PERMANENT DOWNHOLE GAUGE DATA INTERPRETATION

A REPORT SUBMITTED TO THE DEPARTMENT OF PETROLEUM ENGINEERING

OF STANFORD UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE

By

Khong Chee Kin

June, 2001

I certify that I have read this report and that in my opinion it is fully adequate, in scope and in quality, as partial fulfillment of the degree of Master of Science in Petroleum Engineering.

_____

Dr. Roland N. Horne
(Principal Advisor)

Abstract

This work investigated methodology for the interpretation of data from permanent downhole pressure gauges. The algorithms developed previously by Athichanagorn (1999) were evaluated to investigate their effectiveness and weaknesses. Additional data processing steps were introduced to resolve deficiencies encountered while evaluating the robustness of existing algorithms. The specific issues that required attention were: step outlier removal from the data, setting processing parameters in the *Wavelet* and *Window* programs, refinement of break points from those detected by the *Wavelet* program and the initial estimates of unknown flow rates. With these additions, the algorithms were used to manipulate actual field data with sufficient reliability.

Acknowledgments

I would like to express my gratitude toward Professor Roland N. Horne for his continuous guidance, advice, encouragement and tremendous patience throughout the course of the study.

I would like to thank my colleagues in the SUPRI-D research group for their suggestions and comments in this project.

I am really grateful to my wife, Chong Aik Kean, my daughter Alicia Khong Qian Yu and her grandparents Chong Boo Chock and Ong Sooi Hong for their constant support during my studies. Also not to be left out are my parents, Khong Shee Weng and Lai Ah Looi for their encouragement.

I would also like to thank member companies of the SUPRI-D Research Consortium for Innovation in Well Test Analysis for providing financial support for this work.

Contents

List of Tables

List of Listings

List of Figures

Chapter 1

1. Introduction

Permanent down hole gauges have been installed in more than 1000 wells worldwide. The continuous pressure measurement allows the operator to make adjustments to the well to optimize recovery. Past experience indicates that permanent down hole gauges are cost-effective even for the single use of well monitoring to assist operational management (Athichanagorn, 1999). Additional value could be derived by analyzing the pressure data for reservoir information or using it during history matching.

An operator will attempt to maximize the value from the investment in a permanent down hole gauge acquisition system. However, in practice lots of permanent down hole gauge data have been archived or even discarded because of the lack of tools to process the data. This is because permanent down hole long term measurements are prone to different kinds of errors than data from a short well test.

In the traditional well test, the pressure response of the reservoir is measured carefully under a strictly controlled environment. In the case of long term reservoir monitoring using permanent pressure gauges, the well and the reservoir and its fluid composition may undergo dynamic changes. The well may be stimulated or worked over causing the gauge to record invalid measurements. The pressure data may also be stored at low precision or the system may malfunction creating superfluous outliers and noise. Flow rate data are often not available in the permanent down hole gauge acquisition system. Athichanagorn (1999) developed an interpretation methodology for long-term pressure data records from permanent down hole gauges. This methodology is useful when flow rates data are not available or not complete. If flow rates prior to a transient are available, a transient can be analyzed using conventional well testing methods but the data may still be preprocessed using the data processing tools developed by Athichanagorn (1999).

Athichanagorn (1999) implemented algorithms to remove outliers and noise from a data set using wavelet transform signal processing. Since the amount of data collected by the permanent down hole gauges is very large, an algorithm to reduce the number of data to a manageable size by eliminating redundant information was also implemented. In order to interpret the permanent long-term data, a complete history of flow rates and the time of flow rate changes are needed, and these records are often unavailable. So Athichanagorn (1999) developed algorithms to detect the times at which flow rate changes occur (the break points) and to reconstruct unknown or uncertain flow rates from the pressure data.

Since long-term monitoring is under an uncontrolled environment, inconsistent abnormal transients may appear in the record and need to be removed prior to interpreting the data. Reservoir properties may change in the long term, so it may not be appropriate to interpret the data all at once. Athichanagorn (1999) implemented a procedure to interpret one window of data at a time, the interpretation window being stepped forward successively until the end of the data. The moving window interpretation method allows the determination of local values of reservoir parameters and also overcomes computer memory limitations if the entire transient were to be interpreted at the same time.

The algorithms in Athichanagorn (1999) were coded into two programs, the *Wavelet* program and the *Window* program. The processing steps for these two programs can be summarized in a seven-step procedure as follows:

Wavelet program

- outlier removal
- denoising
- transient identification
- data reduction

Window program

- flow history reconstruction
- behavioral filtering
- data interpretation

1.1. Problem Statement

The source codes of the algorithms from Athichanagorn (1999) were compiled in the UNIX operating system environment. The first step in this research project was to port the source codes to the Windows operating system so that the programs could be used more easily in the field.

A short user guide was written to assist users to learn how to use the *Wavelet* and *Window* programs.

During the work of porting the source codes to the Windows operating systems, experimenting with the programs and writing the short user guide, experiences gained were documented and the difficulties and deficiencies encountered identified were subjects for this research work. Recommendations for future work were also suggested based on the results.

One difficulty encountered was in setting the processing parameters in the *Wavelet* program to denoise the data and identify all the true break points with no false break points. The break points from the *Wavelet* program often did not fall exactly at the beginning of the transient and had to be adjusted. It was also found that the *Wavelet* program did not accept data that overlap in time and that step outliers (described in Section 3.2.) were not properly removed by the *Wavelet* program.

Good initial estimates of the unknown flow rates are needed by the *Window* program regression algorithm to estimate the unknown rates. Initial estimates of the unknown flow rates had previously been estimated manually, so an automatic rate estimation algorithm was implemented during in this research work.

The estimated parameters from the *Window* program were found to vary more than seemed reasonable. Research effort was spent to investigate the optimal processing parameters to be used with the *Window* program and also to evaluate the robustness of the program.

1.2. Report Outline

Chapter 2 summarizes the interpretation steps of the *Wavelet* and *Window* programs from Athichanagorn (1999) with an interpretation example on a simulated data set.

Chapter 3 discusses the inability of the *Wavelet* program to handle data with time overlaps and to remove step outliers. The noise level in the data is estimated by removing the trend from the data and the appropriate denoising method and denoising threshold use are discussed. The adjustment of the processing parameters in the *Wavelet* program to select the true break points and the procedure to correct the detected break times are also discussed.

Chapter 4 presents the histogram and Fourier transform methods to select the true break points. As this research used the existing wavelet transform break point detection algorithm, the break points detected from various combinations of the wavelet sample spacing and slope detection threshold processing parameters were screened further using the histogram and Fourier methods.

Chapter 5 shows how to adjust the break points detected by the *Wavelet* program using a straight line intersection method and also describes the inability of the algorithm to correct the break point of one transient based on the break point from another transient.

Chapter 6 elaborates on the method to estimate unknown flow rates based on the area under a transient in the *Inrate* program. Chapter 7 describes the *Window* program and the processing parameters appropriate to obtain estimates that vary reasonably with time.

Chapter 8 presents interpretation case studies using simulated data and real field data. Simulated data were used to check estimates from the *Window* program and the field data were used to explore the robustness of the existing and newly introduced algorithms.

Chapter 9 concludes on the results of this research and suggests areas for additional work in the future. Appendix 1 includes a short user guide for the *Wavelet* and *Window* programs from Athichanagorn (1999) and information on the new algorithms introduced.

Chapter 2

2. Summary of Interpretation Steps

Permanent downhole pressure data are recorded over long periods of time, during which gauge or acquisition systems may degrade or fail, introducing noise and outliers into the measurements. Flow rate data are often not available for the entire duration. The interpretation algorithm implemented to estimate completion and reservoir parameters like skin, permeability-thickness and geometry requires time of flow-rate changes to be identified correctly.

To accommodate these issues, Athichanagorn (1999) developed a sequence of procedures to manipulate and interpret data from permanent downhole gauges. Athichanagorn's approach can be summarized as a seven-step procedure as follows:

- outlier removal
- denoising
- transient identification
- data reduction
- flow history reconstruction
- behavioral filtering
- data interpretation

These seven steps are implemented in two processing stages: *Wavelet* algorithm and *Window* algorithm. Figure 2.1 shows the data processing path.

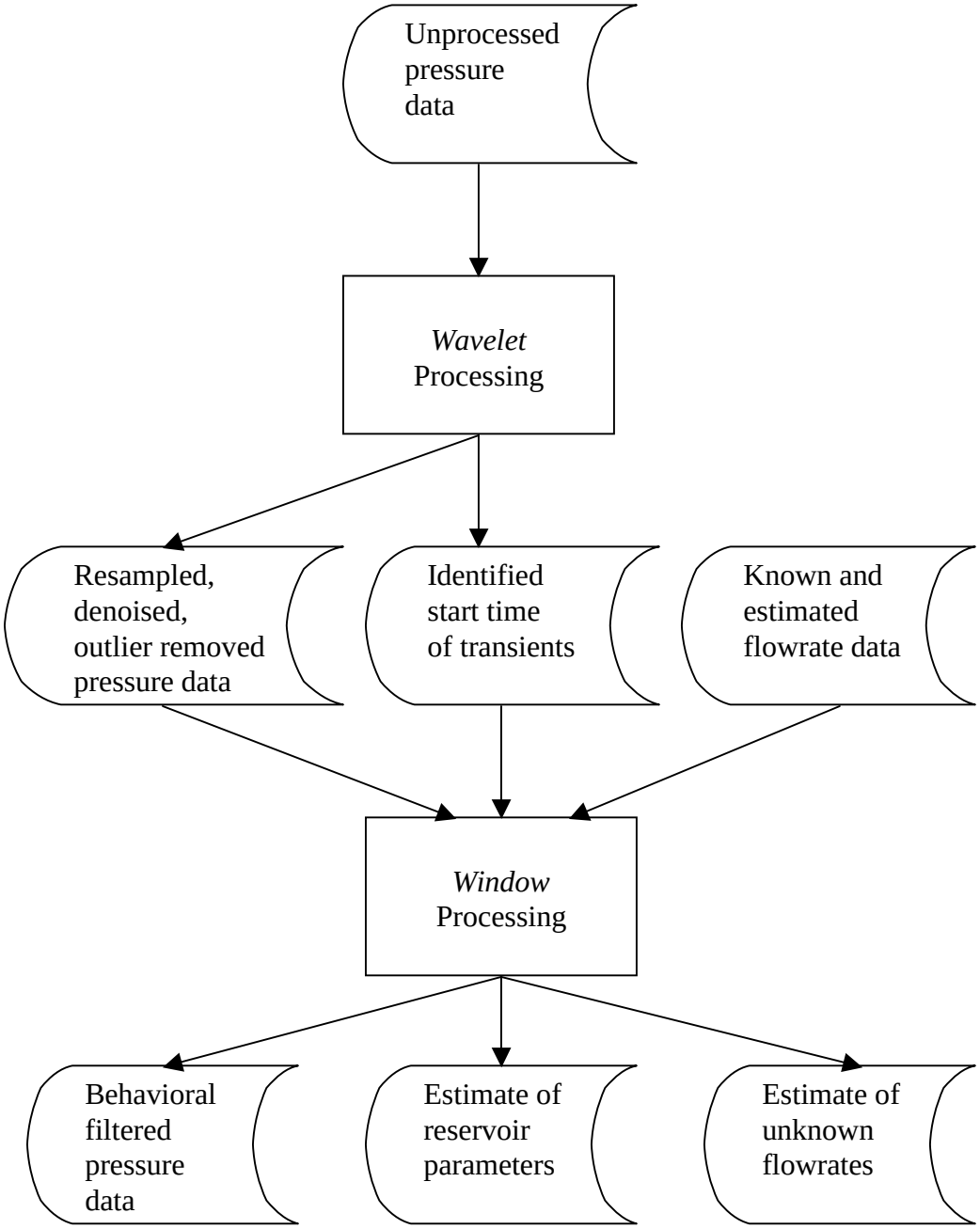


Figure 2.1: Permanent downhole gauge data interpretation steps.

2.1. *Wavelet* Algorithm

The pressure data are decomposed to wavelet signals at various levels of detail using the wavelet decomposition algorithm. Various processing are performed on the wavelet signals to remove outliers and noise. The processed wavelet signals at various levels of detail are then recombined to form filtered pressure signal. The processed pressure data are then resampled at lower pressure and time sampling intervals to reduce the size of the data set.

2.1.1. Outlier Removal

Measurement errors can be classified as noise or outliers. Noisy data scatter around the trend of the overall data. “Outliers” on the other hand are data points that lie away from the trend of the data. An “outlier” causes discontinuities in the data stream creating two consecutive singularities. The detail wavelet signal first changes sharply in one direction, and then changes again in the opposite direction. The singularities created by the outliers can be detected by screening for two large magnitudes of the detail wavelet signal with opposite directions (Athichanagorn, 1999). Figure 2.2 compares outlier-removed data to the original data.

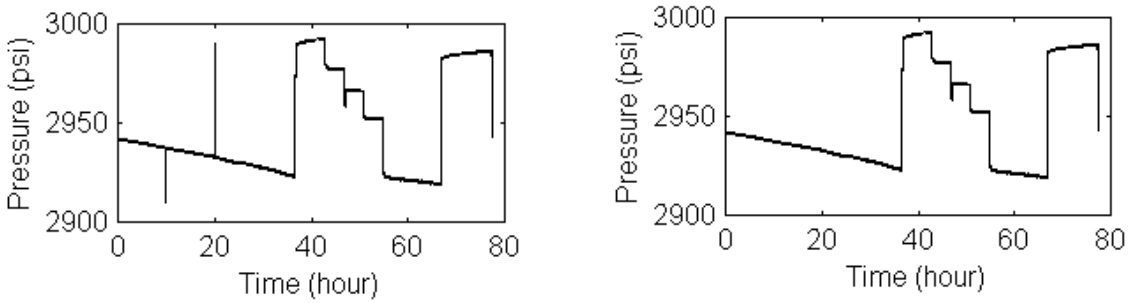


Figure 2.2: Original signal (left) and after outlier removal (right).

2.1.2. Denoising

The denoising process is applied to the data to reduce scattering and fluctuation in order to extract the most representative features in the data. The detail wavelet signals whose magnitudes are smaller than a certain threshold are set to zero and the denoised data set is constructed using the thresholded signals. Figure 1.3 compares data before and after denoising.

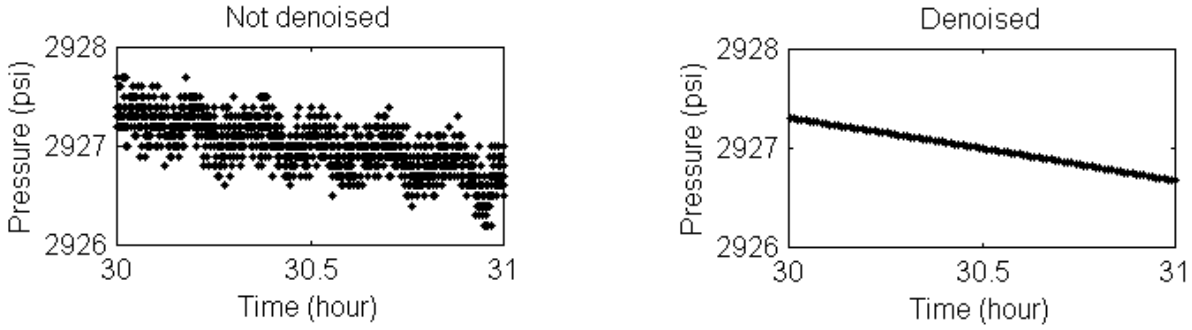


Figure 2.3: Denoising – original (left) and denoised (right) data.

2.1.3. Transient Identification

The times at which flow rates change are determined by identifying sudden changes in pressure. These changes cause singularities in the data. The wavelet modulus maxima which indicate the neighborhood of singularity are used to determine the time at which flow rate changes (Athichanagorn, 1999). A proper identification of these "break points" is critical to the interpretation of the data. Figure 2.4 shows transients identified by the *Wavelet* algorithm.

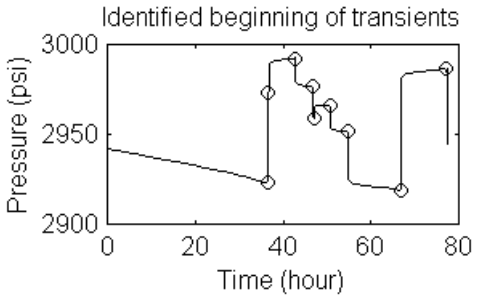


Figure 2.4: Transients identified by the *Wavelet* algorithm.

2.1.4. Data Reduction

The number of data points is reduced using the pressure thresholding and time thresholding methods. Points are sampled from the original data set when a certain change of pressure has occurred or whenever the time span between samples becomes higher than a maximum preset time threshold. For noisy data that are collected at high frequency, it is usually necessary to denoise the data before undertaking the data reduction step so that the representative points can be identified in the data set (Athichanagorn, 1999). A set of data distributed at even time interval can be generated by setting the pressure threshold to a very high level and selecting the desired time threshold. Figure 2.5 shows an example of a dense and a reduced data set.

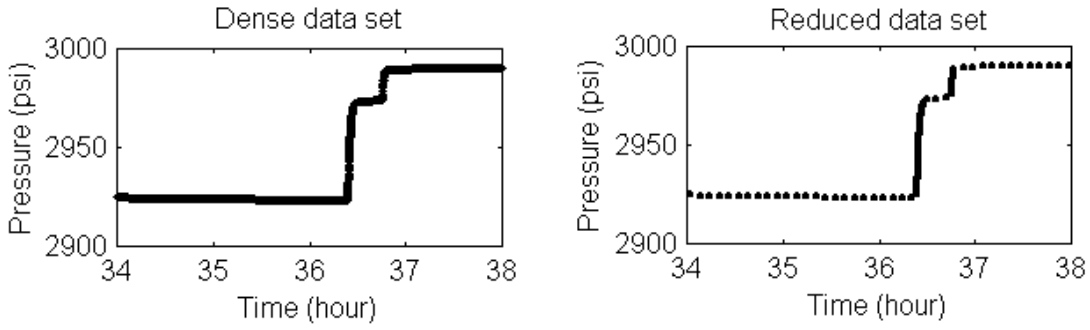


Figure 2.5: Dense (left) and reduced (right) data set

2.2. *Window* Algorithm

As it is possible for reservoir properties and conditions to change, one constant property model should not be used to fit the entire data set (Athichanagorn, 1999). Sections of the data should be analyzed by using the moving window technique. The data in the transients that are grouped together form a window of data. Nonlinear regression is used to estimate reservoir parameters and unknown flow rates. The estimated parameters are associated with the time at the center of the window. The window of data is then stepped forward in time and regression analysis is performed again. Stepping is carried out until the final window covers the end of the data. A wider window gives smoother trends of parameter variation over time.

2.2.1. Flow Rate Reconstruction

When all the flow rates are known, each transient can be analyzed separately using local values of reservoir and fluid properties. However, this traditional pressure transient analysis cannot be used when some of the flow rates are unknown. Unknown flow rates are parameters to be estimated in the regression analysis. It is important to have good initial estimates of the unknown flow rates to ensure that the regression algorithm converges correctly. Figure 2.6 shows an example of the estimation of unknown flow rates.

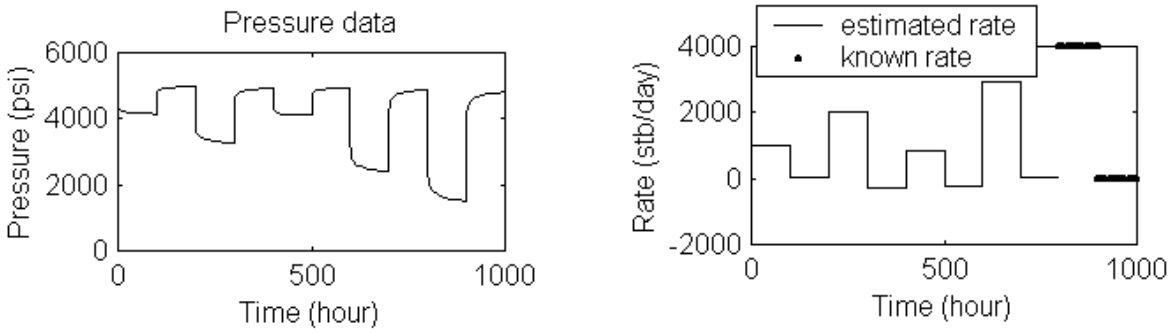


Figure 2.6: Flow rate reconstruction.

2.2.2. Behavioral Filtering

The pressure history data may exhibit strange behavior that does not follow the general trends that are caused by sudden changes in conditions in the well or reservoir. Abnormal transients increase the uncertainty of the regression match. The variances of abnormal transients are generally unusually high because they are not well matched by the regression (Athichanagorn, 1999). The transients with the highest variance are excluded in the average variance calculation. The variances of each transient (including the maximum variance) are then compared to the average variance, and transients whose variances that are at least three times higher than the average variance are considered abnormal and excluded from the analysis. Repeated regression removes more abnormal transients until there are none remaining. Figure 2.7 shows example of abnormal transients removed by this method.

