression is performed with the same selections for pressure and rate signals as in signal de-noising in chapter 5.3. Hence the selection of which detail coefficients to consider as "large" and store, and which detail coefficients to consider as "small" and not worth storing, is determined in the same way as in signal de-noising. Also note that we cannot apply the stationary wavelet transform in wavelet signal compression. This since the number of detail coefficients at each resolution level is kept constant and redundancy is introduced in the stationary wavelet transform. The number of "important" detail coefficients in the stationary wavelet transform is often larger than the original number of signal measurements due to the redundancy introduced in the stationary wavelet transform. In addition, the number of remaining approximate coefficients is equal the number of original signal measurements. Consequently, the signal representation obtained in the stationary wavelet transform is not well suited for signal compression. Thus we need to compress the signal measurements by the use of the dyadic wavelet transform as described in chapter 4.7. However, as already mentioned, the wavelet signal compression is well suited for data compression and efficient data storage of the signal measurements, while thresholding is preferred if the objective is to reduce the number of signal measurements.

Next we will demonstrate some valuable application of signal filtering in the E&P industry.

5.5 Applications

Case 1. De-noising and Data reduction

The following example consists of 5335 pressure measurements generated using Eclipse reservoir simulator[79] and a user defined rate history. One standard deviation of white noise was added to both rate and pressure measurements. The pressure measurements are however generated based upon exact and noise-free rate measurements. Random outliers were added to both pressure and rate measurements as described in chapter 5.1. The pressure and rate signals including noise and outliers are shown in *Figure 5.44.* A zoomed in section of the signals is shown in *Figure 5.45.*

The first step is outlier removal using the previously described median filter. This is done completely automatically with the median filter described in chapter 5.2 and with noise estimation from RMAD noise estimator. As a result, all outliers were successfully removed from the pressure and rate measurements as illustrated in a limited time interval in *Figure 5.46*. After outlier removal, the rate measurements are de-noised as described in chapter 5.3 and data reduction is applied as described in section 5.4. Out of the initial 5335 rate measurements, only 25 remain after data reduction. This is illustrated in *Figure 5.47.* Ideally our data reduction method should have removed four additional rate measurements (four of the constant rate intervals contain two remaining measurements). However, we chose to keep the data reduction method like this to minimize the risk of removing measurements with important information content. The pressure measurements are then de-noised as described in chapter 5.3. A zoomed in section of the filtered pressure signal is shown in *Figure 5.48.*

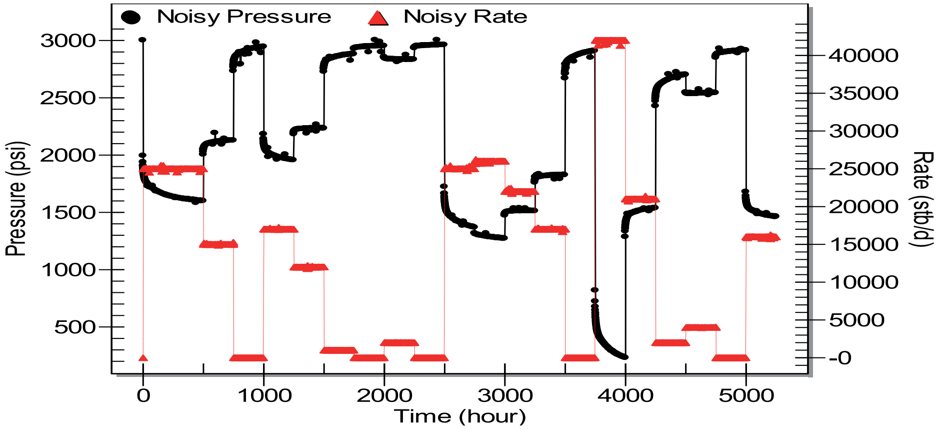


Figure 5.44: *Pressure and rate measurements added white noise and random outliers.*

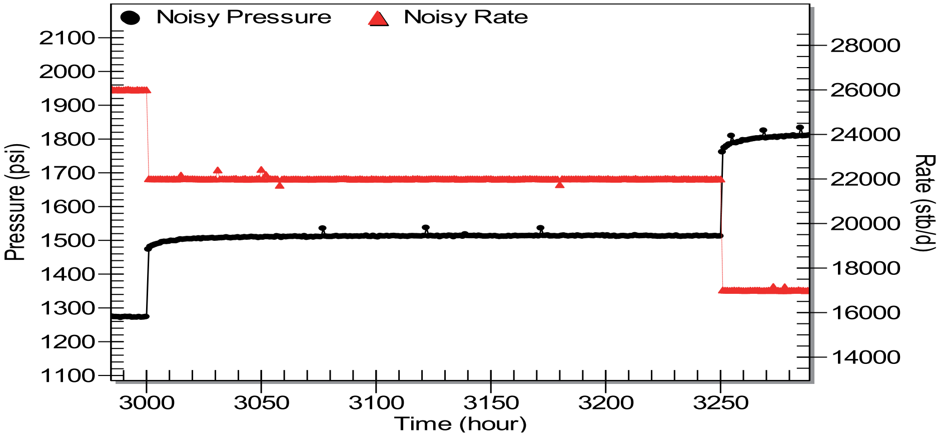


Figure 5.45: *Zoomed in section of pressure (black curve) and rate (red curve) measurements added white noise and random outliers.*

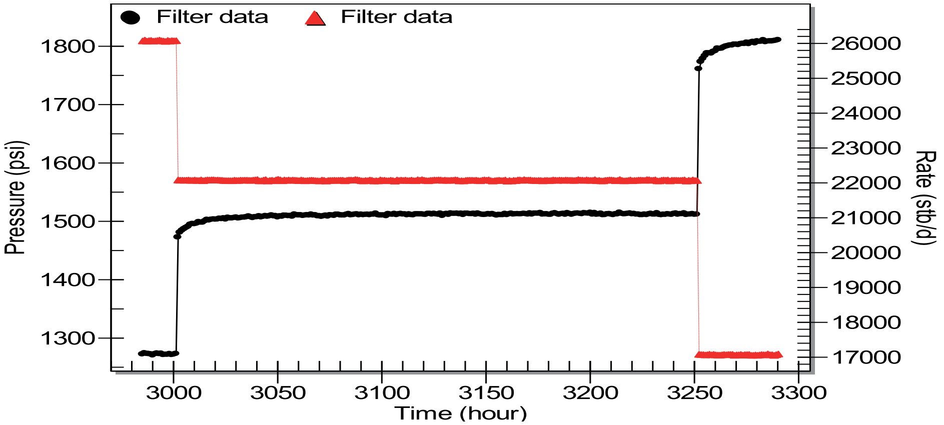


Figure 5.46: *Outliers removed from pressure (black curve) and rate (red curve) measurements with proposed outlier filter.*

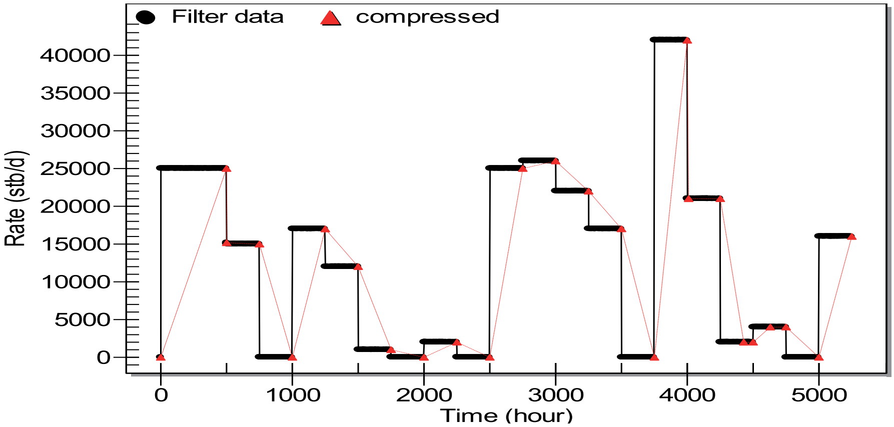


Figure 5.47: *De-noised and reduced number of rate measurements (red curve) along with original rate measurements (black curve).*

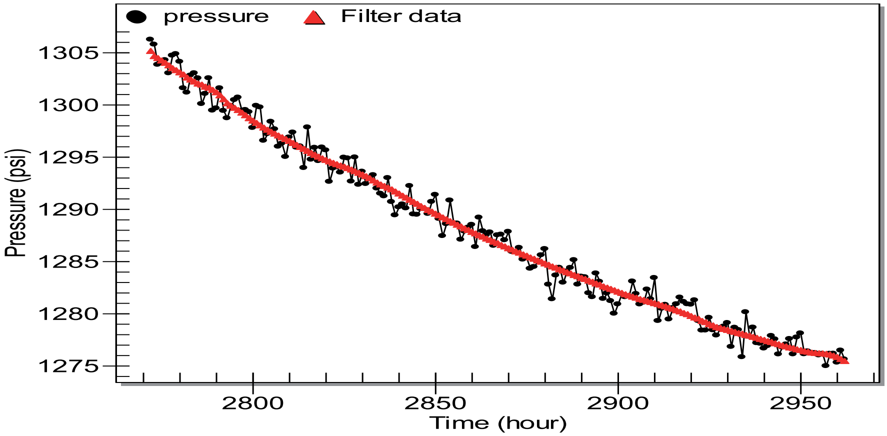


Figure 5.48: *Zoom in on de-noised pressure measurements (red curve) along with original noisy measurements (black curve).*

In automatic well-test analysis, as will be described in chapter 7, a different approach than described above is chosen for rate signals. In data preparation to well-test analysis, only pressure signals are filtered. Next pressure transients are detected and average rates between the detected pressure transients is calculated. Thus outlier removal, de-noising and data reduction is not required for rate signals in our proposed automatic well-test analysis. However, as this is developed as a general signal filtering module, other applications than well-test analysis might require rate signal filtering. Thus the purpose of this case was to illustrate that the proposed rate signal filtering works well if desired in other applications than well-test analysis.

Case 2. Derivative Analysis

One of the areas where wavelet de-noising can be very useful, is for generation of derivative plots(13) in well-testing. Derivative plots are used in well-testing for model identification, i.e.

identify if the reservoir is homogenous, fractured or layered just to mention a few. These plots utilize numerical pressure derivatives and are thus very sensitive to noise. Derivative plots were previously described in chapter 3.8.2. By de-noising the pressure measurements with a wavelet filter before calculating the numerical derivatives, better derivative plots might be obtained. Notice that the derivative plots already contains the *differentiation interval*(42; 46) used in order to smooth noisy derivatives. Instead of choosing measurements next to each other in calculation of numerical derivatives, measurements apart from each other are used. The distance between the measurements are measured along the x-axis in the derivative plots and is typically between 0.2 and 0.4. In general, the use of differentiation intervals might distort the shape of the derivative plot if chosen too high, or it might not smooth the derivative plot at all if chosen too low. In the case where wavelet filtering is applied with the measured pressure signal, neighbouring measurements are used in the calculation of numerical derivatives in the derivative plot.

Figure 5.49 shows a derivative plot for a dual porosity reservoir with square boundaries (42; 46) where the use of the differentiation interval fails to suppress the noise even when chosen as high as 0.7, i.e. measurements with a distance 0.7 along the x-axis are used in the calculation of the numerical derivatives. If chosen higher than this, it might completely distort the shape of the derivative plot. In some cases, the combination of noise and utilization of the differentiation interval can alter the derivative shape in a way that can make model selection hard or even wrong. The wavelet filtering suppresses the noise far better then the differentiation interval. The wavelet filter utilized is the one previously determined as best for pressure signals in chapter 5.3. The pressure measurements were de-noised before used in the derivative plot and numerical derivatives were calculated using measurements next to each other. This is shown in *Figure 5.50.*

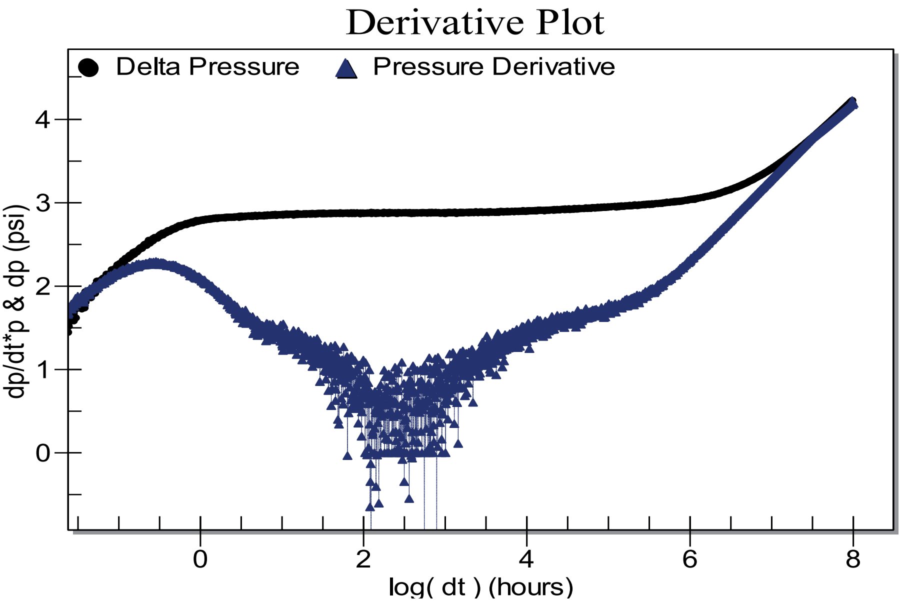


Figure 5.49: *Derivative plot where pressure derivatives are smoothed using a differentiation interval of 0.7.*

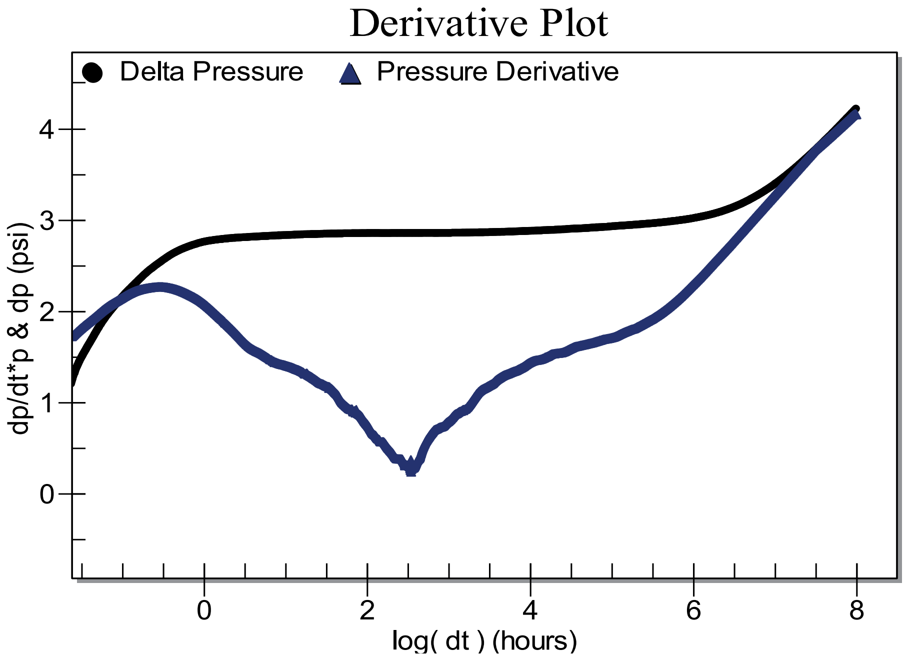


Figure 5.50: *Derivative plot where pressure measurements are de-noised in Waveshrink.*

Case 3. Computational Time, Regression Analysis Well Testing

In non-linear regression analysis one seeks to minimize the square difference between the measured signal and some mathematical model assumed to adequately describe the process model. In well-testing the process model describes the flow of fluid through a porous media towards a single well (42; 46). The output from the non-linear regression analysis is the process state corresponding to the smallest square difference between the measurements and the process model, thus providing an estimate of the state variables. Non-linear regression analysis was described in chapter 3.

We would like to remove outliers before non-linear regression analysis. The reason for this is that single outliers with a large magnitude can give a significant contribution to the square difference and thus contribute to a poor estimate of the process state. Normal distributed noise with a given standard deviation and zero mean will not distort the results from the regression analysis. When de-noising the signal measurements, there is always a certain possibility of removing significant information from the signal measurements, thus shifting the results that otherwise would have been obtained in the well-test non-linear regression analysis. Thus de-noising a signal prior to non-linear regression analysis is not preferred. This also applies to data reduction. To make sure that no significant information is removed from the signal measurements, all noisy signal measurements in general should be included in well-test non-linear regression analysis, of course with the exception of potential outliers. Nevertheless, sometimes in terms of obtaining the results from well-tests regression analysis within a reasonable amount of time, we are required to reduce the number of measurements, and in particular the number of rate measurements (42; 46). As a result, we would like to de-noise and reduce the number of rate measurements in order to speed up the non-linear regression analysis, although we would still like to obtain the same results as if all the signal measurements had been applied in a computational costly analysis.

A rate history with 4000 measurements containing five shut-in periods is generated as described in chapter 5.1. Based upon a selected mathematical reservoir model, pressure measurements are generated based upon the noise free rate measurements. The model used to generate the pressure response is a vertical well in a homogenous reservoir with infinite boundaries as described in chapter 3.3 and given by *eq. (3.22)*. Skin factor and wellbore storage are included by *eq. (3.28)* and variable rates by *eq. (3.29)*. The rate and pressure measurements are shown in *Figure 5.51*. Four standard deviation of white noise is added to both rate and pressure measurements. The shut-in periods are assumed free from noise. No outliers are added to the measurements. Note that due to the large variations in pressure and rate, the scale on the y-axis in *Figure 5.51* is large. Hence a zoomed in section of the rate and pressure measurements is shown in *Figure 5.52* and *Figure 5.53*. The rate measurements are de-noised and the number of measurements reduced, while the pressure measurements are only de-noised. In well-testing, it is usually preferred to keep most available pressure measurements during shut-in periods to get good results from non-linear regression analysis and consequently data reduction is not considered in this case for the pressure measurements. The pressure measurements during shut-in periods should not be reduced unless high sampling rates are used (e.g. every second or every ten second) or if the shut-in periods are very long (e.g. duration of weeks or months). This is not the case for the pressure measurements we consider in this case. However, even if pressure de-noising is not recommended for pressure measurements utilized in non-linear regression, it is still interesting to see if the use of de-noised pressure measurements will influence the results from non-linear regression analysis. Recall the pressure de-noising is required before data reduction if a large number of pressure measurements are considered, or to facilitate model identification by the use of derivative plots. The methods and selections applied in filtering of pressure and rate signals were given in chapter 5.2-5.4. The reduced number of rate measurements is shown in *Figure 5.54* while the de-noised pressure measurements are shown in *Figure 5.55*.

After de-noising and data reduction of rate measurements and de-noising of pressure measurements, an algorithm that runs non-linear regression analysis on each shut-in period estimating the unknown parameters (in this case skin factor, wellbore storage and permeability) is applied. This is referred to as build-up testing[21] and was described in chapter 3.7. The non-linear regression analysis is initially applied with the noisy pressure and rate measurements and then to the de-noised and reduced number of rate measurements along with the denoised pressure measurements. Then the average estimated parameter values for the five shut-in periods are calculated from the non-linear regression analysis with the noisy pressure and rate measurement, and with the de-noised and reduced number of rate measurements along with the noisy pressure measurements. *Figure 5.56* shows the difference in average estimated permeability, skin and wellbore storage in the two cases. Recall that the goal is to obtain the same results from the de-noised and reduced number of rate measurements along with the de-noised pressure measurements as from the original and noisy pressure and rate measurements, only much faster. The difference is less than 2% for skin factor and around 0.5% for permeability and wellbore storage. The time is reduced from 343 seconds to only 4 seconds, and the computational speed is increased with a factor of 86. This is illustrated in *Figure 5.57*. This shows that it is possible to obtain good results in well-test regression analysis using only a small fraction of the original rate measurements while increasing the computational speed significantly. In addition, the results indicate that even if de-noised pressure measurements are utilized, the results from non-linear regression analysis is not significantly affected.

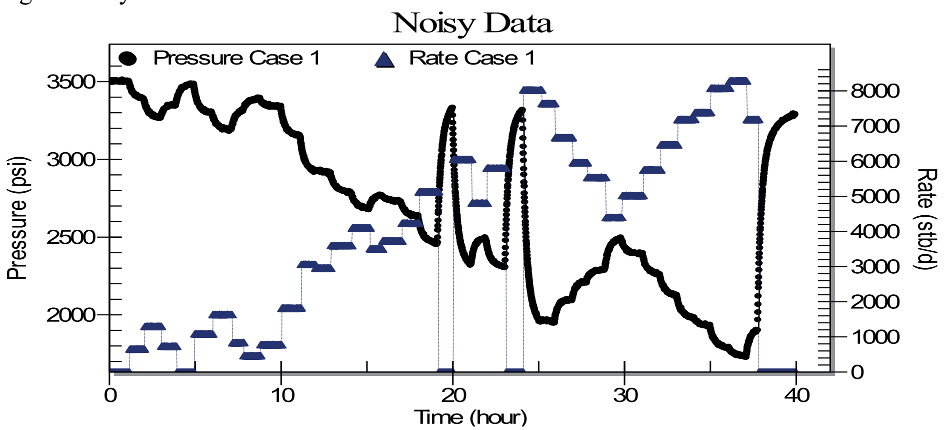


***Figure 5.51:** Pressure and rate signal used in case 3. 4000 measurements with a time interval of 0.01 hours are considered.*

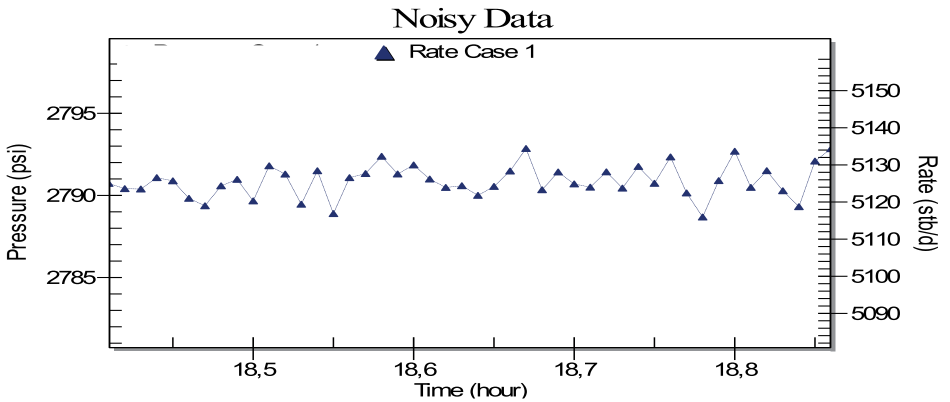


***Figure 5.52:** Zoom in on rate signal to illustrate the noisy rate measurements.*

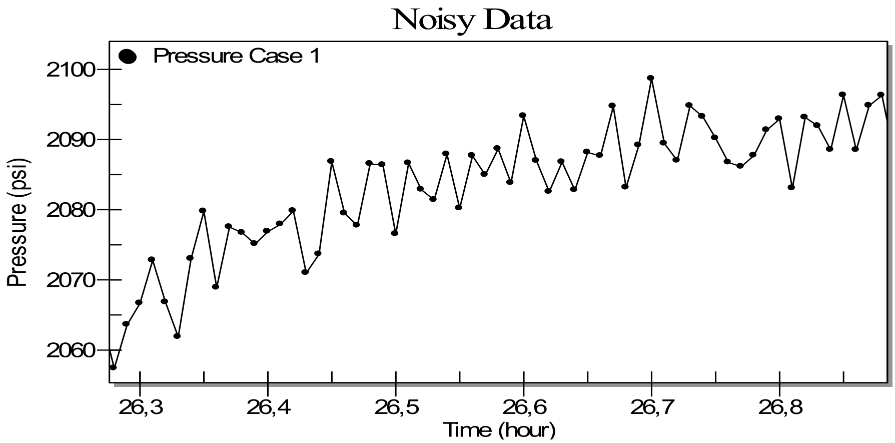


***Figure 5.53:** Zoom in on pressure signal to illustrate the noisy pressure measurements.*

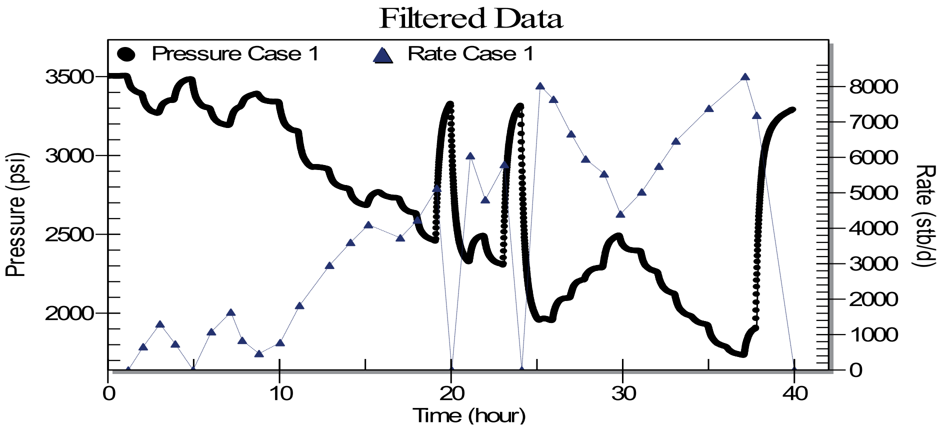


Figure 5.54: *Data reduction of rate measurements.*

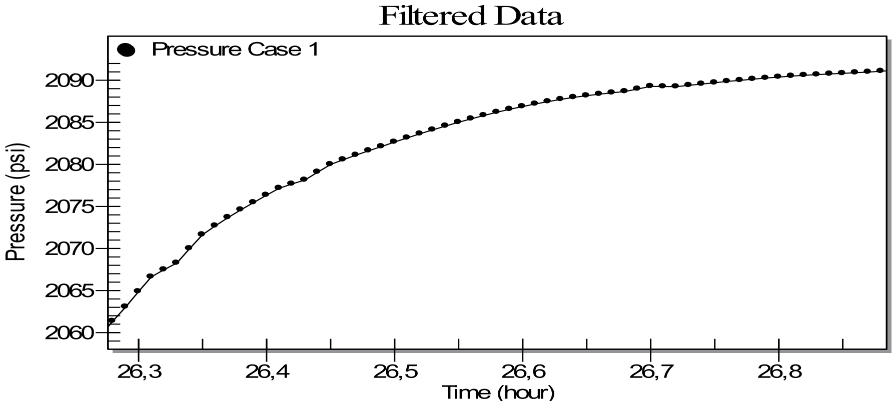


Figure 5.55: *Zoom in on de-noised pressure signal to illustrate the pressure de-noising. Same zoom-in as in Figure 5.53.*

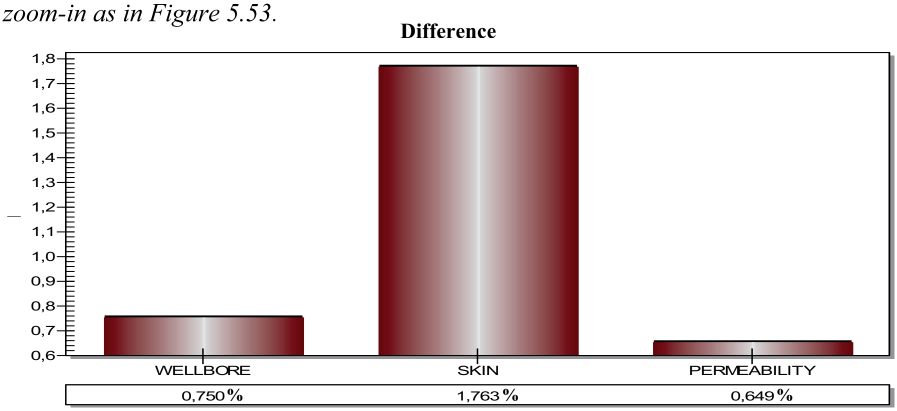


Figure 5.56: *Average difference in estimated wellbore storage, skin factor and permeability from regression analysis when applying noisy and original number of pressure and rate measurements versus de-noised and reduced number of rate measurements along with de-noised pressure measurements. The difference is shown in percent.*

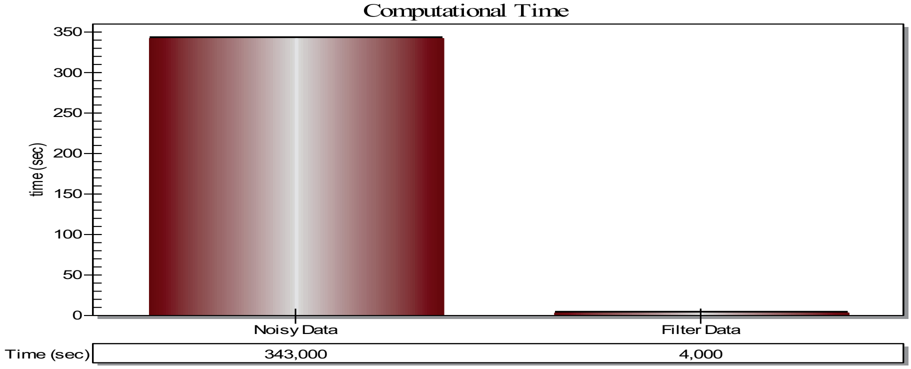


Figure 5.57: *Computational time running non-linear regression analysis on noisy and original number of pressure and rate measurements versus de-noised and reduced number of rate measurements along with de-noised pressure measurements.*

5.6 Summary

The proposed method for outlier removal in pressure and rate measurements will start by estimating the noise level using the RMAD noise estimator in *eq. (5.7)*. This requires no user input. Then the window size for the median filter is chosen, i.e. we specify the largest number of successive outliers to remove. We choose to remove up to nine successive outliers for pressure signals. We are aware that some of the smallest pressure transients might be removed as well, but it was shown that if this is the case, it is unlikely that important information is removed from the pressure measurements in the scope of well-test analysis. For rate signals, as these frequently are stored as averaged or aggregated values, only single outliers is removed. The removal of outliers is then accomplished without any required user input.

After outlier removal, the signals will be de-noised with Waveshrink. For pressure signals, this is accomplished with noise estimator RMAD *eq. (5.7)*, primary resolution level from *eq. (5.10),* wavelet Spline39, SCAD shrinkage rule *eq. (4.59)* and RED threshold estimator *eq. (5.9)*. For rate signals, de-noising is accomplished with noise estimator RMAD *eq. (5.7),* wavelet spline4246, primary resolution level from *eq. (5.10)*, SCAD shrinkage rule *eq. (4.59)* and RED threshold estimator *eq. (5.9).* Since all selections required in Waveshrink are predefined or calculated based upon signal properties, de-noising can be performed automatically under the assumption that the selections used in Waveshrink can be considered universal best for the type of signals considered. Through the large number of filtering exercises completed, we have shown that the recommended selections in Waveshrink on average will work best for the signals considered.

Finally, the number of signal measurements can be reduced by the proposed data reduction method after outlier removal and de-noising. Again data reduction requires no user input and can be done automatically. All the methods and selections required for signal filtering is consequently predefined or estimated based upon signal properties. Excellent performance was demonstrated for outlier removal, de-noising and data reduction for the kind of signals considered.

6 Real-time Signal Filtering and Process Monitoring

The measurements of signals will typically be performed continuously in real-time and it will require some special attention to utilize automatic signal filtering in such an environment. In general, a better estimate of the underlying signal can be obtained in wavelet filtering when the number of signal measurement becomes large[(55)]. Therefore, all available signal measurements should be included in wavelet filtering to obtain a better signal approximation. Of course, this will in most cases be impossible due high signal sampling rates and limited computer processing facilities. Consequently we are required to utilize the wavelet transform and subsequently the wavelet filtering with a limited number of the latest signal measurements. A window approach will be used in this case, i.e. the wavelet filtering will be performed on a given number of the most recent signal measurements. How large this window should be will be discussed later. However, in general it can be stated that if chosen too small, the result from wavelet filtering might be relatively inaccurate while a too large window will result in poor computational performance. Both outlier removal and data reduction require an estimate of the noise level acquired from wavelet filtering. Therefore a similar window approach is required for outlier removal and data reduction.

The case where signal filtering is performed on continuously updated signals is referred to as *real-time signal filtering*. ***The objective with real-time signal filtering is to filter a large number of signal measurements computationally efficient, continuously and without user interactions.*** In the previous chapter, we showed that it would be possible to determine methods and selections to facilitate automatic signal filtering of typical signals encountered in the E&P industry. As this is possible, it is also possible to have a continuous filtering of signal measurements without any user interaction. Continuously filtered signals will then be available to the engineers at any time.

A system is said to be in *a transient state* if a state variable has been changed and the system yet again has not reaches steady state. The detection of the start of the transient is referred to as *transient detection.* The detection of a transient is important not only to alert the engineers about changes in process variables, but also as a tool to simplify and automate different kinds of analyses, such as well-testing. (Automatic well-testing will be discussed in chapter 7). Transient detection will be the key component in what we refer to as *real-time process monitoring*.

The case where process monitoring is performed on continuously measured signals is referred to as real-time process monitoring. ***The objective with real-time process monitoring is to monitor a large number of signal measurements computationally efficient, continuously and without user interactions.*** As in real-time signal filtering, a window approach will be used. The window size will determine how fast and accurate changes in the signal measurements can be detected. A small window will result in fast detection of possible changes, but might in turn result in a higher frequency of "false" detections. On the other hand, a large window size will be better for an accurate detection of changes but might also result in a slower detection.

The real-time signal filtering and process monitoring module can be used to directly monitor the process state and provide the user with alarms in the cases where changes are detected in the signal measurements, or the real-time signal filtering and process monitoring module can be used to data preparation for use with analysis modules such as well-testing. An illustration of real-time signal filtering and process monitoring is shown in *Figure 6.1*. This includes the signal itself, real-time signal filtering, real-time signal trend estimation, real-time transient detection, real-time noise estimation and alarms. Various alarms can be triggered if deviations from the normal trends are detected.

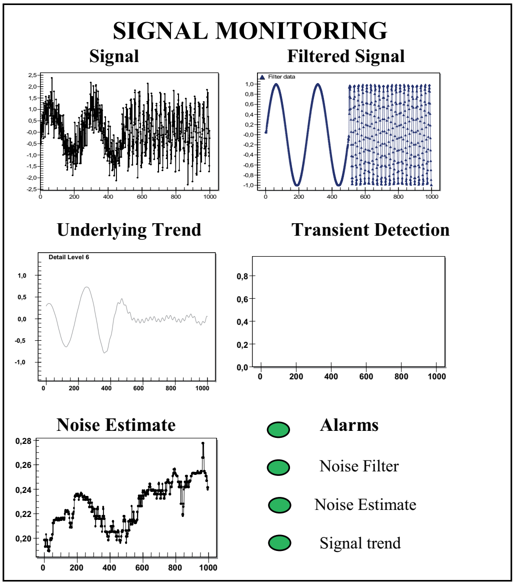


Figure 6.1: *Illustration of real-time signal filtering and process monitoring.*

In this chapter, we will develop a module for real-time signal filtering and process monitoring. First, we propose a computational optimized wavelet transform for use with signals measured in real-time. Thus we can utilize the signal filtering described in the previous chapter with signals measured in real-time. This is accomplished with a sliding window approach. A pattern recognition approach for transient detection is proposed next. Finally we propose a process monitoring module. This processing monitoring module contains the following elements; *continuously updated filtered signals, continuously updated noise estimation, continuously updated transient detection and continuously updated signal trend.* All elements in process monitoring need to be performed on a window basis in order to minimize the requirement for computer processing capacity. Discussion regarding the window size will be given for each of the elements in process monitoring.

The results presented in this chapter are previously published in Olsen et al. [(2)].

6.1 Real-time Wavelet Transform

Until now, the wavelet transform has been considered with a finite and fixed number of signal measurements. Modifications are required if the wavelet transform is utilized with continuous signal measurements, i.e. if the number of signal measurements is continuously increased. Applying the normal wavelet transform in the case of real-time signals would require great computational effort since an updated wavelet transform would require rapid and extensive recalculations as new measurements were available. In addition, depending upon the boundary conditions chosen in the wavelet transform, this might result in some disturbance in the latest (or newest) measurements. This should be avoided since the latest measurements are the ones normally viewed by a user. Consequently, it will be valuable to wavelet transform and de-noise the latest measurements without the influence of boundary conditions.

In *Figure 6.2*, eight signal measurements at times $t_0 - t_7$ are shown in the first row. The second row shows the approximate coefficients at the first decomposition level. The third row shows the approximate coefficients at decomposition level 2. As discussed in chapter 4.7, the estimation of approximate coefficients might require knowledge of what happens beyond the latest signal measurement and before the first signal measurement. As further discussed in chapter 4.7, we might choose to assume a periodic signal or a signal mirrored/folded around the first and last measurement. This might however result in a poor signal approximation for the first and the last signal measurements if this assumption is wrong (which it usually will be). Although other boundary conditions exist to minimize this problem, a sounder approach will be to eliminate those calculations in the wavelet transform that are influenced by the assumed boundary conditions.

In the *Figure 6.2,* the arrows indicate which measurements at a finer resolution level that is required in the estimation of the approximate coefficients at the next and coarser resolution level. The relationship between approximate coefficients at different resolution levels was give by *eq. (4.27)*. We see that the assumption made regarding the signal behavior beyond the last measurement, i.e. if the signal is periodic or mirrored, will affect more and more approximate coefficients at higher decomposition level (or coarser resolution level). Ignoring the detail and approximate coefficients in the wavelet transform affected by the assumption regarding the signal behavior beyond the boundaries, will give a "new" wavelet transform as shown in *Figure 6.3*. This is referred to as the real-time wavelet transform. We will see a delay in the estimation of the approximate coefficients at decomposition level 1, i.e. we can only calculate 7 approximate coefficients at decomposition level 1 without the knowledge of what happens beyond the last signal measurement. A slightly larger delay in estimation of approximate coefficients will appear at decomposition level 2 where only 5 approximate coefficients can be calculated. Thus the estimation of approximate and detail signals at higher decomposition levels will only be close to real-time, while approximate and detail signals at lower decomposition levels will be in real-time.

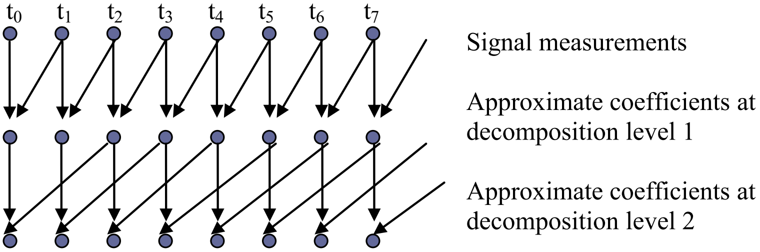

t_0
t_1
t_2
t_3
t_4
t_5
t_6
t_7
Signal measurements
Approximate coefficients at decomposition level 1
Approximate coefficients at decomposition level 2

Figure 6.2: *Traditional calculation of approximate coefficients applying 2 decomposition levels and two scaling coefficients (Haar wavelet).*

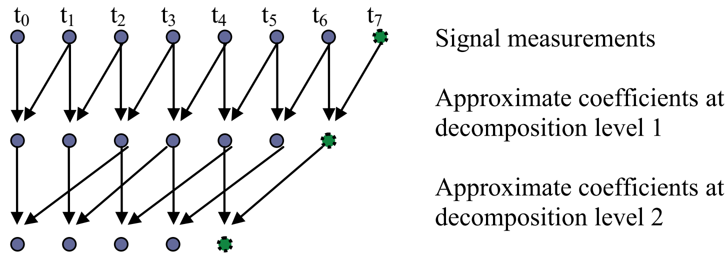

t0
t1
t2
t3
t4
t5
t6
t7
Signal measurements
Approximate coefficients at decomposition level 1
Approximate coefficients at decomposition level 2

Figure 6.3: *Calculation of approximate coefficients with eight signal measurements applying two decomposition levels and two scaling coefficients (Haar wavelet).*

In general, the number of signal measurements required to calculate approximate or detail signals at a given decomposition level is found by;

$$(6.1) \quad \#measurements = \#WC\ 2^j$$

where *#WC* is the number of wavelet coefficients in the wavelet considered and *j* is the decomposition level consider. If a wavelet with four wavelet and scaling coefficients is considered, a sample size of $4*2^5$=128 is required in order to calculate any approximate coefficients at decomposition level 5. This will in addition indicate the delay in the calculation of approximate and detail coefficients at a given decomposition level. By specifying the number of resolution levels to filter, or equally the maximum number of measurements the filtering should contain (in order to reduce the need for computer processing capacity), one can start eliminating old measurements from the beginning of the wavelet transform as illustrated in *Figure 6.4*.

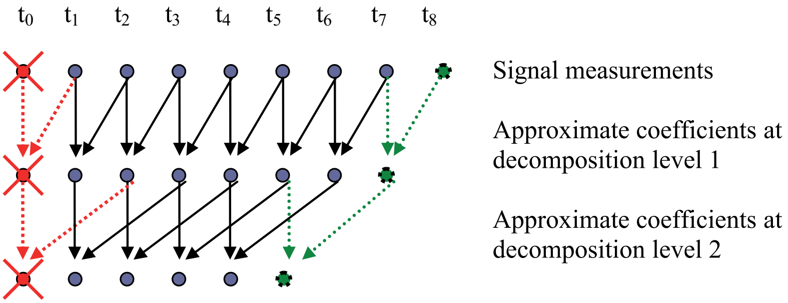

t0
t1
t2
t3
t4
t5
t6
t7
t8
Signal measurements
Approximate coefficients at decomposition level 1
Approximate coefficients at decomposition level 2

Figure 6.4: *Status after adding a new measurement applying two decomposition levels and a wavelet with two wavelet coefficients.*

If for instance the maximum number of measurements was chosen to eight or equally, the maximum number of decomposition levels was chosen equal two, *Figure 6.4* illustrates what will happen as a new measurement becomes available. The new measurement will result in calculation of two new approximate coefficients as shown to the right in *Figure 6.4*. In addition, the approximate coefficient and signal measurement with index zero are deleted. This is shown to the left in *Figure 6.4*. The real-time wavelet transform will thus only result in a small number of new calculations compared to the normal wavelet transform, and it will eliminate the influence of boundary effects in the latest measurements.

Real-time filtering with Waveshrink requires the calculation of the inverse wavelet transform. The real-time inverse wavelet transform is slightly more complicated than the real-time forward wavelet transform. This since modifications of detail coefficients at low resolution levels will cause changes in many signal approximations and thus many updates are required each time a detail coefficient is changed in Waveshrink. However, since most modifications of detail coefficients will occur at low decomposition levels, wavelet filtering in general will cause a limited number of updates in the signal approximation.

In real-time wavelet filtering, the forward wavelet transform is performed as described above when new measurements become available. This requires only a limited number of new calculations. In wavelet filtering, the new detail coefficients are then tested to determine whether or not they should be kept, reduced or killed depending upon the shrinkage rule and threshold estimator considered. If some of the new detail coefficients are changed by Waveshrink, the calculation as shown in *Figure 6.5* must be performed in order to update the filtered signal. In this figure, a detail coefficient at decomposition level 2 was changed in Waveshrink. Thus when performing the inverse wavelet transform, two approximate coefficients at decomposition level 1 will be affected, while four of the filtered signal measurements will be affected. As detail coefficients at higher decomposition levels are included in Waveshrink, a gradual signal filtering will be observed.

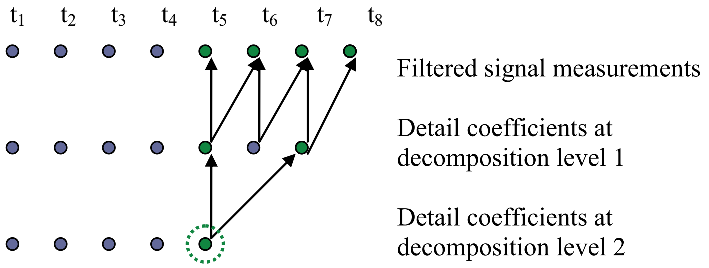

t_1
t_2
t_3
t_4
t_5
t_6
t_7
t_8
Filtered signal measurements
Detail coefficients at decomposition level 1
Detail coefficients at decomposition level 2

Figure 6.5: *Changing a detail coefficient at decomposition level 2 will require several calculations to update the filtered signal. A wavelet with two coefficients is applied (Haar).*

The approach described above for real-time wavelet transform and real-time wavelet filtering is computationally efficient and allows signal de-noising in real-time. Next we will describe a simple but efficient approach for transient detection.

6.2 Transient Detection

The term *transient* is used in many fields and in general can have many different meanings. In our case the term transient refers to an event that will pass with time and is relative short-lived. This can be illustrated by what will happen if a choke in an oil/gas well is opened and fluid starts to flow. This will result in an instant decrease in the pressure measured downhole in the well. A region of pressure drawdown starts to develop near the well caused by the fluid withdrawal. The drawdown area around the well will increase as the well continues to produce. However, after a while (if no process variables including the choke size is changed), steady state flow conditions will start to develop and the transient will pass. The duration of the transient will depend upon the process and the process state. Thus it can be said that a system is in *a transient state* if a process variable has been changed and the process yet again has not been stabilized. The detection of the start of the transients is referred to as *transient detection* and will be the topic of this section.

We would like to have a method for automatic pressure transient detection for use with signals measured in real-time. Thus we require a reliable and computationally efficient method for pressure transient detection. In addition, no user defined input should be required in pressure transient detection. The transients are detected from the pressure measurements only as these are considered more accurate and reliable than the rate measurements.

In the E&P industry, efforts have been devoted to the detection of *pressure transients* for use within well-testing. Pressure transients origin from changes made in the choke opening in a well. Thus it might be believed that one simply could detect the pressure transients by detecting when changes in the choke opening takes place. However, the problem is that the choke will use some time to open or close and this time delay might be of several minutes. It will thus be impossible to infer that the pressure transient will occur at the same time as the choke opening changes. In addition, the magnitude of this time delay normally will be unknown. Thus in order to have the exact location of the pressure transients, they need to be detected from the measurements of the pressure signal.

Athichanagorn *et.al*[(5)] identified pressure transients by the use of modulus maxima at intermediate decomposition levels using the dyadic wavelet transform (wavelet transform with down-sampling as described in chapter 4.7). The wavelet modulus maximum consists of all detail coefficients with an absolute value greater then the previous and the next detail coefficient, i.e. $|d_j(k-1)|<|d_j(k)|$ and $|d_j(k)|>|d_j(k+1)|$ at a given decomposition level j. The further identification of transients would depend on the sampling rate and a user defined slope threshold. The transients were typically detected from the detail coefficients at decomposition level 3 - 5 in the wavelet transform applying a user defined slope threshold.

Khong[(6)] discovered that the method proposed by Athichanagorn was sensitive to signal sampling rate and was not suited for automatic transient detection. In addition, Khong suggested screening for false transients by the use of Fourier analysis. Ougang *et.al*[(11)] further studied the mechanism behind transient detection by applying a formula to predict which transients that would be detected. The method proposed by Athichanagorn was also tested by Viberti et al. [(8)] They also concluded that this method required improvements, and suggested a non-wavelet based method based on the analysis of pressure slopes. This method requires two different thresholds, one for pressure build-up and one for pressure drawdowns. The thresholds need to be determined by the user with a trial-and-error procedure. Rai et al. [(18)] again considered that method proposed by Athichanagorn and concluded that the method was not adequate as the required slope threshold was difficult to determine. Rai et al. further tested the method proposed by Athichanagorn with the Haar wavelet (instead of the Spline4246 as used by Athichanagorn) and with the stationary wavelet transform (instead of the dyadic wavelet transform). Again the results were disappointing as this method still requires the determination of a slope threshold. They further studied an approach based upon Savitzky-Golay Smoothing Filters. This method fits the measured data to polynomials of various degree and transients were detected from the derivatives of the fitted polynomials. This method will in addition work well with noisy measurements. They finally studied a method called the Segmentation method. Rai et al. reported that transient detection based upon the Segmentation method gave good results. Numura et al. [(19)] considered transient detection as a non-parametric regression problem. The initial guess of the unknown parameters was obtained from the method proposed by Athichanagorn. Suzuki et al. [(20)] recently published a method for automatic transient detection based upon pattern recognition in the pressure derivatives. A threshold is required to eliminate the smallest detected transients assumed to origin from noise. Their method seems to be promising and works well with the tested signals. Thomas[(7)] applied pattern recognition methods in order to identify aberrant transients. This approach was shown to work well for the cases considered.

The transient detection methods presented in the literature can roughly be divided into four; wavelet based methods, regression based methods, threshold based methods and pattern recognition based methods. The wavelet based methods appears to have several disadvantages. Let us consider the method proposed by Athichanagorn. This method has two main weaknesses. Athichanagorn proposed to detect the transients from the wavelet modulus maxima signal at intermediate decomposition levels. Recall that the wavelet modulus maximum consists of all detail coefficients with an absolute value greater then the previous and the next detail coefficient. However, the wavelet modulus maxima signal will be shifted compared to the original signal. This is shown below in *Figure 6.6*. The original signal is shown in *Figure 6.6(a)* while the wavelet modulus maxima signal for decomposition level 3-5 is shown in *Figure 6.6 (b)-(d)*. The transient is located at *t=2501* in the original signal while the wavelet modulus maximum corresponding to this transient, is located at *t =2497* at decomposition level 3, at *t=2493* at decomposition level 4 and at *t=2486* at decomposition level 5. This shifting makes it difficult to relate the detected transients in the wavelet modulus maxima signal, to the transients in the original signal. In addition, the selection of a slope threshold to determine the "large" wavelet modulus maximum is difficult to select as reported by several authors. [(6; 10; 18)] Thus we conclude that the wavelet based methods proposed in the literature are not suited for automatic transient detection.

The regression based methods appears to work fairly well but are computational expensive[(19; 18)]. Based on this author's experience, severe convergence problems will be encountered if treating the location and number of transients as unknowns in a non-parametric regression problem as done by Numura et al. [(19)] The threshold based methods have a disadvantage as they require the determination of signal specific thresholds[(8)] and are therefore not suited for automatic transient detection. Finally, pattern recognition is reported to work well. The method proposed by Suzuki et al. [(20)] only recently came to this author's attention and it appears to work very well. The advantage with pattern recognition methods is that they require little information about the reservoir and few, if any at all, user defined limits. In addition, a pattern recognition approach is simple to implement and can be performed computationally efficient with real-time signals. Hence we chose to detect transients by pattern recognition. We have chosen a method slightly different from the one presented by Thomas[(7)]. As opposed to Thomas, we supply the system with predefined patterns for the transients as will be described in the next section.

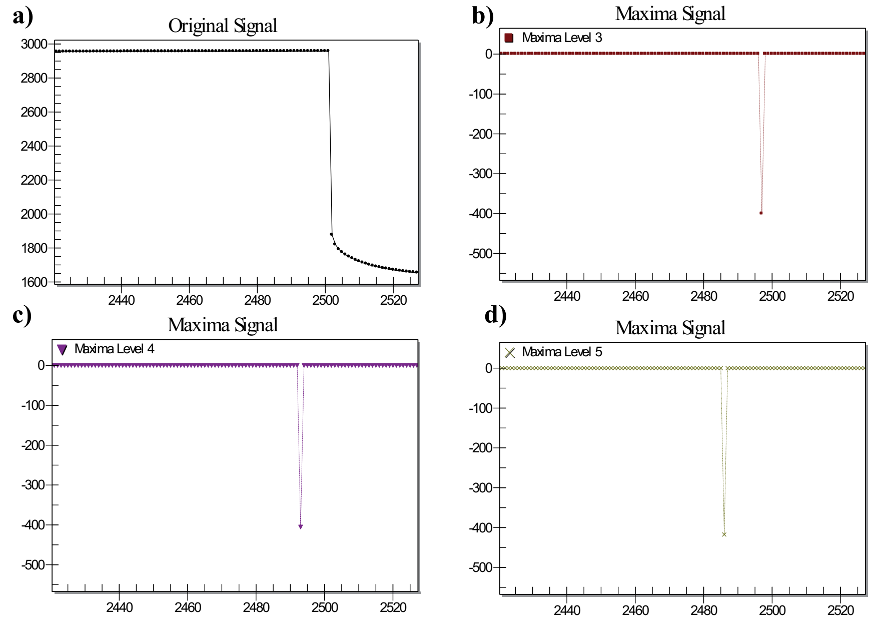


***Figure 6.6**: Wavelet maxima signal shifted a various decomposition levels in the wavelet transform.*

Finally note that transient detection was included as the third step in the procedure by Athichanagorn et al. [(5)]. This means that transients were detected after outlier removal and de-noising but before data reduction. In our work, transient detection will be performed subsequent to data reduction. Athichanagorn et al. applied wavelet analysis for transient detection and hence it was beneficial to detect transients before data reduction as wavelet analysis requires equally spaced data. A typical data reduction algorithm removes signal measurements from periods with small and assumed insignificant signal changes while it keeps the signal measurements in periods with large and assumed significant signal measurements. Hence data reduction tends to create unequally spaces signal measurements. In this chapter, transient detection by a pattern recognition approach is proposed. Consequently, transients can equally good and more computationally efficient be detect after data reduction. Therefore we chose to detect pressure transients after data reduction.

Note that we in this section will detect the transients from the available signals measurements. However, since we are dealing with finite sampled signals, the signal might not be sampled at the exact time when a transient truly occurs. This needs to be corrected for in well-testing and will be discussed in chapter 7. However, it is outside the scope of this chapter.

6.2.1 Pattern Recognition

We chose to detect transients by a simple pattern recognition approach. We chose to represent the transients as pixel based images. The images will contain binaries, either 0 or 1. A window approach screening for pressure transients is utilized. We choose to include several measurements both before and after the pressure transients. In this work we have chosen to apply a window with 20 measurements, i.e. ten measurements before and ten measurements after the start of the pressure transient. This appears to be a good choice based on a trial-and–error approach. A larger window size will prevent detection of transients close to each other, while a smaller window will make the detection more uncertain, i.e. it is more likely that we might end up detecting false transients. In addition, we did chose to remove up to nine successive outliers by the median outlier filter in chapter 5.2.1. Thus if a pressure transient contains nine or less measurements, it will be removed by the outlier filter. This means that if the outlier filter works properly, all remaining transients at least will have ten measurements. It was illustrated in chapter 5.2.1 that it was unlikely that important information was removed from the pressure measurements by removing up to nine successive outliers.

To represent the measurements as an image, we first we need to determine the vertical resolution. The vertical resolution is simply determined by dividing the maximum pressure difference in the window considered by the window size. The index of the pressure measurements is used as the horizontal resolution. The pressure measurements in each window will be represented as a 20x20 image. ***Note that since a window size of 20 is used in the transient detection, at least 10 measurements need to separate two transients to ensure that both are detected.*** Let us illustrate this with an example. Consider the 20 pressure measurements as shown in *Figure 6.7*. The minimum pressure measurement is 8665 psi and the maximum pressure measurement is 9179 psi. This gives a pressure difference of 514 psi and consequently a vertical resolution of 26 psi. The pressure measurements can then be represented as a pixel based image as shown in *Figure 6.8*.

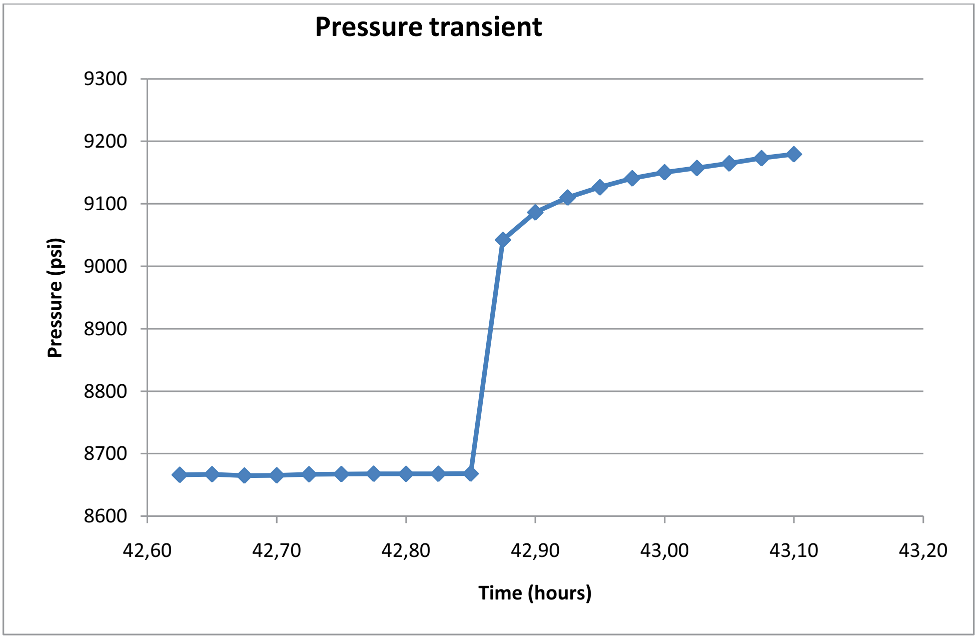
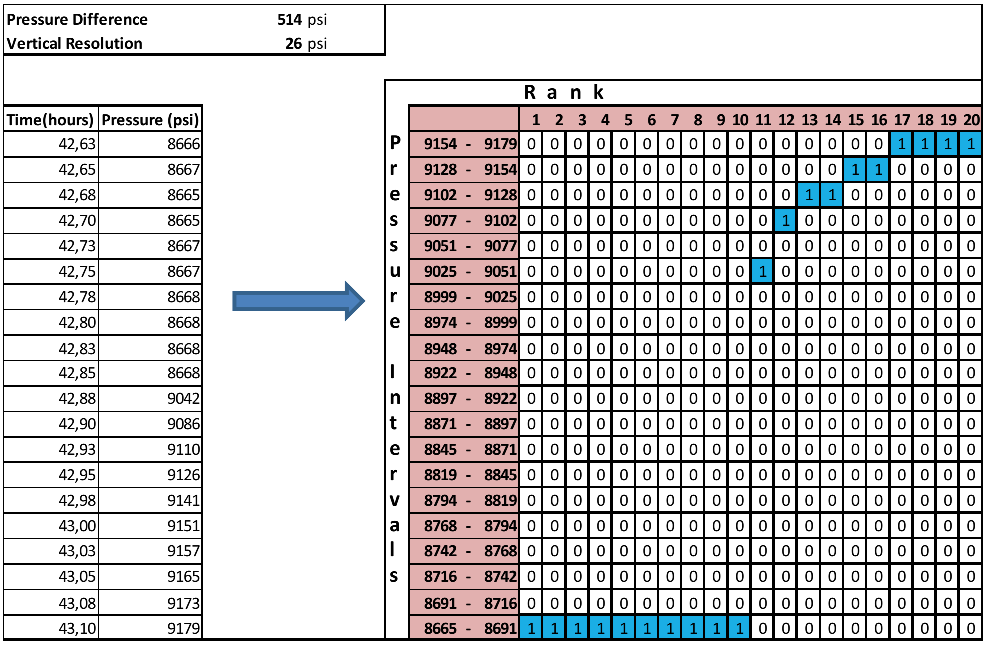


Figure 6.7: *Pressure build-up transient used in pattern recognition.*



Pressure Difference	514 psi
Vertical Resolution	26 psi

Time(hours)	Pressure (psi)
42,63	8666
42,65	8667
42,68	8665
42,70	8665
42,73	8667
42,75	8667
42,78	8668
42,80	8668
42,83	8668
42,85	8668
42,88	9042
42,90	9086
42,93	9110
42,95	9126
42,98	9141
43,00	9151
43,03	9157
43,05	9165
43,08	9173
43,10	9179

		Rank																			
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
P	9154 - 9179	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
r	9128 - 9154	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
e	9102 - 9128	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
s	9077 - 9102	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
s	9051 - 9077	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
u	9025 - 9051	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
r	8999 - 9025	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
e	8974 - 8999	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	8948 - 8974	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I	8922 - 8948	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	8897 - 8922	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
t	8871 - 8897	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
e	8845 - 8871	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	8819 - 8845	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v	8794 - 8819	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
a	8768 - 8794	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
l	8742 - 8768	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
s	8716 - 8742	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	8691 - 8716	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	8665 - 8691	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0

Figure 6.8: *Pressure measurements from Figure 6.7 represented as a 20x20 image.*

In pattern recognition, it is possible to provide the system with a predefined pattern or to use some sort of a system learning process, like neural networks [(7)], to train the system for pattern recognition. We choose to provide the system with predefined patterns of the transients, two for build up transients and two for drawdown transients. In this context we refer to a pressure build-up as a rate change that cause a pressure increase, while a pressure drawdown is a rate change that cause a pressure decrease. The pressure prior to a pressure build-up can either increase or decrease depending on the previous rate changes. In the same manner, the pressure prior to a pressure drawdown can either increase or decrease. To us it appears highly beneficial to distinguish between these two scenarios and to provide different patterns for these two cases. The two predefined patterns for build-up transients are shown in *Figure 6.9* - *Figure 6.10* while the two predefined patterns for drawdown transients are shown in *Figure 6.11* - *Figure 6.12*. These typepatterns were generated by considering a large number of both synthetic and real pressure signals.

A simple computer algorithm compares the estimated transient patterns, as illustrated in *Figure 6.8*, with the predefined patterns as shown in *Figure 6.9* - *Figure 6.12*. If the estimated transient pattern matches one of the four predefined patterns, a transient is detected. ***A transient is detected if and only if the entire transient image falls upon one of the predefined transient patterns.***

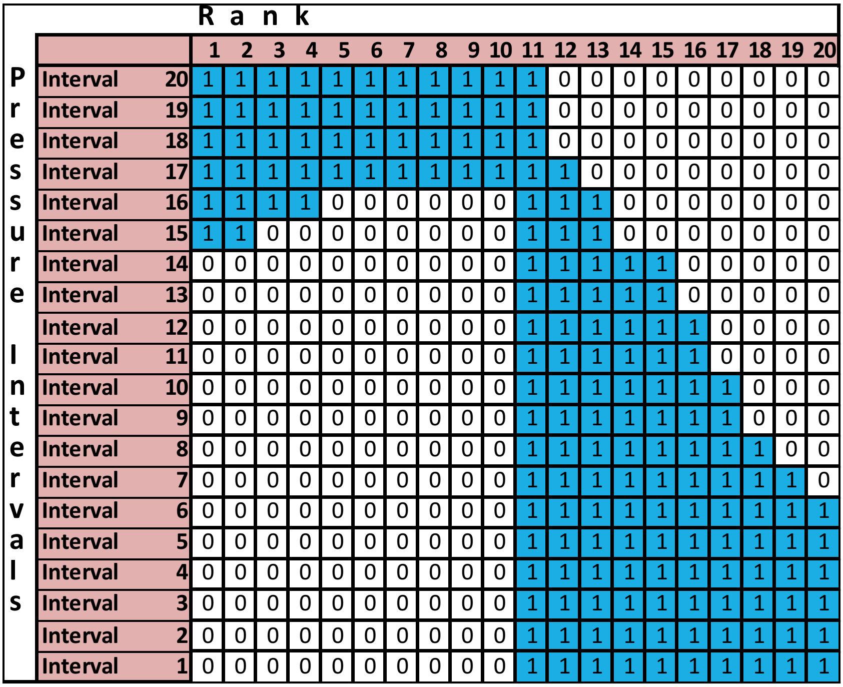
It will be important to remove outliers from the signal before transient detection. It is also recommended to remove signal noise, which might be critical if the noise level is high. On the other hand it is important not to use a noise filter that tends to over-smooth the signal and consequently the transients, since this again will make transient detection more difficult. We believe that proper outlier removal and de-noising can be achieved with the proposed signal filtering in chapter 5. As a general rule, outliers and noise should be removed before transient detection.

Pressure Intervals	Rank																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Interval 20	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
Interval 19	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
Interval 18	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
Interval 17	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
Interval 16	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
Interval 15	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0
Interval 14	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0
Interval 13	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0
Interval 12	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0
Interval 11	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
Interval 10	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
Interval 9	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
Interval 8	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
Interval 7	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0
Interval 6	1	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
Interval 5	1	1	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
Interval 4	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
Interval 3	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
Interval 2	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
Interval 1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0

Figure 6.9: *Predefined pattern for pressure build-up with prior pressure decrease.*

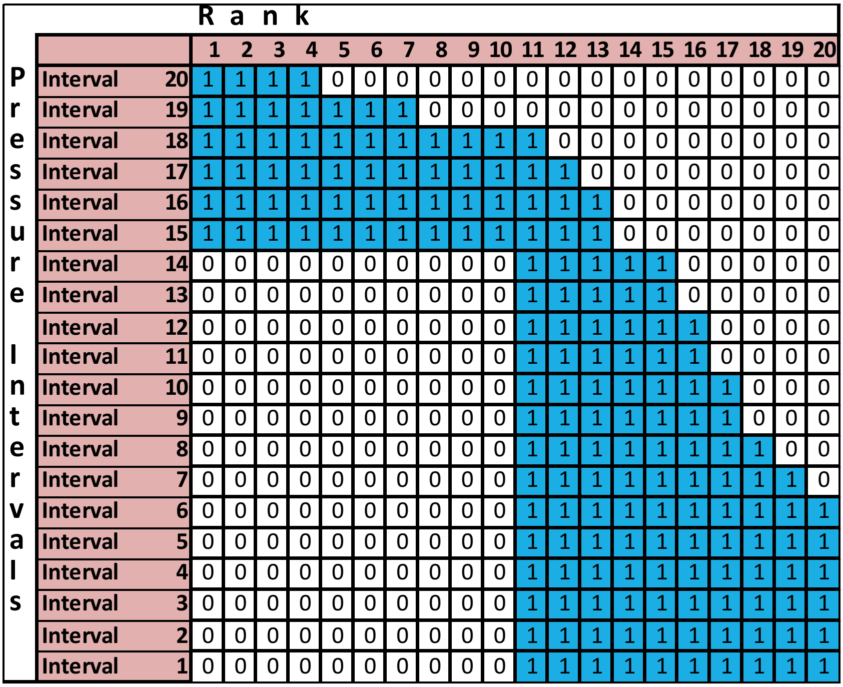
Pressure Intervals	Rank																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Interval 20	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
Interval 19	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
Interval 18	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
Interval 17	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
Interval 16	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
Interval 15	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0
Interval 14	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0
Interval 13	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
Interval 12	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
Interval 11	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
Interval 10	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
Interval 9	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
Interval 8	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
Interval 7	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0
Interval 6	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
Interval 5	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
Interval 4	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
Interval 3	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
Interval 2	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Interval 1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 6.10: *Predefined pattern for pressure build-up with prior pressure increase.*



	Rank																			
Pressure Intervals	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Interval 20	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
Interval 19	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
Interval 18	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
Interval 17	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
Interval 16	1	1	1	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0
Interval 15	1	1	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0
Interval 14	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
Interval 13	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
Interval 12	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0
Interval 11	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0
Interval 10	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0
Interval 9	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0
Interval 8	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0
Interval 7	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0
Interval 6	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
Interval 5	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
Interval 4	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
Interval 3	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
Interval 2	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
Interval 1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1

Figure 6.11: *Predefined pattern for pressure drawdown with prior pressure increase.*



	Rank																			
Pressure Intervals	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Interval 20	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Interval 19	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Interval 18	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
Interval 17	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
Interval 16	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
Interval 15	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
Interval 14	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
Interval 13	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
Interval 12	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0
Interval 11	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0
Interval 10	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0
Interval 9	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0
Interval 8	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0
Interval 7	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0
Interval 6	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
Interval 5	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
Interval 4	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
Interval 3	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
Interval 2	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
Interval 1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1

Figure 6.12: *Predefined pattern for pressure drawdown with prior pressure decrease.*

A pattern recognition approach will not provide us with the magnitude of the detected transients. We can have very small pressure changes with the predefined patterns as shown in *Figure 6.9 - Figure 6.12* or we could have large pressure changes. What we would like to do is to detect the large pressure changes and ignore the smallest ones. The smallest transient patterns are assumed to be caused by an underlying noise structure which is not suppressed by de-noising. We chose to eliminate the smallest transient patterns in the same manner as we eliminated the smallest outliers in chapter 5.2. Hence a transient is detected if and only if the pressure difference in the window considered is at least three times the estimated noise level. Thus outlier removal, data reduction and transient detection all rely upon an accurate noise estimate as can be obtained in Waveshrink as shown in chapter 5.3. If the signal is virtually noise free or has been denoised in advance, another criterion is required in order to eliminate the very smallest transients**.** If the standard deviation of an image window with an identified transient is less than a factor 0.1 of the total signal standard deviation, it is discarded. ***Hence a pressure transient is detected if and only if the entire transient image falls upon one of the predefined transient patterns and the pressure difference in the window considered is at least three times the estimated noise level and the standard deviation in the window considered has a standard deviation at least 0.1 of the total signal standard deviation.***

In addition we will calculate the relative size of the detected transients compared to each other. This means that the largest detected transient will have a value equal one while the rest of the detected transients will get a size relative to the largest detected transient. Note that this is simply used as a tool for visualization of the relative transient "strength" and it has no other practical implications. In real-time, such a list will be continuously updated as new transients are detected.

6.2.2 Applications

The proposed method for transient detection was tested with numerous synthetic generated pressure signals and for two cases with real data from two North Sea wells.

Case 1: Transients in synthetic generated signals

In this case a total of 10 synthetic pressure signals were generated. The signals were generated as described in chapter 5.1. Two of the signals considered are shown in *Figure 6.13*. The signals contain both outliers and noise. Outliers and noise are removed before transient detection with the methods previously described in chapter 5.2 and chapter 5.3.

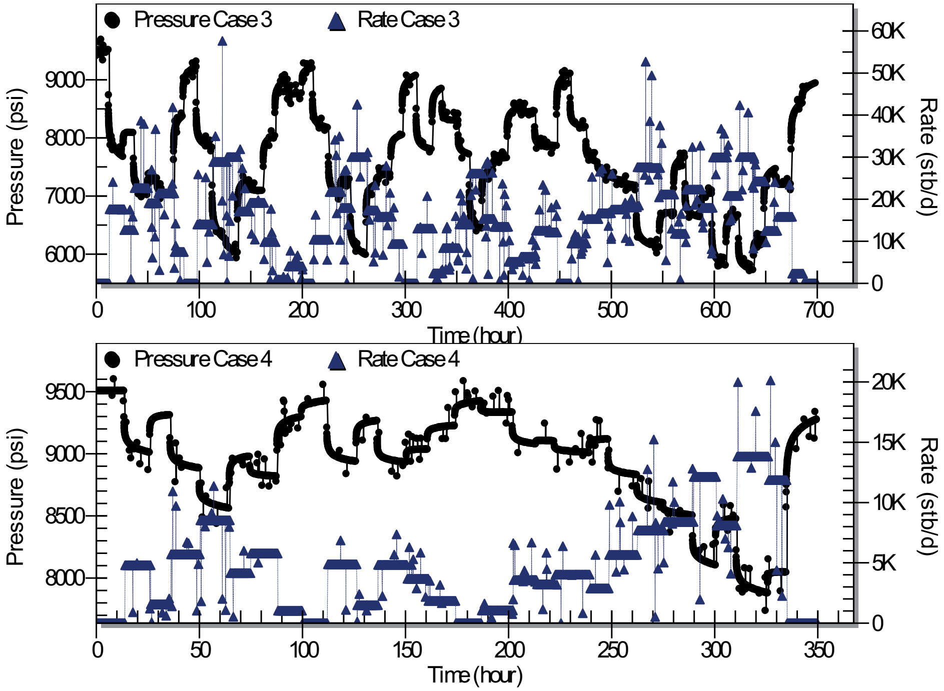


Figure 6.13: *Synthetic generated signals used for transient detection.*

When tested on synthetic signals, more than 86% of the transients were accurately detected. About 13% of the transients were missed. Only the very smallest transients with pressure changes slightly above the estimated noise level were missed. All the “large” transients in terms of pressure change were accurately detected. No erroneously transients were detected in these cases. This is summarized in *Figure 6.14.*

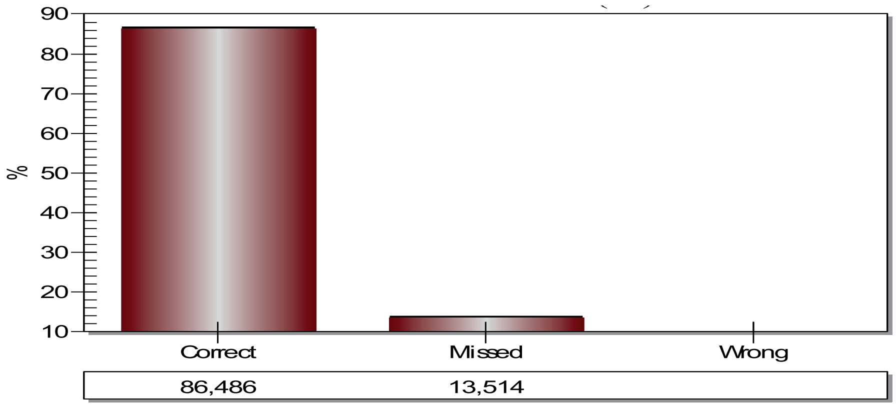


Figure 6.14: *Summery of correctly, missed and wrongly detected transients for 10 synthetic generated pressure signals.*

Case 2: Transients in North Sea well #1

Pressure measurements was been obtained from a North Sea well. As opposed to synthetic data, the use of real measurements requires the author to scan through the measurements to determine whether or not a transient is correctly detected. We thus distinguish between correctly detected transients, those missed due to outliers, those missed because they were too small in terms of pressure change, those missed due to "wrong" patterns, those erroneously detected and finally those pressure changes that were difficult to classify whether or not to consider as transients. The original signal and the detected transients are shown in *Figure 6.15*. Zoom in on *Figure 6.15* are shown in *Figure 6.16 - Figure 6.18.*

The final results are the average values found by the author and co-workers and are shown in *Figure 6.19*. 58 transients where correctly determined, 13 missed because they were too small in terms of pressure change and two missed because of "wrong" patterns. Finally, five transients were difficult to classify and it is uncertain whether or not they should have been detected as transients. This means that 79% of the transients were correctly determined, while 21% were missed, mainly because they had too small pressure changes. All "large" transients were accurately detected. No false transients were detected. A summary of the results are shown in *Figure 6.19*.