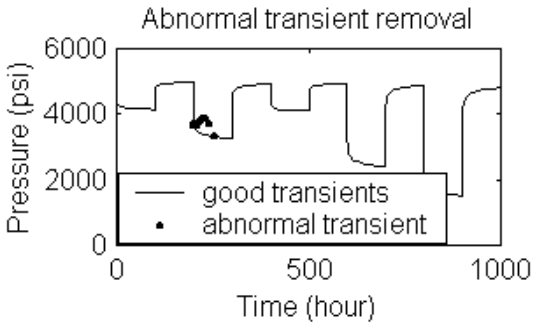


Figure 2.7: Abnormal transient removal.

2.2.3. Data Interpretation

The reservoir model has to be determined by analyzing one of the transients using conventional well test interpretation methods. The moving *Window* analysis can then be based on the interpreted reservoir model.

Using the moving window analysis method, the unknown reservoir parameters and unknown flow rates in each window are estimated by regression. The analysis moves forward to a new window of the same width but the starting point of the new window may lie within the span of the old window. Unknown rates are updated and taken as known for subsequent windows since unknown rates cannot be inferred with high certainty from the response in later windows (Athichanagorn, 1999). Further research work could improve this methodology. Figure 2.8 shows examples of reservoir parameters estimated using moving window analysis.

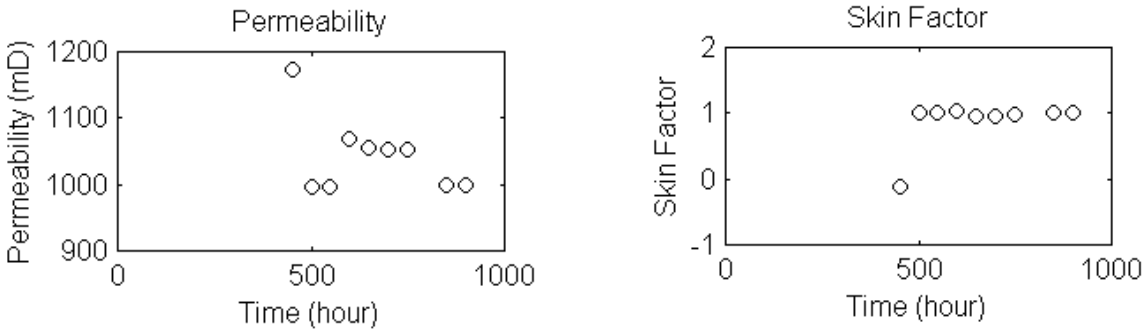


Figure 2.8: Estimates of permeability and skin factor over a sequence of nine windows.

Chapter 3

3. *Wavelet* Algorithm Improvements

3.1. Data Overlap

During the recording or preparation of pressure data, it is possible that there may be an overlap of data in time. The *Wavelet* algorithm cannot accept pressure data with overlaps. Pressure data need to be prescreened and have the overlaps removed before being used in the *Wavelet* algorithm. One simple rule is to delete a time-pressure data pair when its time value is less or equal to the time value of the preceding data point.

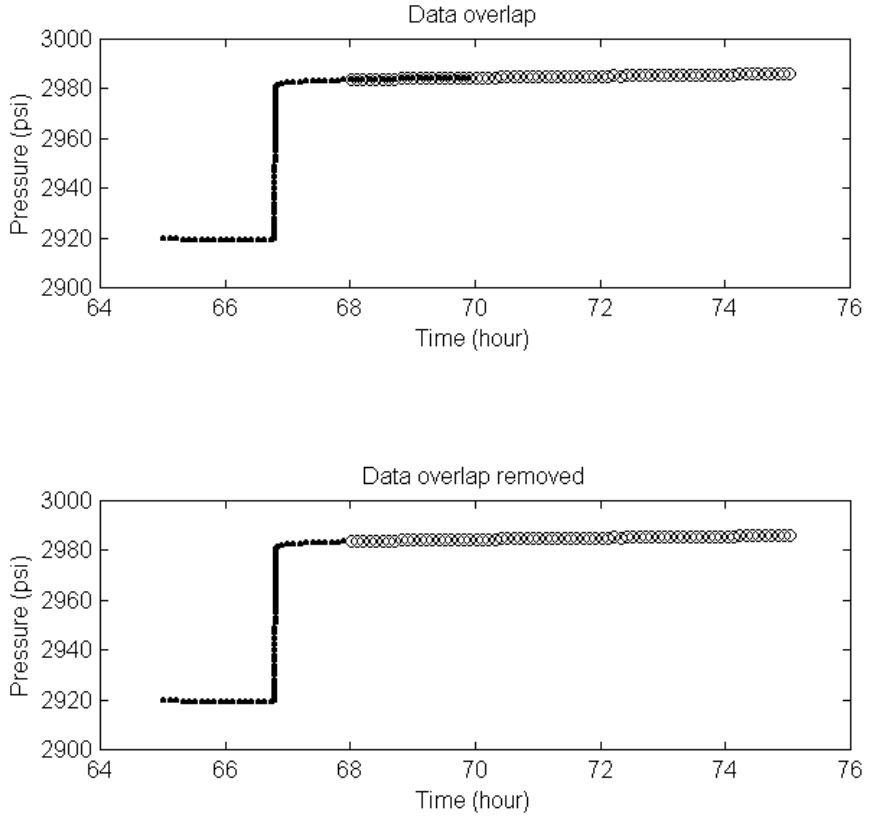


Figure 3.1:Data overlap removal.

3.2. Step Outlier

The *Wavelet* algorithm developed by Athichanagorn (1999) removes single spike outliers successfully. However in some data sets it was found that some outliers occur in steps. Step outliers are created when there is no pressure data available for an interval of time and zero values are assigned to the pressure data points. Either zero or some large negative or positive numbers assigned to the pressure data when no pressure data are available can create these step outliers. Step outliers may also arise when the permanent down-hole gauge acquisition system fails and introduce large positive or negative shift to the pressure data during some time interval.

Step outliers in which a section of the pressure data is ten times lower than the pressure in the neighborhood have been encountered in actual field data. This could be due to data recording error or to the true pressure not being transmitted to the pressure transducer. Figure 3.2 shows a step outlier and Figure 3.3 shows the result from the *Wavelet* algorithm where the step outlier could not be removed. Step outliers have to be removed by deleting time and pressure data in the step before wavelet processing. This will create gaps where no data are available however the *Window* algorithm can handle such situations.

Step outliers also cause two false break points to be interpreted by the *Wavelet* algorithm. If the step outlier is not removed before *Window* processing, this step will be detected as a pair of abnormal transients and will be removed by the behavioral filtering algorithm but the two false break points interpreted remain and this is not correct. Figure 3.4 shows the pressure data with the step outlier removed creating an interval without data.

Figure 3.5 shows a second set of real field data where the step outliers consist of pressure steps with zero pressure values. It was again found that these step outliers could not be removed by the *Wavelet* algorithm and they had to be removed before *Wavelet* processing. Figure 3.6 shows a close up view of this set of field data. Figure 3.7 shows the step outlier is removed from the data creating an interval without data.

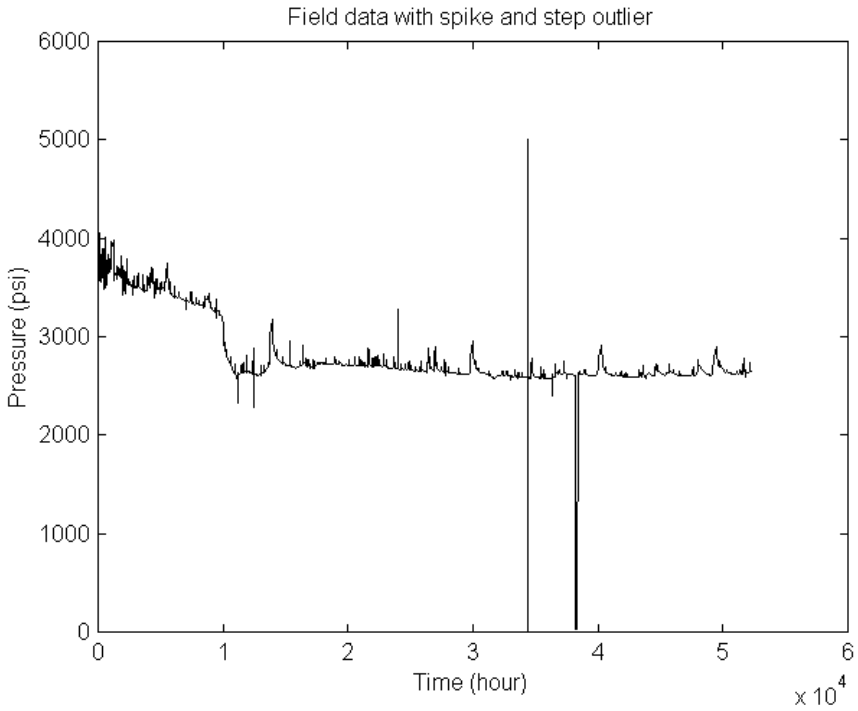


Figure 3.2: Field data (Set 1) with spike and step outliers.

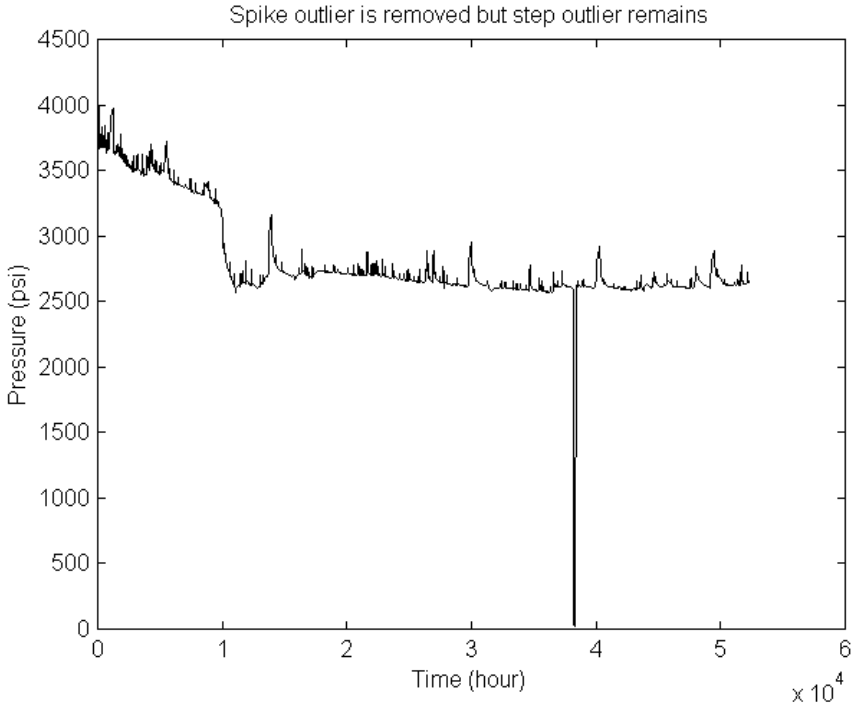


Figure 3.3: Only spike outliers are removed by the *Wavelet* algorithm.

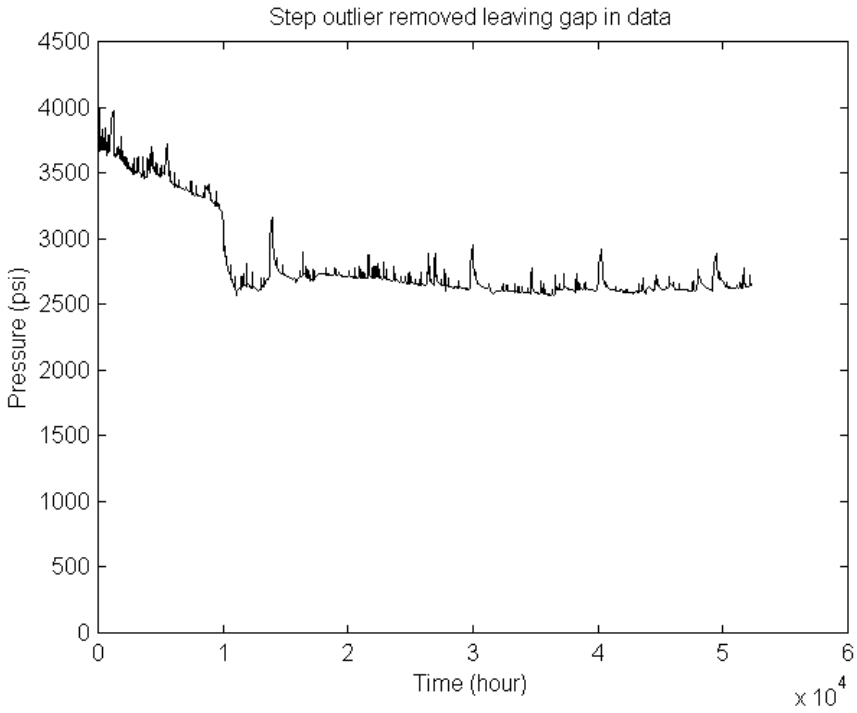


Figure 3.4: The step outlier is removed, leaving a gap in the data.

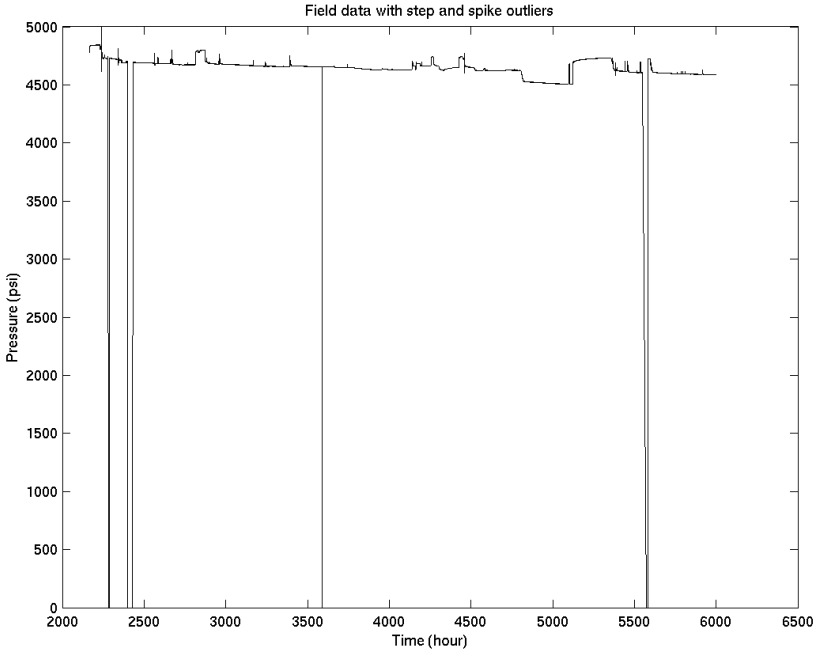


Figure 3.5: Field data (Set 2) with spike and step outliers.

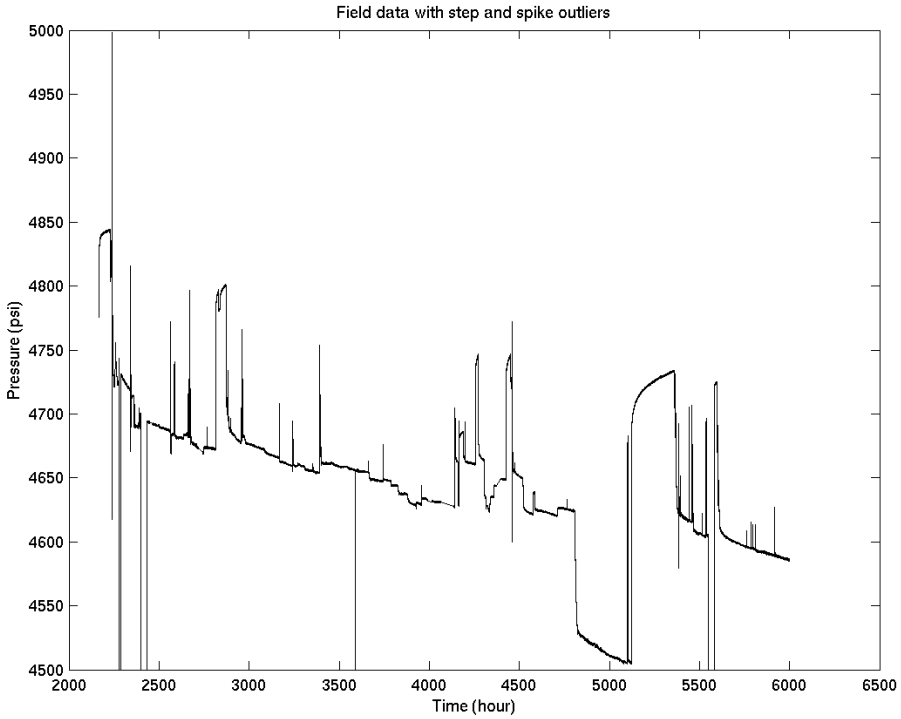


Figure 3.6: Close up view of second field data (Set 2) with spike and step outliers.

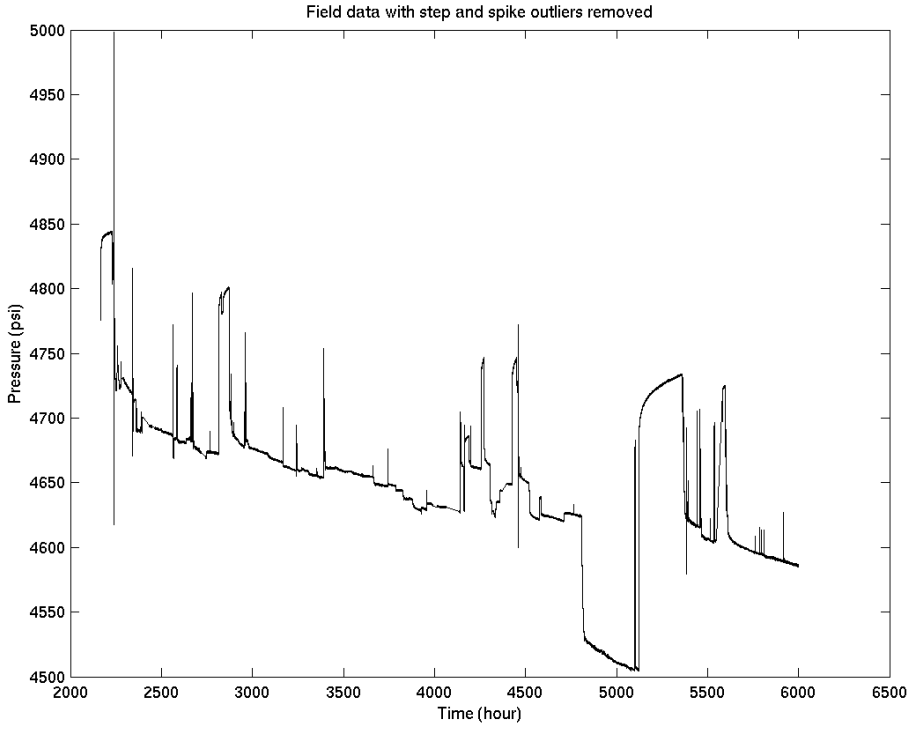


Figure 3.7: Spike and step outliers removed from field data (Set 2).

3.3. Noise Estimation

Noise is filtered as one of the steps of *Wavelet* processing. Athichanagorn (1999) suggested the hybrid noise thresholding method because the soft thresholding method tends to shrinks all the detail signals toward zero while hard thresholding may lead to artifacts that roughen the appearance of the denoised signal. Hybrid thresholding applies the soft thresholding approach in the flat region of the pressure data and applies the hard thresholding approach in the vicinity of discontinuities in the signal. On the other hand, it was found that the computer program for soft thresholding is more robust than that for hybrid thresholding. In the hybrid thresholding program execution can halt at certain combinations of wavelet sampling spacing and slope detection threshold used in the algorithm.

Donoho and Johnstone (1994) proposed a generic value for the universal threshold that can be used for most applications. This threshold value is given in Equation (3.1),

$$\lambda = \sigma\sqrt{2\log(n)} \tag{3.1}$$

where n is the sample size, and σ is the standard deviation of the noise level. The standard deviation of noise in the signal can be estimated in flat regions of the transient.

The trend of the interval where the noise standard deviation needs to be estimated can be found by locating the slope of the least square error straight line fitted to this interval. The pressure trend is subtracted from the data to obtain the residue noise signal. The standard deviation value is then calculated from the residue noise signal.

Consider a linear fit,

$$p = x_0 + x_1 t \tag{3.2}$$

where p is pressure in psi and t is time in hour.

There are n pairs of points in the interval with t_1 and p_1 the first data pair and t_n and p_n the n^{th} data pair. The least square solution of the fitted straight line can be constructed from matrix A and vector b where the solution x_1 is the slope and x_0 is the p -axis intercept of the fitted straight line (Strang, 1988).

$$A=\begin{bmatrix} 1 & t_1 \\ 1 & t_2 \\ . & . \\ . & . \\ 1 & t_n \end{bmatrix} \quad , \quad b=\begin{bmatrix} p_1 \\ p_2 \\ . \\ . \\ p_n \end{bmatrix} \quad , \quad A^T Ax = A^T b \quad , \quad x=\begin{bmatrix} x_0 \\ x_1 \end{bmatrix} \qquad (3.3)$$

The noise N_i for data point i can be calculated by subtracting from each pressure data point the pressure trend:

$$N_i = p_i - (x_0 + x_1 t_i) \qquad (3.4)$$

The standard deviation of the noise, σ can be calculated using Equation (3.5) where n is the number of data points in the interval.

$$\sigma = \left(\frac{1}{n-1} \sum_{i=1}^{n} N_i^2 \right)^{1/2} \qquad (3.5)$$

The noise level can be determined automatically by finding the values of fitted straight line slope in adjacent one half hour intervals and selecting five percent of the intervals with the lowest slope and averaging the noise levels in these lower slope intervals. Figure 3.8 shows an interval of noisy pressure data fitted with a least square straight line, Figure 3.9 shows the noise residue left when the pressure trend is subtracted from the noisy pressure data and Figure 3.10 plots the histogram distribution of the noise. The noise standard deviation is 0.1673 psi.