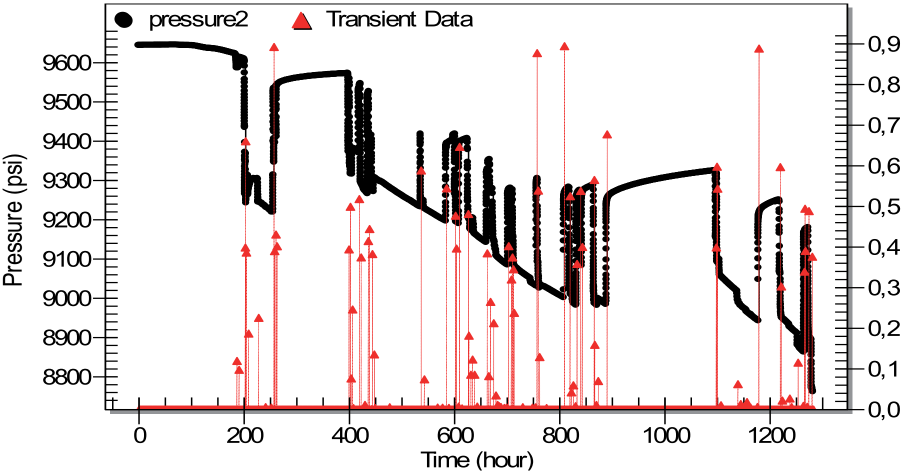


Figure 6.15: *Illustrates the transients detected along with the measured pressures from a North Sea well.*

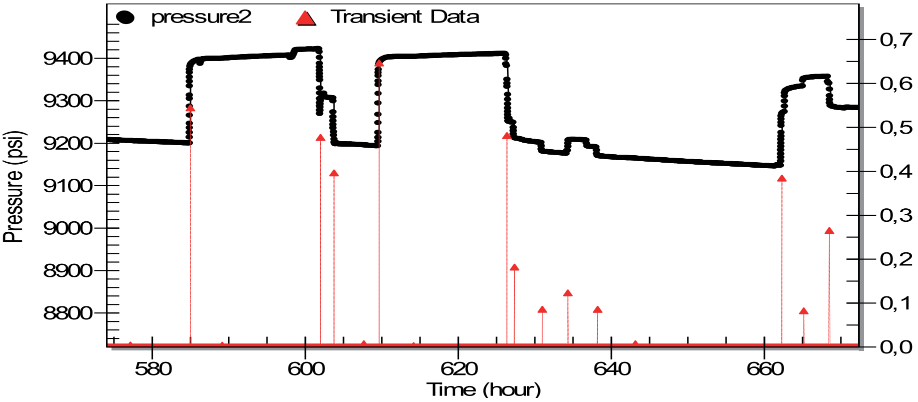


Figure 6.16: *Illustrates the detected transients along with the measured pressures from a North Sea well. Zoom in from Figure 6.15.*

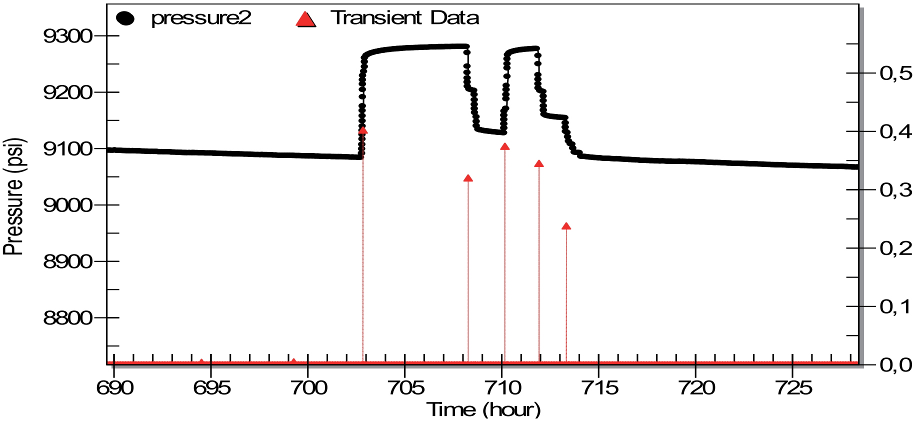


Figure 6.17: *Illustrates the detected transients along with the measured pressures from a North Sea well. Zoom in from Figure 6.15.*

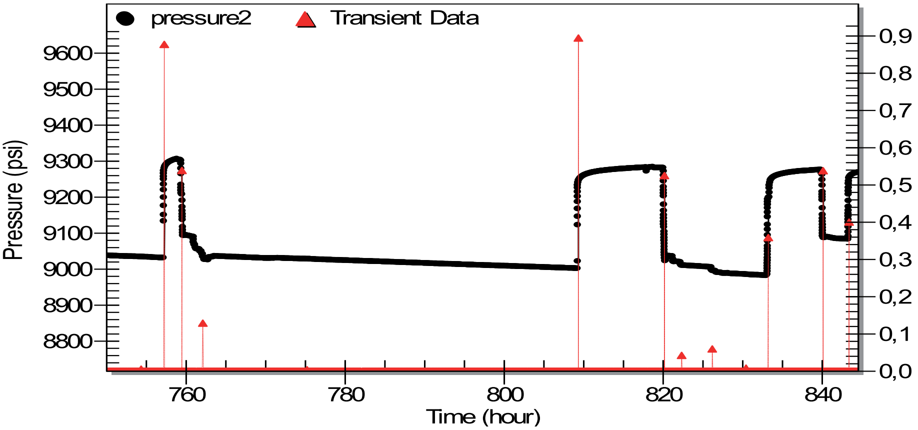


Figure 6.18: *Illustrates the detected transients along with the measured pressures from a North Sea well. Zoom in from Figure 6.15.*

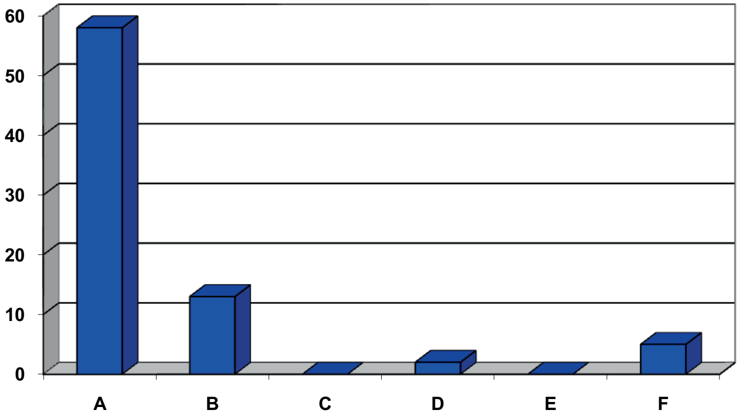


***Figure 6.19:** Summary of **(A)** correctly detected transients, **(B)** missed transients with small pressure changes, **(C)** missed transients due to outliers, **(D)** missed transients with "wrong" patterns, **(E)** erroneously detected transients and **(F)** transients difficult to classify for North Sea well #1.*

Case 3: Transients in North Sea well #2

The following pressure history was obtained from another North Sea well. The transients were detected and classified as in the case above. The pressure signal is shown in *Figure 6.20* while the results from the detection of transients are shown in *Figure 6.21.*

55 transients where correctly detected, 12 missed since they had to small pressure changes and two missed because of wrong pattern. Finally, four transients were difficult to classify and it is uncertain whether or not they should have been detected as transients. This means that 80% of the transient were correctly determined, while 20% were missed mainly because the pressure change was too small. All "large" transients in terms of pressure changes were accurately detected. No false transients were detected.

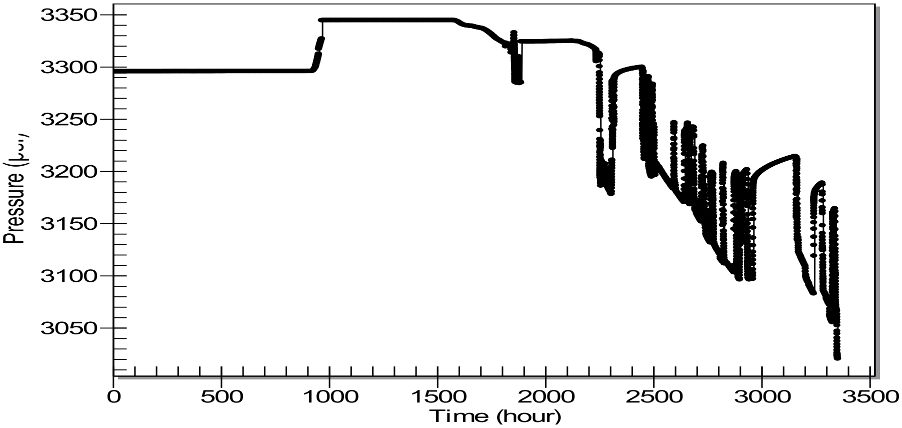


***Figure 6.20**: Pressure data from North Sea well #2.*

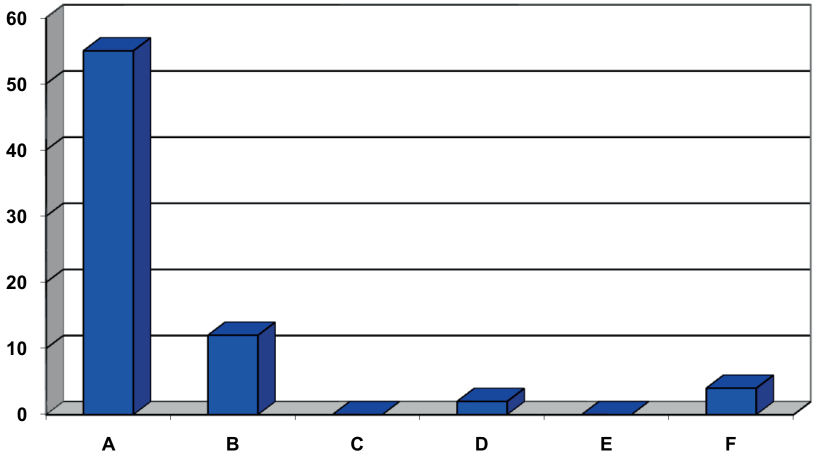


***Figure 6.21:** Summary of **(A)** correctly detected transients, **(B)** missed transients with small pressure change, **(C)** missed transients due to outliers, **(D)** missed transients with "wrong" pattern, **(E)** erroneously detected transients and **(F)** transients difficult to classify for North Sea well #2.*

We believe that the proposed method for transient detection will perform well compared to the other proposed methods for transient detection in the E&P industry*, although no comparison study has been performed.* This claim is based upon the disadvantages reported by several authors[(6; 10; 18)] for the wavelet based methods. In addition the regression based methods are computationally expensive[(18; 19)] and when considered as a non-parametric regression problem[(19)], convergence trouble can be a serious problem. The exception is the pattern recognition methods, such as the one proposed by Suzuki et al. [(20)], which is based upon pattern recognition in the pressure derivative. However, the work by Suzuki was just recently brought to this author's attention.

In summary the proposed method is reliable, it is computationally efficient and requires no user input. In addition, it is well suited for pressure signals measured in real-time. In the cases considered, only small transients, in terms of pressure changes, were missed. All "large" transients were accurately detected and no erroneously transients were detected. This holds the promise for automatic transient detection.

6.3 Real-time Process Monitoring

The proposed module for real-time process monitoring includes real-time noise estimation, real time signal filtering, real-time trend estimation and real-time transient detection. We will start by describing real-time noise estimation.

6.3.1 Real-Time Noise Estimation

The preferred noise estimator in Waveshrink depends upon the median value of the absolute value to the detail coefficients at the first decomposition level in the wavelet transform. By using a moving median window, this estimate can be calculated continuously and in real-time. This window should have some extent in order to minimize the error, as the error decreases as the window size increases. The formula for the moving robust median noise estimation is given as;

$$(6.2) \qquad \sigma = \frac{RMAD_{moving\_window}}{0.6745},$$

where details about the RMAD noise estimator was discussed in chapter 5.3.4. In *Figure 6.22* we have illustrated the estimation of the median value for a window size of five. At time t_5, we have obtained five detail coefficients in the real-time wavelet transform. This gives a median value of 2. At time t_6, one additional measurement is available and one new detail coefficient is estimated in the real-time wavelet transform. Since a window size of five is chosen, the first point is removed and a new median value of 2 is estimated based on five latest (newest) estimated detail coefficients. At time t_7, the same process is repeated, one new detail coefficient is added and a median value of 3 is estimated from the five latest detail coefficients. As new details coefficients are obtained, this process is continuously repeated, providing a continuously updated noise estimate from *eq. (6.2)*. If the detail coefficients in addition are stored in a continuously sorted list, the median value will always be the element in the middle of the list or if even sized, the average value of the two middle elements. By such an approach, the median value is computationally efficiently obtained.

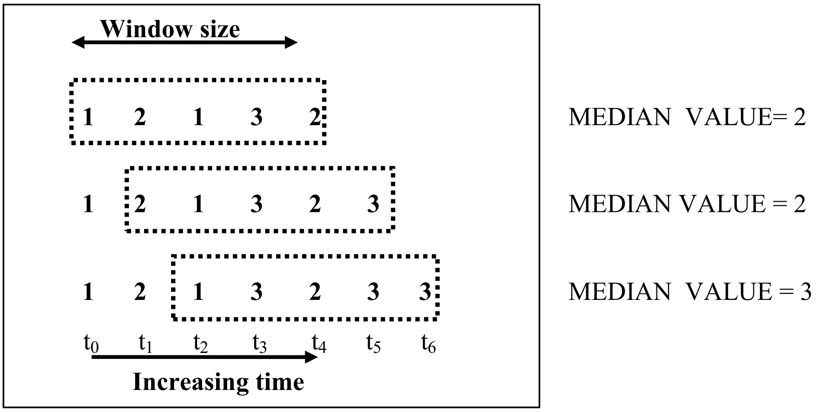


Figure 6.22*: Real-time noise estimation. A continuously updated median value is obtained as new detail coefficients become available.*

By the use of a window based robust median noise estimator *eq. (6.2)***,** an accurate noise estimate as a function of time is obtained. A change in the noise estimate can indicate problems with the measuring device or interference at the signal transmission. White noise with standard deviation 1 on the time interval *t=0* to *t=30* was added to a pressure signal. After time *t=30*, white noise with standard deviation 2.2 was added to the pressure signal. The noise was estimated from the pressure signal by *eq. (6.2)* with a window size of 200, i.e. the

200 latest pressure measurements were used in the continuous estimation of the noise level. The noise estimation was repeated each time a new signal measurement was available. The result from this estimation is shown in *Figure 6.23* along with the "true" noise trend.

A small window size will allow a rapid detection of changes in noise level in the measured signal. On the other hand, a large window will result in a more accurate noise estimate but in a slower detection of changes in the noise level. Accurate noise estimates can be obtained with as little as 200 signal measurements (see *Figure 5.28*) and allows rapid detection of changes in noise level.

Note that this window size should not be confused with the later recommended window size in real-time signal filtering. The aim in real-time signal filtering is to get an accurate noise estimate to achieve accurate signal filtering and it might be beneficial to use a larger window. In real-time noise estimation, the goal is rapid detection of possible changes in noise level. Thus a smaller window size is preferred.

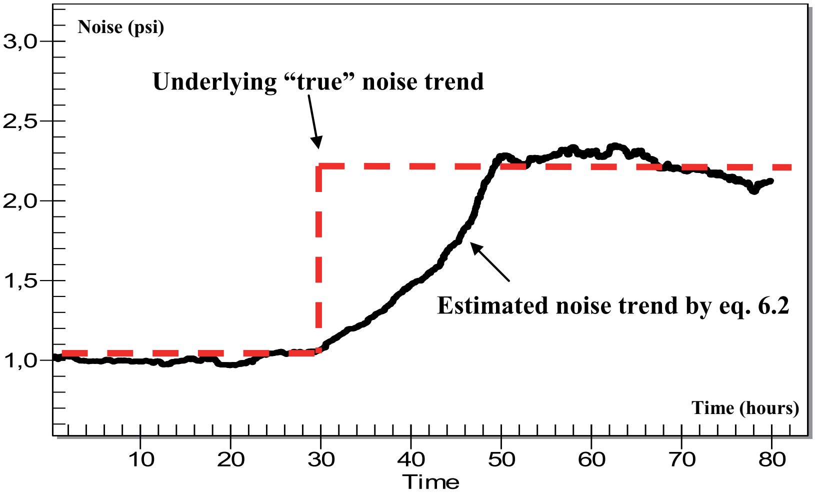


Figure 6.23: *Noise estimate as a function of time from a synthetic generated pressure signal where the noise level is increased at time t=30 hours.*

Next we will consider real-time signal filtering.

6.3.2 Real-Time Signal Filtering

Real-time signal filtering will provide the user with continuously and automatically filtered pressure and rate signals. All selections required for an automatic signal filter were determined in the previous chapter. The filtering could be done automatic as no user input was required. Signal filtering removes outliers, signal noise and reduces the number of signal measurements. This ensures that data is ready for analysis at any time.

De-noising is accomplished with Waveshrink as described in chapter 5.3. In addition a sliding window approach, as described in chapter 6.1, is required for signals measured in real-time. The window size should be chosen as large as possible, but this will of course be limited by computer processing capacity. We have not done any extensive study to determine which minimum window size to consider in Waveshrink, but some simple observations can be made from the comparison study performed in chapter 5.3.

In *Figure 5.36 - Figure 5.38* it was shown highly beneficial, in terms of root-mean-squared error, to include at least 4.096 measurements in Waveshrink. A significant smaller root-mean-squared error is achieved with 4.096 measurements in *Figure 5.37* than with 512 signal measurements in *Figure 5.36.* No significant improvements are achieved in terms of root-mean-squared error in *Figure 5.38* when 32.768 measurements are included in Waveshrink. When utilizing 4.096 signal measurements as a minimum window size in Waveshrink, good computational performance is achieved, and consequently we perform no further study to determine whether or not an even smaller window size could have been utilized. Consequently, at least 4.096 measurements should be included in the window considered for real-time de-noising. This is a minimum recommendation and additional measurements might be included if possible, but this is often limited by available computational facilities. The real-time wavelet filtering is updated each time a new measurement becomes available.

Outlier removal was described in chapter 5.2. The median filter used for outlier removal is already a sliding window filter and requires no modifications for use with real-time signal measurements. However, the proposed method for outlier removal requires an accurate estimate of the noise level in the signal measurements. An accurate noise estimate for use in outlier removal can be achieved with as little as 200 signal measurements (see *Figure 5.28*). However, for simplicity and computational efficiency, the same window size will be applied in real-time outlier removal as in real-time de-noising. For that reason a window size of at least 4096 signal measurements is applied for noise estimation in real-time outlier filtering. In chapter 5.2 we recommended removal of up to nine successive outliers for pressure signals and only single outliers for rate signals. If the latest (or newest) measurement is far outside the "normal" signal trend, we do not know if this is an outlier or the beginning of a new transient. Thus a delay is required in the outlier filter equal the number of successive outliers that can be removed. The nine latest measurements are therefore never included in the outlier filter for pressure signals and the latest measurement is never included in the outlier filter for rate signals. Outlier removal will be performed each time a new measurement is available.

Data reduction requires a de-noised signal as described in chapter 5. Since at least 4.096 measurements are recommended for de-noising, the same window size will be utilized in real-time data reduction. In addition, it is not meaningful to utilize data reduction each time a new signal measurement is available. Consequently, we will only apply data reduction each time at least 4.096 new measurements are available. Recall that there will be a delay in the de-noising and data reduction should not be utilized with signal measurements until they are completely denoised. Since a window size of at least 4.096 is utilized in de-noising, the latest 4.096 signal measurements should not be included in data reduction. If a larger window size is utilized in de-noising, the delay in data reduction should be adjusted likewise.

It should again be emphasized that the recommended window size of 4096 signal measurements in real-time signal filtering is a minimum recommendation. If possible, additional measurements should be included. However this is in many cases limited by computational facilities.

Next we will illustrate the real-time noise filtering process. Initially, only a small number of signal measurements are available and we can only utilize Waveshrink with a limited number of resolution levels. As more measurements become available, additional resolution levels can be included in Waveshrink to improve de-noising. This was previously illustrated in *Figure 5.36 - Figure 5.38* where the root-mean-squared error decreased as the number of measurements available increased and additional resolution levels could be included in Waveshrink. If only two resolution levels were available in Waveshrink, only a fraction of the noise will be removed. This is illustrated in *Figure 6.24.* In this figure the filtered signal is shown along with the original noisy signal. All the figures *(Figure 6.24-Figure 6.26)* are zoomed in at the same time interval to show the gradual smoothing of the filtered signal. In *Figure 6.25*, additional measurements become available and additional decomposition levels can be included in Waveshrink, four in this case. More noise is removed in the filtered signal which gradually becomes smoother. Note that new signal measurements will be added outside these windows and are therefore not visible.

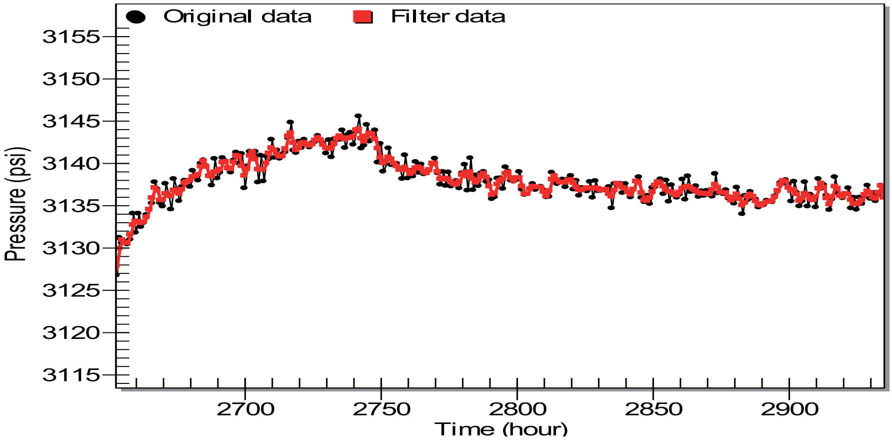


Figure 6.24: *In the beginning, only a few resolution levels are included in Waveshrink and the signal is only slightly de-noised.*

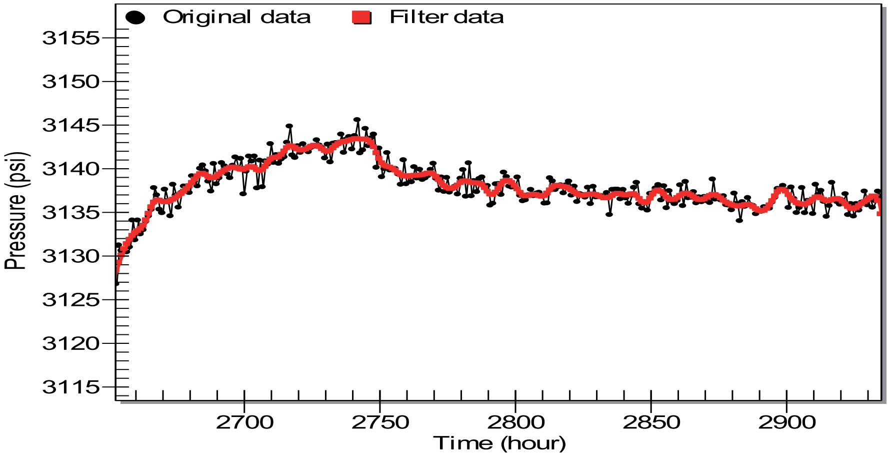


Figure 6.25: *After a while, additional resolution levels are included in Waveshrink as the number of signal measurement increases, and the signal is further de-noised.*

In *Figure 6.26,* five resolution levels are included in Waveshrink and the filtered signal becomes even smoother. The methods and selections, as described in chapter 5.3, are used for pressure signal de-nosing.

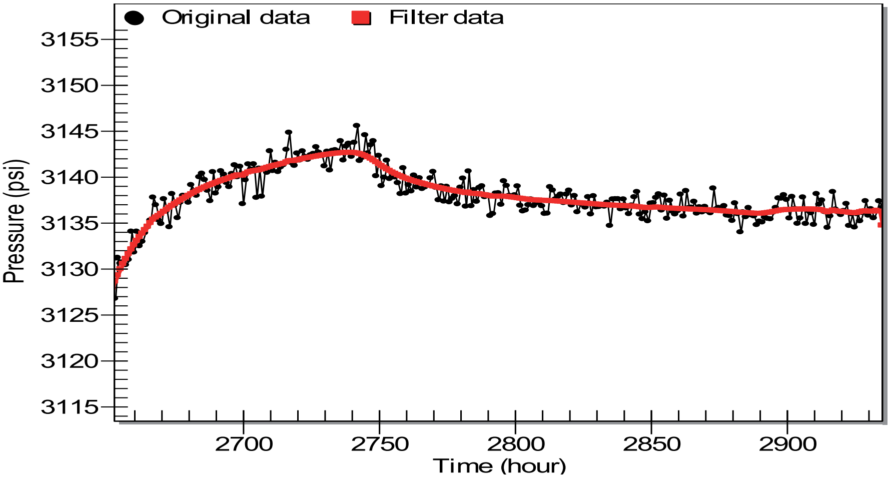


Figure 6.26: *The final resolution levels are included in Waveshrink and the signal is further de-noised.*

Next we will illustrate how the wavelet transform can be used to obtain a real-time signal trend.

6.3.3 Real-Time Signal Trend

The approximate wavelet signals at coarse resolution levels are ideal for investigating underlying signal trends. The approximate signal at coarse resolution levels (or high decomposition levels in the wavelet transform) will give us the low frequency content of the signal as all the high frequency content is removed from the signal approximation and stored in the detail signals. For the signal shown in *Figure 6.27*, the approximate signal at decomposition level 12 in the wavelet transform is shown in *Figure 6.28.* This will provide us with the underlying signal trend. By inspecting the underlying trend in *Figure 6.28*, it is possible to find that the pressure is slightly increasing, which is virtually impossible to see from the original signal in *Figure 6.27* alone. The wavelet Spline39 is used in the wavelet transform.

If the same window size of 4.096 signal measurements is utilized in the real-time trend analysis as in the real-time wavelet filtering, approximate signals at 12 different resolution levels is available. Hence we recommend the same window size in real-time trend analysis as in real-time wavelet filtering. It will be a good choice to monitor the approximate signal a both intermediate and at high decomposition levels. Thus, signal trends are recommended monitored at decomposition level 6 and 12.

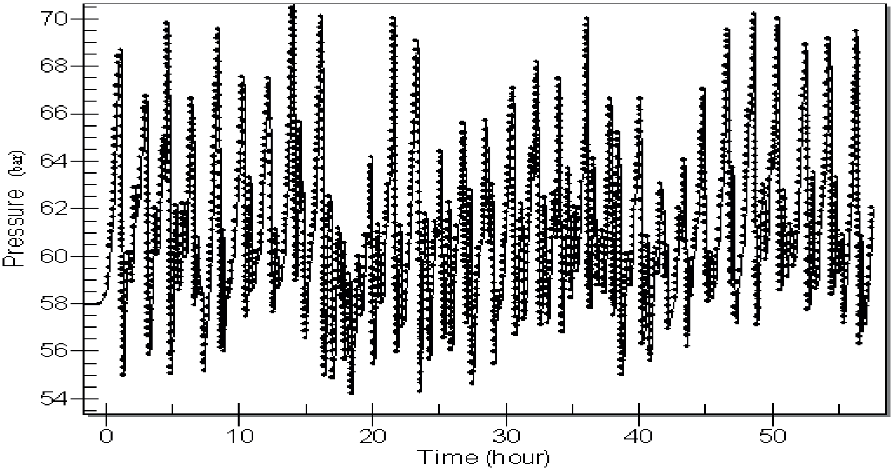


Figure 6.27: Signal used for trend investigation.

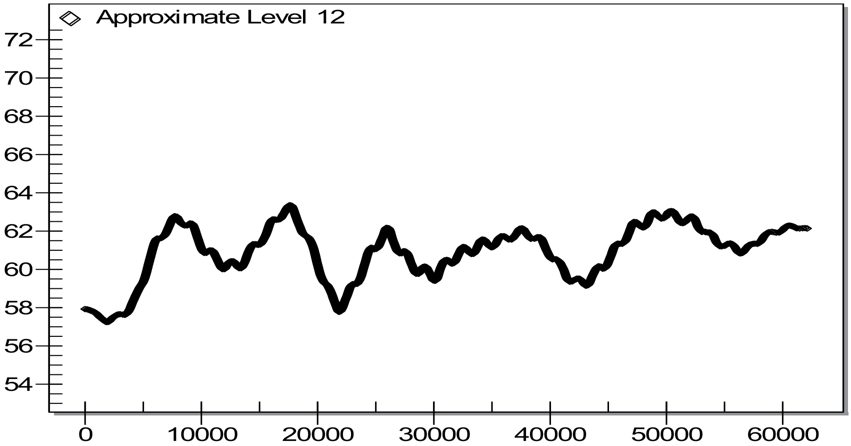


Figure 6.28: Underlying signal trend from wavelet approximate signal at decomposition level 12. Original signal is shown above in Figure 6.27.

The next element in real-time process monitoring is real-time transient detection.

6.3.4 Real-Time Transient Detection

Transients are detected by the pattern recognitions approach described in chapter 6.2. The proposed pattern recognition approach is implemented as a sliding window analysis and is thus well suited for use with pressure signals measured in real-time.

Previously we have stated that outliers and noise should be removed from the signal measurements before transient detection. However, with the proposed window size of 4096 for de-noising, this would result in a significant delay in transient detection if de-noised signal measurements were required. ***Thus for transient detection in the context of real-time process monitoring, noisy signal measurements will be used.*** By applying noisy signal measurements in transient detection, some additional and small transients might be missed, compared to the case where de-noised signal measurements are applied. However, the "large" transients will still be accurately detected in the proposed method for transient detection. Hence we have to compromise between a rapid detection of transients and an accurate detection of "small" transients. We believe that in the context of real-time pressure monitoring, a rapid detection of transients is more important. Note that outliers will be removed before transient detection as the proposed delay in outlier removal is only nine measurements. To ensure that outliers are removed before transient detection, a delay in the transient detection equal to the window size in the outlier median filter is required. Thus the latest (or newest) 19 measurements are not included in transient detection.

The proposed pattern recognition is computationally efficient and is performed each time a new measurement is available. An example illustrating the real-time transient detection is given below.

The original pressure signal is shown in *Figure 6.29.* The transients detected for the first 200 hours in this signal is shown in *Figure 6.30.* Note that the relative magnitude of the detected transients is considered as large in this figure. However, as new transients are detected, the first detected transients will be classified as relatively small with respect to the latest detected transients. In the last two figures, *Figure 6.31* and *Figure 6.32*, the relative magnitude of the first two detected transients from *Figure 6.30* will be relatively small compared to the latest detected transients. In *Figure 6.31* and *Figure 6.32*, the relative magnitude of the detected transients is somewhat constant, thus indicating that the relative magnitude of the detected transients will stabilize as the number of detected transients increase. Recall that the displayed relative magnitude of the detected transients is simply used for visualization.

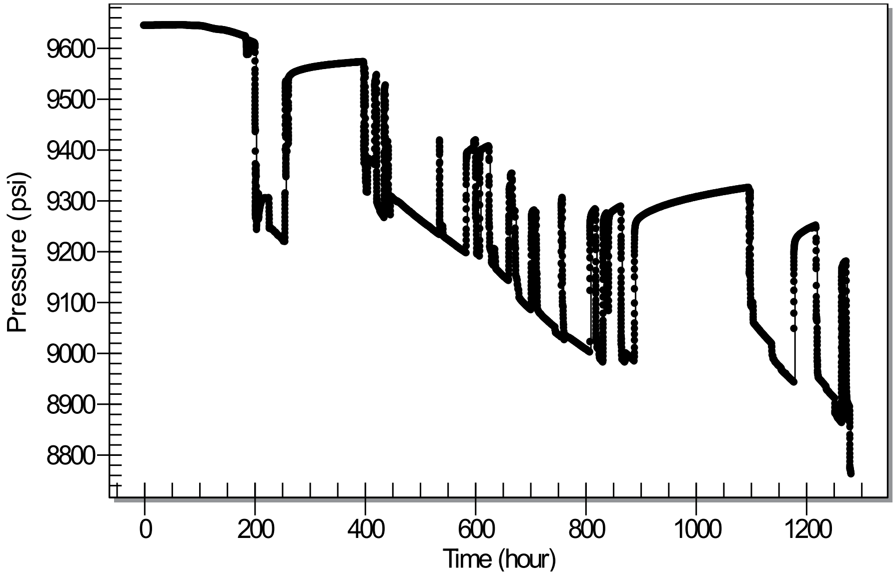


Figure 6.29: *Pressure signal from North Sea well used in transient detection.*

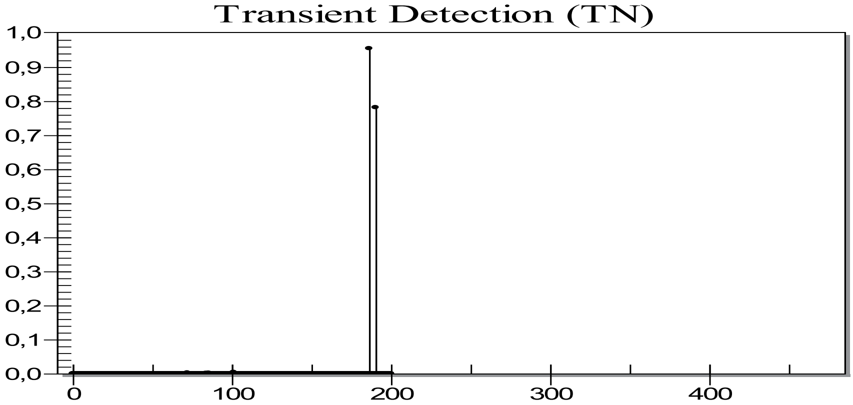


Figure 6.30: *Detected transients after 200 hours of pressure measurements.*

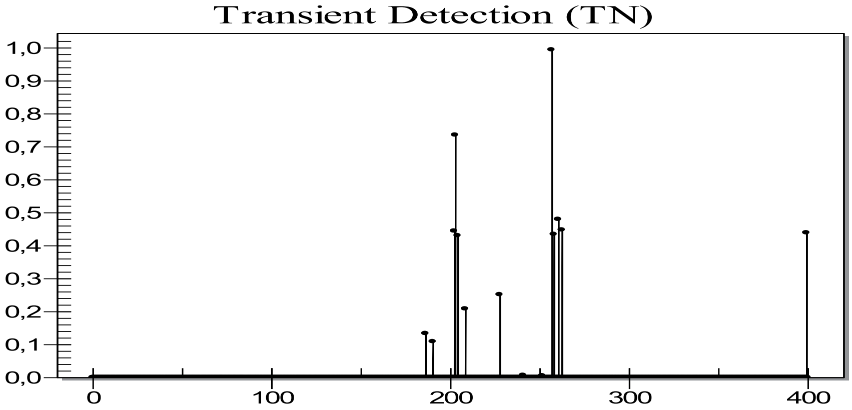


Figure 6.31: *Detected transients after 400 hours of pressure measurements.*

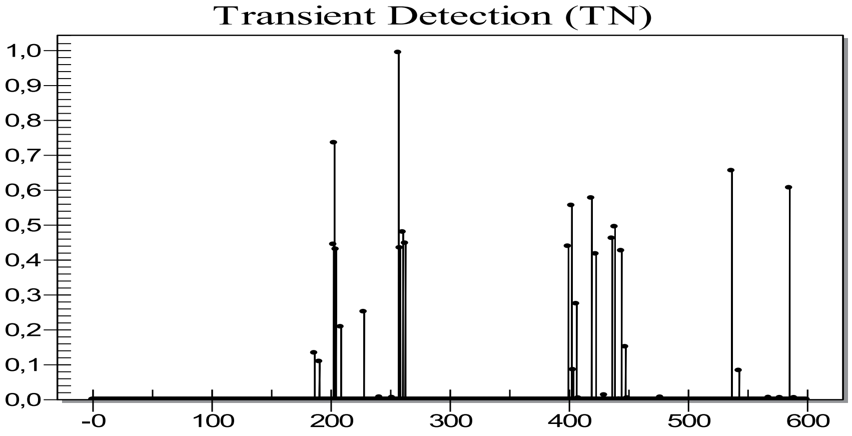


Figure 6.32: *Detected transients after 600 hours of pressure measurements.*

6.4 Summary

We have shown how the real-time wavelet transform might be used to continuously update wavelet transformed data with a minimum of computational effort. The real-time wavelet transform is performed continuously with the measured signals through a window approach in which the boundary problems in the wavelet transform of finite signals is eliminated. The transformed data can be used for real-time wavelet de-noising to ensure continuously de-noised signals ready for analysis at any time. The methods and selections for an automatic wavelet filter were determined in chapter 5 for use with typical signals encountered in the E&P industry as pressure and rate signals.

Pattern recognition is a promising way for detection of transients requiring no user input. A new method for transient detection is proposed based upon pattern recognition. Predefined images for pressure build-ups and pressure drawdowns are provided. In addition, small transients assumed to origin from underlying noise structures not adequately removed in de-noising, are eliminated by requiring the transients to have a certain magnitude. A transient is detected if and only if the current transient image match one of the predefined images and the magnitude of the transient (in terms of pressure change) is at least three time the estimated noise level and the standard deviation in the image window considered is at least 0.1 of the total signal measurements standard deviation. The magnitude of the detected transients relative to the largest detected transient (in terms of pressure difference) is used for visualization. A window size of 20 is applied in transient detection.

In process monitoring, noise level estimation, signal filtering, signal trends and transient detection is continuously estimated. The noise level is continuously updated through a median noise estimator and a sorted list of wavelet detail coefficients at the finest resolution level. A smaller window size is recommended for real-time noise estimation than for real-time signal filtering. This since a rapid detection of possible changes in noise level is desired. Hence a window size of 200 is recommended for real-time noise estimation.

The window size in real-time signal filtering should be chosen as large as possible, but this will of course be limited by computer processing facilities. In the author's experience, at least 4096 measurements should be used and will be sufficient for good results. However, it might be beneficial to include additional measurements if the computational capacity is sufficient.

In real-time transient detection, the requisite for de-noised signal measurements must be removed. This since there will be a significant delay in real-time signal de-noising. If a noisy signal is applied in transient detection, some additional and "small" transients might be missed. Hence we have to compromise between a rapid detection of transients and an accurate detection of "small" transients. We believe that in the context of real-time pressure monitoring, a rapid detection of transients is more important. Thus in the context of real-time process monitoring, noisy signal measurements will be used. However, outliers are removed from the signal measurement before transient detection.

The real-time wavelet transform can be applied for investigation of underlying signal trends located in the approximate signals at lower decomposition levels. Decomposition level 6 and 12 is recommended. Alarms can be added to warn the operator if significant deviation from trends occurs. This will help the operator to easily and efficiently detect changes in the wells considered.

Real-time Signal Filtering and Process Monitoring

We have in this chapter described a module for real-time signal filtering and process monitoring. The aim was automatic signal filtering and process monitoring of signals measured in real-time. This can be used as an input to different kinds of analyses. One such analysis is well-testing. We will in the next chapter describe automatic well-test analysis.

7 Automatic Well-Test Analysis

In well-testing we analyze the pressure response to rate changes in a specific well. As discussed in chapter 3, we prefer to analyze the pressure response during well shut-ins which is referred to as build-up tests. Typical information required for description and understanding of the flow properties of the reservoir (e.g. permeability) and the flow properties of the well (e.g. skin factor) is obtained in well-test analysis. This information is valuable for the engineers for increased understanding of the well and reservoir considered and to ultimately determine and optimize the recovery of oil and gas.

Well-testing is however quite a time consuming exercise that would require significant attention from the engineers if utilized with all pressure and rate measurements available from wells equipped with permanent high resolution gauges. Due to limited engineering time, frequently only a limited number of well-test analyses are completed, thus utilizing only a limited number of the measurements available. We will in this section describe an approach which utilizes all available pressure measurements during shut-in periods and minimizes the engineering time associated by repeatedly performing well-test analyses. As previously explained, we will limit ourselves to only consider well-tests from shut-in periods due to the poor results usually obtained from well-tests during flow periods, where the poor results are caused by inaccurate rate measurements.

The idea behind the proposed method is that once a manual well-test analysis has been performed for a shut-in period, automatic well-test analyses can be performed for later automatically detected shut-ins. By *a manual well-test analysis* is here meant a well-test where the user performs the four steps described for modern well-test analysis in section *3.8,* including data preparation, model identification, non-linear regression analysis and confidence interval estimation. By *an automatic well-test analysis* is meant a well-test analysis performed without user interaction and where the well-test analysis is initialized with the same settings (e.g. same model selection, same method for data filtering etc.) as a previously performed manual well-test analysis. These settings will, as stated in the introduction, be referred to as *run specifications*. Hence, one manual well-test is required for determination of the run specifications required in automatic well-test analysis.

In automatic well-test analysis, one of the main purposes will be ***to detect changes in the reservoir properties*** previously estimated in a manual well-test analysis. If no changes are detected, the other main purpose is ***to obtain smaller confidence intervals of the initial estimated reservoir properties from the manual analysis,*** accomplished by performing additional automatic well-test analyses**.**

By performing well-testing on arbitrary detected shut-ins, we obtain time series of estimated parameters such as permeability and skin factor. These time-series will help us to detect possible changes in the well that might affect the fluid flow and the well performance. For instance, a decrease in permeability and/or an increase in the skin factor will reduce the fluid flow and/or increase the pressure drop in the well. A decreased flow rate in the well will have economical aspects that might need to be accounted for in the overall asset management, while an increased pressure drop might cause operational difficulties such as decreased lift potential in the well.

Ultimately actions might be required in order to deal with these problems; ***however the key element in this work will be proper detection and identification of possible changes in well and reservoir properties that might be identified by automatic well-testing.***

In non-linear regression analysis, it is possible to match pressures during the individual shut-in periods (as we propose to do) or to match the entire pressure history (during both flow and no-flow periods). Non-linear regression analysis of the individual shut-in periods allows us to estimate time-series of parameters which we propose to use for detection and identification of possible changes in well and reservoir properties. If we had chosen to apply the entire pressure history in the well-test regression analysis, only average parameter values would be estimated, i.e. we would not be able to detect possible changes in well and reservoir properties.

Athichanagorn et al. (5) proposed a window approach for use in well-test analysis for calculation of well and reservoir properties, and presented the results as time-series. Also Lee(22), Coludrovich et al. (25), Airlie et al. (23) and Shi-Yi et al. (15) have published work regarding this. Thus there is nothing new in presenting the results from repeatedly well-test analyses as time series. However, the proposed methodology for how to perform repeated automatic well-test analyses contains several new elements. ***The proposed automatic statistical evaluation of the results from repeated well-test analyses is new***. This will help the engineers to spend their time on those well-tests where changes are detected. ***In addition, the proposed methods for automatic signal filtering and process monitoring is new, providing the automatic well-test analyses with continuously updated and filtered data.*** The methods proposed for automatic signal filtering and process monitoring were described in chapter 5 and 6.

Some authors have also proposed automatic model selection in order to do a fully automatic well-test analysis(24; 80), but this would also lead to uncertainties as to whether the correct model is chosen. We believe that the analysis will be improved if an engineer does a setup in advance in order to determine the proper model. It would for instance be unwise not to use the engineer's knowledge of well models (like slanted wells, partial penetrated wells etc.), reservoir models (like double layered reservoir or dual porosity reservoir etc.) and boundary models if possible.

We will start by providing a brief summary of the steps in the proposed method for automatic well-test analysis. The proposed method includes four steps; *setup, data preparation, well-test analysis and possible user intervention*. We will describe the data preparation module, the operations applied in it and their sequence. We have chosen not to have a separate section for the setup module, but will instead describe the run specifications along with the sequence of operations in data preparation. We will continue by describing the well-test module. This module includes both the non-linear regression analysis and a method for evaluation of the results obtained. In the result evaluation, we have chosen to apply hypothesis testing for detection of possible changes in the time series of estimated parameters. As for the data preparation module, any run specifications for the well-test module will be discussed simultaneously. Finally, several examples utilizing the proposed method are presented.

The results presented in this chapter are previously published in Olsen et al. [(3).]

7.1 The Proposed Method

The proposed approach for automatic well-test analysis is a four step procedure as illustrated in ***Figure** 7.1*.

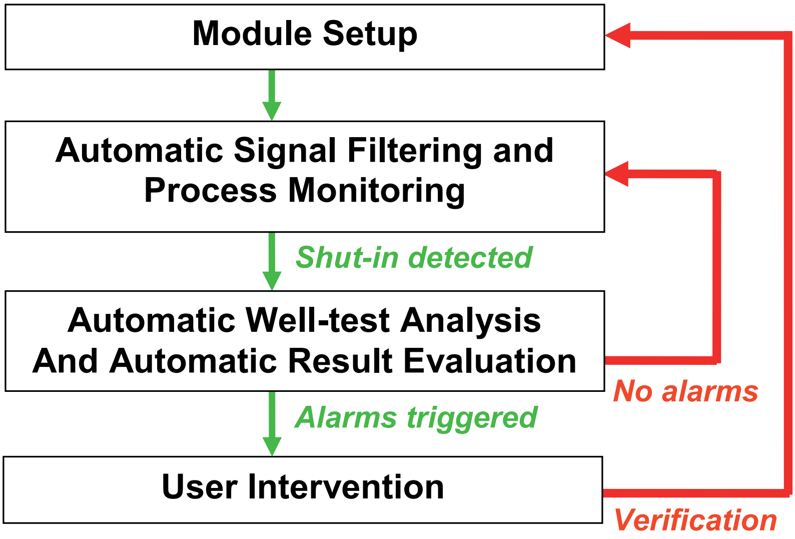

Module Setup
Automatic Signal Filtering and Process Monitoring
Shut-in detected
Automatic Well-test Analysis And Automatic Result Evaluation
No alarms
Alarms triggered
User Intervention
Verification

***Figure** 7.1: Illustration of the four main steps in the proposed automatic well-test analysis.*

Setup

The first step is the setup. The purpose of the setup module is to provide the run specifications for how to perform the operations in each subsequent module. This step includes the setup for both *data preparation and well-test analysis*. In the data preparation setup, we specify those run specifications required to obtain continuously filtered signals and continuously detected events. This includes run specifications on how the pressure signal should be de-noised, how the number of pressure measurements should be reduced, how to detect events such as well shut-ins etc. The operations used in data preparation along with recommended run specifications were discussed and determine through chapter 5 and 6. The idea was that for typical signals encountered in the E&P industry, it was possible to determine methods and selections that would allow automatic signal filtering.

The main objective of the well-test setup is to determine the mathematical model to use in the automatic non-linear regression analysis. In addition, we need to choose which parameters to consider as unknowns, i.e. which parameter to tune in the non-linear regression analysis. Further, we need to provide an initial estimate of the well and reservoir parameters considered as unknowns. As previously discussed in chapter 3, good initial estimates of the parameters will help to improve the convergence in the non-linear regression analysis. Finally, we need to determine the significance level to apply in the hypothesis testing for detection of changes in the estimated time series of parameters.

The setup module also includes elements such as data source connections, i.e. how and from where to obtain the latest available measurements. Usually we will have to connect to a database, often found remotely, to continuously receive the latest measurements.

This *first step is only done once* and provides the initial run specifications for the automatic data preparation and well-test analysis. The next steps, step two and three, are *executed each time a new shut-in is detected.*

Automatic Signal Filtering and Process Monitoring

The ***second step*** in the proposed method is automatic signal filtering and process monitoring. This step includes a sequence of different signal filtering and process monitoring operations where the run specifications for this module are defined in the setup module. The purpose of this module is to provide the well-test module with continuously filtered signals and to detect events such as shut-ins that subsequently will trigger automatic well-tests.

Automatic Well-test Analysis and Automatic Result Evaluation

For each detected shut-in, an automatic well-test analysis is performed including automatic result evaluation. This is the ***third step*** in the proposed method. This is done by non-linear regression analysis and a new estimate of the unknown model parameters along with their confidence intervals is obtained. This will ultimately provide us with time-series of the estimated parameters. In the evaluation of the results from the repeated well-test analyses, the main purposes are to detect changes in the process variables such as skin factor and permeability or to obtain smaller confidence intervals of the initially estimated well and reservoir properties provided in the setup section.

Possible User Interaction

If changes are detected in the estimated parameters from the automatic well-test analyses, an alarm is triggered and the engineer is recommended to verify the results. If the engineer in the verification process confirms a parameter change, this might require some sort of well intervention. This is the ***fourth step*** of the proposed method. If for instance a change in the skin factor is detected (for instance caused by scaling), this will cause the flow rates of the well to decrease. In some cases, the decrease in flow rates might be so severe that the engineer might determine that a well work-over is desired.

We have in this section briefly described the four main steps in the proposed method. We will in the next section provide additional details of the proposed method based upon the more detailed illustration as shown in *Figure 7.2*. However, note that we in the subsequent section will provide details about the sequence of operations and simultaneously provide run specifications for the individual operations, i.e. we do not have a separate section for the module setup.

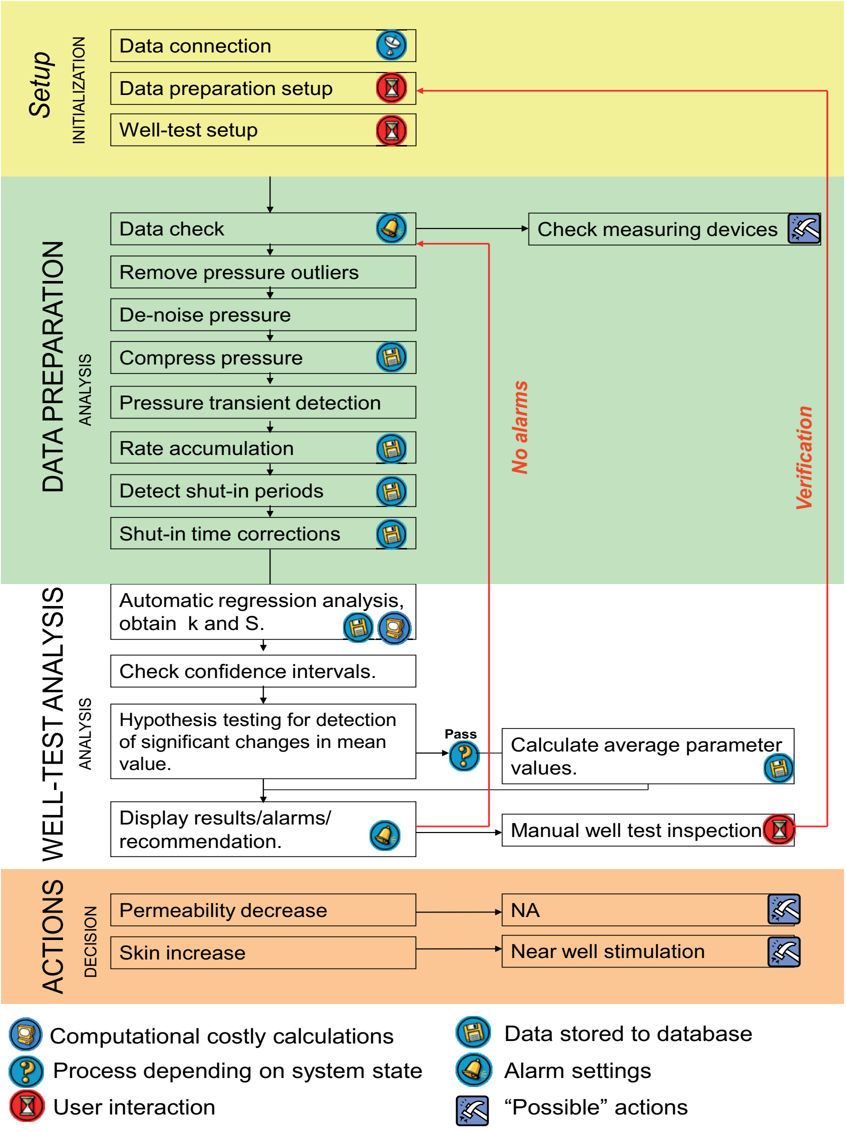


Figure 7.2: *Detailed illustration of the proposed method for automatic well-test analysis. Note that the "recommended" actions are only meant as examples of possible actions.*

7.2 Signal Filtering and Process Monitoring

The data preparation module consists of several operations where the purpose is both signal filtering and process monitoring. In data preparation, we provide a sequence of operations for how typically signals as pressure and rate should be prepared for well-test analysis. All operations are described in the sequence they are performed and as illustrated in *Figure 7.2*.

Data Check

For the moment, the only data check performed is for a changing noise level. A changing noise level in the rate or pressure measurements might indicate troubles with the measuring devices or that the measurements are influenced by a time dependent disturbance. This will increase the uncertainty as to how well the measurements reflect the true process state and this need to be taken into consideration when performing a well-test.

The changing noise level is detected with a noise estimator from wavelet analysis called RMAD and given by *eq. (5.7)*. The wavelets Spline39 and Spline4246 are applied with pressure and rate signals respectively. The noise estimator utilizes a window size of 200 of the latest measurements. The noise estimation is continuously performed for each new measurement. The recommended window size for noise estimation was selected in chapter 6.3.1.

Outliers

Outliers can introduce errors in the non-linear regression analysis. In addition, the presence of outliers might cause the proposed pressure transient detection scheme to fail. Outlier removal is the second step in data preparation in *Figure 7.2.* Outliers are removed with a median filter with window size 19, i.e. we can remove up to nine successive outliers. This was a choice made in chapter 5.2. We remove outliers from the pressure measurements only as a different approach is chosen for rate signals. The noise level estimate required in the outlier filter is obtained from the RMAD noise estimator given by *eq. (5.7).* Wavelet Spline39 is used with pressure signals. A window size of 200 is applied in estimation of the required noise level. The outlier removal is performed continuously for each new measurement. However, we cannot remove outliers from the latest measurements since we do not know if these are actual outliers or just the start of a new transient. Consequently we apply a delay in outlier removal equal the number of successive outliers that can be removed. Thus when utilizing a window size of 19, the nine latest measurements are never included in outlier removal.

Pressure Signal De-noising

De-noising should not be used as a part of automatic well-test analysis unless it is to facilitate data reduction or model identification. De-noising signals for use in non-linear regression analysis might introduce errors and provides no improvements in the analysis. This will typically happen if de-noising removes more than just noise from the signal measurements and over-smoothes the signals by removing some of the signal information. De-noising of pressure signals is thus only utilized if we have a large number of measurements and data reduction is required. Then proper de-noising of the pressure measurements will be critical. De-noising of pressure signals is accomplished by Waveshrink with the selections summarized in chapter 5.6. De-noising is performed continuously as described in chapter 6.

In addition, when it comes to model recognition/data visualization, de-noising is an important and critical tool in order to achieve this. Thus de-noising will also have an important role in the well-test setup as part of model identification.

Data Reduction Pressure Signal

The fourth step in the automatic signal filtering and process monitoring is data reduction. The method for data reduction was described in chapter 5.4. The aim in data reduction is to obtain a smaller subset of the measured signals but with approximately the same information content. The reason for data reduction is to increase the computational speed of the later well-test analysis. However, we would like to obtain the same results as if the original number of signal measurements were used. The rate measurements will be handled at a later stage, and hence we at this stage will consider data reduction of the pressure measurements only. In general, data reduction should be applied if and only if the well-test module cannot handle the number of pressure measurements available in a computationally efficient manner. If a signal is sampled only every minute during a week, it sums up to approximately 10.000 signal measurements. In addition, we would like to have pressure measurements with higher resolution than this in the beginning of the pressure build-up, as for instance every five or every ten second. Thus a great number of pressure measurements can be available and data reduction is therefore enabled by default. However, in some cases and for different reasons, signal sampling rates might be poor. In such cases, the user should consider disabling data reduction during setup. We will choose to do this for two of the cases we have considered in this work, namely in case 3 and case 4 as will be discussed later. Data reduction is performed continuously as described in chapter 6.

Transient Detection

A pattern recognition approach is used for pressure transient detection. The advantage of using pattern recognition is that it is easy and computationally efficient to use. The method used for transient detection was described in chapter 6.2. A window size of 20 is used in the pattern recognition, i.e. we look for transient patterns within the 20 latest pressure measurements. As discussed in chapter 6.3, outliers should be removed before transient detection, and thus a delay in transient detection equal to the window size of the outlier filter is required. Thus the 19 latest (or newest) pressure measurements are not included in transient detection. Transient detection is updated for each new measurement. This is the fifth step in data preparation in *Figure 7.2.*

Rate Accumulation

The number of rate measurements might be reduced and filtered independently of the pressure measurements. This was illustrated in case 1 in chapter 5.5. Although rate signal filtering has been shown to work well, errors might occur. Our investigations show that it is beneficial to simply calculate the average rates between the detected pressure transients. This simple and efficient approach synchronizes the pressure and rate measurements, and in addition, the number of rate measurements is reduced. Thus there is no need for outlier removal or de-noising in the rate measurements. This is step six of the proposed method shown in *Figure 7.2.*

Shut-in Detection

Next, every period with zero rates are detected as shut-in periods and will be analyzed as build-up tests by the well-test module. This is step seven of the proposed method shown in *Figure 7.2***.**

Shut-in Time Correction

Sometimes the pressure is not measured at the exact shut-in time, i.e. only pressure measurements before and after the "true" shut-in is available. In derivative plots, this might be identified from the deviation of the unit slope line in the beginning of the plot, either upwards or downwards [(42)]. This effect can easily be removed by "adjusting" the shut-in time. The time where the true shut-in occurs is found from the intersection of the straight lines of the two measurements before the "true" shut-in and by the two measurements after the "true" shut-in. The shut-ins were detected in the previous step and this step corrects the detected shut-ins times. This is step eight in data preparation. Next the well-test analysis module will be described.

7.3 Well-Test Analysis

The automatic well-test module uses the filtered signals together with the detected shut-ins to perform automatic well-test analysis. *The well-test module is executed each time the end of a new shut-in is detected.* The well-test module consists of two main elements, the non-linear regression analysis and the statistical evaluation of the results.

Automatic Regression Analysis

In non-linear regression analysis we seek to minimize the difference between the measured pressures and the pressures calculated from a mathematical model assumed to adequately describe the physical process we consider, i.e. fluid withdrawal from a porous medium (reservoir) through a well. The method for non-linear regression analysis is the Levenberg-Marquardt method described in chapter 3. The minimization is accomplished by tuning the process variables considered as unknown. Which parameters to consider as unknowns are determined in the well-test setup by a manual well-test analysis. The well, reservoir and boundary models to use in non-linear regression analysis are determined from derivative plots in the manual well-test analysis, while the initial estimate of the unknown parameters are determined from the manual performed non-linear regression analysis. As described in chapter 3, it is important to provide the non-linear regression analysis with good initial estimates of the unknown parameters such as skin factor and permeability. If the initial parameter estimates are far away from the "true" parameters, the non-linear regression analysis might fail due to convergence problems.

In well-testing, the unknown parameters are usually considered to be the skin factor and permeability, but might in addition include other well and reservoir properties such as wellbore storage, drainage area and reservoir heterogeneities. The parameters that might be estimated will depend upon the duration of the shut-in periods considered. For short shut-ins, typically limited to a couple of hours, only well and reservoir properties (such as permeability, skin factor, etc.) might be estimated. For longer shut-ins, typically lasting for days or maybe weeks, boundary properties (such as size and shape of drainage area) might be estimated as well. We expect that most shut-ins considered will be of limited duration from where only well and reservoir properties might be estimated and that only a limited number of the shut-ins will allow estimation of boundary properties. However, note that the non-linear regression analysis is always performed utilizing pressures for the entire shut-in period applying the model identified in the derivative analysis. This means that if the selected model includes boundary properties, these are estimated for each detected shut-in period, even the shut-ins of short duration. Since none of the measured pressures during short shut-ins are influenced by boundary effects, boundary properties will in these cases be estimated with large uncertainties resulting in large confidence intervals. For longer shut-ins, the pressures will be influenced by reservoir boundaries, and boundary properties might be estimated with small uncertainties and small confidence intervals. ***Since we assume that most shut-ins will be of short duration, we will focus on the estimation of well and reservoir properties such as skin factor and permeability, which can be estimated even from short shut-ins periods.***

If a parameter is estimated with a small confidence interval in the non-linear regressions analysis, the result is usually considered good and high confidence can be given to the result. On the other hand, if the parameter is estimated with a large confidence interval, the result is usually considered poor, and little confidence should be given to the result. If for instance, the permeability is estimated to 100 ± 5 mD, this is usually considered a good result, while an estimate of the permeability to 100 ± 90 mD, is usually considered a poor result.

We choose to eliminate estimated parameters with large confidence intervals from the time-series of estimated parameters. The reason for this is simple; little confidence can be given to the results from well-test analysis if the estimated parameters have large confidence intervals. Although we assume that we are able to estimate permeability and skin factor from most shut-in periods, even from short shut-in periods, there are still several reasons why a non-linear regression analysis can fail. This might for instance be that the shut-in period is *very* short, few available signal measurements (poor signal sampling rate) during the shut-in period, convergence failure in non-linear regressions analysis etc. We could choose to check the duration of the shut-in periods and the number of available measurements during the shut-in periods to eliminate shut-in periods that were likely to give poor results. However, it appears more convenient to simply exclude estimated parameters with high confidence intervals from the time-series.

In the book by Horne[46], the following recommendation is given; if the confidence interval for the permeability is maximum ±10% of the estimated permeability value, this is considered a good result, otherwise not. This means that if the estimated permeability is 100 mD, the confidence interval should be less than ±10 mD to consider the result reliable. Similar limits are provided for other estimated parameters such as skin factor, initial pressure etc. However, in our cases, the recommended limits given by Horne appear to be too strict. In this work, we consequently choose to apply broader limits. We chose to double the recommendations given by Horne and these new limits are summarized in *Table 7.1*. The reason why we increased the limits was since many of the performed well-tests in our work had confidence intervals just above the limits given by Horne, and we did find it beneficial to also include these in our analysis.

Table 7.1: *Tolerance limits used in automatic non-linear regression analysis.*

	Tolerance
Permeability	±20%
Skin	±2

Note that the tolerance limits in *Table 7.1* is only applied with a single automatic well-test analysis. Later we will calculate average parameter values based upon parameter estimated from several automatic well-test analyses, and in these cases, we will apply the same tolerance limits as recommended by Horne.

The automatic well-test analysis is performed each time a new shut-in is detected and this will provide us with new values of the parameters considered as unknowns for each detected shut-in. We will therefore obtain time series of the unknown parameters and hypothesis testing will be used to detect possible changes in these time series.

Statistical Result Evaluation

To evaluate the results from automatic well-test analysis, we apply hypothesis testing for detection of *significant changes* in the time series of estimated parameters. If a significant change is detected, a notification will be send to the engineer. If no significant changes are detected, average parameter values with confidence intervals are calculated. Since this average parameter value is calculated based upon many “measurements” of the parameter considered, it is assumed to represent an improved estimate of the parameter considered. In addition, we expect this average parameter value to have a smaller confidence interval than individual estimated parameter values.

Automatic well-test analysis is performed each time a new shut-in period is detected. If *N* different shut-ins are detected, *N* non-linear regression analyses are performed and N estimated parameter values are obtained. The estimated parameters from the non-linear regression analysis will hence form a time series.

Each estimated parameters from the non-linear regression analysis will be associated with a mean value and a variance obtained from the non-linear regression analysis. Based upon the variance, a confidence interval can be constructed as shown in chapter 3. Each of the values estimated from the non-linear regression analysis is assumed to be independent of the other values estimated.

The subject will be to determine whether the last estimated parameter, with index *N,* have a significant different value then the rest of the estimated parameters (*1,…, N-1*) in terms of mean value. To investigate this, hypothesis testing is ideal(81; 82). In hypothesis testing we try to answer the question; assume that the zero hypothesis is true, what is the probability of observing a value that is at least as extreme as the value that we actually did observe? (82) To determine whether or not to reject the zero hypothesis, a significance level is chosen. The significance level tells us the probability for two samples to have the measured difference and still have the same mean, i.e. that the zero hypothesis is true. The chosen significance level in hypothesis testing is often set to 5%, also in this work. This means that if a value is observed and it is less than 5% probability of observing such a value, the zero hypothesis is rejected.

We would like to test the difference in mean value by the following hypothesis;

$$\text{(7.1)} \qquad \mathrm{H}_0\,;\ \bar{x}_A = \bar{x}_B$$

$$\text{(7.2)} \qquad \mathrm{H}_1\,;\ \bar{x}_A \neq \bar{x}_B$$

where x_A and x_B are the two different populations considered and where population x_A contains the estimated parameters with index (*1,….., N-1*) and population x_B contains the single estimated parameter with index *N*. H_0 is referred to as the zero hypothesis while H_1 is the alternative hypothesis. To test the zero hypothesis, we consider the estimated parameters in the time series as being from two independent distributions. The average value of the *N-1* estimated parameters in population x_A is given as;

$$\text{(7.3)} \qquad \bar{X} = \frac{\sum_{i=1}^{N-1} \bar{x}_i}{N-1}$$

However, to obtain an average parameter estimate like this, we need to assume that each parameter have been estimated based upon approximately the same number of signal measurements, i.e. each non-linear regression analysis is performed with approximately the same number of signal measurements. We believe that this assumption will be accurate enough once the number of estimated parameters in the time series increases.

To estimate the variance of the *N-1* measurements assumed independent, we need to sum up all the measured variances and divide it with the squared number of samples in the distribution, *N-1*;

$$\text{(7.4)} \qquad \operatorname{var}\left(\bar{X}\right)=\frac{\sum_{i=1}^{N-1}\operatorname{var}\left(\bar{x}_i\right)}{(N-1)^2}$$

The variance for each estimated parameter is obtained from the diagonal elements in the co-variance matrix given by *eq. (3.42).*

One can test if the two distributions have different mean values by performing a student t-test [51; 52]. The *t* in the student t-test with N_A-N_B-*2* degrees of freedom is given as;

$$\text{(7.5)} \qquad t=\frac{\bar{x}_A-\bar{x}_B}{s_D}$$

and where s_D is given as;

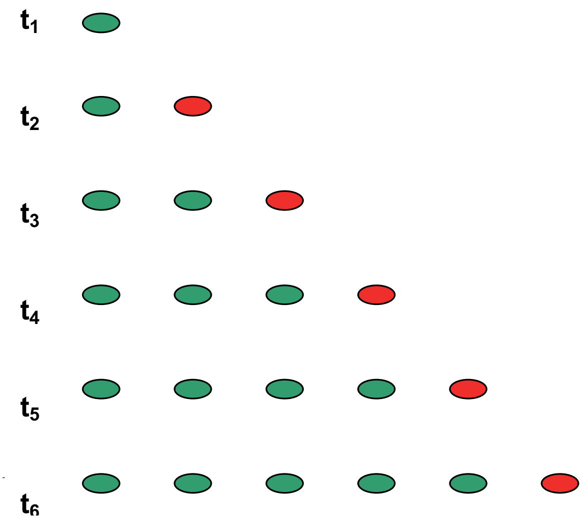
$$\text{(7.6)} \qquad s_D=\sqrt{\frac{\sum_{i\in A}()^2+\sum_{i\in B}(x_i-\bar{x}_B)^2}{N_A+N_B-2}\left(\frac{1}{N_A}+\frac{1}{N_B}\right)}$$

and where N_A is the number if samples in the first distribution and N_B in the number of samples in the second distribution.

The student t-test probabilities are finally found from lookup table or estimated trough a computer program[52] with the given number of samples and degrees of freedom. The zero hypothesis is rejected if the probability from the student t-test is lower than the chosen significance level.

Once a significant change in the time series is detected, the automatic well-test module continues by performing two different hypothesis tests. One test as described above containing two data sets with *N-1* estimated parameters and one estimated parameter respectively. The other test splits the data set in two from where the *first significant change was detected*, and continues by adding new estimated parameters to the second population (initially consisting of only the last estimated parameter). Thus, we would like see if the change in the estimated parameters still is significant when new estimated parameters are added to the second population.

The two different tests are illustrated in *Figure 7.3* and *Figure 7.4.* These figures illustrate the two different populations tested at six different times, t_1 – t_6, where each time will correspond to the time when a new estimated parameter is added to the time series. A significant change in the estimated parameters is assumed to occur at time t_4. In *Figure 7.3* only the last estimated parameter is considered as population B and the hypothesis, whether the last estimated parameter has the same mean value as the rest of the estimated parameters, is tested. Once a significant change is detected in the first test, in this case at time t_4, a second hypothesis test is performed as shown in *Figure 7.4*. In this test, all new estimated parameters are added to population B to test if the difference between the two populations mean value still is significant.



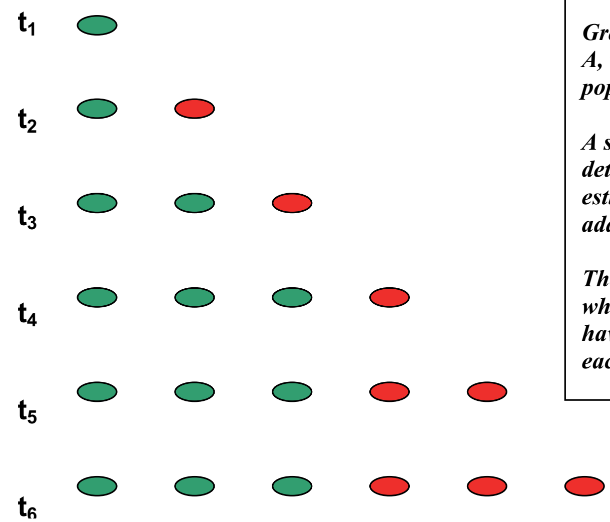
Description test 1

Green circles are population A, while red circles are population B.

A significant change is detected at time t_4. Only the last estimated parameter is considered as population B.

The hypothesis test checks whether population A and B have the same mean value at each time step.

Figure 7.3: *Hypothesis test #1 in well-test result evaluation.*



Description test 2

Green circles are population A, while red circles are population B.

A significant change is detected at time t_4 and all new estimated parameters are added to population B.

The hypothesis test checks whether population A and B have the same mean value at each time step.

Figure 7.4: *Hypothesis test #2 in well-test result evaluation.*

7.4 Applications

We will show four cases to illustrate the use of the automatic well-test module; two cases with synthetic data and two cases with real data. By a *synthetic data case* is meant a case where the pressure response in the well is generated based upon a specified rate history and a specified mathematical model. Noise and outliers are added to the pressure and rate measurements, although the pressure measurements are generated based upon the noise free rate history. A detailed description of signal generation for use in data filtering was given in chapter 5.1, and signal generation for use in automatic well-test analysis is identical. The reason why we consider synthetic data cases is since the true well and reservoir properties are known and consequently we can evaluate the performance of the automatic well-test analysis since we know the "true" answers. By a *real data case* is meant a case where the pressure and rate measurements are obtained from a real oil and/or gas well. The cases with real data were performed to validate that the proposed method was applicable with real data. The latter will typically include challenges as a large number of measurements, noisy measurements, difficulties/uncertainty in well-test model identification etc.

We will start by providing details of the evaluation criteria used to determine the performance of the proposed automatic well-test analysis. Then two cases with synthetic data are presented followed by two cases with real data from a North Sea gas/condensate reservoir.

7.4.1 Performance Analysis

In this section we will describe how we evaluated the performance of the automatic well-test analysis in the cases we considered. We evaluated the proposed method with both synthetic and real data, the later obtained from wells in a North Sea gas/condensate reservoir.

The synthetic cases were completed to verify that the proposed automatic well-test analysis works as required when synthetic pressure and rate measurements are applied. We are then required to choose some success criteria for the performance of the proposed method. In the performance evaluation, we chose to only consider the final calculated average parameter value in the time series. ***If the true parameter value is within the confidence interval of the final estimated average parameter value obtained from automatic well-test analyses, and if the estimated confidence interval is small, the results will be considered a success***. If the estimated confidence interval is less than ± 10% of the estimated permeability value, we choose to consider this a small confidence interval, and the result a success. For the skin factor, we choose to consider the confidence interval small and the result a success, if the confidence interval is less than +/- 1. Note that these confidence intervals are half the confidence intervals used in the evaluation of single automatic well-tests and the same confidence intervals as recommended by Horne [(46)].

Consider the time series of estimated average skin factors as shown in *Figure 7.5.* The last average skin factor value to the right in this figure is estimated to 3.0 ± 0.1. The "true" skin factor of 3 in within the estimated confidence interval and the confidence interval is definitely within the range of an acceptable confidence interval, chosen above to ± 1. This result is hence considered a success.

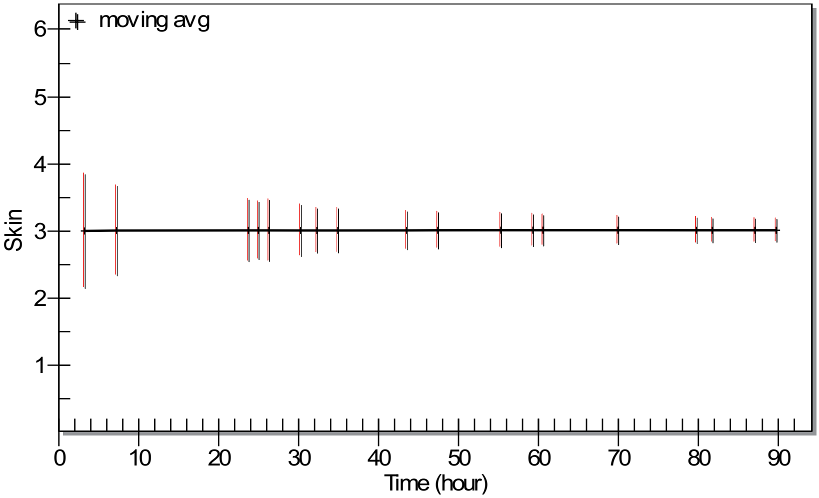


Figure 7.5: *Average estimated skin factors with confidence intervals.*

If we consider a test case with constant parameters, i.e. they do not change with time, the results from the hypothesis testing are evidently considered a success if no changes is detected. On the other hand, if the test case is designed with a changing parameter value, the results from hypothesis testing is evidently considered a success if the change is detected with the selected significance level.

To evaluate the performance in the cases with real data, we choose to compare the results from the automatic well-test analyses with those obtained by performing *manual well-test analyses*. This since we only know the true parameter values when working with synthetic data. In manual well-test analysis, data is carefully prepared and filtered. A manual inspection of the filtered signals is done and clearly erroneous measurements/trends not removed by the filter are excluded from the analysis. The shut-in periods are determined by visual inspections of the pressure and rate signals. Rate and pressure measurements are carefully synchronized if datum-shifting effects are present. Non-linear regression analysis is performed for each detected shut-in period and we determine by visual inspection if the non-linear regression analysis is successful. This is done by examining if the match between the measured and calculated pressure is adequate. Hence we manually perform each step in the well-test analysis and carefully evaluate the results. In the automatic well-test analysis, the scheme in *Figure 7.2* is followed. Besides from the initial run specifications, the proposed method is performed without any user interaction.

If the results from the manual tests and the results from the automatic tests are consistent, this will be a good indication that the automatic well-test analysis works well, at least in the cases we have considered. If no changes are detected in the estimated parameters by hypothesis testing, ***the results are considered a success if the last average parameter values from the time series in the manual and automatic***

well-test analyses have over-lapping confidence intervals. If a change in any of the estimated parameters are detected, the results will be considered a success if the last average parameter values from the manual analyses and the last average parameter values from the automatic analyses, have overlapping confidence intervals. In addition, the last estimated average parameter values from the manual and the automatic analyses before the detected change should have overlapping confidence intervals to consider the result a success.

7.4.2 Case 1

The first case is part of the verification process for the proposed automatic well-test analysis. We apply a synthetic generated pressure signal based upon a known reservoir model and a specified rate history. The pressure signal is generated based upon constant well and reservoir properties. All known fluid and reservoir properties are applied in the automatic well-test module, including our knowledge of which model to use. Hence the purpose of this case is to obtain continuous estimates of the permeability and skin factor close to the true known values and with small confidence intervals. The proposed hypothesis testing for detection of changes in the estimated time series of permeability and skin factor should not reject the zero hypothesis; that the permeability and the skin factor are constants.