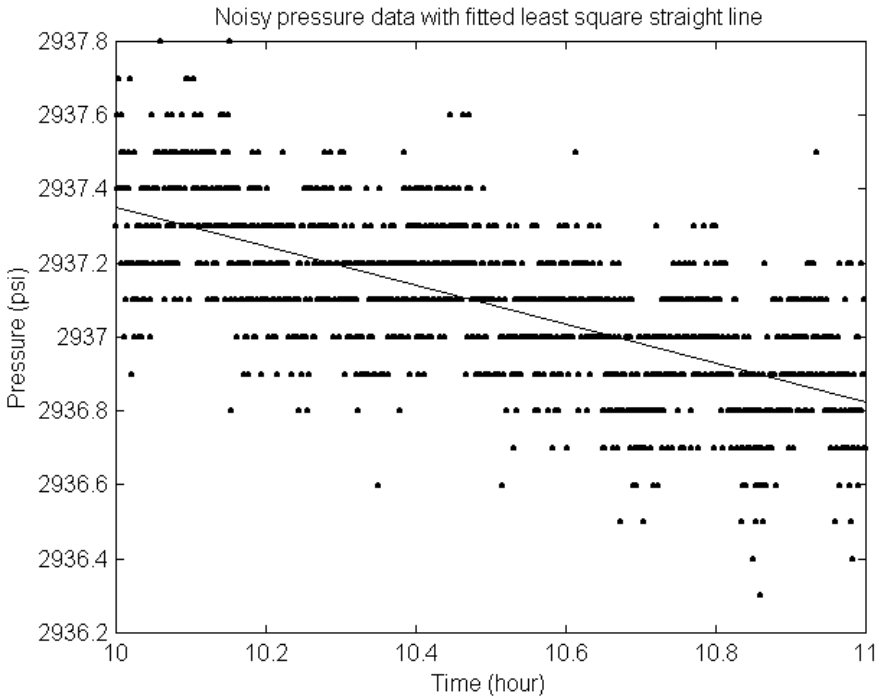


Figure 3.8: Noisy pressure data fitted with least square straight line.

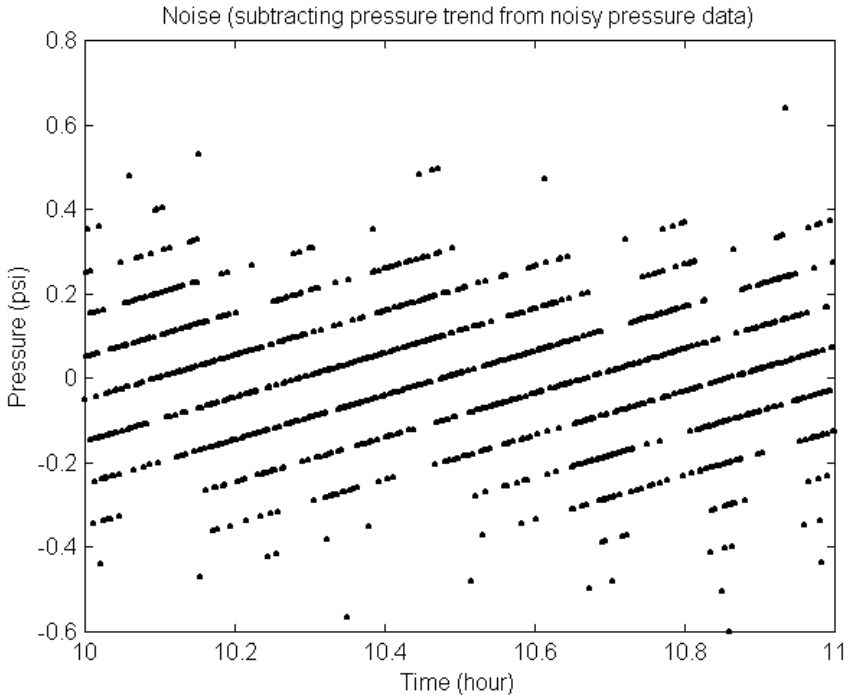


Figure 3.9: The noise signal obtained by subtracting the least square line from the original data in Figure 3.8.

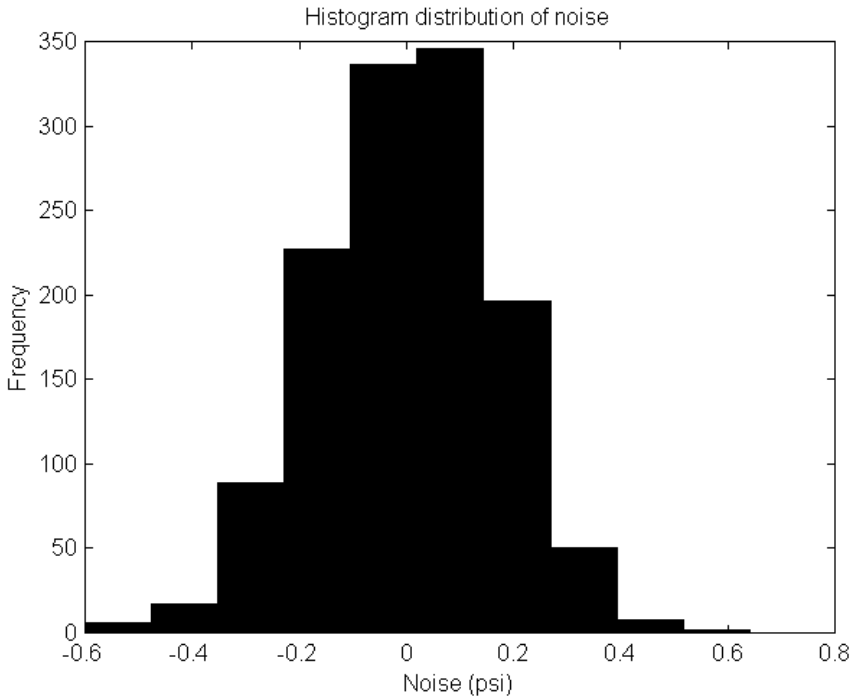


Figure 3.10: Histogram of noise from Figure 3.9.

The effectiveness of denoising at the threshold suggested by Donoho and Johnstone was investigated to verify the guideline given. Figure 3.11 shows the pressure data with 99039 data points before denoising and Figure 3.12 shows a section of the pressure data before denoising. Figure 3.13 shows this section of pressure data denoised using noise threshold of 0.5288 psi based on Donoho and Johnstone suggestion. Figure 3.14 shows the pressure data after denoising.

Denoising based on Donoho's denoising threshold is a good choice. It is prudent to check the denoised data again to verify that a correct denoising threshold has been used. The denoising threshold calculated depends on the number of data points, in this example, the number of data points of 99039 gives a denoising threshold of 0.5288 psi which is larger than three standard deviations of the noise. If the number of data is lower, a minimum value of denoising threshold of three or four noise standard deviations should be used as it covers from left edge to the right edge of the noise histogram.

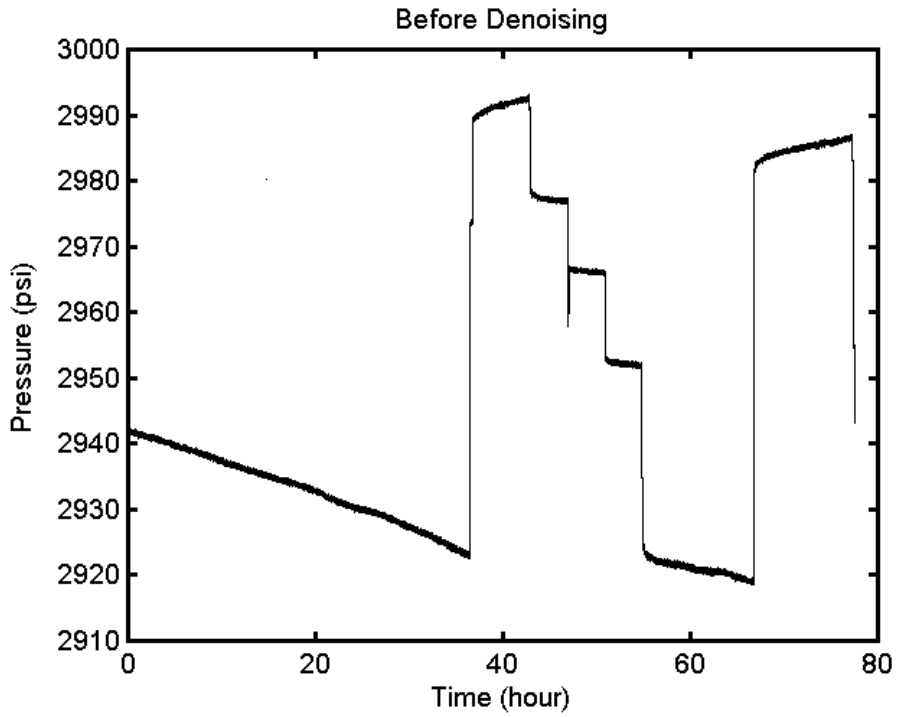


Figure 3.11: Pressure data before denoising.

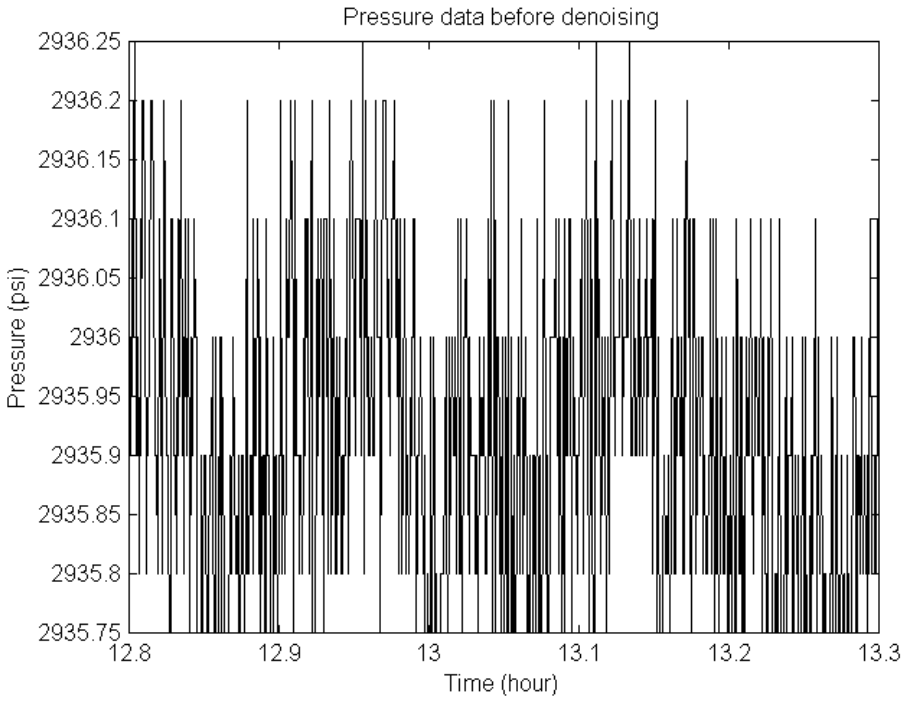


Figure 3.12: Close up view of a section of pressure data before denoising.

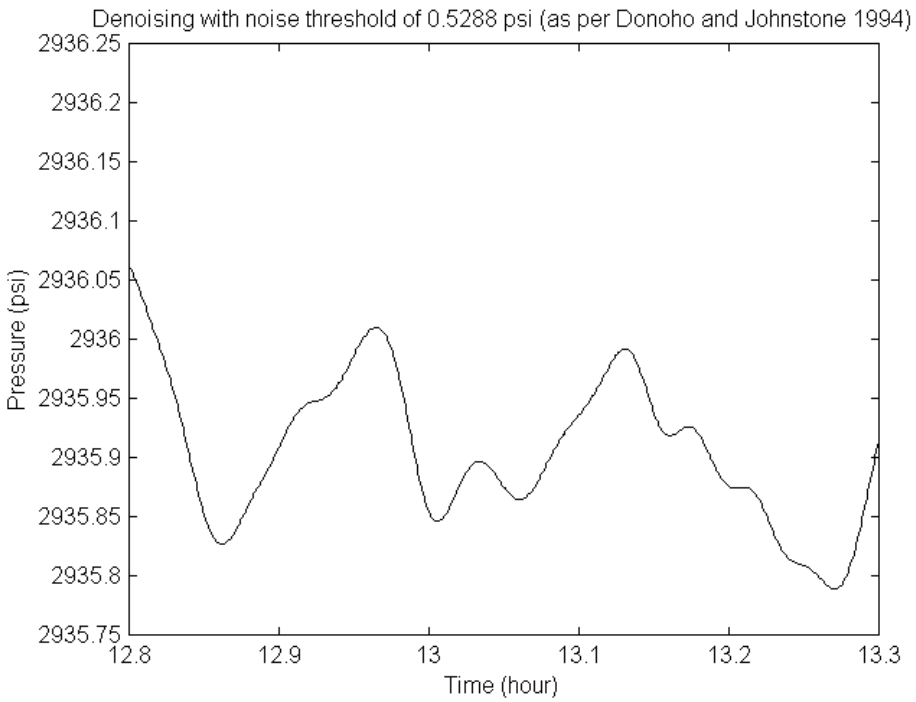


Figure 3.13: Denoising using noise threshold of 0.5288 psi. (as per Donoho, 1994)

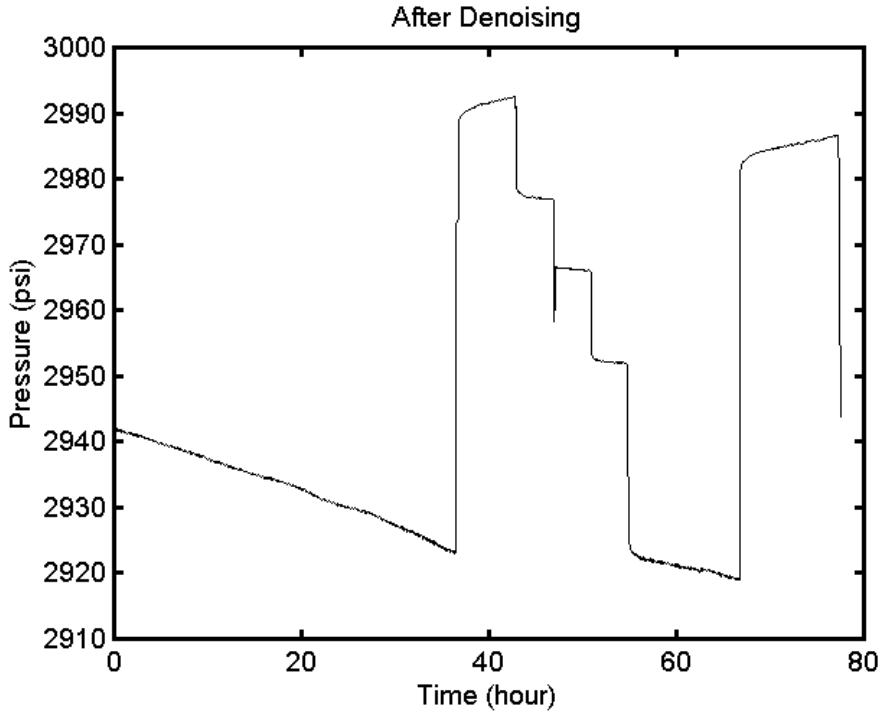


Figure 3.14: Pressure data after denoising.

If the number of data points is not high, the denoising threshold calculated as suggested by Donoho will be lower. Denoising the signal using various values of denoising threshold was also investigated. Figure 3.15 shows the section of data from Figure 3.12 after denoising at a denoising threshold of one standard deviation of the noise. Figure 3.16 shows the same section of data after denoising at a threshold of two standard deviation of the noise. Figure 3.17 shows the section of data after denoising at a threshold of two standard deviation of the noise. It is found that a minimum denoising threshold of three or four standard deviations is needed to properly denoise a set of data. If the denoising threshold calculated as in Donoho's guideline is less than three standard deviations, use a denoising threshold of three or even four standard deviations.

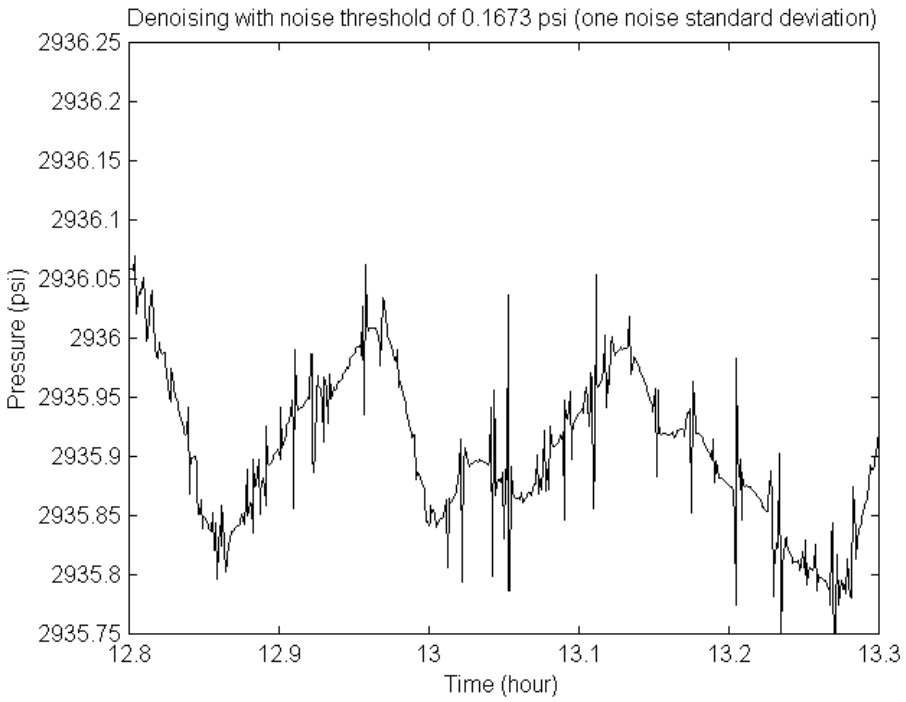


Figure 3.15: Denoising threshold at one noise standard deviation.

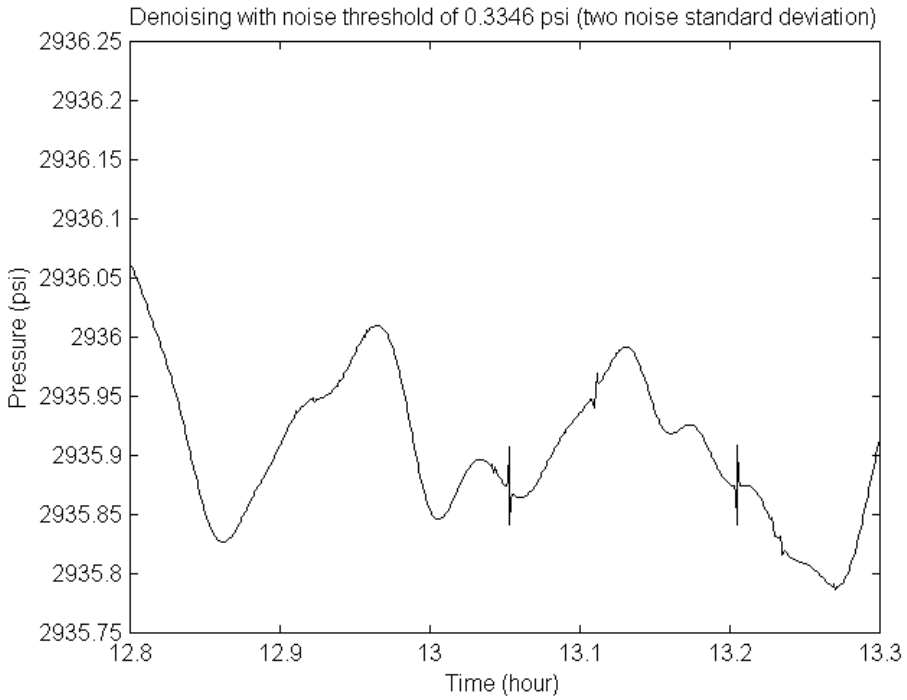


Figure 3.16: Denoising threshold at two noise standard deviation.