The reservoir considered is a homogenous reservoir with wellbore storage and skin effects as described in chapter 3. The rate history was generated based upon a random number generator as described in chapter 5.1 consisting of 10.000 uniformly spaced rate measurements. The mobile phase is assumed to be oil only. The pressure was generated based upon the specified rate history and the homogenous reservoir model given by *eq. (3.29), eq. (3.28)* and *eq. (3.22)*. Other relevant fluid and reservoir properties are listed below;

Reservoir height: 40 ft
Total compressibility: 5.2e-4 1/psi
Porosity: 0.2
Permeability: 60 mD
Skin factor: 3
Oil b-factor: 1.2 rb/stb
Viscosity: 0.6 cp
Well radius: 0.35 ft
Wellbore storage constant: 0.6 std/psi
Initial pressure: 9000 psi

Three standard deviation white noise and random outliers were added to the pressure and rate measurements (although the pressure measurements are generated based upon the noise free rate measurements). The resulting pressure and rate history is shown in *Figure 7.6*. The generated rate history, the fluid properties and the reservoir properties were selected such that they should be representative for what we believe might be encountered in a real oil well.

Before we can run automatic well-test analyses, we need to perform the required setup. The methods and settings for the automatic signal filtering and process monitoring modules were given in chapter 5 and 6 and summarized for automatic well-test analysis in section 7.2. Further we apply all known fluid and reservoir properties in the well-test module, including which well, reservoir and boundary model to use.

The proposed method is then applied with the synthetic pressure and rate signals. First the filter is used for outlier removal, de-noising and data reduction in the pressure signal. Pressure transients are detected and average rates between the detected pressure transients are calculated. Next, shut-in periods are detected and corrected for possible errors in the exact shut-in time. Non-linear regression analysis is performed and an estimate of the skin factors and permeabilities with confidence intervals is obtained for each detected shut-in period. Hypothesis testing is performed on the time series of skin factors and permeabilities to test if any significant changes are present. If no changes are detected, average parameter values are calculated.

The automatic well-test analysis works well for the case considered. The estimated skin factor and permeability are very close to the original values; skin factor of 3 and permeability of 60 mD. The estimated confidence intervals are in addition small. This is shown in *Figure 7.7* and *Figure 7.8.* The hypothesis testing did not detect any significant changes in the estimated time series of permeabilities and skin factors as shown in *Figure 7.9* and *Figure 7.10.* However, one estimated value of the skin factor and one estimated value of permeability had too large confidence intervals (as defined in *Table 7.1*) and were excluded from the hypothesis testing and average parameter value estimation. *Figure 7.9* and *Figure 7.10* show the probability that the H_1 hypothesis is true, i.e. that we have significant changes in the time series of estimated parameters. ***Note that the results from the H_1 hypothesis testing are shown in these figures, i.e. it shows the probabilities for changes in the estimated permeabilities***. Recall that in hypothesis test #2 in *Figure 7.10*, estimated parameters are added to population B once a significant change is detected. Since no significant changes are detected, the values in this graph will be identical to *Figure 7.9*.

For both the estimated skin factors and permeabilities, the probability of the parameters not being a constant is less than 2%, i.e. there is a 98% probability for the parameters being constants. Hence we can calculate average parameter values as shown in *Figure 7.11* and *Figure 7.12* to obtain continuously updated parameter estimates with confidence intervals. The final estimated skin factor and permeability in these time series are well within the success criteria as defined in chapter 8.4.1. We would consider the result a success if the final estimated permeability had a confidence interval less then 10% of the estimated parameter value and if the “true” parameter value was inside the estimated confidence interval. In this case, the final estimate of the permeability is 60.1 ± 1.2 mD, and this can clearly be considered a success since the “true” permeability is 60 mD. We would consider the result as a success if the final estimated skin factor had a confidence interval less than 1 and if the “true” parameter value was inside the estimated confidence interval. In this case, the final estimate of the skin factor is 3.0 ± 0.1, and the result can clearly be considered a success (“true” skin factor is 3).

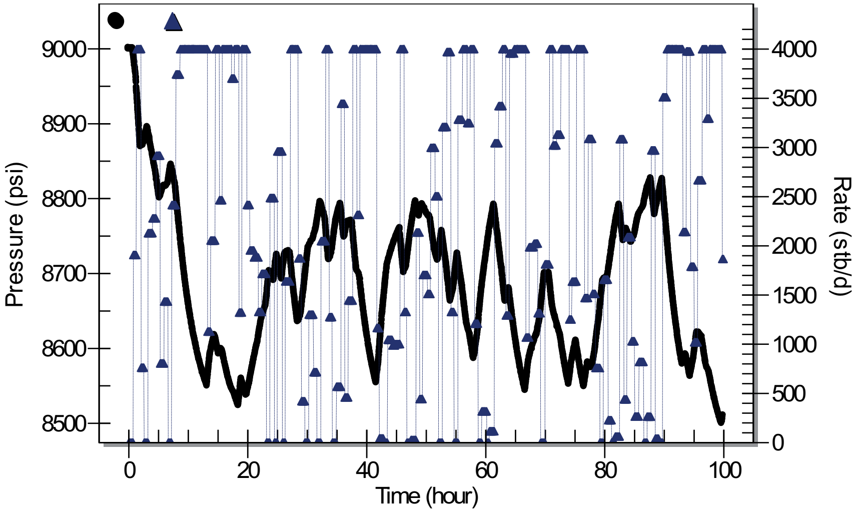


Figure 7.6: *Synthetic pressure and rate history in case 1.*

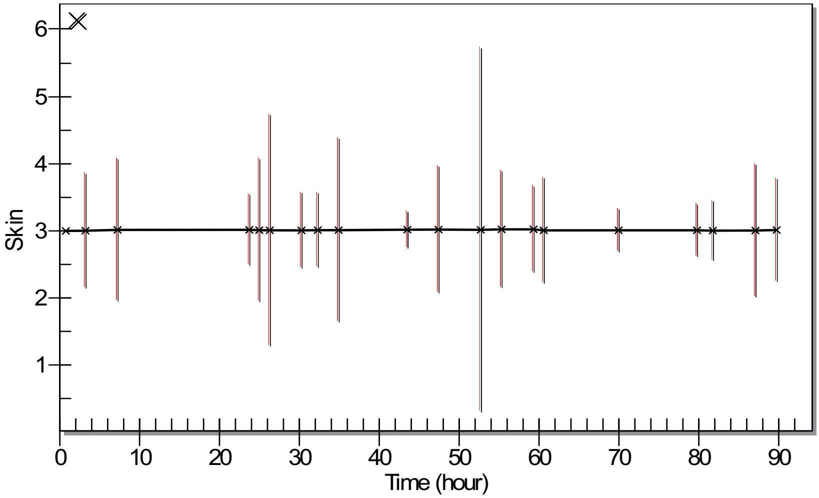


Figure 7.7*: Estimated skin factors from all detected shut-ins with confidence intervals estimated in automatic well-test analysis.*

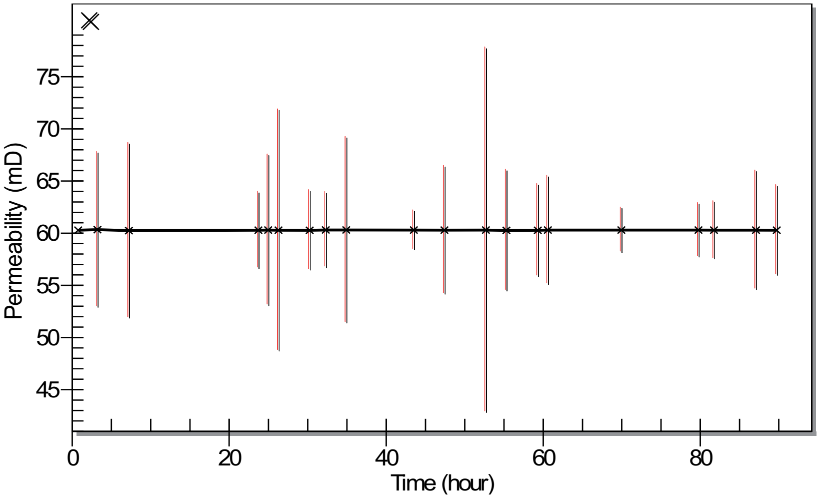


Figure 7.8: *Estimated permeabilities from all detected shut-ins with confidence intervals estimated in automatic well-test analysis.*

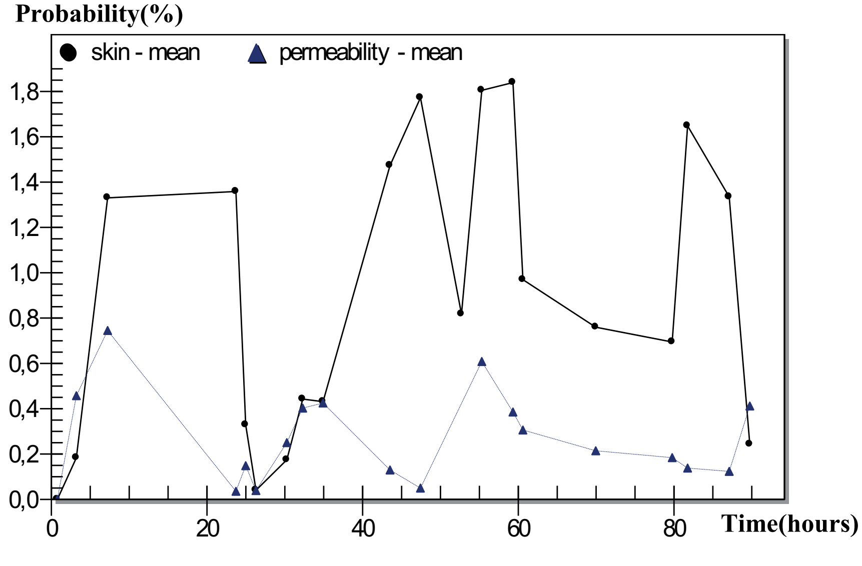


Figure 7.9: *Results from hypothesis test #1 testing for significant changes in skin factor and permeability mean values, i.e. that the H_1 hypothesis is true. Recall that only the last estimated parameter is used as population B in hypothesis test #1.*

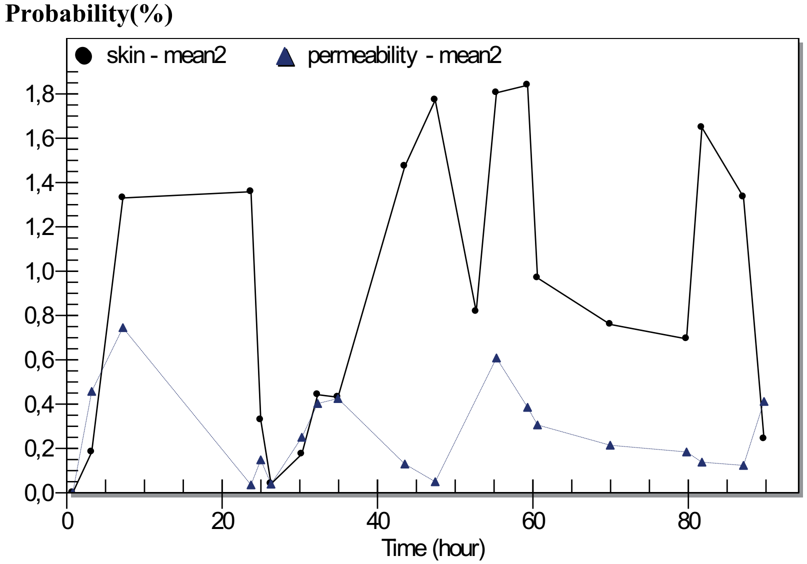


Figure 7.10: Results from hypothesis test #2 testing for significant changes in skin factor and permeability mean values, i.e. that the H_1 hypothesis is true. Recall that all estimated parameters are used as population B in hypothesis test #2 once a significant change is detected.

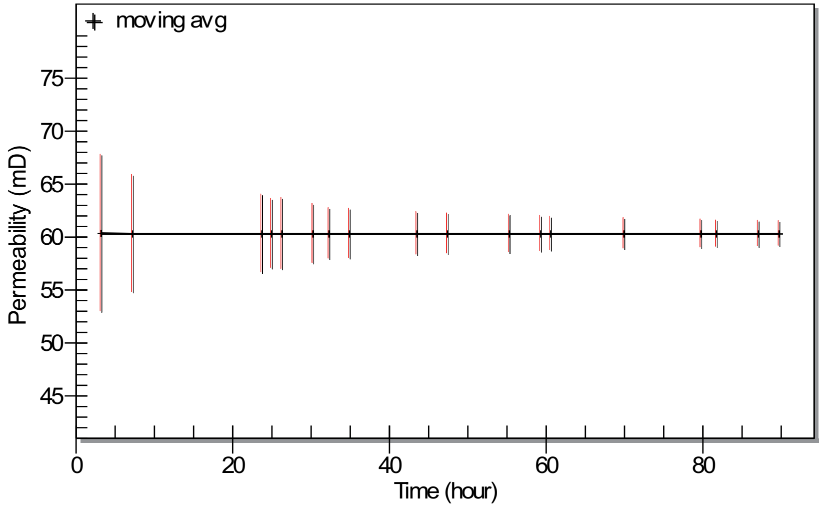


Figure 7.11: Estimated average permeabilities with confidence intervals from the detected shut-ins.

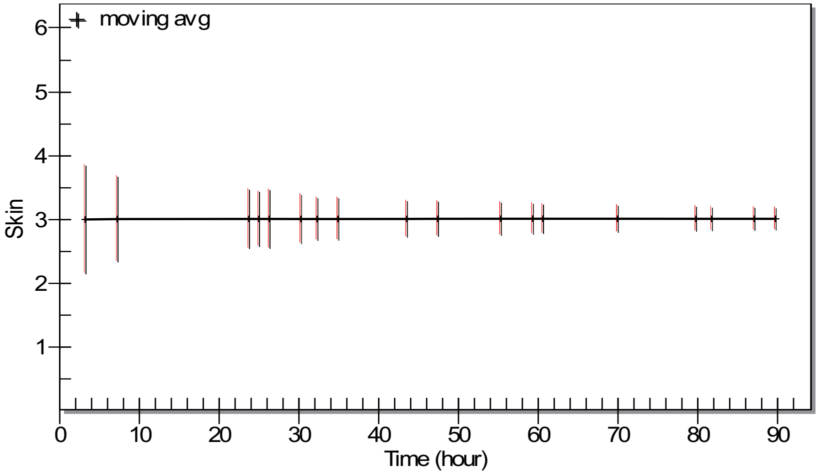


Figure 7.12: Estimated average skin factors with confidence intervals from the detected shut-ins.

7.4.3 Case 2

Case 2 was performed with the same intention as case 1; we would like to test how well the automatic well-test analysis works in a case where we know the true answer. A random rate history of 20.000 measurements is generated as described in chapter 5.1. The pressure is generated based upon a specified rate history and the same reservoir model as described in case 1. However there are two main differences; the permeability is set to 10 mD and the skin factor is changing and increases abrupt from 1 to 4 at time $t = 100$ hours. Hence in this case we would like to investigate if the proposed automatic well-test analysis is capable of detecting a change in the skin factor. Once again, three standard deviation white noise and random outliers are added to both rate and pressure measurements. (Again the pressure is generated based upon the noise free rate measurements.) The pressure and rate history in case 2 are shown in *Figure 7.13.*

In *Figure 7.14,* the permeabilities with confidence intervals from all detected shut-ins are shown. *Figure 7.15* shows the results from the hypothesis testing of whether significant changes are detected in the estimated permeabilities. *Note that the results from the H_1 hypothesis testing are shown in this figure, i.e. it shows the probabilities for changes in the estimated permeabilities.* Although the probabilities for changes in the estimated permeabilities are quite high, the zero hypothesis that the permeability is constant cannot be rejected with a significance level of 5%. Hence a continuously updated average value of the permeability with confidence intervals can be estimated and is shown in *Figure 7.16*. Again the true permeability is within the confidence interval of the final estimated permeability and the estimated confidence interval is significantly smaller than the limit for an acceptable confidence interval as discussed in chapter 7.4.1 (less than 10% of the estimated average parameter value). Hence the result is again considered a success.

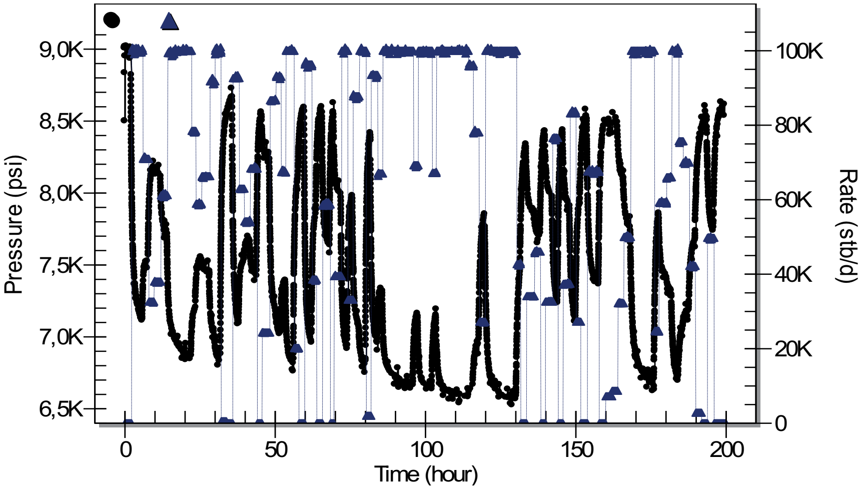


Figure 7.13: *Pressure and rate history in case 2.*

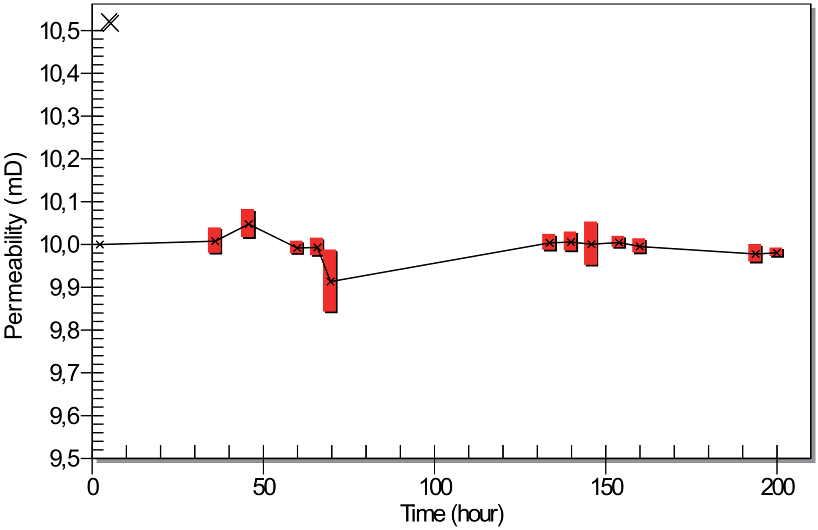


Figure 7.14: *Estimated permeabilities from all detected shut-ins in automatic well-test analysis.*

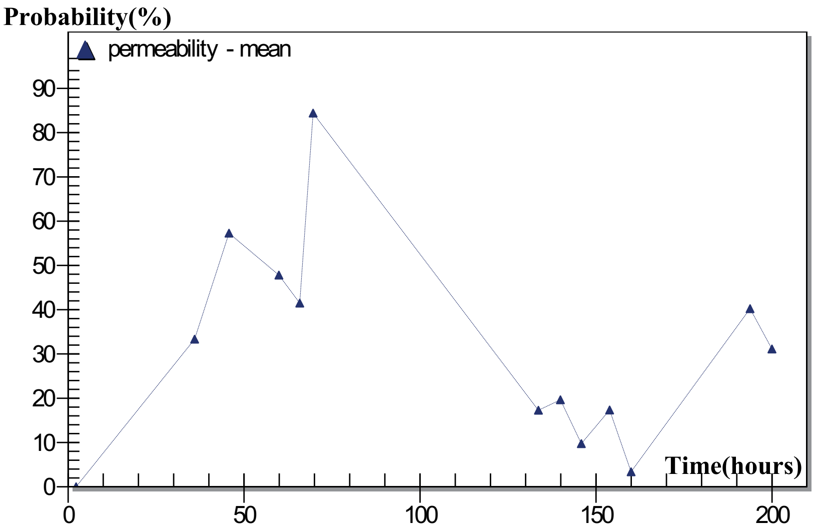


Figure 7.15: Results from hypothesis test #1 testing for significant changes in permeability mean values, i.e. that the H_1 hypothesis is true. Recall that only last estimated permeability is used as population B in hypothesis test #1.

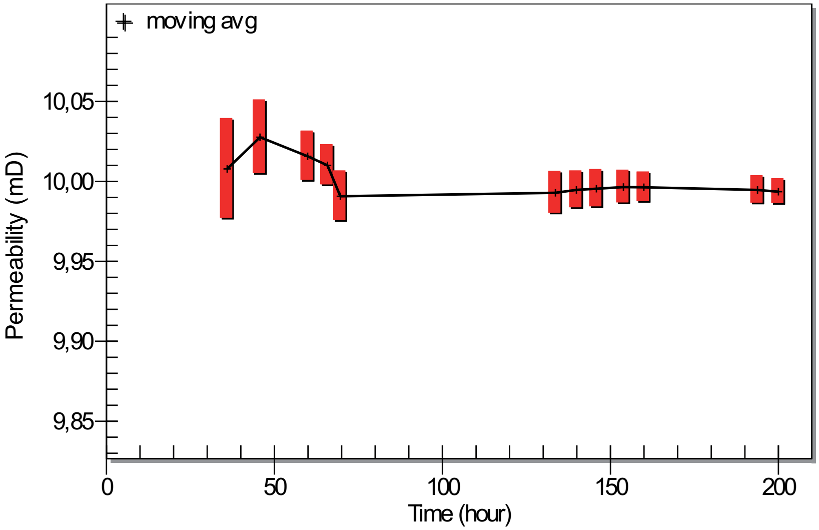


Figure 7.16: Estimated average permeabilities continuously updated in automatic well-test analysis. Only permeabilities estimated with small confidence intervals (i.e. less than 20% of the estimated permeability) are included.

Regarding the skin factor, the results from the individual well-tests are shown in *Figure 7.17*. By observing the estimated skin factors, we can clearly see the change taking place at t=100 hours.

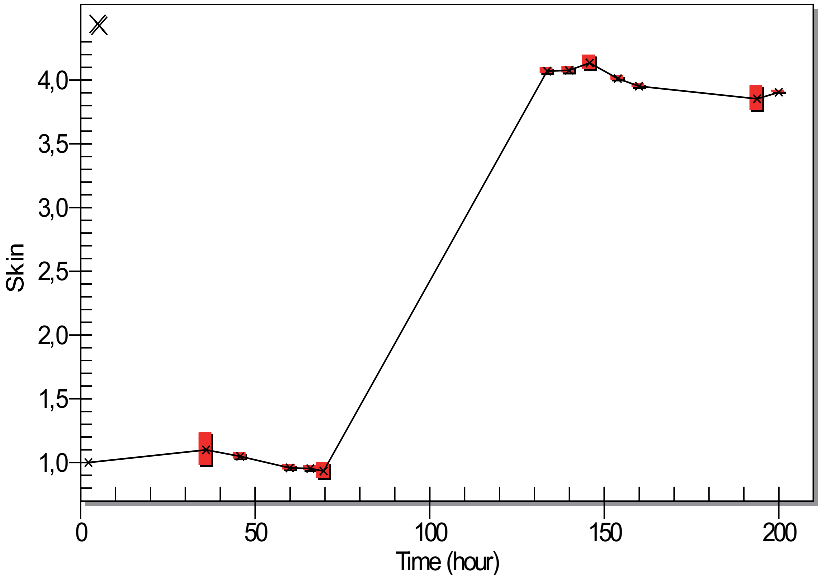


Figure 7.17: *Estimated skin factors from all detected shut-ins in the automatic well-test analysis.*

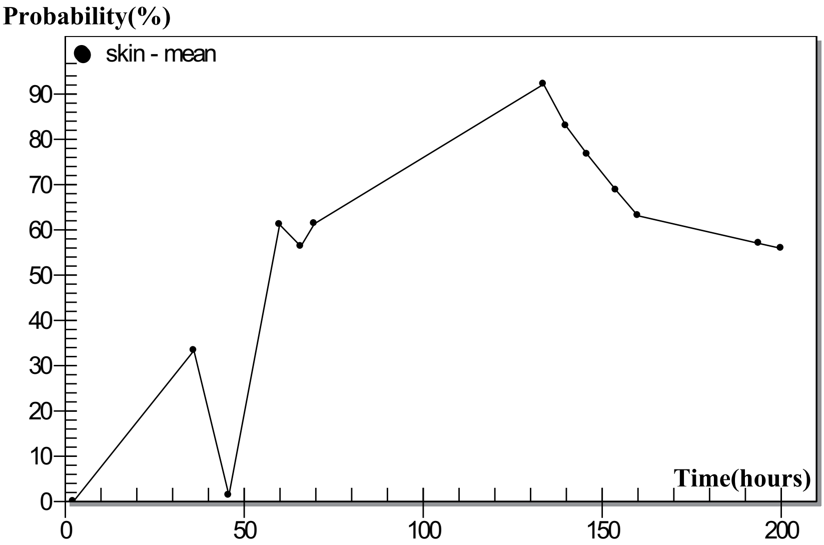


Figure 7.18: *Results from hypothesis test #1 testing for significant changes in skin factor mean values, i.e. that the H_1 hypothesis is true. Recall that only the last estimated parameter is used as population B in hypothesis test #1.*

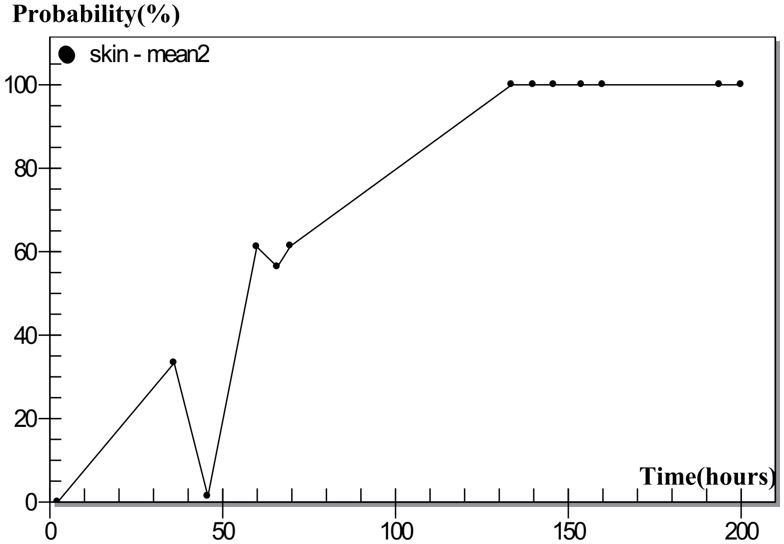


Figure 7.19: Results from hypothesis test #2 testing for significant changes in skin factor mean values, i.e. that the H_1 hypothesis is true. Recall that all estimated parameters are used as population B in hypothesis test #2 once a significant change is detected.

In *Figure 7.18,* only the last estimated parameter value is used as population B in the hypothesis test. Once a significant change in the estimated skin factor is detected, as happens at *t=100* hours, all new estimated skin factors are added to population B in a second hypothesis test. The result from this hypothesis test is shown in *Figure 7.19*. This to verify that the first detected change still is significant when new estimated parameters are added to population B. In *Figure 7.19***,** it is shown that the change in skin factor is detected with a probability above 99.9% and the hypothesis that the skin factor is constant must clearly be rejected. Recall that we use a significance level of 5% for the zero hypothesis that the skin factor is constant. Less then 0.1% probability for a constant skin factor simply means that the chance of doing a "measurement" like this and still have constant skin factor, is less then 1 to 1000. This example illustrates the capability of the proposed automatic well-test analysis to detect changes in time series of estimated parameters such as skin factor.

7.4.4 Case 3

Case 3 is from a North Sea gas/condensate well. We would like to perform automatic well-test analysis on production data from this well in order to obtain time series of estimated well and reservoir properties such as skin factor and permeability.

Approximately two month of production data is analyzed. The data is obtained from a database where pressure sampling has been performed only every 15 minute or slightly more frequently in periods with larger pressure changes. This also applies to the rate data. This means that the pressure resolution is quite poor. From the data base, an amount of 12.000 pressure measurements are stored giving an average sampling rate of only 8 measurements/hour for the two month considered. Sampling rates higher than 1 measurement/min are rare. This means that the wellbore storage period will contain few measurements, if any at all. Thus it should be expected that the skin factor and wellbore storage parameter estimates are poor. Recall that in the cases where only pressure signals with poor sampling rates are available, pressure de-nosing and data reduction is disabled. Even if the data resolution is poor, this is often the kind of data available for analysis. It is thus very interesting to check what results it is possible to obtain using such data. The pressure and rate measurements obtained from this well are shown in *Figure 7.20.*

The known well and reservoir parameters for the well considered are;

Total Compressibility: 4e-5 1/psi
Well Radius: 0.354 ft.
Porosity: 0.14
Viscosity gas: 0.64 cp.
B-factor gas: 0.003 rb/stb
Height: 200 ft.
Temperature: 293 F

The well considered is a gas/condensate well where the oil rate contributes less than 2-3 percent to total reservoir flow rate and can thus be ignored. The well is located in a high pressure/high temperature reservoir meaning that the well can be analyzed without the use of pseudo-pressures[(21)]. Numerous unplanned and planned shut-ins occur during this two month allows automatic well-test analysis to be performed.

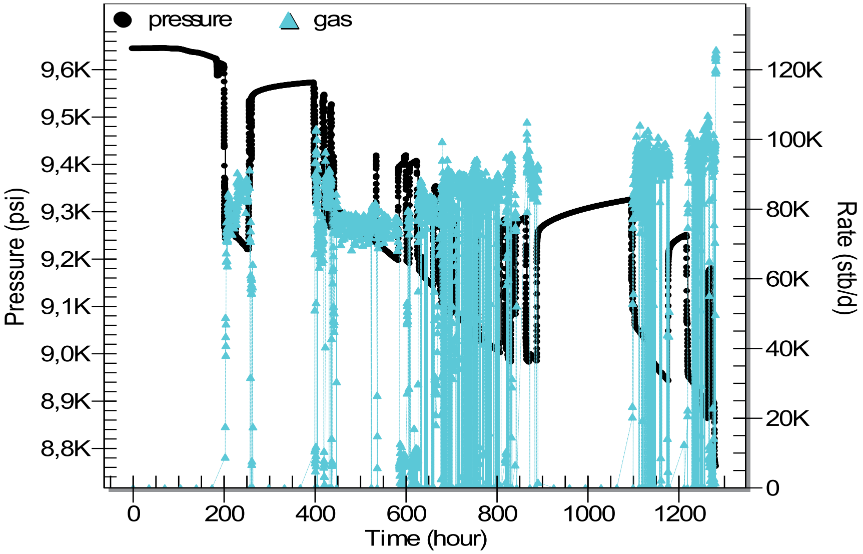


Figure 7.20: *Pressure and rate data for well in case 3.*

Initially one well-test needs to be analyzed manually to determine which wellbore, reservoir and boundary model to apply. As discussed in chapter 3, model identification is usually accomplished by the use of derivative plots. The shut-in period ending at approximately *t=400* hours is analyzed and the derivative plot for this shut-in period is shown in *Figure 7.21.* As seen from the derivative plot, the reservoir behaves as a homogenous reservoir with wellbore storage and skin effects in the start. At the end, boundary effects start to influence the derivative plot. The half unit slope line at the end of the derivative plot indicates that the well is placed inside a channel.

The results obtained from the manual and automatic tests indicate as expected that the only parameter that can be estimated from some of the tests with small enough confidence interval, i.e. where the confidence intervals are within the limits given in *Table 7.1*, is the permeability. The results from all well-tests are shown in *Table 7.2* and the continuously estimated permeability from the automatic well-test analysis is shown in *Figure 7.22*. Some of the permeabilities are estimated with large confidence intervals indicating poorly estimated parameters and are eliminated from the analysis. All the estimated permeabilities with small confidence intervals, i.e. within the limits given in *Table 7.1*, are marked in yellow and bold in *Table 7.2.*

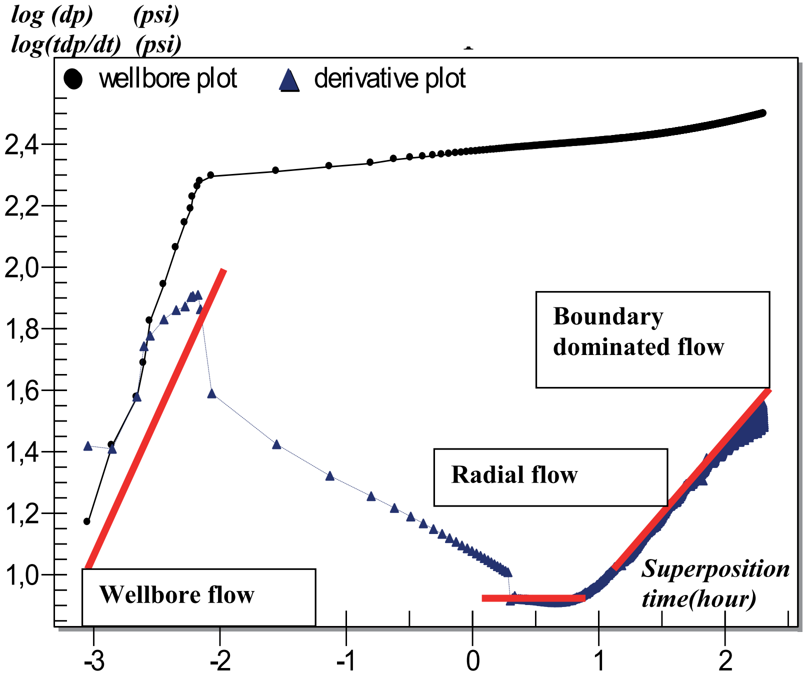


Figure 7.21: *Derivative plot for shut-in ending at time 400 hours.*

***Table** 7.2: Results from non-linear regression analysis in automatic well-test analyses compared with manual well-test analyses. The estimated permeabilities with confidence intervals within the tolerance limits in Table 7.1, are marked in bold and yellow.*

	Manual Analysis		**Automatic Analysis**	
Time (hours)	**Permeability (mD)**	**95% Confidence interval (mD)**	**Permeability (mD)**	**95% Confidence interval (mD)**
257.936	***90.2***	***6.1***	-----	----
262.428	***92.5***	***5.1***	-----	----
282.79	***89.6***	***7.8***	*37.8*	*11.6*
422.326	*91.1*	*51.5*	*131.2*	*149.8*
438.385	***79.4***	***6.2***	***79.2***	***2.5***
601.904	*77.7*	***20.8***	***84.2***	***13.6***
758.779	***77.6***	***1.2***	*138.1*	*79.5*
817.561	***82.1***	***1.9***	***84.4***	***5.5***
839.769	***85.9***	***3.8***	*130.2*	*93.6*
863.236	***81.8***	***2.6***	***82.2***	***2.8***
910.09	***80.1***	***1.1***	***80.9***	***1.7***
1199	***84.1***	***2.7***	***79.0***	***1.3***

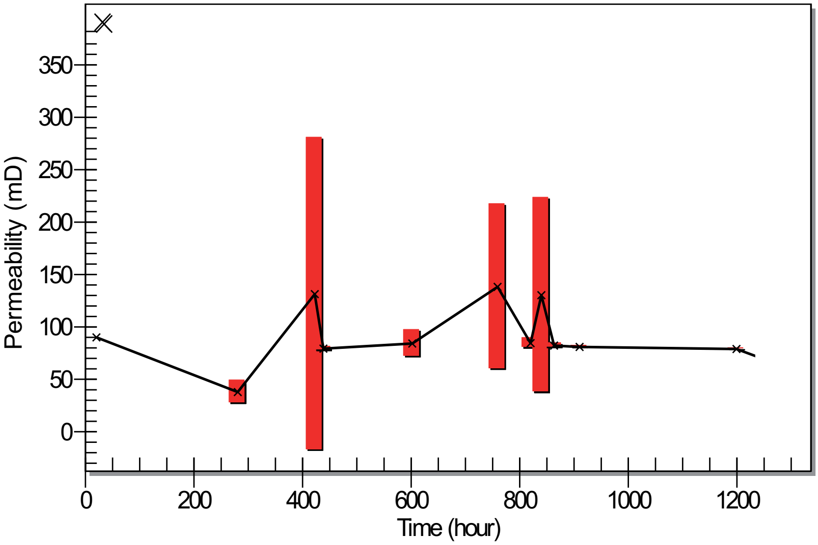


***Figure** 7.22: Permeabilities with confidence intervals estimated in automatic well-test analyses.*

Based upon the continuously estimated permeabilities, hypothesis testing is performed each time a new estimate of the permeability is obtained. The hypothesis testing did not detect any significant permanent changes in the estimated permeabilities for tests that did pass the confidence interval limit of maximum ±20% of the estimated permeability applying a significance level of 5%. In *Figure 7.23* the probability that the permeability is changing is shown, i.e. the inverse of the permeability being a constant. No significant changes are detected, and this allows us to obtain a continuously updated average permeability with confidence intervals as shown in *Figure 7.24*. This will in general provide us with an increasingly improved permeability estimate with a smaller and smaller confidence interval.

In the manual analysis, 10 out of 12 shut-in periods had small enough confidence intervals to be accepted while only six out of 12 well-tests did pass in the automatic analysis. The other well-tests failed due to convergence failure in the well-test non-linear regression analysis and/or high confidence intervals of the estimated parameter values. It should however be noted that those well-tests that passed in the automatic analysis, were the tests with the largest number of measurements and often, but not always, the well-tests with the smallest confidence intervals in the manual analysis. The final estimated average value obtained from the automatic well-test analyses was 83.7 ± 1.5 mD versus 81.1 ± 1.8 mD in the manual well-test analyses. The confidence intervals are overlapping, and thus within the uncertainty of the estimated values. We consequently consider the result a success.