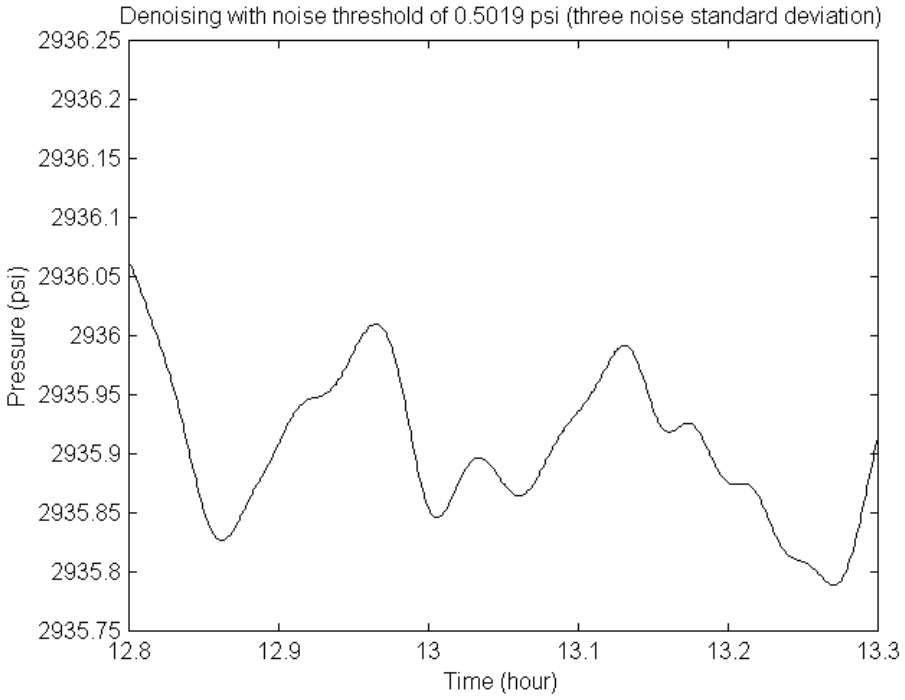


Figure 3.17: Denoising threshold at three noise standard deviation.

3.4. Break Point Selection

It was found that the number of break points detected by Athichanagorn's algorithm is sensitive to the wavelet sample spacing and slope detection threshold used. The number of break points detected is higher for a low value of wavelet sample spacing when the slope detection threshold is held constant. Similarly, the number of break points detected is higher for a low value of slope detection threshold as the wavelet sample spacing is held constant.

Each combination of wavelet sample spacing and slope detection threshold may give a nonunique set of detected break points. Setting both processing parameters to high values is likely to cause the *Wavelet* algorithm to fail to detect some break points. Setting both processing parameters to low values will cause too many break points to be detected where some of them are false break points detected between the true break points.

The algorithm needs to be improved, as it is not robust enough as an automatic transient detection algorithm in its present form. One solution could be to combine all the different break points detected by various combinations of wavelet sample spacing and slope detection threshold so it would be likely to have all the true break points detected. The false break points detected would then be screened out using an additional signal processing. The possibility of using Fourier transform to screen out false break points was investigated in this project and this discrimination method will be discussed further in Chapter 4.

Valuable information is ignored if only the combination of unique break points is selected from the break points obtained by using various combinations of wavelet sample spacing and slope detection threshold. The frequency with which a given break point is detected provides valuable information of its likelihood of being a true break point. Statistical methods to distinguish true break points from false break points were investigated in this project and will also be presented in Chapter 4.

Figure 3.18, Figure 3.19, Figure 3.20 and Figure 3.21 show the possibility of detecting different sets of break points when the combination of wavelet sample spacing and slope detection threshold used is different.

It can be seen that the number of break points detected is highest when both the wavelet sample spacing and slope detection threshold used are low and many false break points are found. On the other hand, the number of break points detected is lowest when both the wavelet sample spacing and the slope detection threshold used are high and some valid break points are missed.

It is difficult to find the combination of wavelet sample spacing and slope detection threshold pair that will detect all the valid break points and avoid any false break points in just one attempt. Therefore, additional break point discrimination methodologies such as a Fourier discrimination method and a statistical break point discrimination method were investigated in this project.

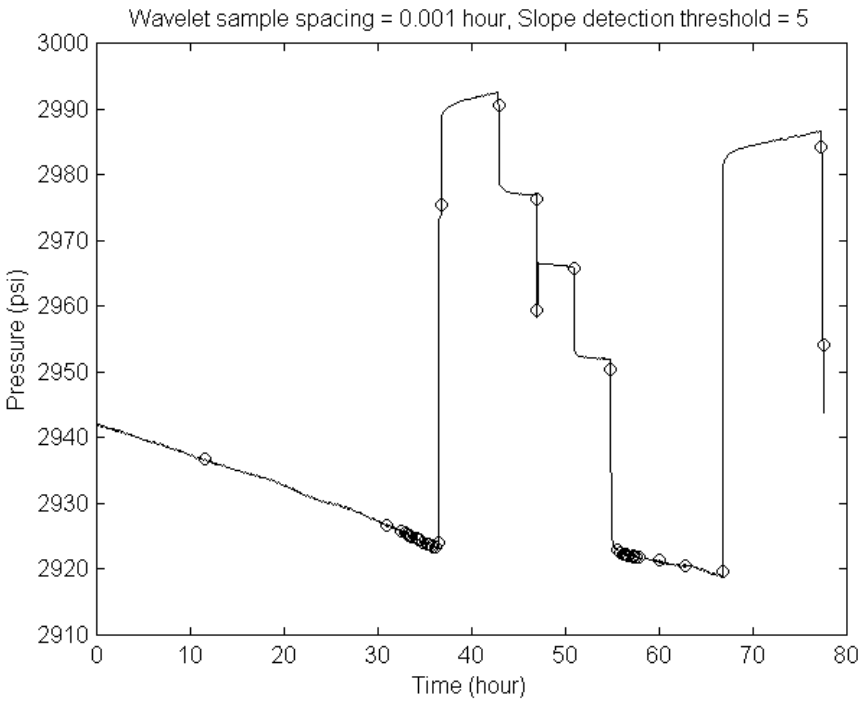


Figure 3.18: Break points for sample spacing 0.001 hour and detection threshold 5.

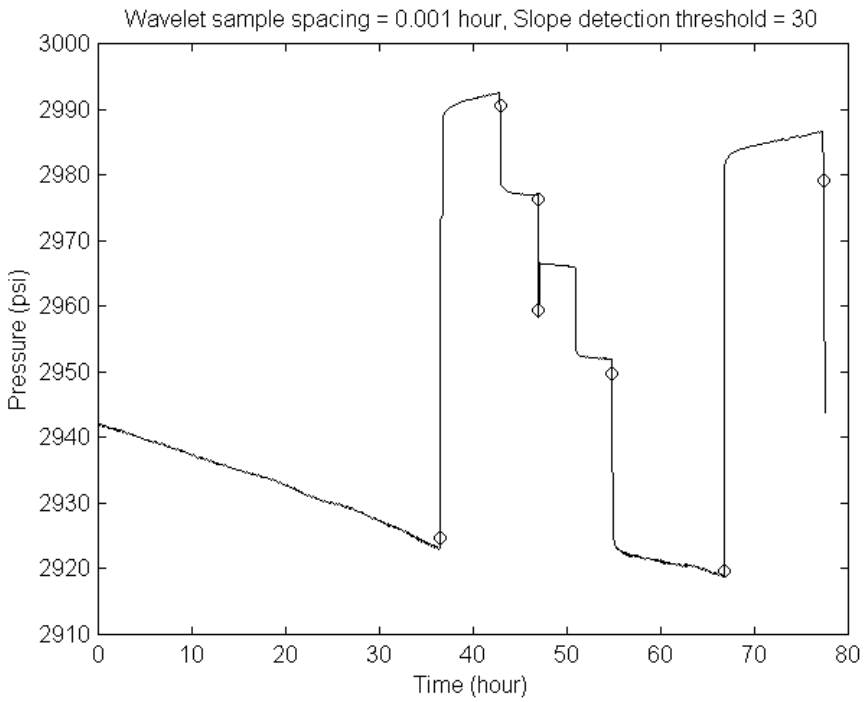


Figure 3.19: Break points for sample spacing 0.001 hour and detection threshold 30.

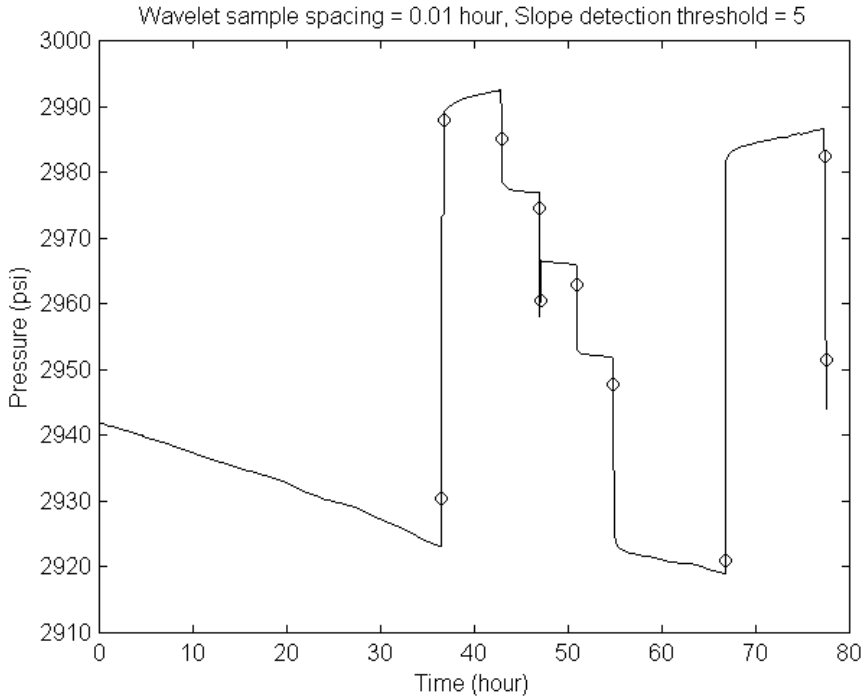


Figure 3.20: Break points for sample spacing 0.01 hour and detection threshold 5.

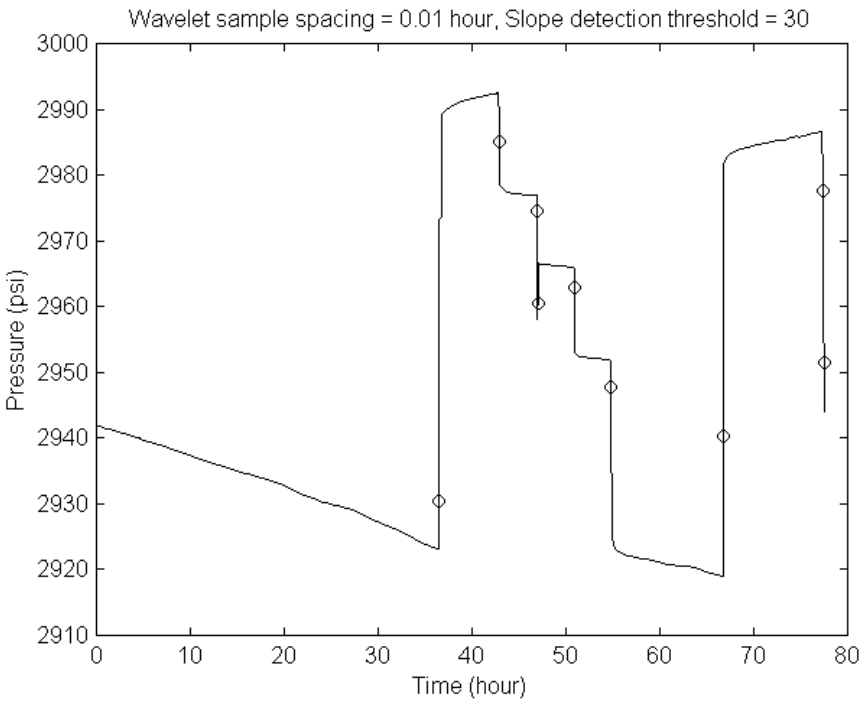


Figure 3.21: Break points for sample spacing 0.01 hour and detection threshold 30.

3.5. Break Point Adjustment

It can be seen from Figure 3.22 that break points given by the *Wavelet* algorithm do not fall exactly at the beginning of a transient. The break points may fall slightly after the beginning of a transient. The error is more severe if large wavelet sampling spacing is used. This error needs to be corrected because the *Window* algorithm requires a break point to be specified exactly at the beginning of a transient, otherwise the reservoir model cannot be matched at early time.

This error can be reduced by estimating a better break point by using the intersection of two least-square fitted straight lines, one to the left of the true break point and the other to the right of the break point. The implementation of this algorithm is discussed further in Chapter 5. This approach is illustrated in Figure 3.23. An attempt to further correct a break point estimated by straight lines intersection method by comparing one transient having perfect break point with a second transient having a break point with some error was also studied and the result is also presented in Chapter 5.

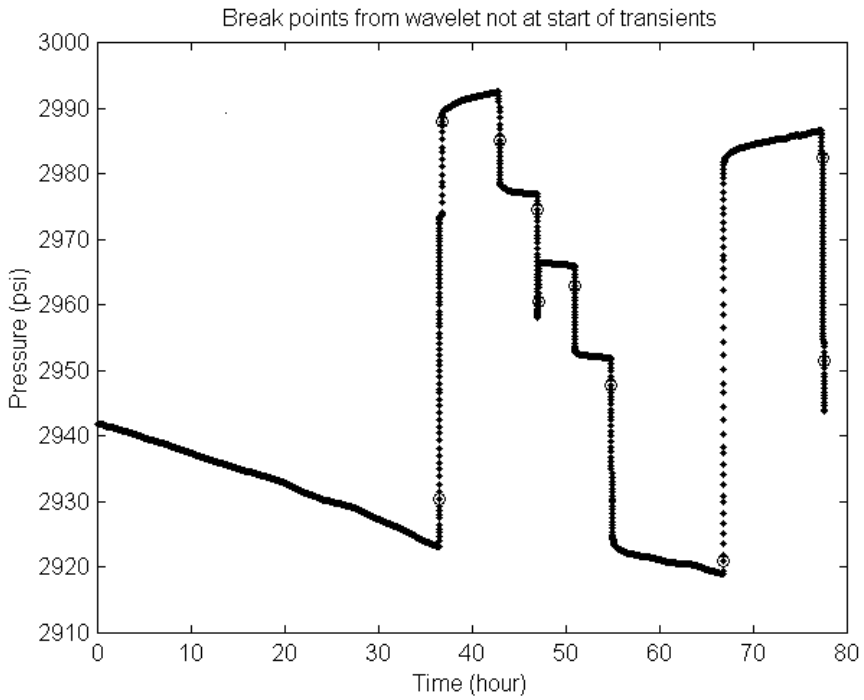


Figure 3.22: Break points from *Wavelet* algorithm.

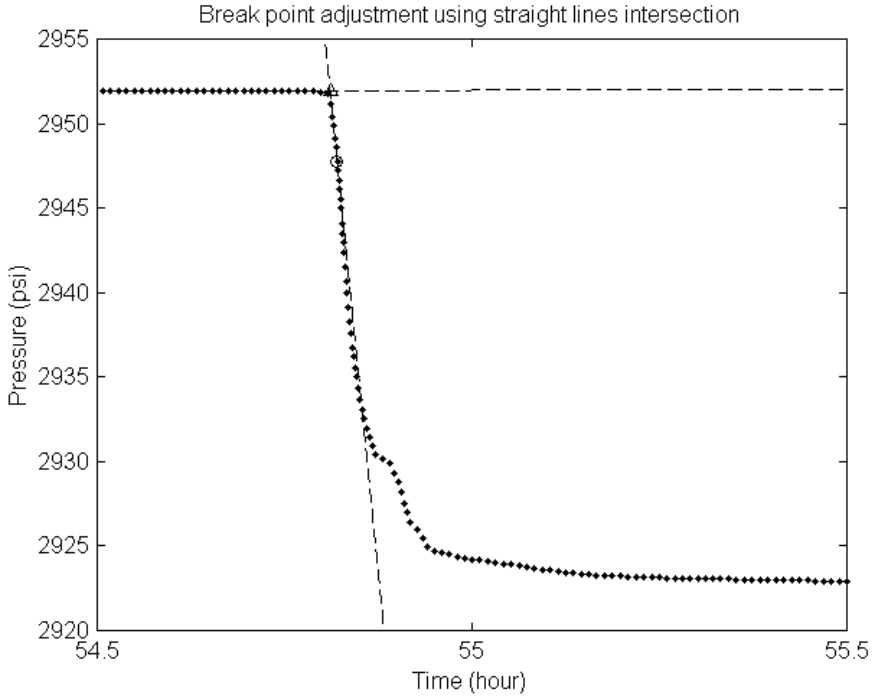


Figure 3.23: Break point adjustment using straight lines intersection.

3.6. Initial Estimate of Unknown Rate

The *Window* algorithm needs an estimate of the unknown rate if the flow rate is not available. A good initial estimate of the unknown rate is essential for the *Window* algorithm to work. A poor initial estimate will cause the regression routine to fail to converge to the correct answer or even fail to converge to any answer at all.

A change of flow rate and the resulting pressure drop are proportional to each by Darcy's Law. Hence the pressure change of each transient is proportional to the rate change in that transient. This proportionality should be equal to the ratio of the pressure change to the rate change of the transient and this ratio should be almost the same for transients close to each other. The proportionality constant of one transient can be used to estimate the unknown flow rate of its neighboring transients. On the other hand, it is not practical to use just one pressure point to calculate this proportionality constant because it is possible to select a noisy point. In this work the area under a transient was used and was found to be a more robust approach. This approach is described in Chapter 6.

3.7. Summary of Improvements to the *Wavelet* Algorithm

Before application of Athichanagorn's *Wavelet* algorithm, the data need to be checked for data overlap and step outliers and also the noise level needs to be determined. After *Wavelet* processing, the true break points need to be discriminated from false break points, as it is difficult to find the combination of wavelet sampling spacing and slope detection threshold to pick all the true break points without selecting any false break point at all. The break points also need to be adjusted because they may not fall exactly at the beginning of transients. A good initial estimate of the unknown rates is also essential. Figure 3.24 shows the pre-*Wavelet* and post-*Wavelet* processing steps.

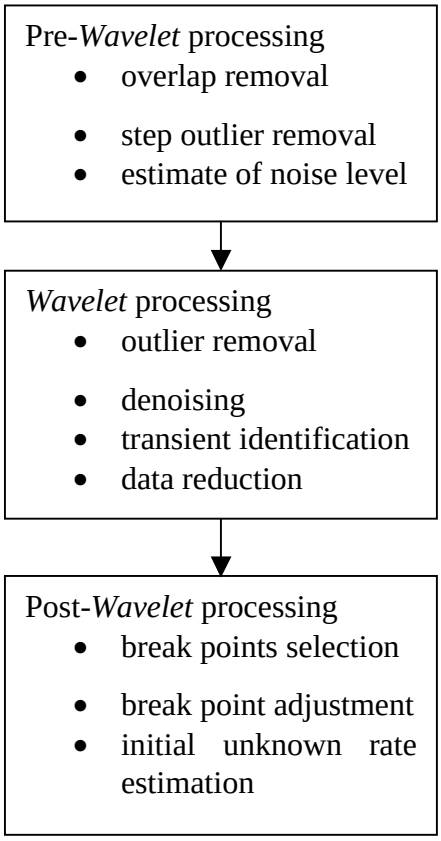


Figure 3.24: Pre-*Wavelet* and Post-*Wavelet* processing steps.

Chapter 4

4. Break Point Selection

4.1. Fourier Discrimination

The detection of break points depends on the setting of the values of the wavelet sample spacing and the slope detection threshold used. The difficulty in choosing the combination of wavelet sample spacing and slope detection threshold that will select all the valid break points while avoiding any false break point motivated the effort to investigate an alternative method to pick break points.

Time series data can be transformed into the frequency domain and for a discrete signal, the data have to be sampled at even intervals. The *Wavelet* algorithm generates an evenly sampled signal from field data that may be sampled at even time intervals. The pressure threshold for data reduction should be set to a very high value so that data reduction is only constrained by the time threshold.

A section of evenly sampled pressure transient data is shown in Figure 4.1 with three break points. Break point one is at the beginning of the transient caused by an increase in flow rate, break point two is at the flat region of a transient and break point three is at the beginning of the transient caused by a decrease in flow rate. The time between data points is checked to make sure that the wavelet algorithm gives evenly sampled data. An interval of 0.01 hour between data samples was used in the *Wavelet* algorithm and the histogram of time between samples (Figure 4.2) shows the data is indeed sampled at regular time intervals. Two-hour data windows centered at each of the break points were used for the windows of a discrete Fourier transform.

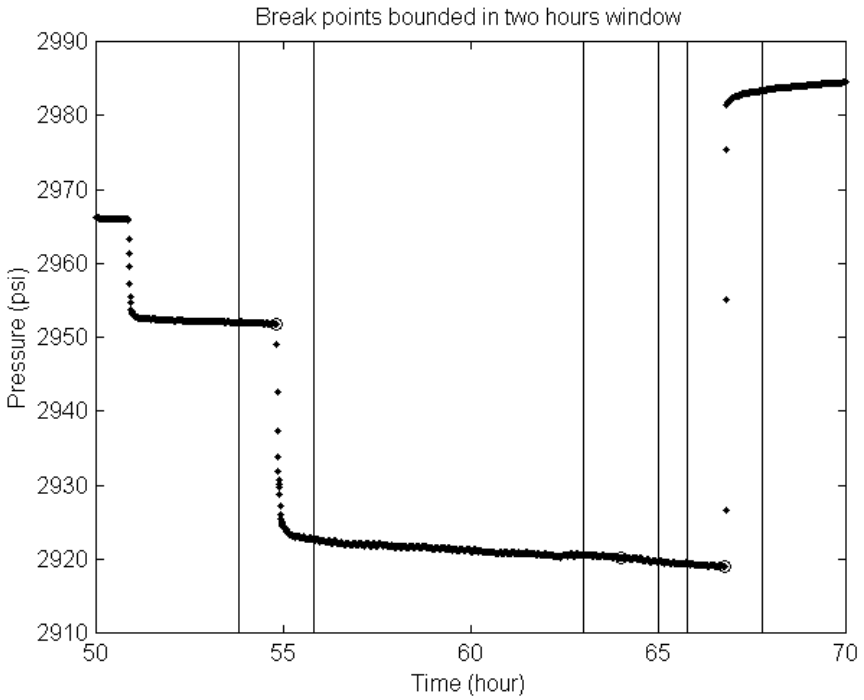


Figure 4.1: Break points in two hours wide data window.

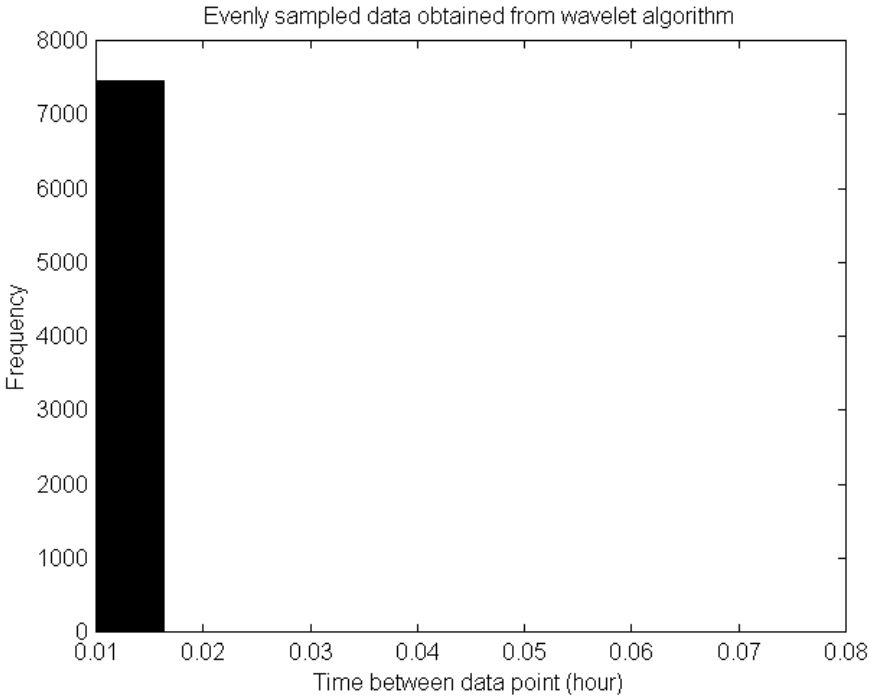


Figure 4.2: *Wavelet* algorithm can provide evenly sampled data.

A continuous signal $h(t)$ sampled at a finite number of sample points can be represented as h_k with sampling interval Δ (Press et al., 1994).

$$h_k = h(t_k), \qquad t_k = k\Delta, \qquad k = 0, 1, 2, ..., N-1 \tag{4.1}$$

The Fourier transform of $h(t)$, $H(f)$ at all the values of f in the range - f_c to f_c estimated at discrete values f_n is $H(f_n)$ where f_c is the Nyquist critical frequency. The integral in Equation (4.3) is approximated by a discrete sum.

$$f_c = \frac{1}{2\Delta}, \qquad n = -\frac{N}{2}, ..., \frac{N}{2} \tag{4.2}$$

$$H(f_n) = \int_{-\infty}^{\infty} h(t) e^{2\pi i f_n t} dt \approx \sum_{k=0}^{N-1} h_k e^{2\pi i f_n t_k} \Delta = \Delta \sum_{k=0}^{N-1} h_k e^{\frac{2\pi i k n}{N}} \tag{4.3}$$

The final summation in Equation (4.3) is called the discrete Fourier transform of the N points h_k. The discrete Fourier transform maps N complex numbers (the h_k) into N complex numbers (the H_n). The transform does not depend on any dimensional parameter, such as the time scale Δ.

$$H_n = \sum_{k=0}^{N-1} h_k e^{\frac{2\pi i k n}{N}} \tag{4.4}$$

The pressure value of the break point is subtracted from the discrete pressure data in each interval before transforming the signal into the frequency domain. The real and imaginary values of the N complex numbers of the Fourier transform are plotted to find out differences for each of the windows for each break point. Figures 4.3, 4.4 and 4.5 show the discrete data for break point one, Figures 4.6, 4.7 and 4.8 for break point two and Figures 4.9, 4.10 and 4.11 for break point three.

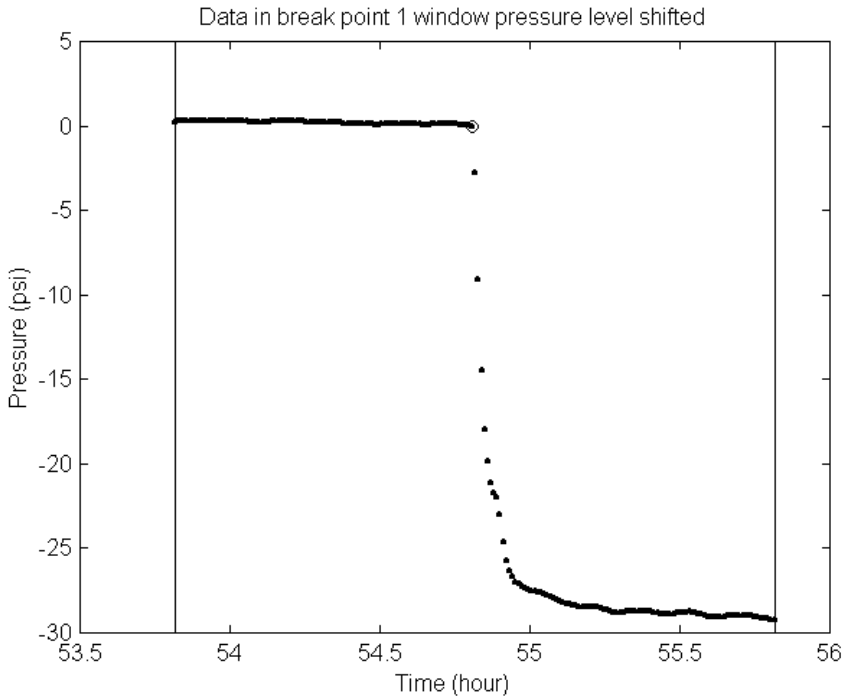


Figure 4.3: Discrete pressure data in break point 1 window.

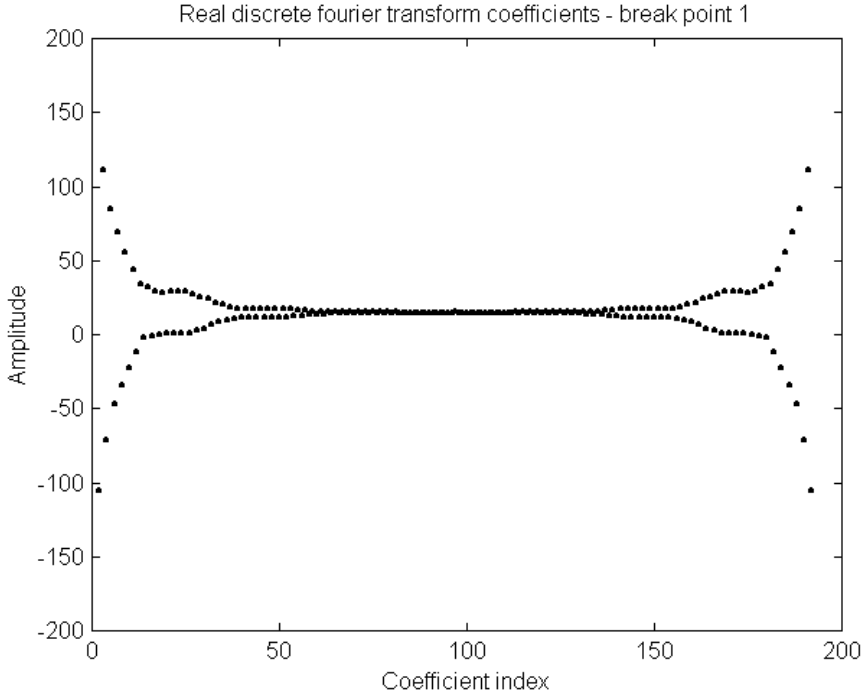


Figure 4.4: Real discrete Fourier transforms coefficients for break point 1.

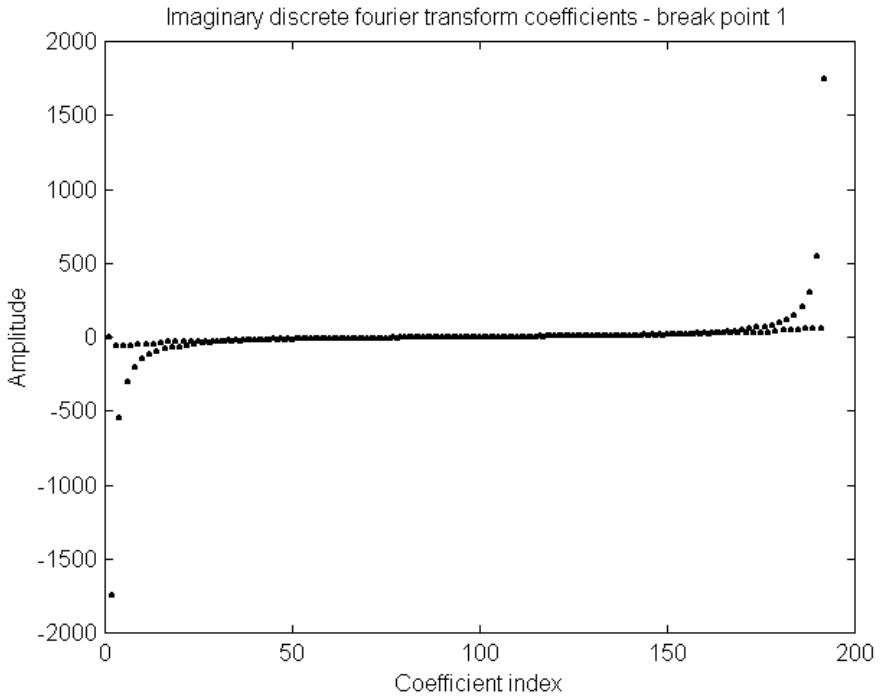


Figure 4.5: Imaginary discrete Fourier transforms coefficients for break point 1.

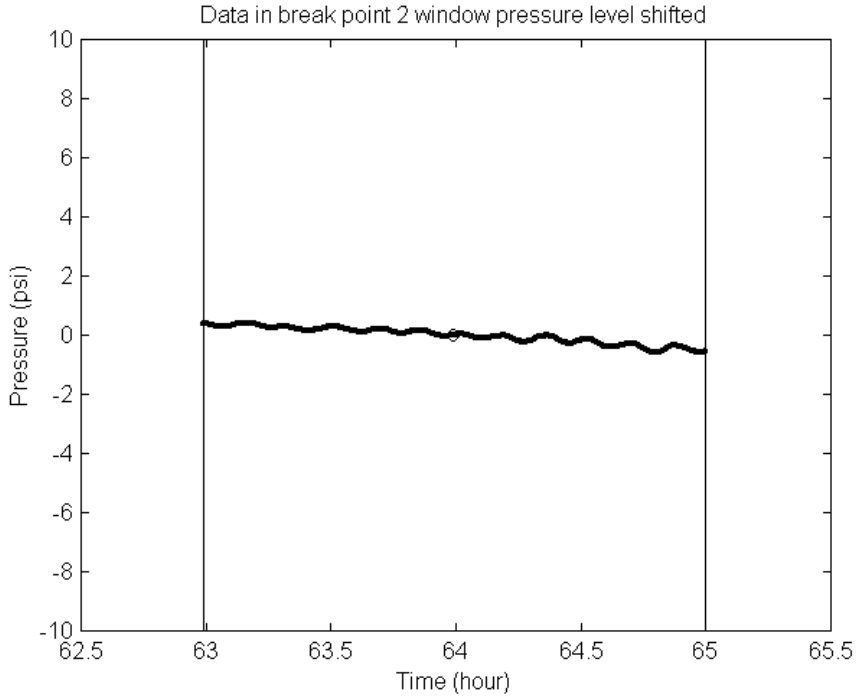


Figure 4.6: Discrete pressure data in break point 2 window.

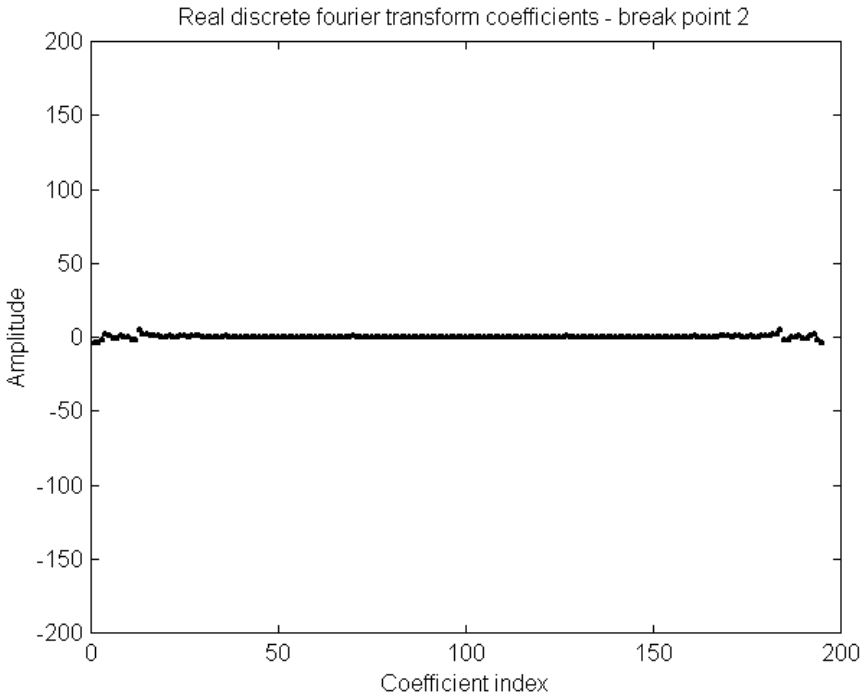


Figure 4.7: Real discrete Fourier transforms coefficients for break point 2.

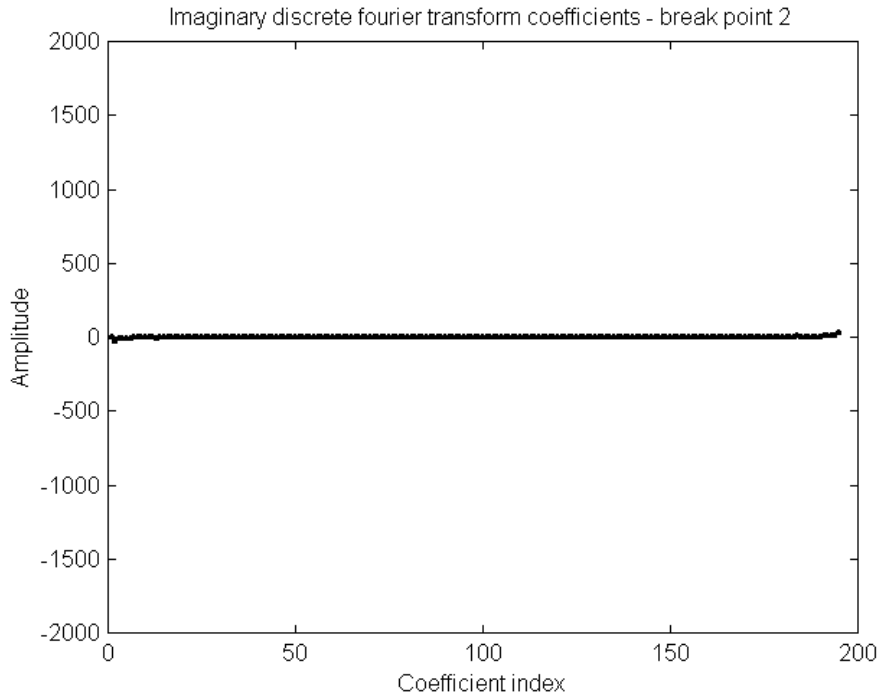


Figure 4.8: Imaginary discrete Fourier transforms coefficients break point 2.

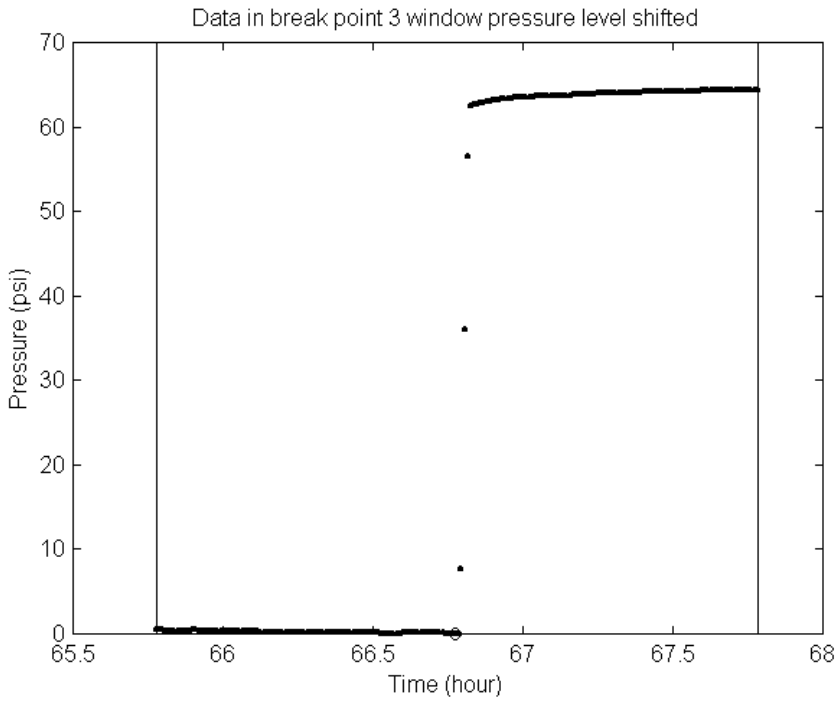


Figure 4.9: Discrete pressure data in break point 3 window.

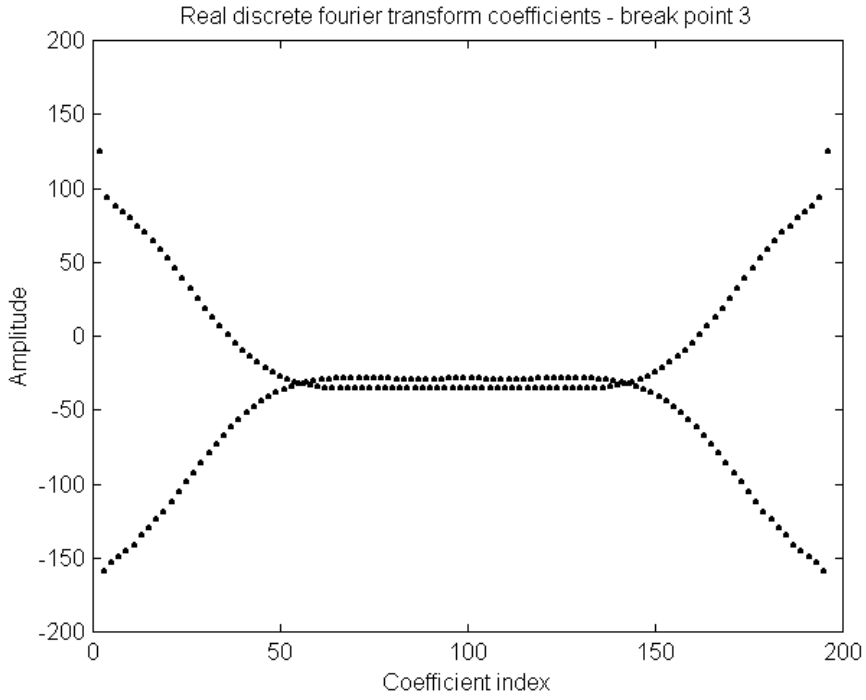


Figure 4.10: Real discrete Fourier transforms coefficients for break point 3.