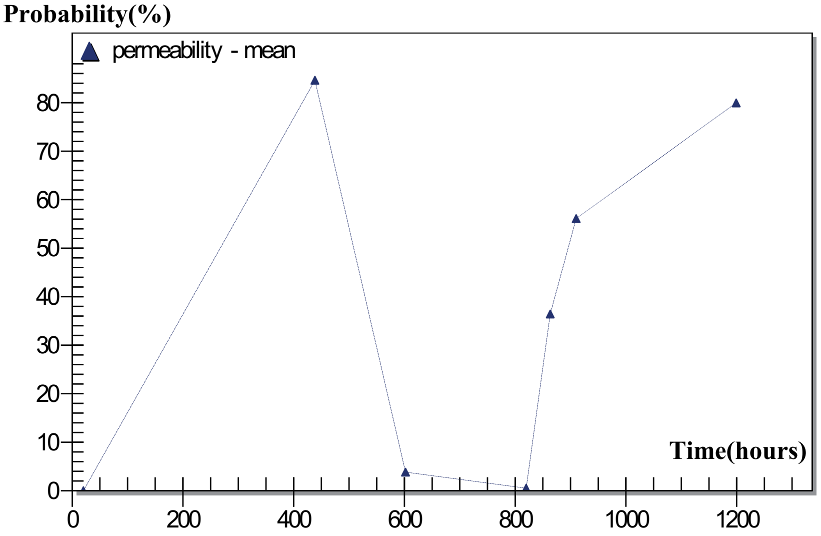


Figure 7.23: *Results from hypothesis test #1 testing for significant changes in permeability mean values, i.e. that the H_1 hypothesis is true. Recall that only the last estimated parameter is used as population B in hypothesis test #1.*

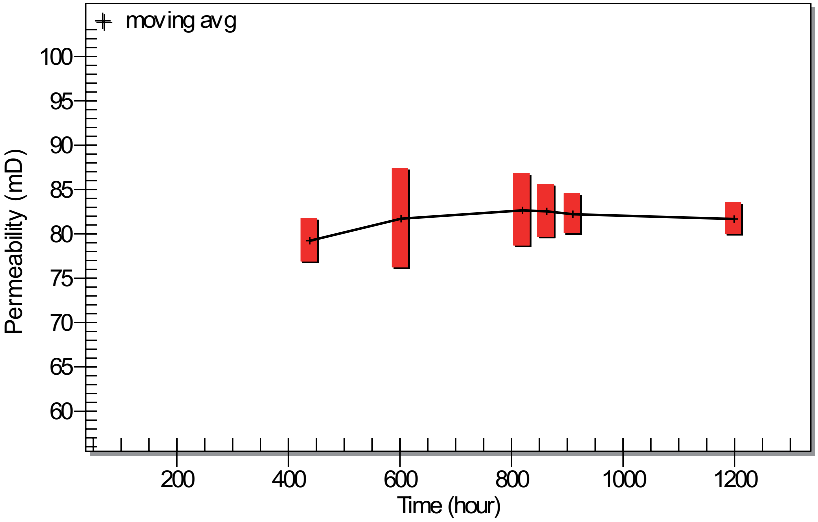


Figure 7.24: *Average permeability estimated from automatic well-test analysis where the estimated permeabilities passed the confidence interval limits in Table 7.1.*

Usually the confidence intervals in manual well-test analysis are smaller than in the automatic well-test analysis. This seems logical since we in manual well-test analysis continuously have the opportunity to eliminate errors. However, in some cases the confidence interval of the automatic well-test analysis is smaller. Actually one well-test in the automatic analysis with small enough confidence interval, had not small enough confidence interval in the manual analysis (within ±20% of estimated permeability value). This test is at time *t=601* hours. There might be several reasons for this including different pressure signal filtering, poor convergence in the manual non-linear regression analysis, different initial estimate of parameters used in non-linear regression analysis etc. Although we should expect to obtain better results in the manual analysis than in the automatic analysis, we believe exception can be expected, as seen in this case.

Finally, we will in *Figure 7.25* show an example of one well-test performed in the automatic well-test analysis with a good match between the measured pressures and the calculated pressures. One well-test with a poor match between the measured pressures and the estimated pressures in the automatic well-test analysis is shown in *Figure 7.26*. Unfortunately, skin factors and wellbore storage constants are not available from any of these well-tests due to large confidence intervals. The results are summarized in *Table 7.3 and Table 7.4.*

Table 7.3: *Results from regression analysis of build-up at t=817 hours. The bold and yellow estimated parameters had small enough confidence intervals to be accepted in automatic well-test analysis.*

	Parameter estimates	**95% confidence interval (+/-)**
Permeability (mD)	**84.4**	**5.5**
Wellbore Storage (stb/psi)	0.51	0.21
Skin Factor	10.6	5.3
Initial pressure (psi)	**9305**	**1**

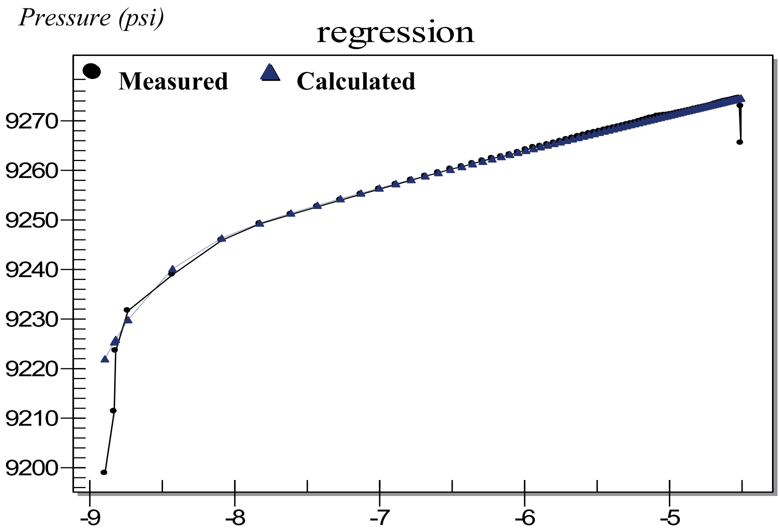


Figure 7.25: *Non-linear regression match on radial plot for build-up at t=817 hours.*

Table 7.4: *Results from regression analysis of build-up test at t=759 hours. All estimated parameters had too large confidence intervals to be accepted in automatic well-test analysis.*

	Parameter estimates	**95% confidence interval (+/-)**
Permeability (mD)	138.1	79.5
Wellbore Storage (stb/psi)	3	4.4
Skin Factor	20	666
Initial pressure (psi)	9488	2049

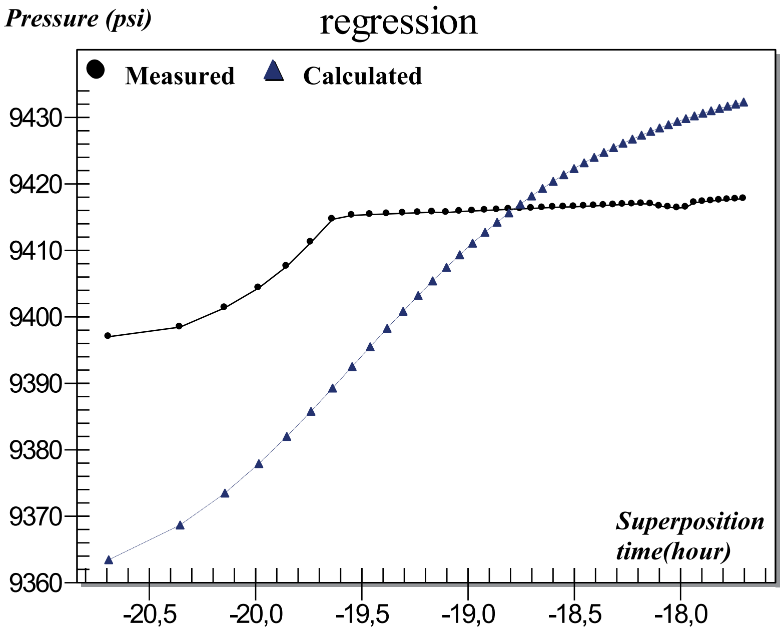


Figure 7.26: Non-linear regression match on radial plot for build-up test at t=759 hours.

7.4.5 Case 4

The fourth case is a well from the same North Sea gas/condensate reservoir as in case 3. Again we would like to utilize automatic well-test analysis to obtain time series of estimated permeabilities, skin factors and wellbore storage constants.

Approximately two month of production is analyzed. Again it is assumed that the only parameter that it is possible to estimate is the permeability due to poor pressure sampling rate as discussed in the previous case. Recall that in the cases where only pressure signals with poor sampling rates are available, pressure de-nosing and data reduction is disabled. Again a relatively large number of planned and unplanned shut-ins occur making it possible to obtain time series of the estimated parameters. In *Figure 7.27,* the pressure and rate history for this well is shown.

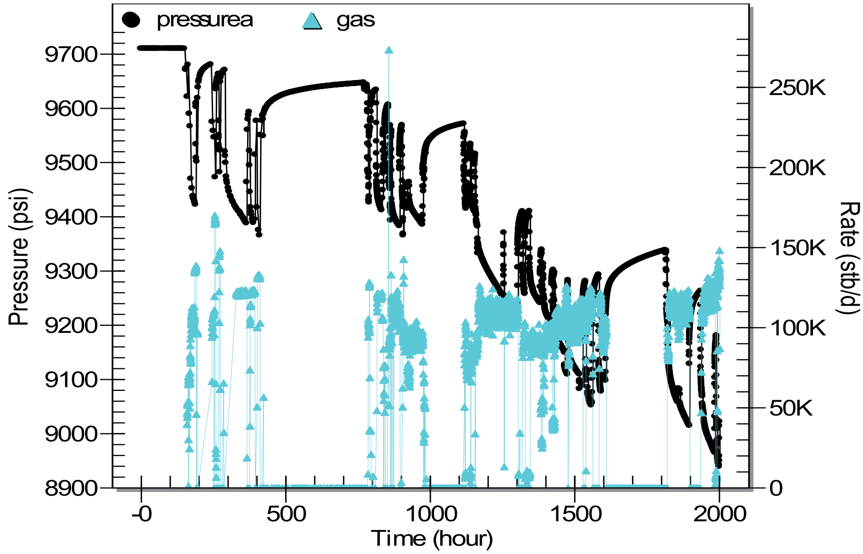


Figure 7.27: *Pressure and rate history for well in case 4.*

All known reservoir and fluid properties are shown below;

Total Compressibility: 4e-5 1/psi
Well Radius: 0.354 ft.
Porosity: 0.14
Viscosity gas: 0.64 cp.
B-factor gas: 0.003 rb/stb
Height: 200 ft.
Temperature: 293 F

Once again we need to perform one manual well-test analysis to determine the model for use in the non-linear regression analysis. The shut-in starting at time *t=430* hours is used and the derivative plot for this build-up test is shown in *Figure 7.28.* The derivative plot for this well indicates a well with wellbore storage and skin. The half unit slope line at the end of the derivate plot indicates that the well is placed inside a channel.

After the module setup is completed, automatic well-test analysis is performed with the rest of the rate and pressure measurements. Once again the estimated skin factors and the wellbore storage constants have too large confidence intervals, while acceptable confidence intervals are obtained for many of the estimated permeabilities. Eight automatic well-tests have estimated permeabilities with small enough confidence intervals while ten manual well-tests have estimated permeabilities with small enough confidence intervals. The results are shown in *Table 7.5.* The permeabilities with confidence intervals obtained in automatic well-test analysis are shown in *Figure 7.29.* One well-test in the automatic analysis with small enough confidence interval was estimated with large confidence intervals in the manual well-test analysis. The reason for such a result was briefly discussed in the previous case. We were not able to analyse the shut-ins detected at times *t=1142* and *t=1158*, not even in the manual well-test analysis. Too few measurements were available during these shut-in periods.

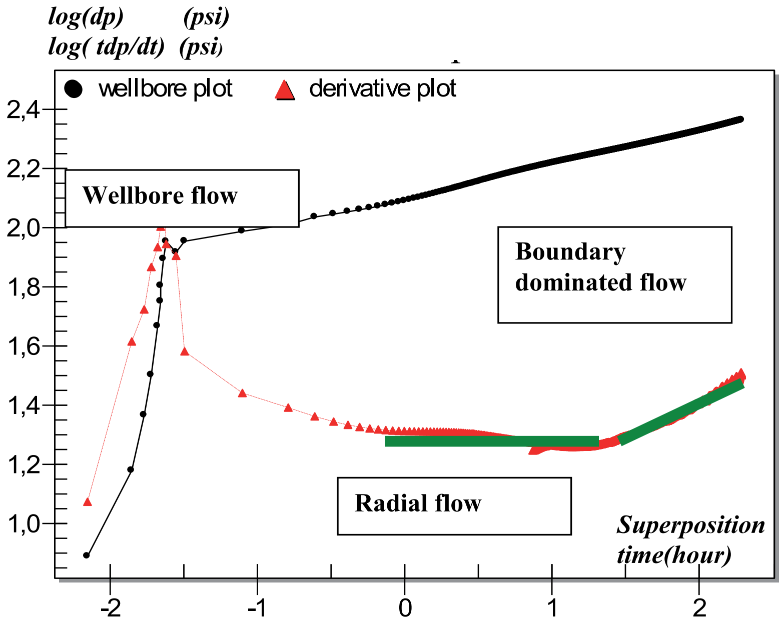


Figure 7.28: *Diagnostic plot for build-up test starting at time 430 hours.*

Table 7.5: *Results from regression analysis of automatic well-tests compared with manual performed well-tests. The estimated permeabilities with confidence intervals within the tolerance limits in Table 7.1 are shown in yellow and bold text.*

	Manual Analysis		Automatic Analysis	
Time (hours)	Permeability (mD)	95% confidence interval (mD)	Permeability (mD)	95% confidence interval (mD)
431.844	***27.0***	***0.2***	*15.9*	*39.3*
815.827	***30.3***	***1.1***	***20.0***	***2.4***
855.961	***30.6***	***1.1***	***30.7***	***2.8***
904.302	*28.9*	*9.8*	***33.8***	***5.1***
1001.72	***38.6***	***1.2***	*20.1*	*5.2*
1141.84	----	----	----	----
1157.66	----	----	----	----
1320.85	***37.4***	***2.2***	***32.0***	***4.5***
1351.21	***35.2***	***2.1***	***32.5***	***3.0***
1478.78	*49.4*	*8.5*	----	----
1538.86	***31.1***	***2.6***	*41.5*	*11.5*
1582.43	***28.7***	***0.1***	***29.3***	***2.8***
1629.66	***33.0***	***0.5***	***29.5***	***1.7***
1918.51	***31.6***	***1.6***	***26.3***	***1.8***

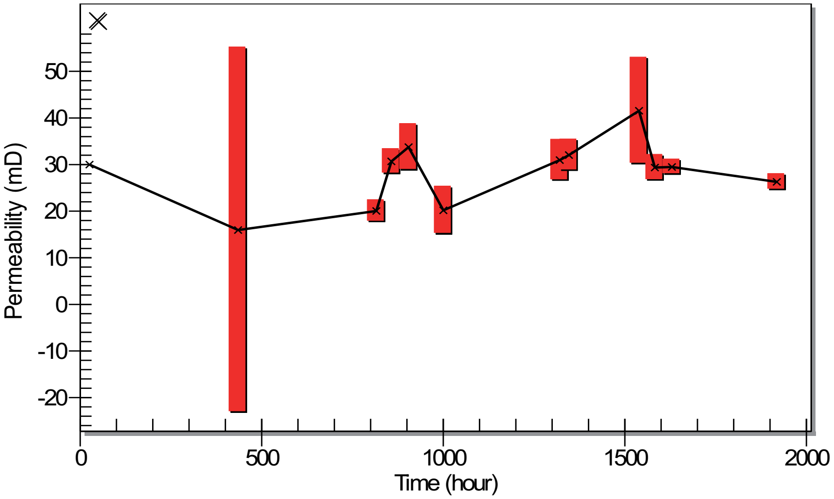


Figure 7.29: *Estimated permeabilities with confidence intervals from automatic well-test analysis.*

Hypothesis testing is performed each time a new estimate of the permeability is available and the results are shown in *Figure 7.30.* This figure shows the probability for a change in the estimated permeabilities, i.e. for the H_1 hypothesis being true. No significant changes are detected, and once again a continuously updated average permeability with confidence interval can be estimated. This is shown in *Figure 7.31.*

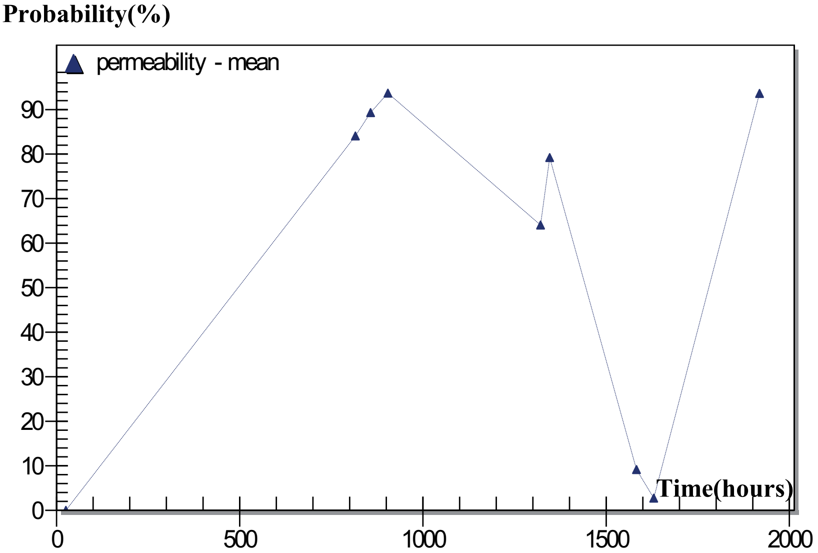


Figure 7.30: *Results from hypothesis test #1 testing for significant changes in permeability mean values, i.e. that the H_1 hypothesis is true. Recall that only the last estimated parameter is used as population B in hypothesis test #1.*

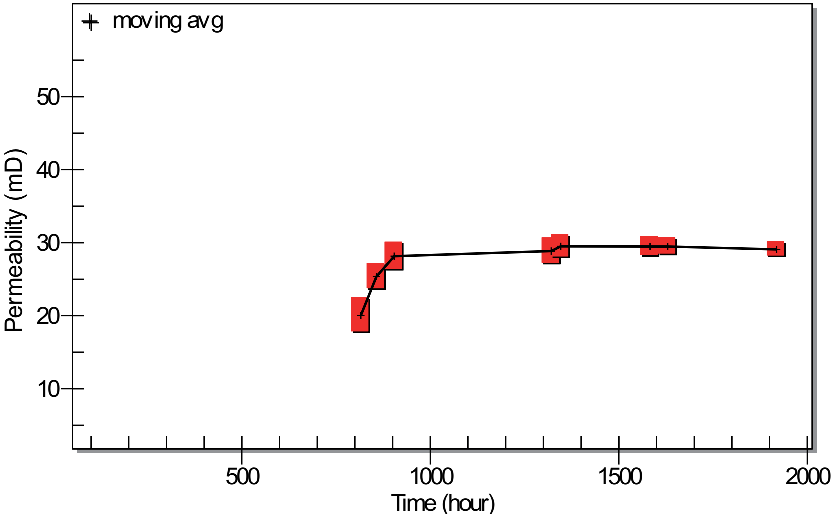


Figure 7.31: *Estimated average permeabilities with confidence intervals from automatic well-tests that passed the confidence limits from Table 7.1.*

The final average permeability estimated in the manual and automatic well-test analyses are very close. The permeability is estimated to 30.3 ± 0.8 mD in the automatic well-test analyses against 32.0 ± 1.1 mD in the manual well-test analyses. Again we have overlapping confidence intervals and the results can be considered identical within the uncertainties of the estimated values. Consequently we once again consider this result a success.

For this well, we also show the wellbore storage constant ands the skin factors estimated in the manual and the automatic well-test analyses. These are shown in *Table 7.6 and Table 7.7.* As previously mentioned, due to poor pressure data resolutions, poor estimates of the parameters along with large confidence intervals were obtained, both in the manual well-test analysis and in the automatic well-test analysis. In most cases, the estimated skin factors equal the initial estimate of 0 or the upper limit of 20 used in the non-linear regression analysis. And in most cases, the estimated wellbore storage constants equal the initial estimate of 0 or the upper limit of 3 used in the non-linear regression analysis. No limitation is put on the confidence intervals, and thus very high confidence intervals are in some cases seen due to very large uncertainties in the estimated parameter values.

Table 7.6: *Skin factors estimated from manual and automatic well-test analyses.*

	Manual Analysis		**Automatic Analysis**	
Time (hours)	**Skin**	**95% confidence interval**	**Skin**	**95% confidence interval**
431.844	*0*	*0.8*	*20*	*538*
815.827	*0*	*932.889*	*0*	*17.0*
855.961	*0*	*654*	*0*	*72*
904.302	*0*	*2192.13*	*0*	*121*
1001.72	*20*	*49.0501*	*20*	*4.0*
1141.84	----	----	----	----
1157.66	----	----	----	----
1320.85	*0*	*17.4683*	*0*	*369*
1351.21	*0*	*15.212*	*0*	*102*
1478.78	*0*	*159.673*	----	----
1538.86	*0*	*176254*	*20*	*12*
1582.43	*20*	*1.93884*	*20*	*16.4*
1629.66	*6.7*	*9.1*	*20*	*7.4*
1918.51	*20*	*3.1*	*0*	*430*

Table 7.7: *Wellbore storage constants estimated from manual and automatic well-test analyses.*

	Manual Analysis		**Automatic Analysis**	
Time (hours)	**Wellbore storage (std/psi)**	**95% confidence interval (std/psi)**	**Wellbore storage (std/psi)**	**95% confidence interval (std/psi)**
431.844	*3*	*0.5*	*3*	*69*
815.827	*0.223029*	*31.3524*	*3*	*15*
855.961	*0*	*11*	*0*	*0.1*
904.302	*0*	*32.9027*	*0*	*0.1*
1001.72	*0*	*0.43330*	*0*	*0.7*
1141.84	----	----	----	----
1157.66	----	----	----	----
1320.85	*0*	*0.510381*	*0*	*17*
1351.21	*0*	*0.0212*	*0*	*0.1*
1478.78	*0*	*7.42722*	----	----
1538.86	*0*	*12.9172*	*0*	*0.1*
1582.43	*0*	*0.015*	*0*	*0.1*
1629.66	*0*	*0.034*	*0*	*0.1*
1918.51	*0*	*0.10*	*0*	*10*

7.5 Summary

We have in this chapter described the proposed method for automatic well-test analysis. This method consists of four main step, setup, data preparation, well-test analysis and possible user intervention. The data preparation consists of several elements from the automatic signal filtering and process monitoring module, where the goal is to provide the well-test module with continuously filtered data and event detection. For each detected shut-in, the well-test module is executed. The well-test module contains two main elements, the non-linear regression analysis and the result evaluation. From the non-linear regression analysis, time series of the parameters considered as unknown are obtained. The result evaluation, based upon hypothesis testing, was used for detection of possible changes in the time series of estimated parameters. A notification is sent to the engineer if changes are detected. If no changes are detected, average parameter values are calculated. Four cases illustrating the application of the proposed method was shown, two cases with synthetic data and two cases with real data from a North Sea gas/condensate reservoir.

In chapter 7.4.1, we specified the criteria used to evaluate the proposed automatic well-test analysis with both synthetic data and real data. For all cases considered the proposed automatic well-test analysis performed well. In the synthetic cases, the true parameter values were within the confidence intervals of the estimated parameter values. In addition, the estimated confidence intervals were small and well within the success criteria discussed in chapter 7.4.1. Unfortunately, poor data resolution in the real data cases made skin factor estimation difficult. However, the permeabilities estimated from the automatic well-test analyses compared well to those estimated from the manual well-test analyses. In the third case considered, the final estimated average permeability was 83.7 ± 1.5 mD in automatic well-test analyses versus 81.1 ± 1.8 mD in manual well-test analyseis. In the fourth case, the final average permeability was 30.3 ± 0.8 mD in automatic well-test analyses against 32.0 ± 1.1 mD in manual well-test analyses. Based upon the cases we have considered, we conclude that the proposed automatic well-test analyses works excellent for the cases considered. Unfortunately, we did not have the opportunity to test the automatic well-test analysis with high resolution signals measured in real-time.

Finally note that we do not claim that automatic well-test analysis is better or more accurate than manual well-test analysis in any ways. However, additional well-tests will be completed if performed automatically and we have shown that some of the information obtained by repeatedly performing manual well-test analysis might be obtained in automatic well-test analysis.

8 Conclusions

In this work, new methods for automatic signal filtering, process monitoring and well-test analysis is proposed. Automatic signal filtering saves the engineers for significant effort in signal filtering and data processing. Measured signals can easily be utilized in different kinds of analyses. The fact that automatically filtered measurements are available at any time might in fact increase the use of different types of analyses as they might be performed faster. The process monitoring approach provides the engineers with increased and improved process monitoring. This provides the engineers with faster and better detection of events such as transients, changing noise level and changing signal trends. Automatic well-test analysis ensures that all available rate and pressure measurements are analyzed and the information within the measurements is revealed. This leads to faster and better decisions. Actions can be implemented based upon these decisions where the aim is to improve the asset management. ***An initial setup is required for the automatic modules, but when completed, it is shown that signal filtering, process monitoring and well-test analysis can be performed automatically and in real-time with the signals considered in this work.***

More specifically we have developed methods for automatic signal filtering including;

- Outlier removal
- De-noising
- Data reduction

The method for outlier removal is based upon a median filter and removes single as well as multiple outliers without user intervention. We conclude that the proposed outlier removal is well suited for automatic outlier removal as all settings are predefined or estimated from the measured signals. Excellent performance was demonstrated with the signals considered in this work.

Signal de-noising was accomplished with Waveshrink. Through the large number of filtering exercise done by this author, the selections and methods in Waveshrink that on average works best for the signals considered were determined. This holds the promise for automatic signal de-noising.

Next, the number of signal measurements can be reduced by thresholding in an iterative minimization process. We conclude that the proposed method for data reduction is well suited for automatic data reduction as all settings are predefined or estimated from the measured signals. Again, good performance was demonstrated for the proposed method with the signals considered in this work.

From the results in signal filtering, we conclude that automatic signal filtering is possible for the kind of signals we have considered with the methods proposed in this work.

A pattern recognition approach is proposed for pressure transient detection.

When tested on synthetic pressure signals, in general around 85% of the pressure transients were correctly identified. For the cases considered with real pressure signals, around 80% of the pressure transients were correctly identified. In general only the transients with the smallest pressure changes were missed. All the “large” transients in terms of pressure change were accurately detected and no erroneously transients were detected in the cases considered. In summary, good performance was demonstrated for the proposed transient detection and the method is well-suited for automatic transient detection.

Additional challenges will occur if signal filtering and transient detection should be utilized with continuously measured signals. A process monitoring approach is consequently proposed. This includes;

- Real-time noise estimation
- Real-time signal filtering
- Real-time transient detection
- Real-time trend estimation

A window approach is required for signal filtering and process monitoring with continuously measured signals. Recommendation of window size for use in real-time signal filtering and process monitoring was given. In addition, a real-time wavelet transform is proposed to continuously update noise estimation, signal filtering and trend estimation with a minimum of computational effort. Continuously updated filtered signals will be ready for analysis at any time and the process monitoring is well suited for continuous pressure transient detection, signal noise estimation and signal trend estimation, ***We conclude that with the proposed approach, a large number of signal measurements can be handled in a computationally efficient manner in real-time for use in automatic signal filtering and process monitoring.***

In this work, we have also proposed a method for automatic well-test analysis from continuously measured pressure and rate signals. The proposed method contains four main step;

- Setup
- Automatic data preparation
- Automatic well-test analysis
- Possible user interaction

In initial setup is required, but once this is done, well-test analysis is performed automatically. The performance of the automatic signal filtering and process monitoring module for data preparation in well-test analysis was very good in the cases we considered.

The estimation of well-test parameters can be best done from shut-ins and this can be performed continuously and automatically for each detected shut-in. This will provide us with time series of estimated parameters provided that enough shut-ins occur. However, measurements with a sufficiently high sampling rate must be available, otherwise the estimation of different parameters becomes highly uncertain. In our cases with real data, poor pressure sampling rates prevented the estimation of skin factor and wellbore storage. With a high degree of certainty, the non-linear regression analysis will fail for some of the detected shut-ins and results in erroneous parameter estimates. High confidence intervals rules out many of the erroneous parameter estimates and provides us with an excellent tool for quality checking the results. It is further shown that hypothesis testing can be used to detect changes in estimated parameters. This was tested with synthetic data and worked very well.

When hypothesis testing indicates that the parameters mean value is constant, a continuously updated average parameter value with variance is estimated. It is shown that even if some of the shut-in periods are missed or give erroneous results in the automatic well-test analyses, the estimation of average parameter values can still be successful when compared to the results from manual well-test analyses. In our two cases with real data, the final estimated parameters from the manual and the automatic well-test analyses were close, *81.1 mD* versus *83.7 mD* (case 3) and *30.3 mD* versus *32.0 mD* (case 4). We will not claim that automatic well-test analyses is more accurate than manually performed well-test analyses, but additional well-tests will be completed if automatic well-test analyses are available and additional information about the well and reservoir considered will be available. ***We conclude that the use of the proposed automatic well-test analysis for calculation of average parameter values works very well.***

9 Further work

Improved and better use of real-time data has been a major concern in the E&P industry for some time now. However, the main challenge is to utilize these measurements to obtain more information about wells and reservoirs. For engineers with limited available time, efficient and automatic tools for extraction of information are critical. We have in this work considered automatic signal filtering, automatic process monitoring and automatic well-test analysis. A natural extension of this work will be to consider other types of analyses than well-testing. This includes material balance calculation and decline curve analysis. Material balance is simply a volume balance which equates the total production and injection in the reservoir to the initial volume and the current volume. Decline curve analysis is used to fit production data to a decline curve for prediction of future oil and gas production. Both material balance and decline curve analysis would be good candidates for a similar approach as is described for well-test analysis in this work.

Even in reservoir simulation, a similar approach could be applied. Today, the reservoir engineers spend a lot of time on data preparation for updates of reservoir simulation models. Then the reservoir simulation is performed and the results are inspected by the reservoir engineers. If measured data significantly deviates from predicated data, additional history-matching of the reservoir simulation model is usually considered. However, we can envision that reservoir simulation could be done automatically in the same way as well-test analysis, and where the reservoir engineer only receives a notification if a significant deviation between the measured data and the predicted data is found. Thus the reservoir engineers could spend less time on data preparation and routinely updates of the reservoir simulation model and more time on analysing the cases where a poor history match is obtained.

Some further improvement and extension of the modules developed in this work is recommended as well. The performance of the proposed data reduction method or the proposed transient detection method has not been tested against other methods available in the literature. We believe that they will perform very well compared to the methods found in the literature, but this should be confirmed.

In Waveshrink we simply determined the best selections for use in Waveshrink in terms of root-mean-squared errors. We did not investigate or explain why some selections gave better results than the others. An improvement of this work will be to understand why some selection works better than others for the various signals considered.

The automatic well-test module has been tested using numerous synthetic data sets and four cases with real data (where two synthetic cases and two real-data cases representative for the results obtained were included in this work). However, we still believe that it would be beneficial to test the proposed method with more data, and in particular data where various well-test models would be required (note that the well-test implementation developed in this work already includes numerous other well-test models than the models described in this thesis). It would in addition be beneficial to test the automatic well-test module using real data with higher sampling rates than we had access to. The system has been tested using synthetic data with high sampling rate, but from the experience gained in this work, additional challenges might be identified when using real signals with high sampling rates.

Finally, automatic signal filtering, process monitoring and well-test analysis should be tested within an online operation centre by an asset team, to verify that the recommendations and methods developed in here will prove valuable to the asset team. It should also be verified that the proposed automatic process monitoring provides the asset team with information from the signal measurements faster or with information otherwise missed. Further it should be verified that automatic signal filtering saves time for the asset team in data preparation when preparing data for different kinds of analysis, and that this subsequently results in faster and/or in the completion of additional analyses providing the asset team with valuable information. Next it should be verified that automatic well-test analysis provides the asset team with faster information from well-tests and/or the completion of additional well-tests. Finally it should be verified that the information obtained leads to faster and better decisions, and the value of these decisions should be quantified in terms of increased recovery and/or decreased cost.

Nomenclature: Well-testing

A	Reservoir area or drainage area for the well
B	Formation volume factor
C	Wellbore storage constant
C_D	Dimensionless wellbore storage constant
c	Compressibility (Total compressibility of formation and fluid)
h	Reservoir height
$I_0(x)$	Modified Bessel function of second order
$I_1(x)$	Modified Bessel function of second order
$K_0(x)$	Modified Bessel function of first order
$K_1(x)$	Modified Bessel function of first order
k	Permeability
L	Laplace transform
L^{-1}	Inverse Laplace transform
m	Mass
μ	Viscosity
Δp	Pressure difference
Δp_{skin}	Pressure drop/increase caused by skin factor
P_D	Dimensionless pressure
P_{wD}	Dimensionless pressure with wellbore storage and skin effects
p_{wf}	Flowing well pressure
p_{wfn}	Flowing well pressure influenced by n’th rate change
p_i	Initial reservoir pressure
Φ	Porosity
ρ	Density
r	Radius
r_w	Well radius
r_e	Reservoir extension
r_{eD}	Dimensionless reservoir extension
r_D	Dimensionless radius
S	Skin factor
t	Time
t_D	Dimensionless time
v	Velocity
V	Volume
V_p	Pore volume.
V_b	Bulk volume
V_t	Total volume
Q	Surface flow
q	Sandface flow
q_w	Wellbore flow
z	Laplace variable

Nomenclature: Non-Linear Regression Analysis

$A_{m\times n}$	Sensitivity matrix for *m* parameters and *n* samples
$C_{m\times m}$	Covariance matrix for *m* parameters
c_{ii}	Element in the covariance matrix
ε	Model error
$f(t,v_0)$	Model function with initial parameter vector v_0 at times *t*
$f(t,v)$	Model function with parameter vector *v* at times *t*
$f(t,v)^*$	Approximation of the model estimate
G	Jacobi matrix
H	Hessian matrix
$J(v)$	Objective function
R	Real numbers
Σ	Expected value of residual vector
t_{n-m}	t –distribution of sample with *n* samples and *m* parameter
θ	Risk level in student t-test
v	Parameter vector in regression analysis
v_0	Initial estimate of parameters vector *v*
v_i	Element *i* of the parameter vector
Y	Measurement vector at times *t*
y_i	Measurement at time t_i

Nomenclature: Wavelet Analysis

$c(k)$	Approximate coefficient *k* at resolution level $j=0$
$c_j(k)$	Approximate coefficient *k* at resolution level *j*
$c_{j0}(k)$	Approximate coefficient *k* at primary resolution level j_0
$d_j(k)$	Detail coefficient *k* at resolution level *j*
$f(t)$	Model function
F	De-noised signal
f_i	De-noised signal measurements *i* (*i* is an integer index)
j	Resolution level in wavelet transform
j_0	Primary resolution level
J	Max resolution level
k	Finite or infinite integer index used in function approximation
Λ	Cumulative distribution function
Λ^{-1}	Inverse cumulative distribution function
m	Window size for use in median outlier filter
N	Number of samples taken of a signal
n	Finite or infinite index for use with scaling or wavelet coefficients
p	Signal period
S	Signal
s_i	Signal sample *i* (*i* is an integer index)
	De-noise signal

f_i	Denoised signal sample *i*, *(i* is an integer index)
T	Sampling interval of signal S(t)
W	Wavelet transform
W^{-1}	Inverse wavelet transform
α	Noise level
δ	Thresholding operator for wavelet coefficients
δ_λ	Thresholding operator for wavelet coefficients applying a threshold
λ	Threshold in wavelet filtering
μ	Mean value
$\phi_{j,k}(t)$	Scaling function with dilation *j* and translation *k*
$\phi\ (t)$	Basic scaling function with k=0 and j=0.
$\psi_{j,k}$	Wavelet function with dilation *j* and translation *k*
$\psi(t)$	Basic wavelet function with k=0 and j=0.
Φ_j	Vector of all scaling functions at level *j*
Ψ_j	Vector of all wavelet functions at level *j*
Y	Noise free signal
y_i	Noise free signal sample *i*, *(i* is an integer index)

k will in the context of wavelet and scaling functions refer to the translation of these functions.
j will in the context of wavelet and scaling functions refer to the dilation of these functions.