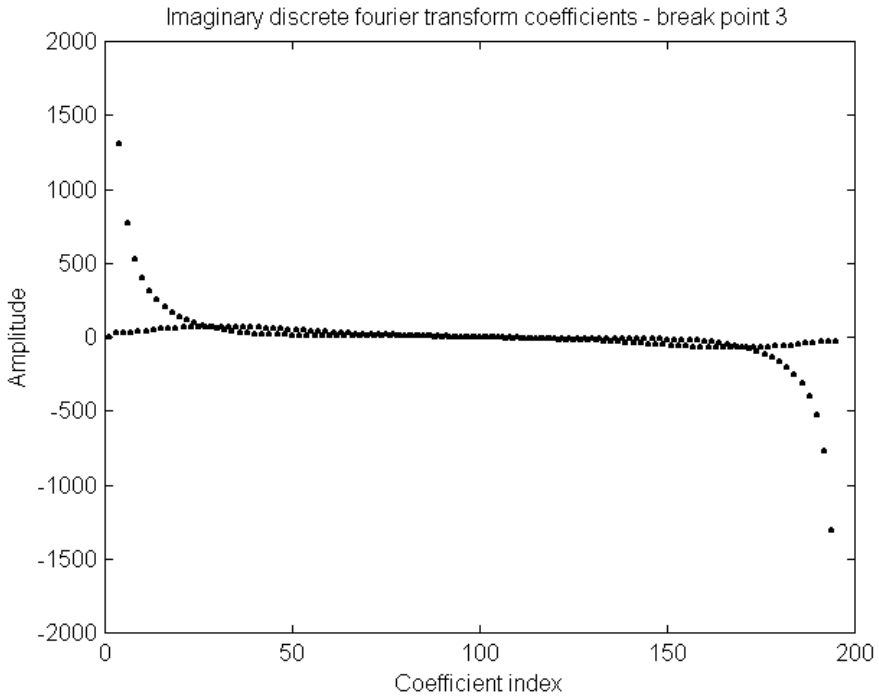


Figure 4.11: Imaginary discrete Fourier transforms coefficients break point 3.

The real component of the discrete Fourier transform coefficients amplitude at both ends of the spectrum is high when the break points are real as for break point 1 and break point 3. Break point 2, which is in the flat region of a transient, has low values of real discrete Fourier transform coefficients. One method to make use of this feature is by comparing the magnitudes or square of the real coefficients at the beginning or end of the spectrum (say by taking five percent of the points) of the discrete Fourier transform to help pick true break points. Figure 4.12 shows break points of a pressure data record and Figure 4.13 shows the sum of the squares of the real part of the discrete Fourier transform coefficients at the beginning and end of the spectrum.

The imaginary parts of the discrete Fourier transform coefficients are highly negative at the beginning and highly positive at the end of the spectrum when the transient is caused by an increase of flow rate as in break point 1. On the other hand, the imaginary part of the discrete Fourier transform coefficients are highly positive at the beginning and highly negative at the end of the spectrum when the transient is caused by a decrease of flow rate as in break point 3.

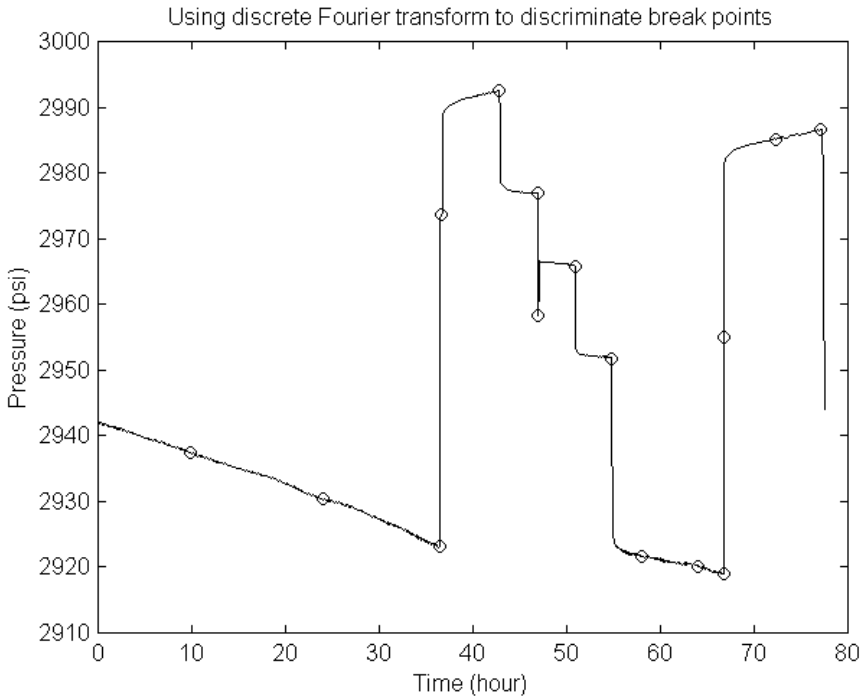


Figure 4.12: All identified break points (true and false).

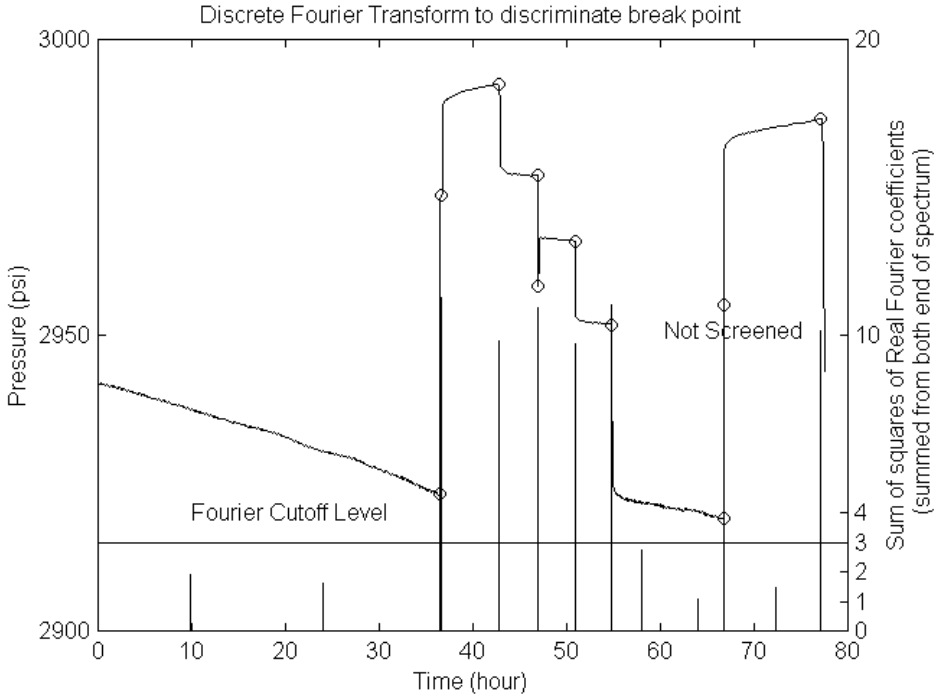


Figure 4.13: Using sum of squares of the real Fourier coefficients (from the beginning and end of a spectrum) to identify true break points.

It was found that the discrete Fourier transform method could be used to screen out false break points. In Figure 4.13, one false break point could not be screened. This break point is at a place of rapidly increasing pressure, just 0.01 hour after the true break point. One remedy is to combine break points that are within a few sampling intervals of each other. It was also found that another limitation of the discrete Fourier transform method is the requirement of having a window of data. Data windows may overlap one another if the Fourier transform window is too wide or the break points detected by the *Wavelet* algorithm are close to one another. Figure 4.14 shows that break points detected by *Wavelet* algorithm can be close to each other.

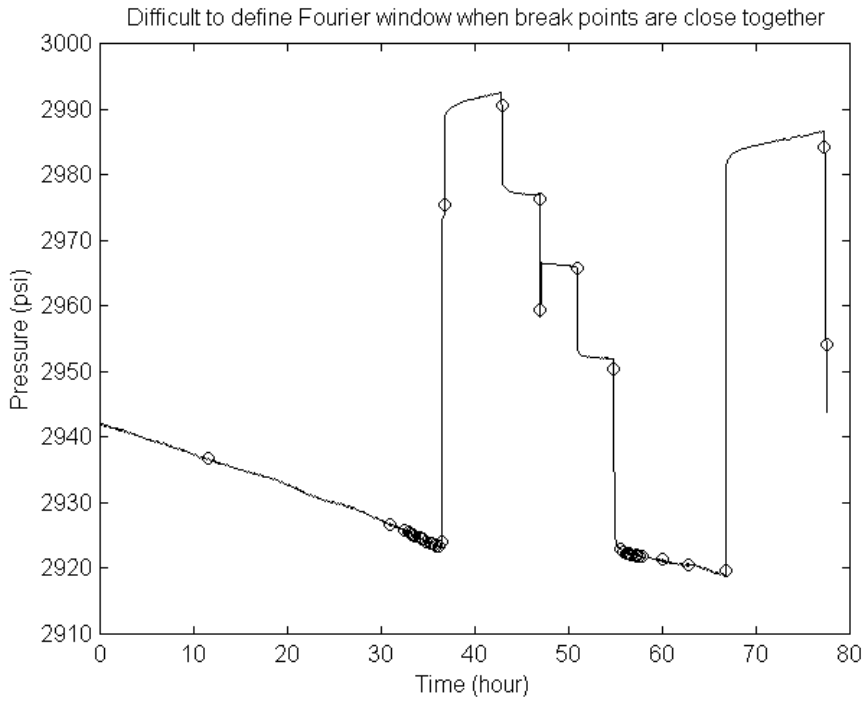


Figure 4.14: Break points detected by wavelet algorithm can be close to each other.

The ability to screen false break points from all the break points detected from the combination of wavelet sample spacing and slope detection threshold is limited because of the requirement of having a window of data in the Fourier transform. On the other hand, if the break points detected are binned at a certain bin width and screened statistically before using the Fourier method, this limitation can be overcame. Statistical screening will be discussed in the next section.

4.2. Statistical Discrimination

The detection of break points based on the wavelet transform coefficient in the *Wavelet* algorithm depends on the setting of the value of wavelet sample spacing and slope detection threshold used. It was found that for each data set there are lower and upper bounds of wavelet sample spacing within which the *Wavelet* algorithm would be able to execute.

The lower bound is very close to the minimum wavelet sample spacing suggested by the algorithm and can be found by performing a binary search in the vicinity of the minimum wavelet sample spacing. The upper bound has been found to be lower than the average wavelet sample spacing suggested by the *Wavelet* algorithm most of the time. The search for the upper bound can start with the average wavelet sample spacing and is refined using the binary search method until the upper bound is found.

Athichanagorn (1999) suggested that the hybrid noise thresholding method is able to retain the good features of both soft thresholding and hard thresholding but it was found here that hybrid thresholding sometimes cause the *Wavelet* algorithm to fail. The soft thresholding method usually works in situations when the hybrid thresholding method fails.

The break points detected using different combinations of the wavelet sample spacing and the slope detection threshold for a simulated data set and a real field data set were investigated. It was found that the *Wavelet* algorithm is able to pick all the true break points with no false break points for the simulated data set. The effect of changing the wavelet sample spacing and the slope detection threshold is illustrated in Figure 4.15. Medium sized filled circles are plotted in Figure 4.15 when all true break points are detected with no false break point (this represents all cases shown).

Figure 4.16 shows the frequency of the break points detected for the simulated pressure transient when all the detected break points are combined and binned into a histogram.

Figures 4.17 and 4.18 shows the characteristics of the detected break points for the real field data as a function of wavelet sample spacing and slope detection threshold values.

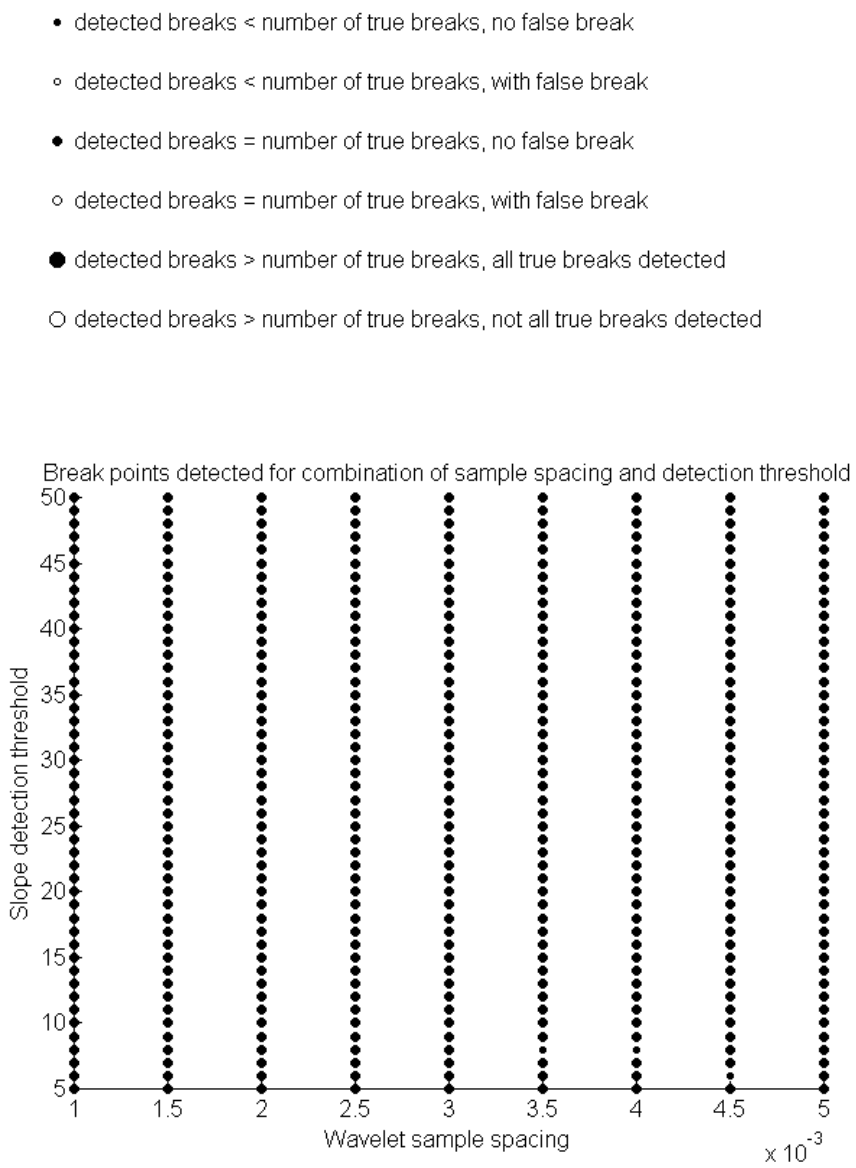


Figure 4.15: Characteristics of break points detected for a simulated pressure transient.

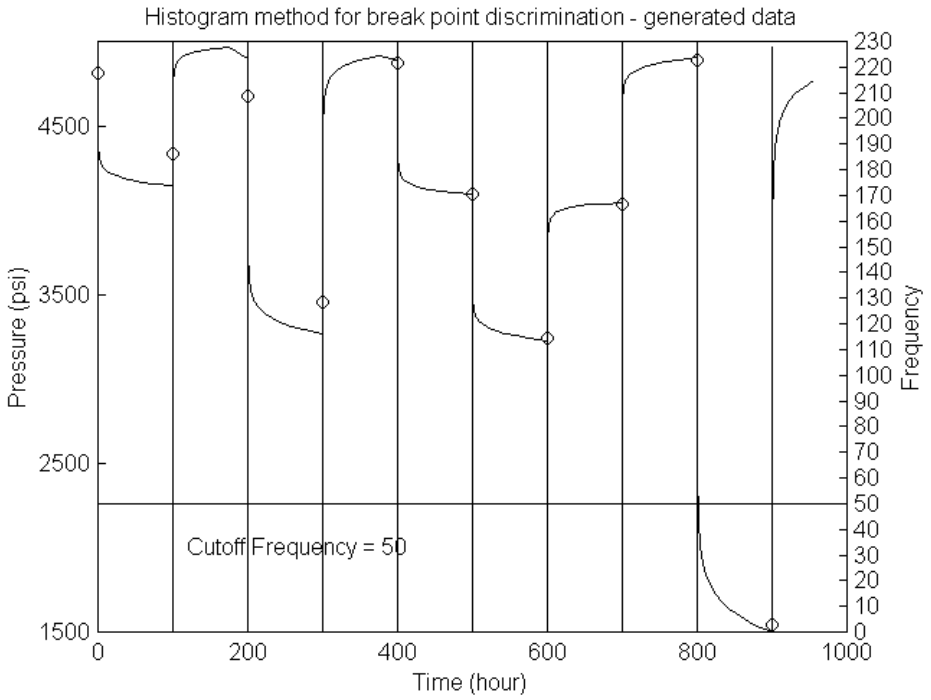


Figure 4.16: Histogram of detected break points for simulated pressure transient.

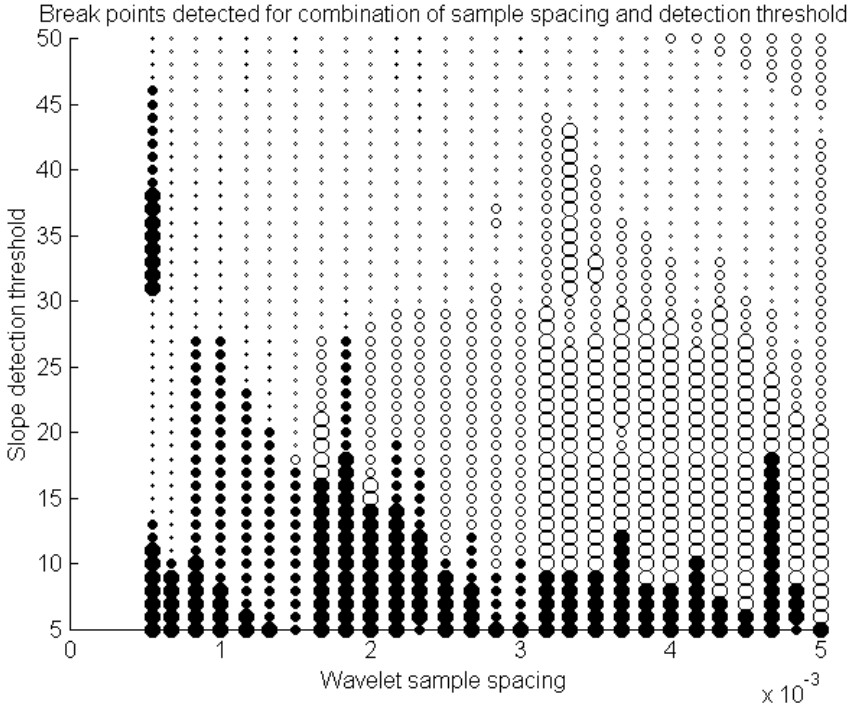


Figure 4.17: Characteristics of detected break points for real field data.
(Legend as in Figure 4.15 and data as in Figure 4.19).

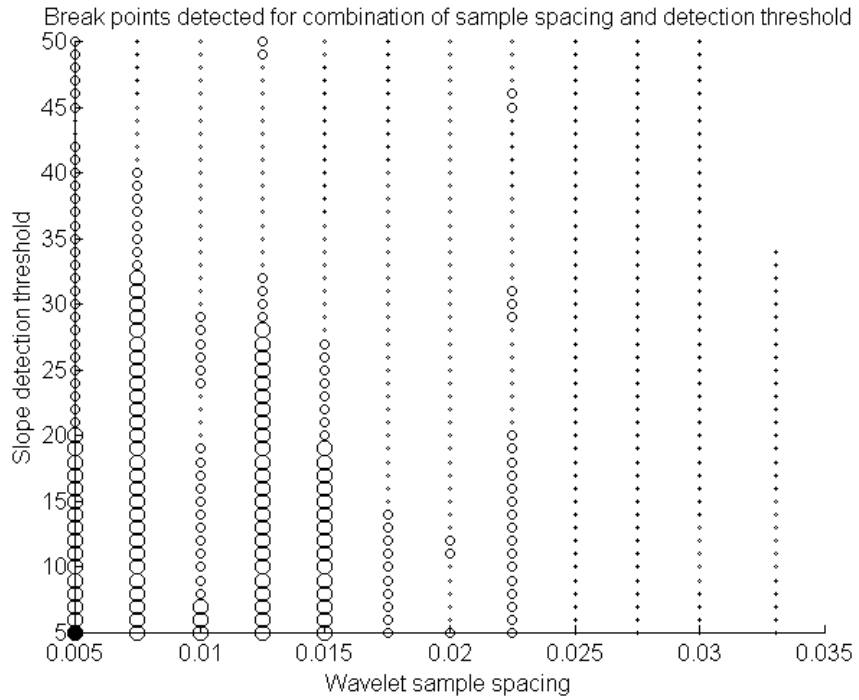


Figure 4.18: Characteristics of detected break points for real field data
(Legend as in Figure 4.15 and data as in Figure 4.19).