Nomenclature: Hypothesis Testing

H_0	Zero hypothesis in hypothesis testing
H_1	Alternative hypothesis in hypothesis testing
$\bar{x}_i$	Mean value sample *i*
var_i	Variance sample *i*
$\overline{X}$	Average mean
$\mathrm{var}(\overline{X})$	Average variance
N	Number of samples
N_A	Number of samples in population A
N_B	Number of samples in population B
$\bar{x}_A$	Mean value population A
$\bar{x}_B$	Mean value population B

Works Cited

1. ***Improved Wavelet Filtering and Compression of Production Data.*** Olsen, S. and Nordtvedt, J.-E. **Presented at Offshore Europe, Aberdeen, Scotland, Sept. 2005. Paper SPE 96800.**

2. ***Automatic Filtering and Monitoring of Real-Time Reservoir and Production Data.*** Olsen, S. and Nordtvedt, J.-E. **Presented at Annual Technical Conference and Exhibition, Dallas, Texas, Oct. 2005. Paper SPE 96533.**

3. ***Experience from the use of Automatic Well-Test Analysis.*** Olsen, S. and Nordtvedt, J.-E. **Presented at Annual Technical Conference and Exhibition, San Antonio, Texas, USA, Oct. 2006. Paper SPE 102290.**

4. ***Multi-Resolution Analysis of Long-Term Pressure Transient Data Using Wavelet Methods.*** Kikani, J. and He, M. **Presented at 1998 SPE Annual Technical Conference and Exhibition, New Orleans, Lousiana, USA, Sept. 1998. Paper SPE 48966.**

5. ***Processing and Interpretation of Long – Term Data Acquired From Permanent Pressure Gauges.*** Athichanagorn, S., Horne R. N. and Kikani, J. **SPEREE, 384-390, Oct. 2002. Paper SPE 56419.**

6. ***Permanent Downhole Gauge Data Interpretation.*** Khong, K. C. **Master Thesis, Stanford University, June 2001. .**

7. ***The Data as the Model; Interpreting Permanent Downhole Gauge Data without Knowing the Reservoir Model.*** Thomas, O. **Master Thesis, Stanford University, June 2002. .**

8. ***An Improved Treatment of Long-Term Pressure Data for Capturing Information.*** Viberti, D., Verga, F. and Delbosco, P. **SPE Journal, 359-366, August 2007. Paper SPE 96895.**

9. ***Application of Wavelet Transform to the Analysis of Pressure Transient Data.*** Soliman M. Y., Ansah, J., Stephenson, S and Manda, B. **SPEREE, 89-99, April 2003. Paper SPE 83670.**

10. ***Use of wavelet Transform in Pressure-Data Treatment.*** Riberi, P. M., Pires, A. P., Oliveira, E. A. P., Ferroni, J. G. **SPE Journal, 24-31, February 2008. Paper SPE 100719.**

11. ***Improving Permanent Downhole Gauge (PDG) Data Processing via Wavelet Analysis.*** Ouyang, L. and Kikani, J. **Presented at the SPE 13th European Petroleum Conference, Aberdeen, U.K, Oct. 2002. Paper SPE 78290.**

12. ***Wavelet filtering of permanent downhole gauges data.*** Ortiz, C.E.P, Aguiar, R.B., Pires, A.P. **Presented at SPE Latin American and Caribbean Petroleum Engineering Conference in Cartagena, Columbia, June 2009. Paper SPE 123028.**

13. ***Truncation De-Noising in Transients Pressure Tests.*** Gonzalez-Tamez, F., Camacho-Velazquez, R. and Escalante-Ramirez, B. **Presented at 1999 SPE Annual Technical Conference and Exhibition, Houston, Texas, USA, Oct. 1999. Paper SPE 56422.**

14. ***Application of Wavelet Transforms to Reservoir-Data Analysis and Scaling.*** Panda, M. N., Mosher, C. C. and Chopra, A. K. **SPE Journal, 92-101, March 2000. Paper SPE 60845.**

15. ***Analyzing Transient Pressure From Permanent Downhole Gauges (PDG) Using Wavelet Method.*** Shi-Yi, Z. and Xiao-Gang, L. **Presented at SPE Europec/EAGE Annual Conference and Exhibition held in London, UK, June 2007. Paper SPE 107521.**

16. ***Wavelet in Petroleum Industry; Past, Present and Future.*** Guan, L., Du, Y. and Li, L. **Presented at the Annual Technical Conference and Exhibition, Houston, Texas, USA, September 2004. Paper SPE 89952.**

17. ***Wavelets in Non Parametric Regression; A Comparative Simulation Study.*** Antoniadis, A., Bigot, J. and Sapatinas, T. **Journal of Statistical Software, 6, 2001. .**

18. ***Analyzing Simultaneous Rate and Pressure Data From Permanent Downhole Gauges.*** Rai, H. and Horne, R. N. **Presented at ATCE in Anaheim, California, USA, 11-14 November 2007. Paper SPE 110097.**
19. ***Data Processing and Interpretation of Well Test Data as a Nonparametric Regression Problem.*** Nomura, M. and Horne, R. N. **Presented at SPE Western Regional Meeting in San Jose, California, USA, 24-26 March 2009. Paper SPE 120511 .**
20. ***Automatic Detection of Pressure-Buildup Intervals from Permanent Downhole Pressure data Using Filter Convolution.*** Suzuki, S. and Chorneyko, D. **Presented at ATCE New Orleans, USA, 4-7 October 2009. Paper SPE 125240 .**
21. Dake, L.P. ***Fundamentals in reservoir engineering.*** **s.l. : Elsevier Scientific Publishing Company, 1978. Seventh Edition.**
22. Lee, J. H. ***Estimating Time-Dependent Reservoir Properties by Analyzing Long-Term Pressure Data.*** **Master Thesis, Stanford University, June 2003. .**
23. ***Intelligent Asset Management; Successful Integration of Modeling Tools and Workflow Processes.*** Airlie, C.J. and Lemanzcyk, Z.R. **Presented at Asia Pacific Conference on Integrated Modeling for Asset management in Kuala Lumpur, Malaysia, March 2004. Paper SPE 87019.**
24. ***PanSystem from EPS.*** **Research Park, Riccarton, EDINBURGH, EH14 4AP UK, . www.epsedin.co.uk.**
25. ***The Boris Field Well Management Philosophy – The Application of Permanent Downhole Flowmeters to Pressure Transient Analysis; An Integrated Approach.*** Coludrovich, E. J., McFadden, J. D., Palke, M. R., Roberts, W. R. and Robson, L. J. **Presented at Annual Technical Conference and Exhibition, Houston, Sept. 2004. Paper SPE 90316.**
26. ***Numerical Inversion of Laplace Transform.*** Stehfest, H. **Communication of the ACM, 13, 47-49, January 1970. .**
27. ***Business Value From Intelligent Fields.*** van den Berg, F., Perrons, R. K. and Moore, I. **Presented at SPE Intelligent Energy Conference and Exhibition in Utrecht, The Netherlands : s.n., 23-25 March 2010. Paper SPE 128245.**
28. ***Real-Time Asset Management: From Vision to Engagment – An Operator's Experience.*** Unneland, T and Hauser, M. **SPE Annual Technical Conference and Exhibition, Dallas, Texas, USA : s.n., 9-12 October 2005. Paper SPE 96390.**
29. ***Evolution Of Decision Environments: Lessons Learned From Global Implementations And Future Direction Of Decision Environments.*** Hauser, M. and Gilman, H. **Presented at SPE Intelligent Energy Conference and Exhibition, Amsterdam, The Netherlands : s.n., 25-27 February 2008. Paper SPE 112215.**
30. ***Strategies for Training Digital Petroleum Engineers.*** Ershaghi, I., Paul, D. and Mahdavi, M. **Presented at SPE digital Energy Conference and Exhibition in Houston, Texas, USA : s.n., 7-8 April 2009. Paper SPE 123896.**
31. ***Integrated Reservoir and Production Management: Solutions and Field Examples.*** Bøe, A., Nordtvedt, J-E, Schmidt, H. S. **SPE European Petroleum Conference, Paris, France, 24-25 October 2000. SPE 65151.**
32. ***Development of a Marginal Gas-Condensate Field Using a Novel Integrated Reservoir and Production Management Approach.*** Nygaard O., Kramer C., Kulkarni R., Nordtvedt J.E. **SPE Asia Pacific Oil and Gas Conference and Exhibition, Jakarta, Indonesia, 17-19 April 2001. Paper SPE 68734.**
33. Epsis homepage. ***www.epsis.no.*** **[Online]**
34. ***Implementing "I Field" Initiatives in a Deepwater Green Field, Offshore Nigeria .*** Adeyemi, O.S., Shryock, S.G., Sankaran, S., Hostad, O. and Gontijo, J. **Presented at ATCE in Dallas, Colorado, USA : s.n., 21-24 September 2008. Paper SPE 115367.**

35. ***Installation and Implementation of "Smart Fields Foundation" on a Brown Field Asset, Adding Value Without Major Capital Investment.*** Gerrard, C. A., McCabe, H., and Beck, A. D. **Presented at SPE Intelligent Energy Conferance and Exhibition in Utrecht, The Netherlands : s.n., 23-25 March 2010. Paper SPE 127849.**
36. ***Continous Improvement the step after IO implementation.*** Sandstad, B.:. **IO conference Stavanger, Norway : s.n., 2007.**
37. ***The use of Real-Time Data at the Statfjord Anno 2005.*** Militer. J., et al.:. **Paper SPE 99257.**
38. ***BP Norways's FIELD OF THE FURTURE Implementation - A case Study.*** Hocking, P., and Shahly, M.:. **Paper SPE 112147.**
39. ***Enablers for the Succesful Implementation of Intelligent Energy: The Statoil Case.*** Henriquez, A., Fjærtoft, I, Johnsen, C., Yttredal, O. and Gabrielsen, T. **Presented at Intelligent Energy Conference and Exhibition held in Amsterdam, The Netherlands : s.n., 25-27 February 2008. Paper SPE 111470.**
40. OLF. **Oppdatert verdipotensiale for Integrerte Opersjoner på norsk sokkel.** ***http://www.olf.no/getfile.php/zKonvertert/www.olf.no/Rapporter/Dokumenter/080125%20Oppdatering%20av%20verdipotensialet%20i%20IO.pdf.*** **[Online] December 2007.**
41. ***A Method for Determination of Average Pressure in a Bounded Reservoir.*** Matthews, C. S., Brons, F. and Hazebroek, P. **Trans. AIME, 201, 182-191, 1954. .**
42. Bourdet, D. ***Well Test Analysis; the use of Advanced Interpretation Models.*** **Amsterdam, Netherlands : Elsevier Science B.V., 2002. first edition.**
43. Adams, Robert A. ***Calculus, A complete Course.*** **third edition : Addison-Wesley Publisher Limited.**
44. ***New Solutions for Well-Test-Analysis Problems: Part2 – Computational Considerations.*** Ozkan, E. and Raghavan, R. **SPEFE, 369-378, September 1991. .**
45. Sabet, M.A:. ***Well test Analysis.*** **Houston, Texas : Gulf Publishing Company, 1991. ISBN 0-87201-584-X.**
46. Horne, R. ***Modern Welltest Analysis; A Computer-Aided Approach.*** **s.l. : Petroway, Palo Alto, CA, USA, 2000. second edition.**
47. ***Use of Pressure Derivative in Well-Test Interpretation.*** Bourdet, D., Ayoub, J. A. and Picard, Y. M. **SPEFE, 293-302, June 1989. Paper SPE 12777.**
48. Marquardt, D. W. ***An algorithm for Least-Square Estimation of Non-Linear Parameters.*** **JSIAM 11, 431-441, 1963. .**
49. Netlib, Free Online Software library. ***www.netlib.org.*** **[Online]**
50. ***None-Linear Regression Algorithm.*** Burton, S., Garbow, Kenneth, E., Hillstrom and Jorge, J. **Argonne National Laboratory, Mar. 1980. .**
51. Abramowitz, M., and Stegun, I.A. ***Handbook of Mathematical Functions, Applied Mathematics Series, Volume 55.*** **Washington: National Bureau of Standards: reprinted 1968 by Dover Publications, New York, §9.8, 1964. .**
52. Numerical Recipes Software. ***www.library.cornell.edu/nr/.*** **[Online] Published by Cambridge University Press. www.library.cornell.edu/nr/.**
53. ***Adapting to Unknown Smoothness via Wavelet Shrinkage.*** Donoho, D. L. and Johnstone, I. M. **J. Am. Statist. Ass., 90, 1200-1224, 1995 . .**
54. ***Ideal Spatial Adaptation by Wavelet Shrinkage.*** Donoho, D. L. and Johnstone, I. M. **Biometrika, 81, 425-455, 1994 . .**
55. ***Wavelet Shrinkage: Asymptopia? (with discussion).*** Donoho, D. L., Johnstone, I. M., Kerkyacharian, G. and Picard, D. **J. R. Statist. Soc. B, 57, 301-337, 1995. .**
56. ***Wavelets and their Applications.*** Ruskai, M.B, Beylkin, G, Coifman, R., Daubechies, I., Mallat, S., Meyer, Y. and Raphael, L.:. **Boston MA USA : Jones and Bartless, 1992.**

57. ***Wavelets, Time-Frequency Methods and Phase Space.*** Combes, J.M., Grossman, A., and Tchamitchian, P.:. **Berlin : Springer-Verlag, 1989.**
58. Mallat, S. ***A Wavelet Tour of Signal Processing.*** **Academic Press, San Diego, CA 92101-4495, 2001. Second edition.**
59. ***Orthonormal bases of compactly supported wavelets.*** Daubechies, I. **s.l. : Communications on Pure and Applied Mathematics, 41:909-996, November 1988.**
60. Burrus C. S., Gopinath R. A. and Guo H. ***Introduction to Wavelets and Wavelets Transforms, A Primer.*** **Prentice Hall, Upper Saddle River, NJ 07458 : s.n., 1998.**
61. Gao, H.-Y. and Bruce, A. G. ***Waveshrink with Firm Shrinkage.*** **Statist. Sinica, 7, 855-874, 1997. .**
62. ***Wavelet Shrinkage Denoising Using the Non-Negative Garrote.*** Gao, H.-Y. **J. Comp. Graph. Statist., 7, 469-488, 1998. .**
63. ***Variable Selection via Nonconcave Penalized Likelihood and its Oracle Properties.*** Fan, J. and Li. R. **J. Am. Statist. Ass., 96, 1999. .**
64. ***Incorporating Information on Neighboring Coefficients into Wavelet Estimation.*** Cai, T. T. and Silverman, B. W. **Sankhya, Serie A, 63, 2001. .**
65. ***Wavelet Threshold Estimators for Data with Correlated Noise.*** Johnstone, I. M. and Silvermann, B.W. **J. Roy. Statist. Soc. Series B, 59, 319-351., 1997. .**
66. ***Comparing Automatic Smoothers (a Public Service Enterprise).*** Breiman, L. and Peters, S. **Int Statist. Rev., 60, 271-290, 1992. .**
67. ***Adaptive Wavelet Estimation: a Block Thresholding and Oracle Inequality Approach.*** Cai, T. T. **Ann. Statist., 27, 898-924, 1999. .**
68. ***Understanding Waveshrink: Variance and Bias Estimation.*** Bruce, A. G. and Gao, H.-Y. **Biometrika , 83, 727-745., 1996. .**
69. ***Estimation of the Mean of a Multivariate Normal Distribution.*** Stein, C. **Ann. Statist., 9, 1135-1151., 1981. .**
70. ***From Model Selection to Adaptive Estimation.*** Birgé, L. and Massart, P. **D. Pollard (ed), Festchrift for L. Le Cam, Springer, 55-87., 1997. .**
71. ***Thresholding of Wavelet Coefficients as Multiple Hypothesis Testing Procedure.*** Abramovich, F. and Benjamini, Y. **Wavelets and Statistics, Antoniadis, A. & Oppenheim, G. (Eds.), Lecture Notes in Statistics, 103, Springer-Verlag NY, 5-14., 1995. .**
72. ***Change-Point Approach to Data Analytic Wavelet Thresholding.*** Ogden, R. T. & Parzen, E. **Statist. Comput., 6, 93-99, 1996. .**
73. ***Wavethresh.*** Nason, G. P. **Software package available from Statlib, 1993. .**
74. ***Wavelet, Approximation, and Statistical Applications.*** Härdle, W., Kerkyacharian, G., Pikard, D. and Tsybakov, A. **Lecture Notes in Statistics, 129, NY, Spriner Verlag, 1998. .**
75. ***A real-time algorithm for signal analysis with the help of the wavelet transform.*** Holschneider, M., Kronland-Martinet, R. Morlet, J. and Tchamitchian, P. **In Wavelets, Time-Frequency Methods and Phase Space, pp. 289–297. Springer-Verlag , 1989. .**
76. ***On the Representation of Operators in Bases of Compactly Supported Wavelet.*** Beylkin, G. **SIAM Journal of Numerical Analysis, 29, 1716-1740, Dec. 1992. .**
77. ***The Discrete Wavelet Transform: Wedding the a Trous and Mallat Algorithms.*** Shelsa, M. J. **IEEE Transactions on Information Theory, 40, 2464-2483, 1992. .**
78. ***Extending the Scope of Wavelet Regression Methods by Coefficient-Dependent Thresholding.*** Kovac, A. and Silverman B. W. **J. Am. Statist. Ass., 95, 172-183, 2000. .**
79. Eclipse. **http://www.oilfield.slb.com/content/services/software/reseng/eclipse_simulators/index.asp.** ***Reservoir Simulator.*** **[Online] Schlumberger.**

80. ***Automated Reservoir Model Selection in Well Test Interpretation.*** Guyaguler, B., Horne, R. N. and Tauzin, E. **Presented at SPE ATCE in New Orleans, Lousiana, USA, September 2001. Paper SPE 71569.**
81. Rice, J. A. ***Mathematical Statistics and Data Analysis.*** **Second Edition, Duxburg Press, Belmont, CA. : s.n.**
82. *The Sage Dictionary of Statistics.* Cramer, Duncan and Howitt, Dennis. **p. 76, 2004. ISBN 076194138X.**
83. Mathworks, Matlab from. **Mathematical computational software .** ***www.mathworks.com.*** **[Online]**
84. ***Computing and applied mathematics laboratory.*** Saunders, B. V. and Boisvert, R. F. **National Institute of Standards and Technology, Gaithersburg, MD 20899 (NIST), . .**
85. 2003, Microsoft Visual Studio. **Microsoft .NET Framework.** ***www.microsoft.com.*** **[Online]**
86. Donoho, D., Duncan, M. R., Huo, X. and Levi, O. **Wavelab 802 for Matlab5.x. . [Online]**
87. ***Compression of Downhole Data.*** Bernasconi, G., Rampa, V., Abramo, F. and Bertelli, L. **Presented at SPE/IADC Drilling Conferance, Amsterdam, 9-11 March 1999. Paper SPE/IADC 52806.**
88. ***The Digital Oil Field of the Future: Enabling Next Generation Reservoir Performance.*** CERA. **s.l. : Cambridge Energy Research Associates, May 2003.**
89. ***The Art of Intelligent Energy (iE) - Insights and lessons Learned from the Application of iE.*** Edwards, T., Mydland, Ø. and Hneriquez, A.:. **Presented at SPE Intelligent Energy Conference and Exhibition held in Utrecht, The Netherlands : s.n., 23-25 March 2010. Paper SPE 128669.**
90. ***Wavelet and Applications.*** Meyer, Y. **Proceedings of the Marseille Workshop on Wavelet, France, May 1989 : Springer Verlag, Berlin, 1992.**

List of Figures

List of Tables

Appendix A - Bessel Functions

The Stehfest routine described in chapter 3 adds together several successive large and small values as given by *eq. (3.23) - (3.25)*. This summation will require an accurate estimate of the P_D function in *eq. (3.22)*. Since the P_D function in well-test analysis usually contains Bessel functions **(42; 44)**, it is very important to have Bessel functions with high accuracy in order to get accurate solutions with the Stehfest algorithm. In addition, the large number of Bessel function evaluations requires that the Bessel functions can be calculated computationally efficient.

We will in this appendix describe how to obtain both fast and accurate modified Bessel function for use in well-test analysis. Finally, we will provide a summary of the derivatives for all the Bessel functions that will be useful in the calculation of the pressure derivatives in non-linear regression analysis.

A.1 Equations

The modified Bessel functions are series that are solutions of the differential equation;

$$\text{(A.1)} \qquad x^2 \frac{d^2 y}{dx^2} + x\frac{dy}{dx} - (x^2 + n^2)y = 0 \qquad n \in \Re \quad ,$$

where the solution of the above equation is given as;

$$\text{(A.2)} \qquad y = c_1 I_n(x) + c_2 K_n(x) \quad .$$

$I_n(x)$ and *$K_n(x)$* are referred to as the modified Bessel function of first and second order. *c_1* and *c_2* are constants. As a series expansion, the modified Bessel function of first order may be written as;

$$\text{(A.3)} \qquad I_n(x) = \left(\frac{1}{2}x\right)^n \sum_{k=0}^{\infty} \frac{(\frac{1}{4}x^2)^k}{k!\Gamma(n+k+1)}$$

where *n* is given in *eq. (A.1)* and Γ is the gamma function**(51)**. The modified Bessel function of second order is given as;

$$\text{(A.4)} \qquad \begin{aligned} K_n(x) = (-1)^{n+1}\left[\ln\left(\frac{1}{2}x\right)\right] I_n(x) + \\ (-1)^n \frac{1}{2}\left(\frac{1}{2}x\right)^n \sum_{r=0}^{\infty} \frac{\left(\frac{1}{4}x^2\right)^r}{r!(n+r)!}\left(\sum_{m=1}^{n+r} m^{-1} + \sum_{m=1}^{r} m^{-1}\right) + . \\ \frac{1}{2}\left(\frac{1}{2}x\right)^{-n} \sum_{r=0}^{n-1} (-1)^r \left(-\frac{1}{4}x^2\right)^r \frac{(n-r-1)!}{r!} \end{aligned}$$

Bessel Functions

The main problem by applying a straightforward implementation of the modified Bessel functions is the summation of infinite series. It will in general be very computational expensive to calculate these series with high enough precision. Thus, it is very common to use approximations for faster calculations of Bessel functions. Very often one looks at the evaluations of small and high function arguments separately. For K_n(x) and $I_n(x)$, the difference between low and high functions arguments are normally chosen to be around 2. For small function elements (<2) the modified Bessel functions can be approximated as;

(A.5) $$I_n(x) \approx \frac{\tfrac{1}{2}x}{\Gamma(n+1)} \qquad n \neq -1,-2,-3..... \qquad ,$$

(A.6) $$K_0(x) \approx -\ln(x)$$

and

(A.7) $$K_n(x) \approx \tfrac{1}{2}\Gamma(n)(\tfrac{1}{2}x)^{-n} \qquad n \in \Re, n > 0 \qquad .$$

For large function elements (>2) the modified Bessel functions can be approximated as follow;

(A.8)

$$K_n(x) \approx \left(\frac{\pi}{2x}\right)^{1/2}\left[1+\frac{4n^2-1^2}{1!8x}+\frac{(4n^2-1^2)(4n^2-3^2)}{2!(8x)^2}+\frac{(4n^2-1^2)(4n^2-3^2)(4n^2-5^2)}{3!(8x)^3}+......\right]$$

and

(A.9)

$$I_n(x) \approx \left(\frac{e^x}{\sqrt{\pi 2x}}\right)\left[1-\frac{4n^2-1^2}{1!8x}+\frac{(4n^2-1^2)(4n^2-3^2)}{2!(8x)^2}-\frac{(4n^2-1^2)(4n^2-3^2)(4n^2-5^2)}{3!(8x)^3}+......\right]$$

The number of terms in the summation will determine the precision of the approximations.

Three sources with implementations of the modified Bessel functions were originally considered in this work; Matlab**(83)**, Netlib**(49)** and Numerical Recipes**(52)**. Common for all three implementations is that they use approximations as described above for evaluation of the modified Bessel functions. Numerical Recipes use the implementation by Abramowitz and Stegun**(51)** while Matlab use the implementation by Amos (which they actually have obtained from Netlib as well). The final implementation considered is the one in Netlib (and referred to as the Netlib implementation) by Saunders and Boisvert**(84)**.

Since the well-test module will be written in C++**(85)** (computer programming language), the Bessel functions are required to be written in C or C++, they need to be converted to C or C++ or they can be linked into C++ as an external numerical library.

The Matlab code (Amos) was tested by linking Matlab to C++ while the Numerical Recipes code (Abramowitz and Stegun) is already written in C. The Netlib code (Saunders and Boisvert), written in Fortran, is converted to C by a converter found in Netlib[(49)] called *f2c*.

A.2 Precision Considerations

The Bessel functions found in Netlib, Matlab and Numerical Recipes are all approximations used to increase the computational speed of the Bessel functions. This since the full series implementation of the Bessel functions (*eq. (A.3)* and *eq. (A.4)*) is very computational expensive and would be too computational expensive for use in well-test analysis. However, to test the precision of these approximations, we test the approximations against the full series implementations of the Bessel functions. The full series implementation of the Bessel functions is highly accurate but computational expensive.

To test the various implementations of modified Bessel functions found in Netlib, Matlab and Numerical Recipes, modified Bessel functions with function arguments on the interval 0.01 to 100 with a step length of 0.01 were considered. The relative and absolute errors for the modified Bessel functions were calculated on the interval considered.

A.2.1 Matlab

First we test the Matlab implementation of the Bessel functions. The Matlab modified Bessel functions appear to be highly accurate and give a full precision answer (sixteen correct digits) for the modified Bessel functions. This holds for all modified Bessel functions considered, K_0, K_1, I_0, and I_1. The performance was tested against the complete series implementation given by *eq. (A.3)* and *eq. (A.4)*.

A.2.2 Netlib

The modified Bessel functions K_0, K_1, I_0, and I_1 from Netlib by Saunders and Boisvert were considered next. The relative and absolute errors were obtained for each function and the results are discussed in the subsequent sections.

Bessel K_0

The Netlib algorithm calculates the modified Bessel function $K_0(x)$ correct up to 7 digits on average. The absolute error is largest for function arguments around 8, while the relative error is more uniform. This is shown in *Figure. A.1.* The average, minimum and maximum error for the Netlib $K_0(x)$ function is summarized in *Table A.1*.

Table A.1: *Average absolute and relative error for Netlib $K_0(x)$ function.*

	Absolute	Relative (%)
Average error	*3.60796e-14*	*1.06061e-5*
Minimum Error	*0*	*0*
Maximum Error	*4.00624e-12*	*1.45921e-5*

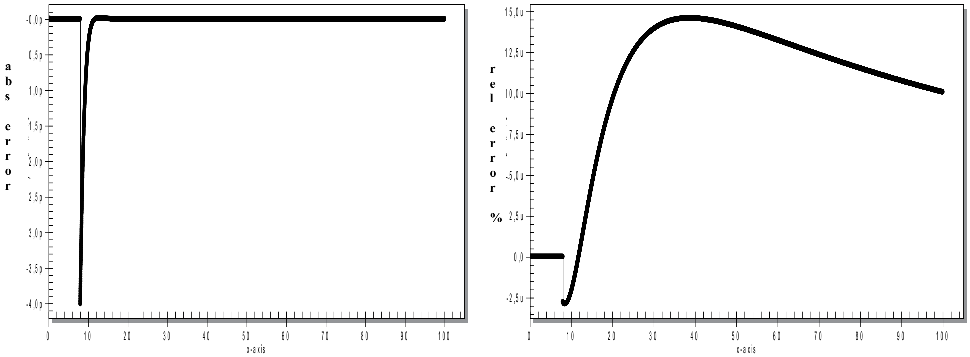


Figure A.1*: Absolute and relative error for Netlib Bessel $K_0(x)$ function for different function arguments. The y-axis unit in the left plot is $p=10^{-12}$ while the y-axis unit in the right plot is $\mu=10^{-6}$. (Note that the relative error is shown in percent, i.e. multiplied with 100.)*

Bessel K$_1$

As for the K_0 function, the Netlib $K_1(x)$ function calculates the Bessel function correct up to 7 digits on average. The absolute error is largest for function arguments around 8, while the relative error again is more uniform. This is shown in *Figure A.2.* The average, minimum and maximum error for Netlib $K_1(x)$ function is summarized in *Table A.2*.

Table A.2: *Average absolute and relative error for Netlib $K_1(x)$ function.*

	Absolute	**Relative (%)**
Average error	*2.09082e-012*	*1.65641e-5*
Minimum Error	*0*	*0*
Maximum Error	*3.40915e-010*	*1.42518e-5*

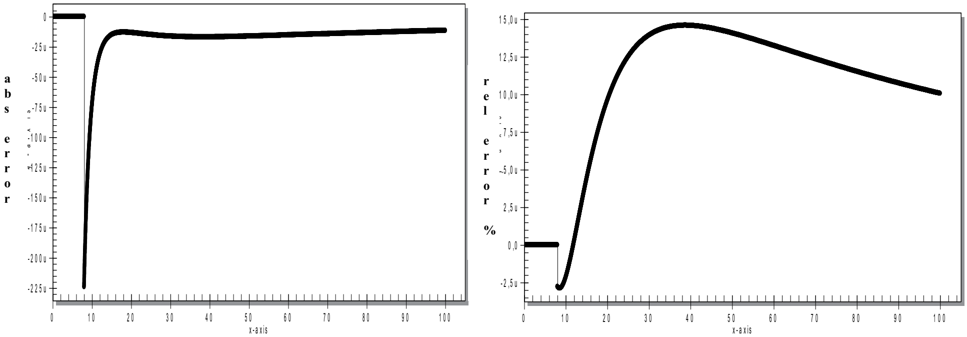


Figure A.2*: Absolute and relative error for Netlib $K_1(x)$ function for different function arguments. The y-axis unit in the left plot and the right plot is $\mu=10^{-6}$. (Note that the relative error is shown in percent, i.e. multiplied with 100.)*

Bessel I_0

The Netlib $I_0(x)$ function is highly accurate and calculates the function correct up to 15 digits in average, and never less than 14 digits. This is not far from the precision to a float number within C++ of 16 digits. Since the $I_0(x)$ function approaches infinite as the function argument becomes large, the absolute error will become higher for large function arguments. This is shown in *Figure A.3.* The average, minimum and maximum error for *Netlib $I_0(x)$* function is summarized in *Table A.3*.

Table A.3: *Average absolute and relative error for Netlib $I_0(x)$ function.*

	Absolute	Relative (%)
Average error	6.91e25	2.91e-13
Minimum Error	0	0
Maximum Error	1.87e28	2.15e-12

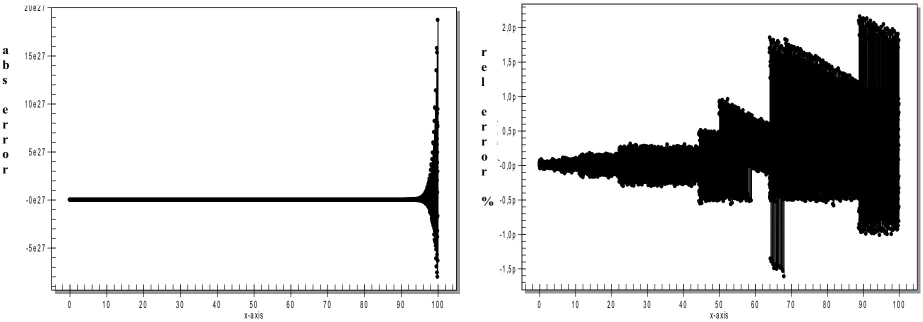


Figure A.3*: Absolute and relative error for Netlib Bessel $I_0(x)$ function for different function arguments. The y-axis unit in the right plot is $p=10^{-12}$. (Note that the relative error is shown in percent, i.e. multiplied with 100.)*

Bessel I_1

As for the $I_0(x)$, the Netlib $I_1(x)$ is highly accurate and calculates the function correctly up to 15 digits in average, and never less than 14 digits. This is shown in *Figure A.3.* The average, minimum and maximum error for *Netlib $I_1(x)$* function is summarized in *Table A.4.*

Table A.4: *Average absolute and relative error for Netlib $I_1(x)$ function.*

	Absolute	Relative (%)
Average error	6.88e25	2.90e-13
Minimum Error	0	0
Maximum Error	1.84e28	2.17e-12

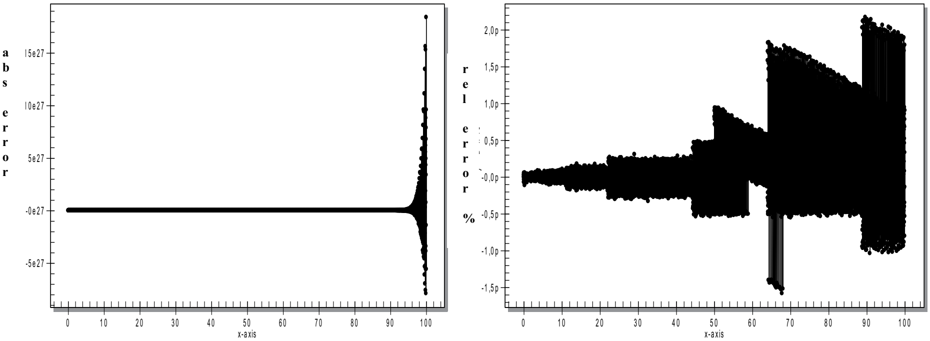


Figure A.4*: Absolute and relative error for Netlib Bessel $I_1(x)$ function for different function arguments. The y-axis unit in the right plot is $p=10^{-12}$. (Note that the relative error is shown in percent, i.e. multiplied with 100.)*

A.2.3 Numerical Recipes

The implementation in Numerical Recipes is very similar to the one found in Netlib and the results are close to those from Netlib. Numerical Recipes in general gives modified Bessel functions with an accuracy of 6-7 digits for $K_o(x)$ and $K_1(x)$ and with an accuracy of 14-15 digits for $I_o(x)$ and $I_1(x)$.

A.3 Computational Performance

If only precision was important, modified Bessel functions in Matlab should have been used, however the speed is equally important as the precision. The different Bessel functions were tested in order to determine their computational speed. All calculations were done at a Dell Dimension 1.8 GHz laptop with 512 MB memory. All functions were calculated four million times with different functions arguments from 0.0001 to 400 with an interval of 0.0001.

A.3.1 Netlib

The results obtained for the Bessel functions in Netlib are shown in *Table A.5*.

Table A.5: *Computational performance for evaluation of Bessel functions applying Netlib's Bessel functions.*

	K_0(x)	**K_1(x)**	**I_0(x)**	**I_1(x)**
Time(secs)	≈1	≈1	≈8	≈8

The results show that the Bessel functions $K_o(x)$ and $K_1(x)$ are fast while the Bessel functions $I_o(x)$ and $I_1(x)$ are considerably slower.

A.3.2 Numerical Recipes

The results obtained from the Bessel functions in Numerical Recipes are shown in *Table A.6*.

***Table A.6**: Computational performance for evaluation of Bessel functions applying Numerical recipe's Bessel functions.*

	$K_0(x)$	$K_1(x)$	$I_0(x)$	$I_1(x)$
Time(secs)	≈1	≈1	≈1.5	≈1.5

The results show that all Bessel functions are computational fast.

A.3.3 Matlab

It is possible to call Matlab's Bessel functions by direct calls from C++ code. The computational performance using direct calls was poor. By comparing the performance of the Bessel functions made by these external calls from our C++ code, and from the calculations performed directly within the Matlab environment, we observe that the overhead of calling Matlab functions from C++ is large. However, it should in fact be noted that the performance in both cases is rather poor as shown in *Table A.7* and *Table A.8.*

***Table A.7**: Computational performance in Matlab for evaluation of modified Bessel functions by linking to C++.*

	$K_0(x)$	$K_1(x)$	$I_0(x)$	$I_1(x)$
Time(secs)	≈103	≈105	≈115	≈113

***Table A.8**: Computational performance in Matlab for evaluation of modified Bessel functions inside Matlab environment.*

	$K_0(x)$	$K_1(x)$	$I_0(x)$	$I_1(x)$
Time(secs)	≈28	≈28	≈33	≈33

The Bessel functions $I_0(x)$ and $I_1(x)$ from Numerical Recipes ensures both high speed and high precision. This is sufficient for the requirements we have for the Bessel functions for use with the numerical Laplace inversion scheme. Also the $K_0(x)$ and $K_1(x)$ performs well for small functions arguments, however the performance for large function arguments are rather poor. The main problem with the implementation found within Netlib by Saunders and Boisvert is the calculation speed of $I_o(x)$ and $I_1(x)$ and the precision of $K_0(x)$ and $K_1(x)$ for large function arguments. The main problem with linking to Matlab is poor performance, although the precision is excellent. The speed is in most cases reduced by a factor 100 when linking to Matlab.

For $I_0(x)$ and $I_1(x)$ we choose to use the implementations found in Numerical Recipes. For $K_0(x)$ and $K_1(x)$ we choose to use the implementations found in Numerical Recipes for small functions arguments, but improvements are required for large function arguments (x>2) and will be described in the next section.

A.4 Improvements

In Numerical Recipes, approximations of the functions $K_0(x)$ and $K_1(x)$, are given on the form:

$$K_n(x) = \left(\frac{\pi}{2x}\right)^{1/2} e^{-x} \left[\sum_{i=0}^{m} \frac{a_n}{x^n}\right] \tag{A.10}$$

However, as mentioned earlier, the precision of the modified Bessel functions implemented in Numerical Recipes is smaller than we would like to have for large function arguments. By simply increasing the size of the polynomial in the approximation above, high precision modified Bessel functions can be obtained. Thus, we chose to use the approximation above with a polynomial of 12 terms. The actual implementation of these Bessel functions is given in Appendix A.6. The absolute and relative error for the new Bessel functions is shown in *Figure A.5* and *Figure A.6.*

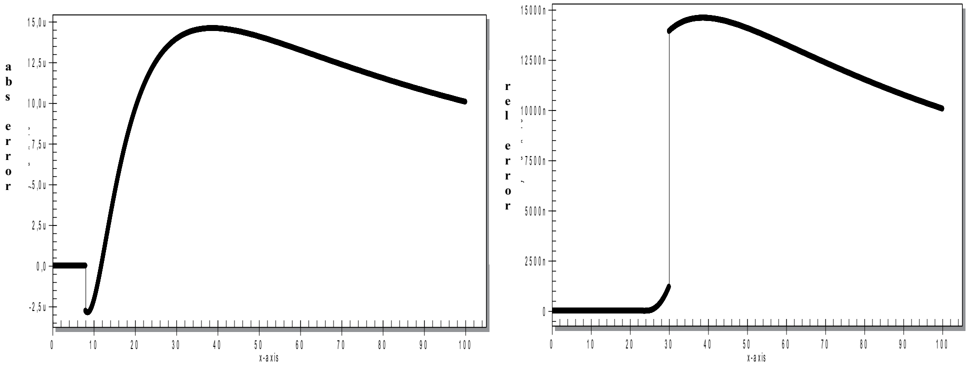


***Figure A.5**: Absolute and relative error in new implementation of $K_0(x)$ Bessel functions for different function arguments. The y-axis unit in the left plot is $\mu=10^{-6}$ while the y-axis unit in the right plot is $n=10^{-9}$. (Note that the relative error is shown in percent, i.e. multiplied with 100.)*

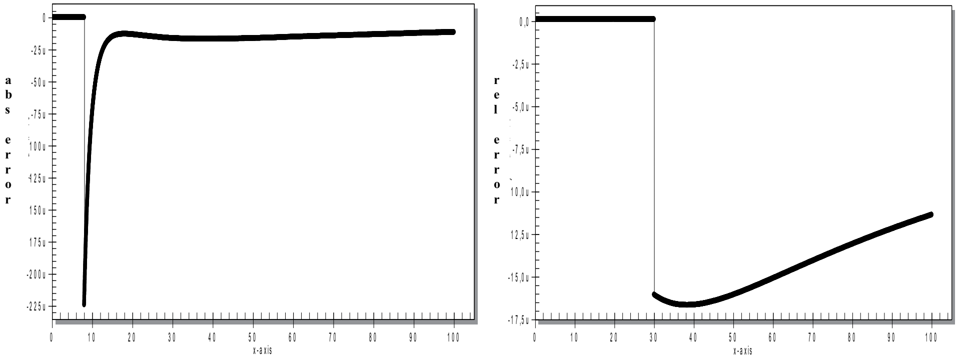


***Figure A.6**: Absolute and relative error in new implementation of $K_1(x)$ Bessel functions for different function arguments. The y-axis unit in the left and the right plot is $\mu=10^{-6}$. (Note that the relative error is shown in percent, i.e. multiplied with 100.)*

Table A.9: *Computational performance for the implemented Bessel functions.*

	K$_1$(x)	**K$_0$(x)**	**I$_1$(x)**	**I$_0$(x)**
Time(secs)	≈1	≈1	≈1.5	≈1.5

With the new Bessel functions, the accuracy is now 12-13 digits for the $K_0(x)$ and $K_1(x)$, for both small and large function arguments. Four millions function evaluations are done in 1-1.5 second using the same computer as previously. The computational performance for the different Bessel functions is summarized in *Table A.9*. The new Bessel functions are thus 100 times faster than the Bessel functions in Matlab, although with a slightly lower precision. The new implementation of $K_0(x)$ and $K_1(x)$ has 5-6 digits higher precision than in Numerical Recipes for large function arguments but are evaluated at the same speed.