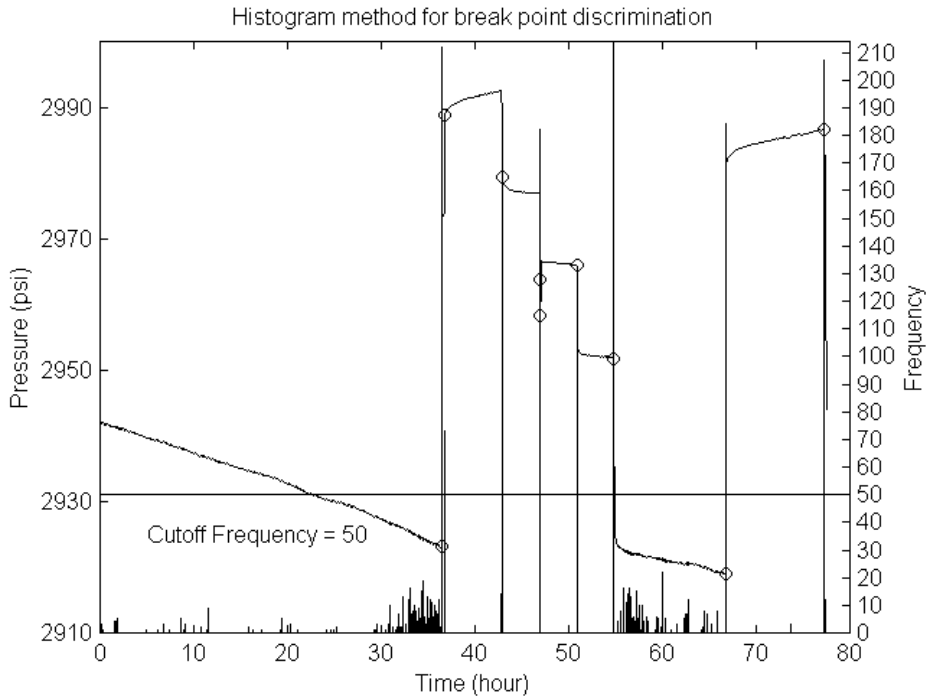


Figure 4.19: Histogram of detected break points for real field data
(for sample spacing and slope detection threshold in Figure 4.17 and 4.18).

As shown in Figure 4.17 and Figure 4.18, it was found that for real field data, the number of break points detected is sensitive to the values of wavelet sample spacing and slope detection threshold used in the *Wavelet* algorithm. All the true break points tend to be detected for low values of wavelet sample spacing and slope detection threshold but false break points are often detected at the same time. At high wavelet sample spacing and slope detection threshold, not all true break points could be detected but there is a lower likelihood of detecting false break points.

It is useful to know the combinations of values of wavelet sample spacing and slope detection threshold that do not completely detect all the true break points but do not pick any false break point since these combinations increase the frequency of detection of the true break points and thus help to screen out false break points statistically.

The research results show that the *Wavelet* algorithm picks break points successfully for an artificial noiseless pressure transient data. The combination of wavelet sample spacing and slope detection threshold that correctly picks all the true breaks and no false break points is not easy to determine for real field data, although the statistical method is a useful tool to screen out false break points. Additional screening algorithms such as the Fourier discrimination method could be applied after using the statistical discrimination algorithm.

Chapter 5

5. Break Point Adjustment

5.1. Least Square Straight Line Intersection

The break points given by the *Wavelet* algorithm often do not fall exactly at the beginning of a transient, these results in an error in the break time and break pressure. It was also found that most of the time the detected break point falls to the right of the true break point. As mentioned earlier, this error can be reduced by adjusting the break point using the intersection of two least-square fitted straight lines, one to the left of the true break point and the other to the right of the break point.

A window of data points is defined at the vicinity of a breakpoint; the number of points in the window depends on the sampling rate of the data set. A window width of 0.1 hour to the right and 0.1 hour to the left of the break point was found to be sufficient most of the time. The minimum number of data points in an adjustment window has to be specified also because time specification alone is not robust for sparsely sampled data. The number of points used to fit the least square straight line depends on the available data points in the window and the noise level of the data set. A minimum of three points is needed to fit a least square straight line but this is only the case for ideal noiseless data. In practice, between five to ten points will be needed to fit the line.

The steepest line with the maximum absolute slope value is sought within a window. This line is the fitted straight line right of a break point. Another straight line that has the minimum absolute slope within the window is sought, this is the straight line left of a break point. The intersection of the left and the right lines defines the adjusted break point within the defined window of data. A denser data set will give better break point adjustment compared to a sparse data set. Figures 5.1 and 5.2 compare break point adjustment using data with high and low sampling rates.

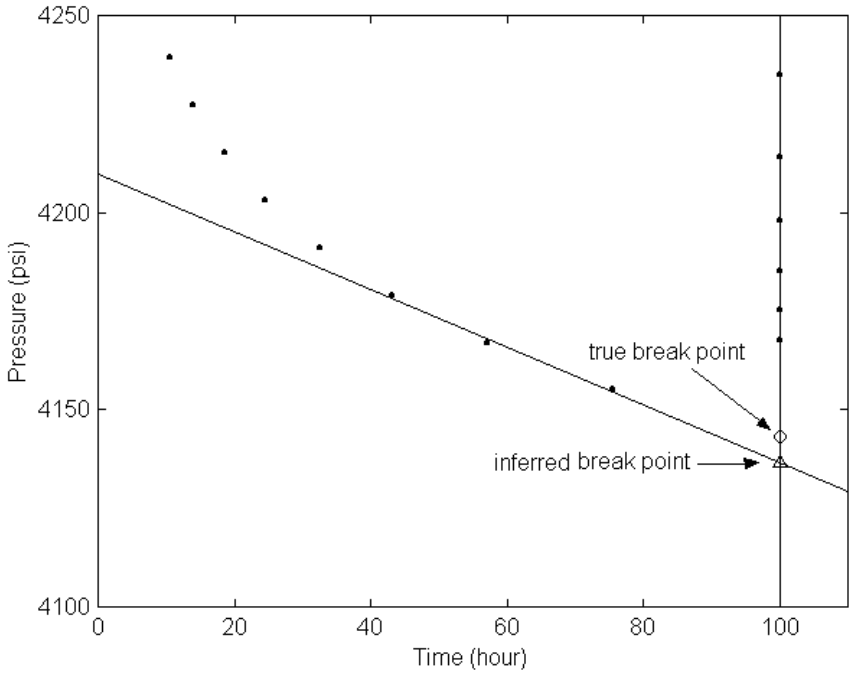


Figure 5.1: Less error in adjusted break point for high data sampling rate.

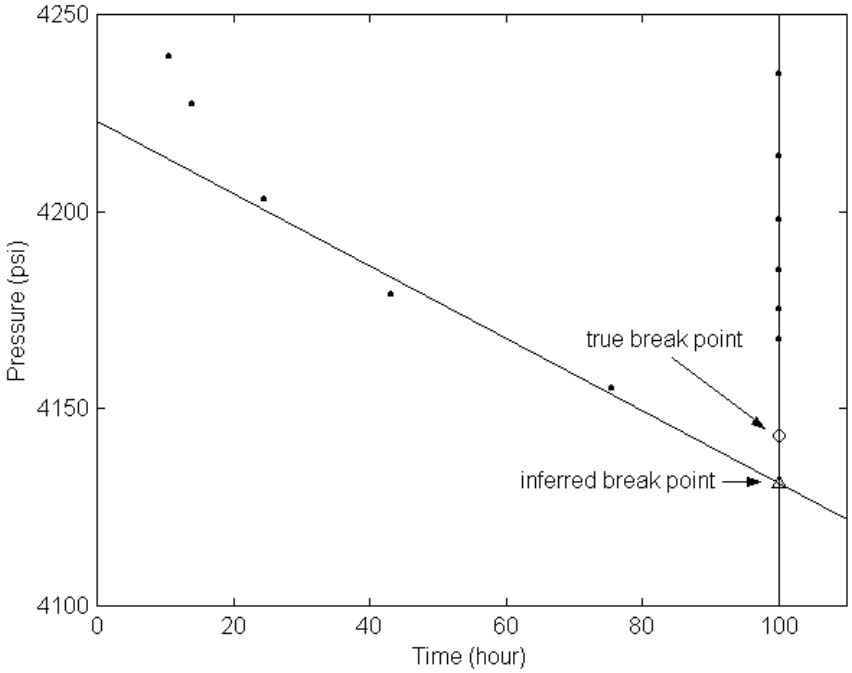


Figure 5.2: More error in adjusted break point for low data sampling rate.

When more points are used to fit the least square straight lines, the difference between the adjusted break point and the true beginning of a transient increases. Figure 5.3 has the same data sampling density as Figure 5.1 but the least square straight lines were fitted using more points and this resulted in a larger error in the adjusted break point. A noisy data set will require more points to fit a straight line reliably but this is in conflict with the least square straight line adjustment method that prefers a smaller number of points. The requirement of the least square straight line method thus requires that the pressure data have to be properly denoised and a high sampling rate data be used in order to reduce the break point adjustment error.

As mentioned earlier, the break point detected by the *Wavelet* algorithm tends to be to the right of the true beginning of the transient and the suggested window of data should be about 0.05 hour to the right and 0.15 hour to the left of the detected break point. The number of data points in the window can be calculated if the data sampling rate is known and the number of points defining a line should be about one tenth to one fifth of the points in the window.

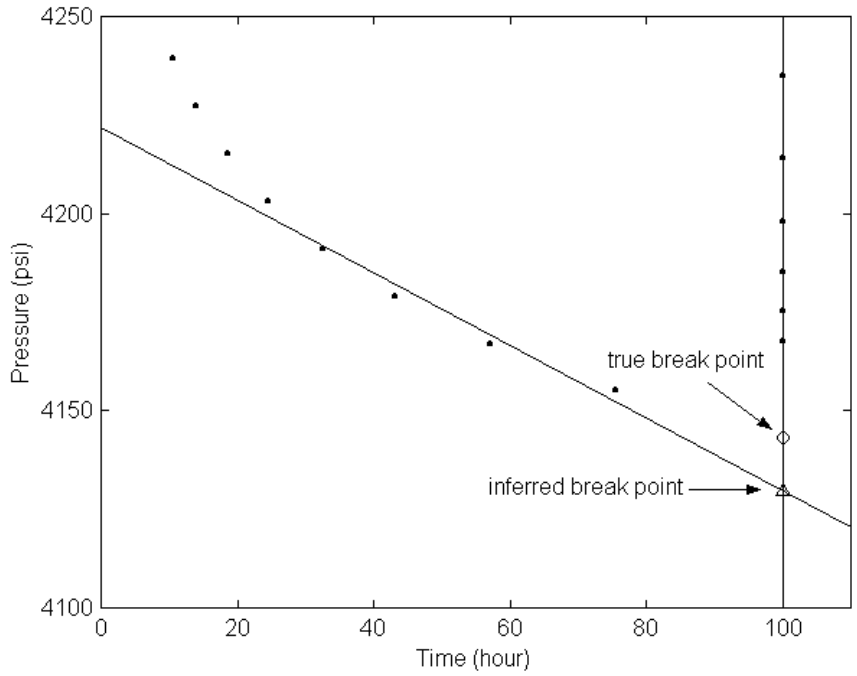


Figure 5.3: Least square line fitted with more points compared to Figure 5.1.

5.2. Nudge Time and Nudge Pressure

Adjusting the break point based on the intersection of two fitted least square straight lines does not perfectly correct the break time and break pressure to the actual beginning of a transient. In this research, the possibility of further correcting break time and break pressure of one transient based on a reference transient was investigated. The two pressure transients have their respective break time and break pressure values subtracted to obtain the time and pressure changes from the beginning of a transient. Negative pressure changes are transformed to positive values by taking the absolute values of the pressure changes. These two transformed pressure transients are then overlaid onto one another over the same time interval. Figure 5.4 shows two adjacent transients with the second transient's break point to be adjusted. Figures 5.5, 5.6 and 5.7 show the two transients overlaid onto one another.

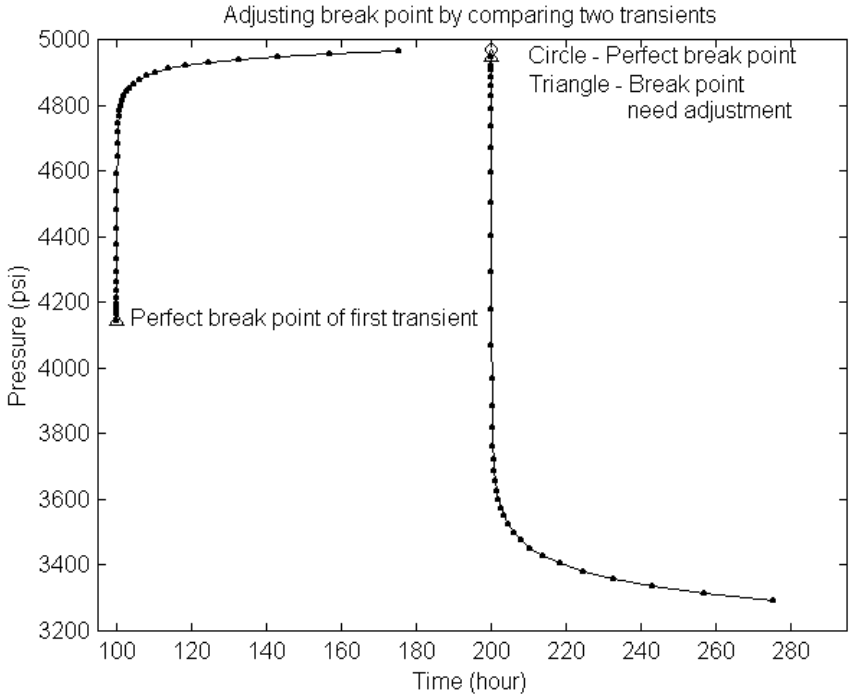


Figure 5.4: The left transient is the reference and the right transient is to be adjusted.

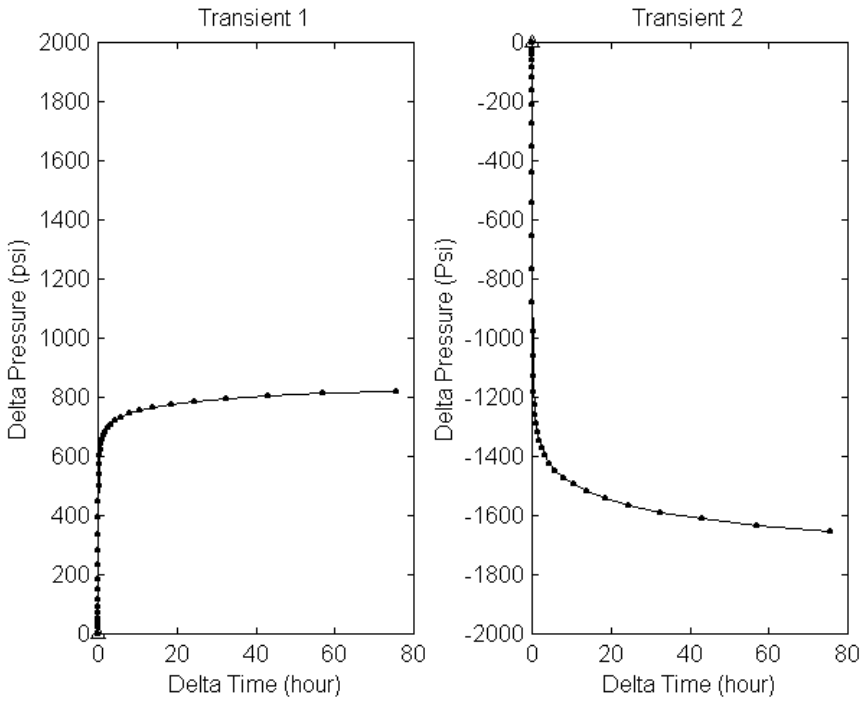


Figure 5.5: Time and pressure changes from the beginning of transients.

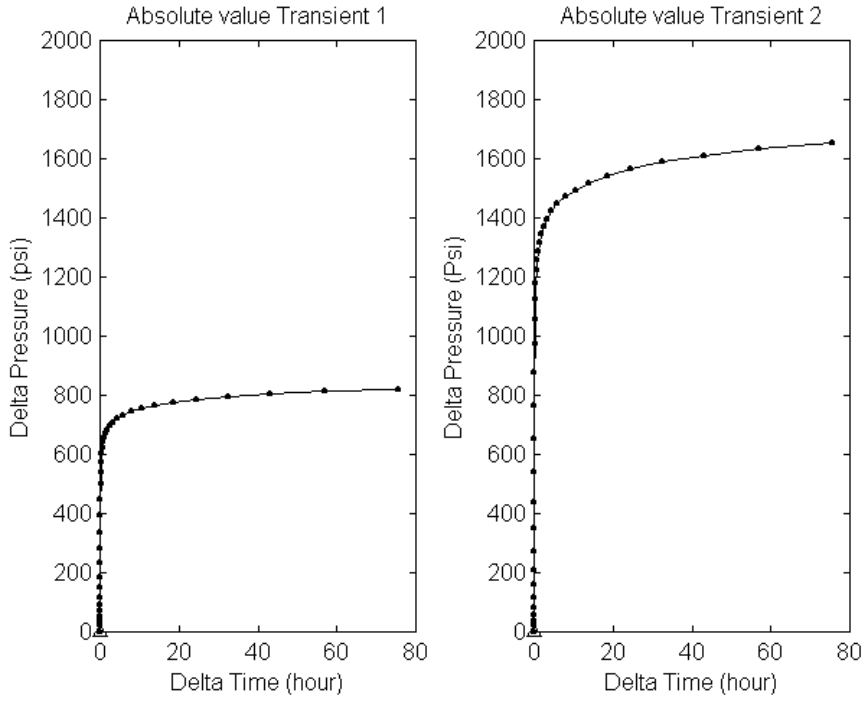


Figure 5.6: Absolute values of pressure changes are taken.

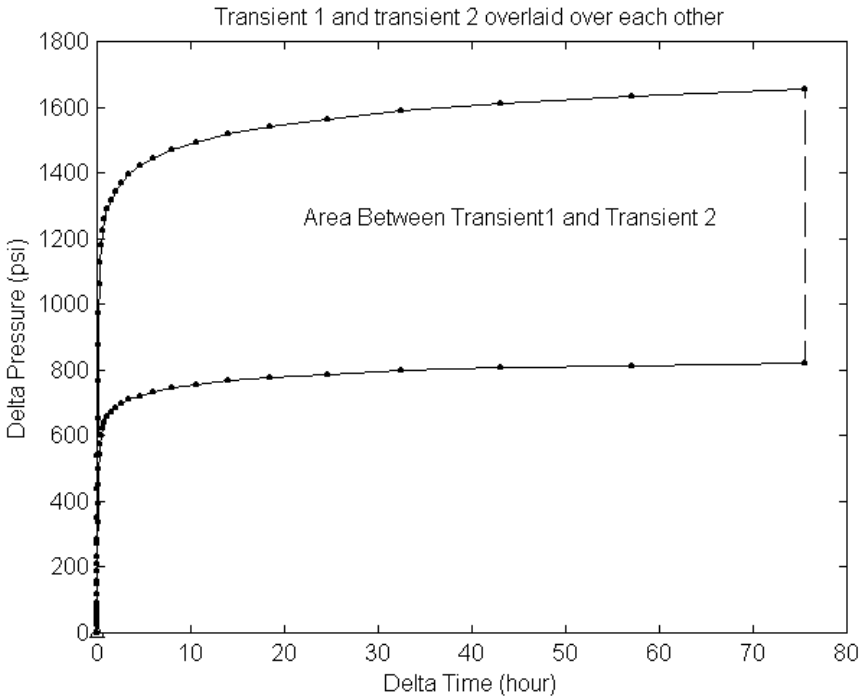


Figure 5.7 Transient 1 and Transient 2 overlaid over each other.

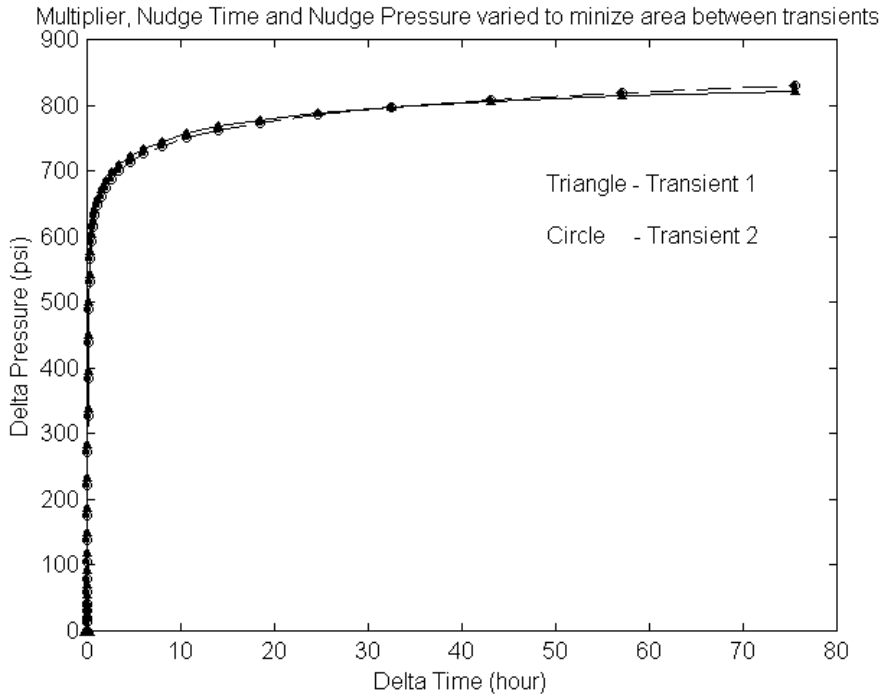


Figure 5.8: Multiplier, nudge time and nudge pressure for Transient 2 are varied to minimize the area between the transients.