A.5 Derivatives of Bessel Functions

When performing well-test non-linear regression analysis, we need analytical derivatives of the Bessel functions to avoid using numerical derivatives that both would both yield a larger error and secondly be slower (since they often include twice the number of function evaluations). The derivatives of the modified Bessel functions can be found in most calculus books(38;73) and are given as:

(A.11) $$\frac{\partial K_0(x)}{\partial x} = -K_1(x)$$

(A.12) $$\frac{\partial K_1(x)}{\partial x} = -K_0(x) - \frac{K_1(x)}{x}$$

(A.13) $$\frac{\partial I_0(x)}{\partial x} = I_1(x)$$

(A.14) $$\frac{\partial I_1(x)}{\partial x} = I_0(x) - \frac{I_1(x)}{x}$$

A.6 Numerical Implementation

This appendix contains some of the implementation of the Stehfest Laplace inversion scheme and the Bessel functions. For practical reasons, only a limited amount of the code is given in here. All implementation is done in C++ applying Microsoft Visual Studio 2003**(85)**.

A.6.1 Inverse Laplace Transform

The following files and C++ code has been written for the numerical implementation of the Stehfest Laplace inversion scheme. A subclass must be written to the *LaplaceBase* class to implement the function *CalculateLaplaceArgument* which evaluates the actual Laplace function.

Bessel Functions

File LaplaceBase.h

```
#pragma once

class SLaplaceBase : public SDerivativeBase
{
public:
    SLaplaceBase( ):
    virtual ~SLaplaceBase( ):
    virtual double CalculateLaplaceArgument(double s) = 0:
}:
```

File Laplace.h

```
#pragma once

#include "SLaplaceBase.h"
#include <vector>

class SLaplace
{
public:
    SLaplace(SLaplaceBase* base):
    virtual ~SLaplace(void):

    double InversStehfestTransform( double s, std::vector<double>& viTable):

protected:
    std::vector<double> m_viTable:

    static void CalculateViTable(int N,std::vector<double>& viTable):

    SLaplaceBase* m_base:

}:
```

File Laplace.cpp

```
#include "slaplace.h"
#include "slaplacebase.h"
#include "smath.h"
#include <math.h>
#include <limits>


SLaplace::SLaplace(SLaplaceBase* base)
{
    m_base = base:


}

SLaplace::~SLaplace(void)
{
}
///////////////////////////////////////////////////////////////////////
//
//
//
////////////////////////////////////////////////////////////////////////
void SLaplace::CalculateViTable(int N, std::vector<double>& viTable )
{
    double sum = 0:
    int q = (int)( N / 2):

    std::vector<double>fak_tab:
    fak_tab.resize( N + 1):
    fak_tab[0] = 1:

    for(int k = 1: k <= N : k++)
        fak_tab[ k ] = fak_tab[ k - 1 ] * k:

    viTable.resize( N ):
```

```
  for(int i = 1: i <= N : i++)
  {
    double sum = 0:
    int K = (int)(( i + 1) / 2) :
    int M = __min( q , i):

    for(int j = K : j <= M : j++)
    {
      sum += pow( j , q + 1) *
        fak_tab[ 2 * j ] /
        fak_tab[ q - j ] /
        fak_tab[ j ] /
        fak_tab[ j ] /
        fak_tab[ i - j ] /
        fak_tab[ 2 * j - i ]:
    }

    //set new vi
    viTable[i-1] = pow(-1, q + i ) * sum:
  }
}


//////////////////////////////////////////////////////////////////////
//
//
//
//////////////////////////////////////////////////////////////////////
double SLaplace::InversStehfestTransform(double s, std::vector<double>& viTable)
{
  double pd = 0,si,pds:

  for( unsigned i = 1: i <= viTable.size() : i++)
  {
    si = log(2.0) * i / s:
                  pds = m_base->CalculateLaplaceArgument( si ):
    pd += viTable[ i - 1 ] * pds:

  return log(2.0) / s * pd:
}
```

A.6.2 Bessel Functions

The implemented Bessel functions for small function arguments in $K_0(x)$ and $K_1(x)$ and for all arguments to $I_0(x)$ and $I_1(x)$ are equal to the implementation in Numerical Recipes. Modifications are introduced for large function arguments in the evaluation of $K_0(x)$ and $K_1(x)$.

```
#include ".\sbesselfunction.h"
#include <math.h>


const double SBesselFunction::polytabK0_2[11] = {    9.33474288285696, 3.04214327567263, -0.351411991966095, -0.558108807018695,
-0.0140618292519901, 0.00582065816805504, 0.0764607816969116, -0.0836772538861002,
0.0875262546036897,-0.156640362498542, 1.25331373336153}:

const double SBesselFunction::polytabK0_3[11] = {0.480114948051918,0.103846117801027,-0.150842128124707, 0.51512251369496, -
0.10338592177391,-0.121327975433006, 0.12028360285608, -0.0902211469515844, 0.0880491093088693,-0.156662294258367,
1.2533141149068 }:

const double SBesselFunction::polytabK0_4[11] ={ 0.872825749408107, 0.460464482628261, 0.280325879465922,-0.352810259427467, -
0.149037838405807,-0.100080483324538, 0.121167762749324, -0.0906639398140896, 0.0880860003561732,-0.156663595449291,
1.25331413231261 }:

const double SBesselFunction::polytabK0_5[11] = {  7.83950456709176, 8.93375021874497,5.35964076581855, 1.19181812702166,
13.0905289572043, 2.55971945564961, -0.10424924636207, -0.0806567849538216,
0.0878412531147571, -0.156660481572338, 1.25331411633321 }:
```

```
const double SBesselFunction::polytabK0_6[11] = { 0.516212169302723,0.225353963750184,0.183819301703172,0.216071222249249,
0.421331952791799,0.884796700680851,0.00421010756282323,-0.0853862798741548,
0.0879728661129094,-0.156662493639418,1.25331412897483}:

const double SBesselFunction::polytabK1_2[11] = { -3.69542031928046, -7.82526935544691, 1.16170179992147, 0.926141255160686,
-0.207204051005454, 0.0730970467785583, -0.117246171581445,0.120278239349953,
-0.14625820246747,0.469967993103062,1.25331455902848 }:


const double SBesselFunction::polytabK1_3[11] = { 9.06205834354389,29.4377721695286,-8.72498963426441,-
2.36542384211555,1.32460846506163, -0.0942394895926431, -0.128760373763683,0.124756486680772,0.146707023134645,
0.469988695216057, 1.2533141812342 }:

const double SBesselFunction::polytabK1_4[11] = { 0.516158228830742,0.225218160392226,0.183327115095412,0.212625150629218,
0.392734004030057,0.71879898369353,-0.292453382201777,0.139074336564685,
0.147348143484178,0.470003289540679,1.25331404583092 }:

const double SBesselFunction::polytabK1_5[11] = {  4.38982009207877, 3.11303384610487,2.08587610524995, 1.39731575465267,
5.37367536164528,  1.5373298745296,-0.298822530074184,  0.134441291001168,
0.147034286688754,0.469995090391121,1.25331412405964}:


const double SBesselFunction::polytabK1_6[11] = { 0.516145962891471,0.225494508374971,0.183809581250699,0.215800013270206,
0.414925879996119,0.806413168188277,-0.244388406667257,0.131657490182756,
0.146948409644669,0.469993702859215,1.25331413304212}:

const double SBesselFunction::polytabK0divK1[6] = { -0.083392410135614, 0.236935274253949,-0.32473168821664,
0.370555356536421, -0.499882244766124,0.99999964263615 }:




/////////////////////////////////////////////////////////////////////////
///
///
///
///
/////////////////////////////////////////////////////////////////////////
double SBesselFunction::BesselK0(double x)
{
        double ans = 0:
        double y:
        const double* tab:


        if( x < 2.0 )
        {
                y = x * x / 4.0:
                ans = ( - log ( x / 2.0 ) * BesselI0( x ) ) + ( - 0.57721566 + y * ( 0.42278420 + y * ( 0.23069756 + y * ( 0.3488590 *
                0.1 + y * ( 0.262698 * 0.01 + y * ( 0.10750 * 0.001 + y * 0.74 * 0.00001 ) ) ) ) ) ):
        }
        else if( x < 50 )
        {
                if( x < 10 )
                        tab = polytabK0_2:
                else if( x < 20 )
                        tab = polytabK0_3:
                else if( x < 30 )
                        tab = polytabK0_4:
                else if( x < 40 )
                        tab = polytabK0_5:
                else if( x < 50 )
                        tab = polytabK0_6:

                double cur_x = 1:

                for( int i = 0: i < 11 : i++ )
                {
                        ans += cur_x * tab[ 10 - i ]:
                        cur_x /= x:
                }

                ans = ans * exp( - x ) / sqrt( x ) :
        }
        else
        {
```

```
        y = 2.0 / x:
        ans = ( exp( -x ) / sqrt( x ) ) * ( 1.25331414 + y * ( - 0.7832358 * 0.1 + y * ( 0.2189568 * 0.1 + y * ( -0.1062446 * 0.1
        + y * ( 0.587872 * 0.01 + y * ( -0.251540 * 0.01 + y * 0.53208 * 0.001 ) ) ) ) ) ):
    }

    return ans:
}


//////////////////////////////////////////////////////////////////////
///
///
///
///
//////////////////////////////////////////////////////////////////////
double SBesselFunction::BesselK1(double x)
{
    double ans = 0:
    double y:
    const double* tab:

    if( x <= 2.0 )
    {
        y = x * x / 4.0:
        ans = (  log ( x / 2.0 ) * BesselI1( x ) ) + ( 1.0 / x ) * ( 1.0 + y * ( 0.15443144 + y * ( -0.67278579 + y * ( - 0.18156897
        + y * ( - 0.1919402 * 0.1 + y * ( - 0.110404 * 0.01 + y * ( - 0.4686 * 0.0001 ) ) ) ) ) ) ):
    }
    else if( x < 50 )
    {
        if( x < 10 )
            tab = polytabK1_2:
        else if( x < 20 )
            tab = polytabK1_3:
        else if( x < 30 )
            tab = polytabK1_4:
        else if( x < 40 )
            tab = polytabK1_5:
        else if( x < 50 )
            tab = polytabK1_6:

        double cur_x = 1:

        for( int i = 0: i < 11 : i++ )
        {
            ans += cur_x * tab[ 10 - i ]:
            cur_x /= x:
        }

        ans = ans * exp( - x ) / sqrt( x ) :
    }
    else
    {
        y = 2.0 / x:
        ans = ( exp( -x ) / sqrt( x ) ) * ( 1.25331414 + y * ( 0.23498619
        + y * ( - 0.3655620 * 0.1 + y * ( 0.1504268 * 0.1 + y * ( -0.780353 * 0.01 + y * ( 0.325614 * 0.01 * y * ( -0.68245 *
        0.001 ) ) ) ) ) ) ):
    }

    return ans:
}




//////////////////////////////////////////////////////////////////////
///
///
///
///
//////////////////////////////////////////////////////////////////////
double SBesselFunction::BesselI1(double x)
{
    double ans:
    double y:
    double ax = fabs( x ):
        if( x < fabs( 3.75 ) )
        {
                y = x / 3.75:
                y = y * y:
                ans = ax * ( 0.5 + y * ( 0.87890594 + y * ( 0.51498869 + y * ( 0.15084934 + y * ( 0.2658733* 0.1 + y * ( 0.301532 *
                0.01 + y * 0.32411* 0.001 ) ) ) ) ) ):
        }
        else
        {
                y = 3.75 / ax:
                ans = 0.2282967 * 0.1 + y * ( - 0.2895312 * 0.1 + y * ( 0.1787654 * 0.1
                          - y * 0.420059 * 0.01 ) ):
                ans = 0.39894228 + y * ( - 0.3988024 * 0.1 + y * ( - 0.362018 * 0.01
                 + y * ( 0.163801 * 0.01 + y * ( - 0.1031555 * 0.1 + y * ans ) ) ) ):
                ans *= exp( ax ) / sqrt( ax ):
        }

        return x < 0.0 ? -ans : ans:
}








//////////////////////////////////////////////////////////////////////////
///
///
///
///
//////////////////////////////////////////////////////////////////////////
double SBesselFunction::BesselI0(double x)
{
        double ans:
        double y:
        double ax = fabs( x ):

  if( x < fabs( 3.75 ) )
        {
                y = x / 3.75:
                y = y * y:
                ans = 1.0 + y * ( 3.5156229 + y * ( 3.0899424 + y * ( 1.2067492
                + y * ( 0.2659732 + y * ( 0.360768 * 0.1 + y * 0.45813 * 0.01 ) ) ) ) ):
        }
        else
        {
                y = 3.75 / ax:
                ans = ( exp( ax ) / sqrt ( ax ) ) * ( 0.39894228 + y * ( 0.13285921 * 0.1 + y * ( 0.225319 * 0.01 + y * ( - 0.157565 *
                0.01 + y * ( 0.916281 * 0.01      + y * ( - 0.2057706 * 0.1 + y * ( 0.2635537 * 0.1 + y * ( - 0.1647633 * 0.1 + y *
                0.392377 * 0.01 ) ) ) ) ) ) ) ):

        }

        return ans:
}


//////////////////////////////////////////////////////////////////////////
///
///
///
///
//////////////////////////////////////////////////////////////////////////
double SBesselFunction::BesselK0divK1(double x)
{
        double ans = 0:
        double cur_x = 1:

        for( int i = 0: i < 6 : i++ )
        {
                ans += cur_x * polytabK0divK1[ 5 - i ]:
                cur_x /= x:
        }

        return ans:
}
```

A.7 Summary

We have in this appendix evaluated the accuracy and computational performance of Bessel functions found within three different sources, Matlab, Netlib and Numerical Recipes. These implementations of the Bessel functions were discovered to be either too slow or too inaccurate. Improvements are suggested in order to have computational efficient Bessel functions with high precision. This is required in order to get accurate solutions from numerical Laplace inversion with the Stehfest Laplace inversion scheme. Some examples of the actual numerical implementation in C++ were given as well.

Appendix B - Numerical Implementation Wavelet Transform

This appendix contains some of the numerical implementation of wavelet analysis used in this work. Out of practical considerations, it is not possible to list all implemented code in this appendix, but important parts of the numerical implementation is shown including all the scaling and wavelet coefficients, the forward stationary wavelet transform and the inverse stationary wavelet transform. The numerical implementation is written in C++ applying Microsoft Visual Studio 2003(85).

The numerical implementation has been verified versus Matlab(83) and Wavelab(86). Our implementation gives the same results as obtained in Matlab and Wavelab.

B.1 Wavelet and Scaling Coefficients

Below the scaling and wavelet coefficients for the wavelets considered in this work are given.

***Table B.1**: Scaling and wavelet coefficients for Haar wavelet.*

h(n)	***h_1(n)***
0.5	*0.5*
0.5	*-0.5*

***Table B.2**: Scaling and wavelet coefficients for Daubechies4 wavelet.*

h(n)	***h_1(n)***
0.3415063509462205	-0.09150635094586698
0.5915063509458669	-0.1584936490537795
0.1584936490537795	0.5915063509458669
-0.09150635094586698	-0.3415063509462205

***Table B.3**: Scaling and wavelet coefficients for Daubechies6 wavelet.*

h(n)	***h_1(n)***
0.2352336038927046	0.02490874986589096
0.5705584579173084	0.06041610415535419
0.3251825002637103	-0.0954672077842601
-0.0954672077842601	-0.3251825002637103
-0.06041610415535419	0.5705584579173084
0.02490874986589096	-0.2352336038927046

***Table B.4**: Scaling and wavelet coefficients for Daubechies8 wavelet.*

h(n)	***h_1(n)***
0.1629017140256182	-0.00749349466512713

0.5054728575456509	-0.02325180053555719
0.4461000691231943	0.02180815023738527
-0.01978751311790877	0.1322535836843695
-0.1322535836843695	-0.01978751311790877
0.02180815023738527	-0.4461000691231943
0.02325180053555719	0.5054728575456509
-0.00749349466512713	-0.1629017140256182

***Table B.5**: Scaling and wavelet coefficients for Daubechies10 wavelet.*

h(n)	***$h_1(n)$***
0.1132094912917304	0.002358713969200755
0.4269717713527106	0.008895935050925998
0.5121634721301556	-0.004413400054325186
0.09788348067375365	-0.05485132932107695
-0.1713283576913299	-0.02280056594204701
-0.02280056594204701	0.1713283576913299
0.05485132932107695	0.09788348067375365
-0.004413400054325186	-0.5121634721301556
-0.008895935050925998	0.4269717713527106
0.002358713969200755	-0.1132094912917304

***Table B.6**: Scaling and wavelet coefficients for Daubechies12 wavelet.*

h(n)	***$h_1(n)$***
0.07887121600143437	-0.0007617669025837149
0.3497519070375683	-0.003378031181505082
0.5311318799412127	0.000391625576034686
0.2229156614650506	0.02233187416547526
-0.1599932994458742	0.01946160485396355
-0.0917590320300334	-0.06894404648719725
0.06894404648719725	-0.0917590320300334
0.01946160485396355	0.1599932994458742
-0.02233187416547526	0.2229156614650506
0.000391625576034686	-0.5311318799412127
0.003378031181505082	0.3497519070375683
-0.0007617669025837149	-0.07887121600143437

***Table B.7**: Scaling and wavelet coefficients for Daubechies20 wavelet.*

h(n)	***$h_1(n)$***
0.01885857879639622	-9.37920788831568e-006
0.1330610913968657	-6.617718319909407e-005
0.3727875357426617	-8.235450295380118e-005
0.4868140553661002	0.0004849739199556974
0.198818870884399	0.00140884329496359
-0.1766681008964704	-0.0009866626824421689
-0.1385549393599319	-0.007589501167678906
0.09006372426665785	-0.002550218483932993
0.06580149355070224	0.02348490704840919
-0.05048328559800522	0.02082962404384584
-0.02082962404384584	-0.05048328559800522
0.02348490704840919	-0.06580149355070224
0.002550218483932993	0.09006372426665785
-0.007589501167678906	0.1385549393599319

0.0009866626824421689	-0.1766681008964704
0.00140884329496359	-0.198818870884399
-0.0004849739199556974	0.4868140553661002
-8.235450295380118e-005	-0.3727875357426617
6.617718319909407e-005	0.1330610913968657
-9.37920788831568e-006	-0.01885857879639622

Table B.8*: Scaling and wavelet coefficients for Symmlet4 wavelet.*

h(n)	$h_1(n)$
0.34150635094622	-0.09150635094586999
0.59150635094587	-0.15849364905378
0.15849364905378	0.59150635094587
-0.09150635094586999	-0.34150635094622

Table B.9*: Scaling and wavelet coefficients for Symmlet6 wavelet.*

h(n)	$h_1(n)$
0.2352336038927	0.02490874986589
0.57055845791731	0.06041610415535
0.32518250026371	-0.09546720778426
-0.09546720778426	-0.32518250026371
-0.06041610415535	0.57055845791731
0.02490874986589	-0.2352336038927

Table B.10*: Scaling and wavelet coefficients for Symmlet8 wavelet.*

h(n)	$h_1(n)$
0.0227851729479749	-0.0535744507089411
-0.008912350720840191	0.0209554825625269
-0.0701588120894228	0.351869534327613
0.210617267101768	-0.568329121704375
0.568329121704375	0.210617267101768
0.351869534327613	0.0701588120894228
-0.0209554825625269	-0.008912350720840191
-0.0535744507089411	-0.0227851729479749

Table B.11*: Scaling and wavelet coefficients for Symmlet10 wavelet.*

h(n)	$h_1(n)$
0.01381607647893	0.01932739797744
-0.01492124993438	-0.02087343221079
-0.12397568130675	-0.02767209305836
0.01173946156807	-0.14099534842729
0.44829082419092	0.51152648344605
0.51152648344605	-0.44829082419092
0.14099534842729	0.01173946156807
-0.02767209305836	0.12397568130675
0.02087343221079	-0.01492124993438
0.01932739797744	-0.01381607647893

Table B.12*: Scaling and wavelet coefficients for Symmlet12 wavelet.*

h(n)	***$h_1(n)$***
-0.00551593375469	0.01089235016328
0.00124996104639	-0.00246830618592
0.03162528132994	-0.08343160770584
-0.01489187564922	0.03416156079324
-0.0513624849309	0.34722898647835
0.23895218566605	-0.55694639196396
0.55694639196396	0.23895218566605
0.34722898647835	0.0513624849309
-0.03416156079324	-0.01489187564922
-0.08343160770584	-0.03162528132994
0.00246830618592	0.00124996104639
0.01089235016328	0.00551593375469

Table B.13*: Scaling and wavelet coefficients for Symmlet20 wavelet.*

h(n)	***$h_1(n)$***
-0.000324794948390885	0.000544585223622193
4.03306014989603e-005	-6.76225099711391e-005
0.00324786418934092	-0.00611032131704497
-0.000568767655336868	0.00103618196027345
-0.0143931159719293	0.0324754623014821
0.0040764083918897	-0.00820943470817067
0.0353517837811451	-0.112779486159984
-0.0226203861521081	0.0501201075064675
-0.0251282701703028	0.333535669214576
0.271406505551405	-0.544125765368736
0.544125765368736	0.271406505551405
0.333535669214576	0.0251282701703028
-0.0501201075064675	-0.0226203861521081
-0.112779486159984	-0.0353517837811451
0.00820943470817067	0.0040764083918897
0.0324754623014821	0.0143931159719293
-0.00103618196027345	-0.000568767655336868
-0.00611032131704497	-0.00324786418934092
6.76225099711391e-005	4.03306014989603e-005
0.000544585223622193	0.000324794948390885

Table B.14*: Scaling and wavelet coefficients for Coiflet6 wavelet.*

h(n)	***$h_1(n)$***
-0.051429728471	-0.011070271529
0.238929728471	0.051429728471
0.6028594569420001	0.272140543058
0.272140543058	-0.6028594569420001
-0.051429728471	0.238929728471
-0.011070271529	0.051429728471

Table B.15*: Scaling and wavelet coefficients for Coiflet12 wavelet.*

h(n)	***h_1(n)***
0.011587596739	-0.000509505399
-0.02932013798	0.001289203356
-0.04763959031	0.003967883613
0.273021046535	-0.016744410163
0.574682393857	-0.042026480461
0.294867193696	0.054085607092
-0.054085607092	0.294867193696
-0.042026480461	-0.574682393857
0.016744410163	0.273021046535
0.003967883613	0.04763959031
-0.001289203356	-0.02932013798
-0.000509505399	-0.011587596739

Table B.16*: Scaling and wavelet coefficients for Coiflet18 wavelet.*

h(n)	***h_1(n)***
-0.002682418671	-2.4465734e-005
0.005503126709	5.0192775e-005
0.016583560479	0.000329665174
-0.046507764479	-0.000790205101
-0.04322076356	-0.001820458916
0.286503335274	0.006369601011
0.56128525687	0.011229240962
0.302983571773	-0.024434094321
-0.050770140755	-0.058196250762
-0.058196250762	0.050770140755
0.024434094321	0.302983571773
0.011229240962	-0.56128525687
-0.006369601011	0.286503335274
-0.001820458916	0.04322076356
0.000790205101	-0.046507764479
0.000329665174	-0.016583560479
-5.0192775e-005	0.005503126709
-2.4465734e-005	0.002682418671

Table B.17*: Scaling and wavelet coefficients for Spline26 wavelet.*

h(n)	$\widetilde{h}(n)$
-0.0625	0
0.0625	-0
0.5	0.5
0.5	-0.5
0.0625	0
-0.0625	-0

Table B.18: *Scaling and wavelet coefficients for Reverse Spline26 wavelet.*

h(n)	$\tilde{h}(n)$
0	-0.0625
0	-0.0625
0.5	0.5
0.5	-0.5
0	0.0625
0	0.0625

Table B.19: *Scaling and wavelet coefficients for Spline35 wavelet.*

h(n)	$\tilde{h}(n)$
0	-0
-0.125	0
0.25	-0.25
0.75	0.5
0.25	-0.25
-0.125	0

Table B.20: *Scaling and wavelet coefficients for Reverse Spline35 wavelet.*

h(n)	$\tilde{h}(n)$
0	0.125
0.25	0.25
0.5	-0.75
0.25	0.25
0	0.125
0	0

Table B.21: *Scaling and wavelet coefficients for spline39 wavelet.*

h(n)	$\tilde{h}(n)$
0	-0
0.0234375	0
-0.046875	-0
-0.125	0
0.296875	-0.25
0.703125	0.5
0.296875	-0.25
-0.125	0
-0.046875	-0

Table B.22: *Scaling and wavelet coefficients for Reverse Spline39 wavelet.*

h(n)	$\tilde{h}(n)$
	2.34375 e-02
	-4.6875 e-02
	-1.25 e-01
2.5 e-1	2.96875 e-01
5 e.1	7.03125 e-01
2.5 e-1	2.96875 e-01
	-1.25e-01
	-4.6875e-02
	2.34375e-02

Table B.23: *Scaling and wavelet coefficients for Spline44 wavelet.*

h(n)	$\widetilde{h}(n)$
-0.25	0.125
0.75	-0.375
0.75	0.375
-0.25	-0.125

Table B.24: *Scaling and wavelet coefficients for Reverse Spline44 wavelet.*

h(n)	$\widetilde{h}(n)$
0.125	-0.25
0.375	-0.75
0.375	0.75
0.125	0.25

Table B.25: *Scaling and wavelet coefficients for Spline48 wavelet.*

h(n)	$\widetilde{h}(n)$
0.046875	0
-0.140625	-0
-0.109375	0.125
0.703125	-0.375
0.703125	0.375
-0.109375	-0.125
-0.140625	0
0.046875	-0

Table B.26: *Scaling and wavelet coefficients for Reverse Spline48 wavelet.*

h(n)	$\widetilde{h}(n)$
0	0.046875
0	0.140625
0.125	-0.109375
0.375	-0.703125
0.375	0.703125
0.125	0.109375
0	-0.140625
0	-0.046875

Table B.27: *Scaling and wavelet coefficients for spline4246.*

h(n)	$\widetilde{h}(n)$	***g(n)***	$\widetilde{g}(n)$
			-3.125e-2
1.25e-2		1.25e-2	-2.1875e-1
3.75e-1	-0.5	3.75e-1	-0.6875
3.75e-1	0.5	3.75e-1	0.6875
1.25e-2		1.25e-2	2.1875e-1
			3.125e-2

B.2 Stationary Wavelet Transform

The source code written in C++ for the forward and inverse stationary wavelet transform is given below.

File SWaveletStat.h

```
#pragma once

#include "swavelet.h"
#include <vector>

class SWaveletStat : public SWavelet
{

public:
        SWaveletStat( std::vector<double>& xtable,
                           std::vector<double>& ytable,
                                     int wavelet,
                                     int boundary,
                                     bool interpolate,
                                     int normalized,
                                     int adjust = S_ADJUST_NONE  );
        virtual ~SWaveletStat(void);

        virtual bool OneStepForwardTrans();
        virtual bool OneStepInverseTrans();


        bool IsAtrous() const;

        bool GetDetailSignal( std::vector< double >& table ,int detailLevel) const;
        bool GetApproximateSignal( std::vector< double >& table,int detailLevel) const;
        bool GetMaximaSignal( std::vector< double >& table,int detailLevel) const;
        bool GetTimeSignal( std::vector< double >& table,int detailLevel) const;

        virtual bool GetDetailSignalIndex( int level, int& startindex, int& endindex )
const;
        virtual bool GetApproximateSignalIndex( int level, int& startindex, int&
endindex ) const;

        virtual void Print();

protected:
        virtual bool RemoveOneDetailLevel();
        virtual bool AddOneDetailLevel();

};
```

File SWaveletStat.cpp

```
#include "StdAfx.h"
#include "SWaveletStat.h"



SWaveletStat::SWaveletStat(std::vector<double>& xtable,
                               std::vector<double>& ytable,
                               int wavelet,int boundary,
                               bool interpolate,
                                                       int normalized,
                                                       int adjust)
    : SWavelet(xtable ,ytable ,wavelet ,boundary ,interpolate, normalized, adjust)
{

}

SWaveletStat::~SWaveletStat(void)
{
}
```

```
//////////////////////////////////////////////////////////////////////////////
///
///
///
//////////////////////////////////////////////////////////////////////////////
bool SWaveletStat::GetDetailSignalIndex( int level, int& startindex, int& endindex )
const
{
        if( level <= m_detailLevel )
        {
                startindex = ( m_boundaryPoint + m_originalPoint + m_adjustPoint ) *
level;
                endindex = ( m_boundaryPoint + m_originalPoint + m_adjustPoint ) *
                               ( level + 1 ) - m_boundaryPoint - m_adjustPoint;
                return true;
        }
        else
                return false;
}

//////////////////////////////////////////////////////////////////////////////
///
///
///
//////////////////////////////////////////////////////////////////////////////
bool SWaveletStat::GetApproximateSignalIndex( int level, int& startindex, int&
endindex ) const
{
        startindex = 0;
        endindex = ( m_boundaryPoint + m_originalPoint + m_adjustPoint );
        return true;
}


//////////////////////////////////////////////////////////////////////////////
///
///
///
//////////////////////////////////////////////////////////////////////////////
bool SWaveletStat::GetDetailSignal( std::vector< double >& table,int detailLevel )
const
{
   if( detailLevel > 0 && detailLevel <= m_detailLevel )
    {
        table.resize( m_originalPoint );
                int start = ( int ) ( ( m_boundaryPoint + m_originalPoint +
m_adjustPoint ) * detailLevel + m_boundaryPoint / 2.0 );

        for(int i = 0 ; i < table.size(); i++)
            table[ i ] = m_ytable[ start + i ];

                return true;
    }
        else
                return false;
}



//////////////////////////////////////////////////////////////////////////////
///
///
///
//////////////////////////////////////////////////////////////////////////////
bool SWaveletStat::GetMaximaSignal(std::vector< double >& table,int detailLevel) const
{
    if( detailLevel > 0 && detailLevel <= m_detailLevel )
    {
        table.resize( m_originalPoint );
                int start = ( m_boundaryPoint + m_originalPoint + m_adjustPoint ) *
detailLevel;

        for(int i = 0; i < m_originalPoint ;i++)
        {
            if( i > 0 && i < m_originalPoint &&
              ( ( m_ytable[ start + i - 1 + m_boundaryPoint / 2 ] < m_ytable[ start +
i + m_boundaryPoint / 2 ]  &&
                  m_ytable[ start + i + 1 + m_boundaryPoint / 2] < m_ytable[  start +
i + m_boundaryPoint /  2 ] ) ||
                              ( m_ytable[ start + i - 1 + m_boundaryPoint / 2 ] >
m_ytable[ start + i + m_boundaryPoint / 2 ]  &&
                  m_ytable[ start + i + 1 + m_boundaryPoint / 2] > m_ytable[  start +
i + m_boundaryPoint /  2 ] ) ) )

            {
                table[ i ] = m_ytable[ start +  i + m_boundaryPoint / 2 ];
            }
            else
                table[ i ] = 0;
        }

               return true;
    }
       else
               return false;
}


//////////////////////////////////////////////////////////////////////////////
///
///
///
//////////////////////////////////////////////////////////////////////////////
bool SWaveletStat::GetTimeSignal( std::vector< double >& table, int detailLevel )
const
{
    if( detailLevel > 0 && detailLevel <= m_detailLevel )
    {
        table.resize( m_originalPoint );

        for(int i = 0; i < table.size(); i++)
            table[ i ] = i;

               return true;
    }
       else
               return false;
}



//////////////////////////////////////////////////////////////////////////////
///
///
///
//////////////////////////////////////////////////////////////////////////////
bool SWaveletStat::GetApproximateSignal(std::vector<double>& table,int detailLevel)
const
{
    if( detailLevel > 0 && detailLevel <= m_detailLevel )
    {
        table.resize( m_originalPoint );

        for(int i = 0; i < table.size(); i++)
            table[ i ] = m_ytable[ i + m_boundaryPoint / 2 ];

               return true;
    }
       else
               return false;
}


//////////////////////////////////////////////////////////////////////////////
///
///
///
//////////////////////////////////////////////////////////////////////////////
bool SWaveletStat::IsAtrous() const
{
    return true;
```

```
}


////////////////////////////////////////////////////////////////////////////////
///
///
///
////////////////////////////////////////////////////////////////////////////////
bool SWaveletStat::AddOneDetailLevel()
{
        int size = m_ytable.size() / (  m_boundaryPoint + m_originalPoint +
m_adjustPoint );

    if( size <= m_maxDetailLevel )
    {
                m_ytable.resize( m_ytable.size() + m_boundaryPoint + m_originalPoint +
m_adjustPoint );
                m_detailLevel++;
                return true;
    }
        else
                return false;

}




////////////////////////////////////////////////////////////////////////////////
///
///
///
////////////////////////////////////////////////////////////////////////////////
bool SWaveletStat::RemoveOneDetailLevel()
{
        int size = m_ytable.size() / (  m_boundaryPoint + m_originalPoint +
m_adjustPoint );

    if( size > 1 )
        {
                m_ytable.resize( m_ytable.size() - m_boundaryPoint - m_originalPoint -
m_adjustPoint );
        m_detailLevel--;
                return true;
        }
        else
                return false;
}



////////////////////////////////////////////////////////////////////////////////
///
///
///
////////////////////////////////////////////////////////////////////////////////
bool SWaveletStat::OneStepForwardTrans()
{
    if (m_detailLevel < m_maxDetailLevel )
    {
        if (m_detailLevel == 0)
            PreWaveletTrans();
        AddOneDetailLevel();

        int n = m_boundaryPoint + m_originalPoint + m_adjustPoint;
        int p = ( int )pow( 2.0, m_detailLevel - 1 );
                int s = ( m_boundaryPoint + m_originalPoint + m_adjustPoint ) * (
m_detailLevel );

        for (int i = 0; i < n; i++)
        {
            m_tempYTable[ i ] = 0;


                      for(int k = 0; k < m_hCoeff.size(); k++)
```

```
                              m_tempYTable[ i ] += m_ytable[ ( n + i + k * p + m_hIndex
* p ) % n ] * m_hCoeff[ k ];


                      for( int k = 0; k < m_gCoeff.size(); k++)
                              m_ytable[ s + i ] += m_ytable[ ( n + i + k * p + m_gIndex
* p ) % n ]  * m_gCoeff[ k ];

               }
        for (int i = 0; i < n; i++)
            m_ytable[ i ] = m_tempYTable[ i ];

               return true;
    }
       else
               return false;
}




////////////////////////////////////////////////////////////////////////////////
///
///
///
////////////////////////////////////////////////////////////////////////////////
bool SWaveletStat::OneStepInverseTrans()
{
    if (m_detailLevel > 0)
    {
        int n = m_boundaryPoint + m_originalPoint + m_adjustPoint;
        int p = ( int )pow(2.0, m_detailLevel - 1 );
               int h = p * 2 * m_igIndex;
               int s = ( m_boundaryPoint + m_originalPoint + m_adjustPoint ) *
m_detailLevel ;

        for (int i = 0; i < n; i++)
        {
            m_tempYTable[ i ] = 0;

                      for(int k = 0; k < m_ihCoeff.size(); k++)
                              m_tempYTable[ i ] += 0.5 * m_ytable[(n + i + p * k + p *
m_ihIndex ) % n] * m_ihCoeff[ k ] ;

                      for( int k = 0; k < m_igCoeff.size(); k++)
                              m_tempYTable[ i ] += 0.5 * m_ytable[ s + ( n + i + p * k
+ p * m_igIndex )% n ] * m_igCoeff[ k ];

               }

        for ( int i = 0; i < n; i++ )
            m_ytable[ i ] = m_tempYTable[ i ];

                 RemoveOneDetailLevel();

        if ( m_detailLevel == 0)
            PostWaveletTrans();

               return true;
    }
       else
               return false
```