The reference pressure transient was used to correct the second pressure transient by comparing the area between the two. The break time and break pressure of the second transient was allowed to change so that the area between the two pressure curves was minimized. A multiplier was also applied to the pressure value of the second pressure transient since the flow rates of the two transients would not necessarily be the same. Figure 5.8 shows how the adjusted second transient matches the reference transient.

Three parameters were adjusted; the break time of the second transient, the break pressure of the second transient and the multiplier applied to pressure value of the second pressure transient. An optimization routine was used to minimize the objective function, which is the area between the two pressure curves by varying the three parameters.

When two transients are overlaid over each other, a data point in one transient may not be available at the sampling time of the other transient and therefore has to be interpolated within each respective transient. Having interpolated the missing data points, both transients will have exactly the same number of data points. The area between the transients and between two consecutive time sampling points is a trapezoid when the transients do not cross each other, this is shown in Figure 5.9. The areas between the two transients and between two consecutive time sampling points are areas of two triangles when the transients cross one another as shown in Figure 5.10. All the areas of the trapezoids and triangles between the two transients are calculated and summed to become the total area between the two transients.

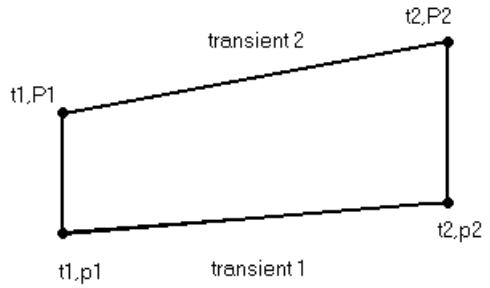


Figure 5.9: Trapezoid area between two transients bounded by two parallel time values.

The area of the trapezoid is given by the Equation (5.1).

$$Trapezoid\ Area = \left(\frac{(P1 - p1) + (P2 - p2)}{2} \right)(t2 - t1) \qquad (5.1)$$

When the transients cross one another between two successive sampling times, two triangles will be formed as shown in Figure 5.10.

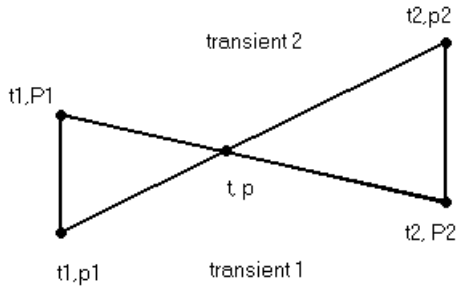


Figure 5.10: Two triangles formed when transient cross one another.

The two transients intersect each other at point *t* and *p*. The sum of the areas of the two triangles is given by Equation (5.2).

$$Area\ of\ triangles = \left(\frac{P1 - p1}{2} \right)(t - t1) + \left(\frac{p2 - P2}{2} \right)(t2 - t) \qquad (5.2)$$

The objective function to be minimized is the total area between the two transients. When there are *N* sampling points including the beginning of a transient, the objective function is given by Equation (5.3).

$$Objective\ Function = f(X) = \sum_{i=1}^{N-1} Area\ between\ t_i\ and\ t_{i+1} \qquad (5.3)$$

$X = [x_1, x_2, x_3]^T$, $x_1 = t_n$ = nudge time, x_2 = nudge pressure, x_3 = multiplier

The nudge time and nudge pressure are used to translate the second transient in relation to the first – this corresponds to finding a new location for the break time of the second transient.

5.2.1. Finite-Difference Approximations to Gradients and Hessian

The gradients and Hessian matrices in the minimization algorithm have to be approximated using the finite difference method. The gradient can be approximated using the forward-difference method (Miller, 2000), given by Equation (5.4),

$$\frac{\delta f(X)}{\delta x_j} = f_j \cong \frac{f\left(X^k + \alpha I_j\right) - f\left(X^k\right)}{\alpha} \tag{5.4}$$

$\alpha > 0$ and I_j is the *jth* column of an *n x n* identity matrix, here *n* equals 3.

The elements of the Hessian matrices h_{ij} are defined by Equation (5.5).

$$h_{ij} = \frac{\delta}{\delta x_i^k}\left(\frac{\delta f(X)}{\delta x_j^k}\right) = \frac{\dfrac{f\left(X^k + \alpha I_i + \alpha I_j\right) - f\left(X^k + \alpha I_i\right)}{\alpha} - \dfrac{f\left(X^k + \alpha I_j\right) - f\left(X^k\right)}{\alpha}}{\alpha}$$
$$= \frac{f\left(X^k + \alpha I_i + \alpha I_j\right) - f\left(X^k + \alpha I_i\right) - \left[f\left(X^k + \alpha I_j\right) - f\left(X^k\right)\right]}{\alpha^2} \tag{5.5}$$

5.2.2. Result of Break Time and Break Pressure Adjustment

It was found that the break time and break pressure of the second transient could not be corrected to the actual break time and break pressure. In order to investigate why this method failed to correctly adjust the break point of the second transient, two transients having perfect break points were overlaid. The break point of the second transient was varied in time and pressure and at each combination the multiplier of the second transient was optimized to give a minimum area between the two transients.

The resulting values of the minimum objective function were plotted in a break time and break pressure plane and it was found that the contour of the objective function is not bowl-shaped where an optimization algorithm can go downhill and converge to the minimum point. Figure 5.11 shows the contour plot of the objective function.

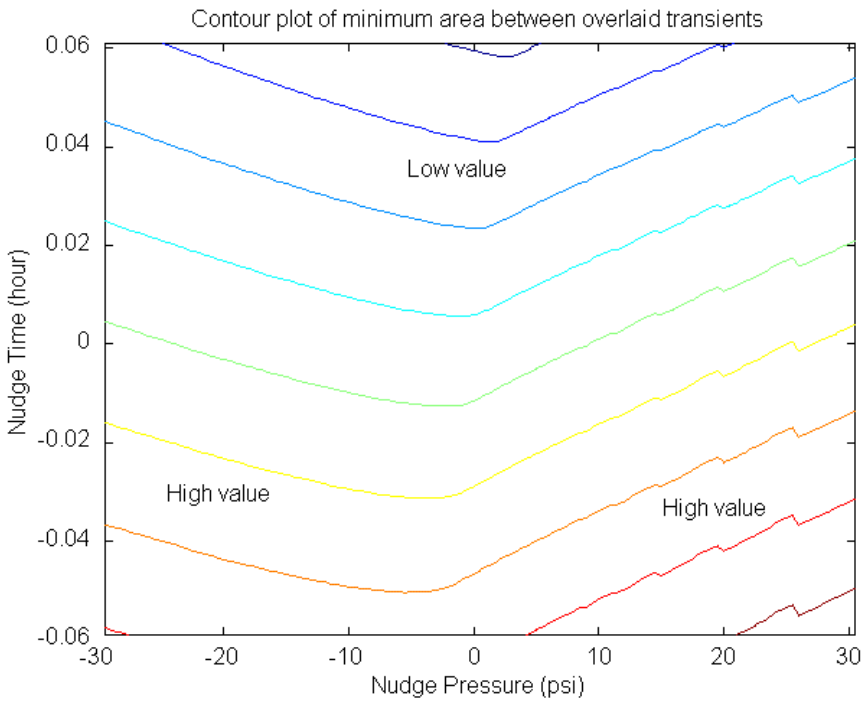


Figure 5.11: The area between two transients for various nudge time and nudge pressure combinations.

The time and pressure errors of the estimated break point using the least square straight lines intersection method cannot be further reduced based on overlaying a reference transient. The objective function (area between the two transients) is not bowl-shaped which makes it impossible for the optimization algorithm to converge to a minimum value. Also, data points are usually sparse at the beginning of a transient, which makes it more difficult to compare two transients.

Further research could discover an objective function that is bowl shaped and allow error of a break point to be corrected. For now, the least square straight line break point adjustment method will not be able to fully correct the error in an estimated break point.

Chapter 6

6. Initial Estimate of Unknown Flow Rate

6.1. Pressure Data to Estimate Unknown Flow Rate

The *Window* algorithm developed by Athichanagorn in 1999 handles the issue of interpreting pressure transient data from permanent downhole gauges when flow rate data are not available. If flow rate data are available, each transient can be analyzed separately. Athichanagorn's *Window* algorithm can estimate the unknown flow rates however an initial guess of the unknown flow rates is required to ensure the *Window* algorithm converges correctly.

Unknown rates have to be estimated from the known flow rates and the pressure data. Equation (6.1) (Horne, 1995) shows how well flowing pressure, p_{wf} is related to the flow rate changes, q at time t starting from an initial pressure p_i.

$$p_{wf} = p_i - 162.6\frac{qB\mu}{kh}\left(\log t + \log\frac{k}{\phi\mu c_t r_w^2} + 0.8686s - 3.2274\right) \qquad (6.1)$$

B is formation volume factor, μ is viscosity, k is permeability, h is reservoir thickness, ϕ is porosity, c_t is total compressibility, r_w is well radius and s is skin factor. For the same elapsed time t, an unknown flow rate change q_2 can be inferred from a transient with known flow rate change q_1 and pressure change $p_{i,1} - p_{wf,1}$. This relationship is shown in Equation (6.2).

$$\frac{p_{i,1} - p_{wf,1}}{q_1} = \frac{p_{i,2} - p_{wf,2}}{q_2} \qquad (6.2)$$

Equation (6.2) can be used to relate the pressure changes and flow rate changes of two consecutive transients as shown in Equation (6.3) and Equation (6.4).

$$\frac{p_{i,k} - p_{wf,k}}{q_k} \cong \frac{p_{i,k+1} - p_{wf,k+1}}{q_{k+1}} \tag{6.3}$$

$$\frac{p_{i,k-1} - p_{wf,k-1}}{q_{k-1}} \cong \frac{p_{i,k} - p_{wf,k}}{q_k} \tag{6.4}$$

This relation can be used to estimate the unknown flow rate of one transient if the flow rate of an adjacent transient is known. This procedure works well when there is no noise in the pressure transient data but it is normal for field data to have noise that cannot be filtered out completely. Choosing a single value of elapsed time t to compare the pressure values of two transients is not robust for noisy field data as the chosen time may be at a noisy section of pressure data. An alternative approach is to take more data points to compare pressure values. The ratios obtained are then averaged. If more points are taken, in the limit when all the pressure data points are taken, the ratio is between the area of the first transient and the area of the second transient.

6.2. Area of Transient to Estimate Unknown Flow Rate

A better way to relate the flow rates and pressure responses of two adjacent transients is by comparing the area between the pressure transient curve and the line of initial pressure at the beginning of the transient for the same time windows. A decrease in flow rate will result in a build-up pressure transient curve while an increase in flow rate will result in draw-down pressure transient curve. The area for a pressure build-up curve will be positive as pressure increases above the initial pressure at the beginning of the transient. The area for a pressure draw-down curve will be negative as pressure decreases below the initial pressure at the beginning of the transient.

Figure 6.1 shows how flow rate changes causes draw-down and build-up transients and how the areas for the two types of transients are defined.

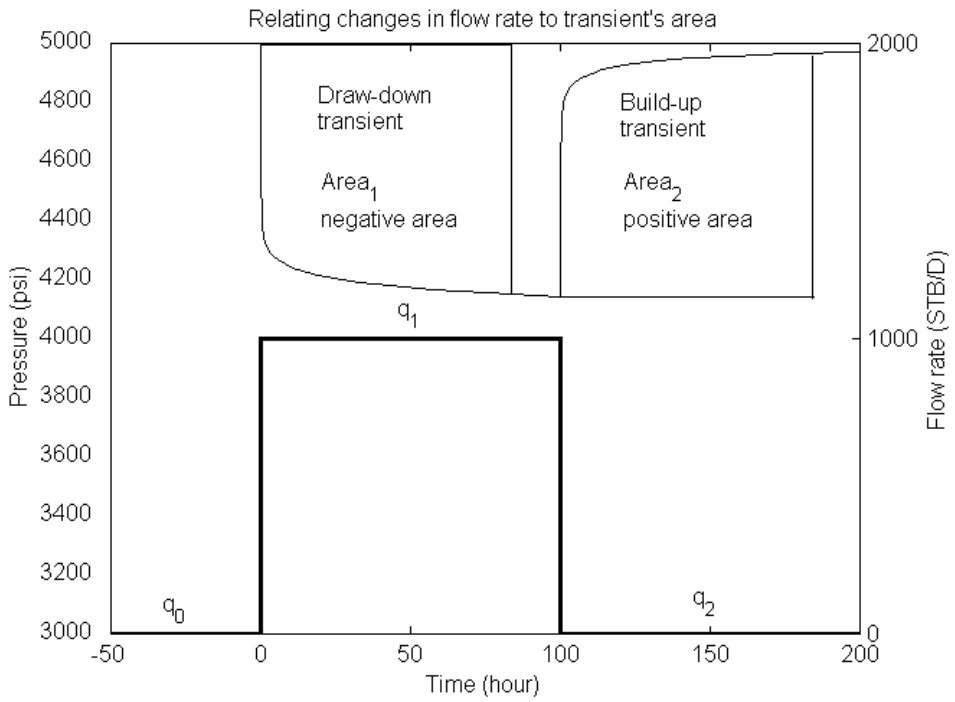


Figure 6.1: Flow rate changes and draw-down and build-up transient areas.

Equation (6.5) relates the areas for the transients and the flow rates. The relation of areas of transients to their respective flow rates can be generalized and given by Equation (6.6).

$$\frac{q_1 - q_0}{q_2 - q_1} \cong \frac{Area_1}{Area_2} \tag{6.5}$$

$$\frac{q_i - q_{i-1}}{q_{i+1} - q_i} \cong \frac{Area_i}{Area_{i+1}} \tag{6.6}$$

6.2.1. Area of a Transient

There will be $n+1$ sample points (including the initial break point found by the least square line method) in a transient for a certain window of data with the beginning of the transient at the break point. In order to estimate the area of a transient, the transient can be approximated by joining data points on the curve with straight lines.

Figure 6.2 shows how a build-up pressure transient is approximated and the area of the transient is evaluated by summing the areas of the triangle and trapezoids. Figure 6.3 shows a draw-down pressure transient. The area of a transient is given by Equation (6.7) where t_i and p_i are the initial time and initial pressure (the break point time and pressure) of the transient.

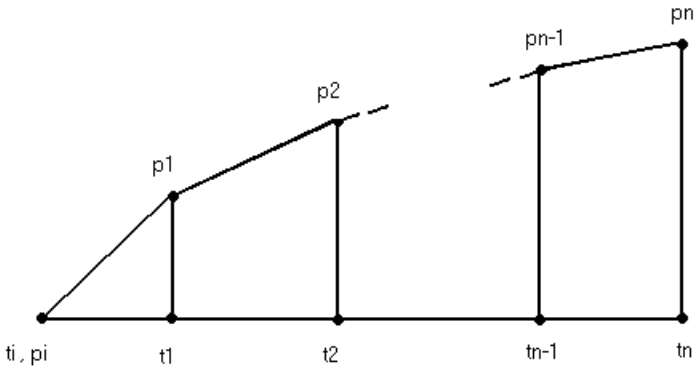


Figure 6.2: Build-up pressure transient approximated by straight lines.

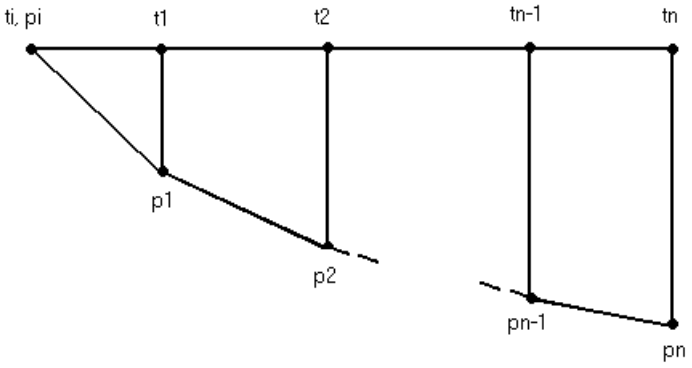


Figure 6.3: Draw-down pressure transient approximated by straight lines.

$$Transient\ area = Triangle\ area + Trapezoid\ areas$$

$$= \frac{1}{2}(p_1 - p_i)(t_1 - t_i) + \left(\frac{(p_1 - p_i) + (p_2 - p_i)}{2}\right)(t_2 - t_1) + \ldots + \left(\frac{(p_{n-1} - p_i) + (p_n - p_i)}{2}\right)(t_n - t_{n-1})$$

$$= \frac{1}{2}(p_1 - p_i)(t_1 - t_i) + \sum_{k=2}^{n}\left(\frac{(p_{k-1} - p_i) + (p_k - p_i)}{2}\right)(t_k - t_{k-1}) \qquad (6.7)$$

6.2.2. Solving the Equations

The initial estimates of the unknown flow rates can found by solving a system of simultaneous equations. For three consecutive rates indexed *0, 1* and *2*, there are eight possible combinations for the three rates from all unknown to all known. Suppose *U* denotes unknown and *K* denotes known, the eight combinations are *UUU, KUU, UKU, UUK, KKU, KUK, UKK* and *KKK*. Denoting $Q_0, Q_1\ or\ Q_2$ for known flow rates and $q_0, q_1\ or\ q_2$ for unknown flow rates, equations relating the known and unknown rates based on Equation (6.5) can be derived for the first seven combinations except the eighth combination of *KKK* where all the flow rates known. When all the three flow rates are known, no equation is needed. The equation relating the flow rates for each of the first seven combinations are given by Equations (6.8) to (6.14) where A_1 is area for transient *1* and A_2 is area for transient *2*.

$$\textbf{\textit{UUU}}: \qquad q_0 - \left(1 + \frac{A_1}{A_2}\right)q_1 + \frac{A_1}{A_2}q_2 = 0 \qquad \textbf{(6.8)}$$

$$\textbf{\textit{KUU}}: \qquad -\left(1 + \frac{A_1}{A_2}\right)q_1 + \frac{A_1}{A_2}q_2 = -Q_0 \qquad \textbf{(6.9)}$$

$$\textbf{\textit{UKU}}: \qquad q_0 + \frac{A_1}{A_2}q_2 = \left(1 + \frac{A_1}{A_2}\right)Q_1 \qquad \textbf{(6.10)}$$

$$\textbf{\textit{UUK}}: \qquad q_0 - \left(1 + \frac{A_1}{A_2}\right)q_1 = -\frac{A_1}{A_2}Q_2 \qquad \textbf{(6.11)}$$

$$KKU: \quad +\frac{A_1}{A_2}q_2 = -Q_0 + \left(1+\frac{A_1}{A_2}\right)Q_1 \qquad (6.12)$$

$$KUK: \quad -\left(1+\frac{A_1}{A_2}\right)q_1 = -Q_0 - \frac{A_1}{A_2}Q_2 \qquad (6.13)$$

$$UKK: \quad q_0 = \left(1+\frac{A_1}{A_2}\right)Q_1 - \frac{A_1}{A_2}Q_2 \qquad (6.14)$$

The first equation of the series of equations is found by deriving the expression that relates a group of three adjacent known and unknown flow rates. The next group of three flow rates is selected by dropping one flow rate at the left and adding one flow rate at the right. No equation will be derived for groups with three known flow rates. The last group of flow rates consists of the flow rates for the last three transients. The resulting system of equations can be over-determined. An over-determined system of equations can be handled by solving a least square problem as in Equation (3.3).

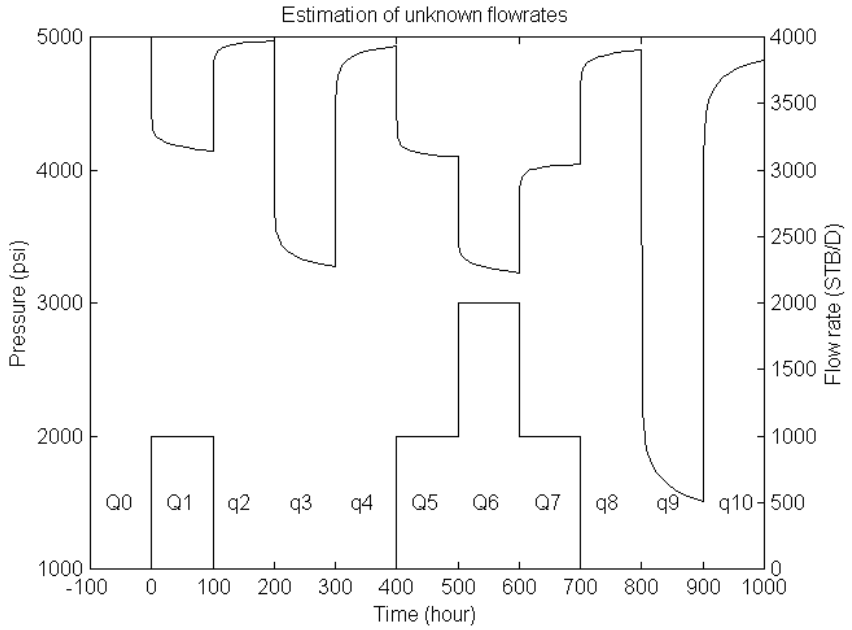


Figure 6.4: Estimation of unknown flow rates from known flow rates.

There are eight equations that can be derived for the example in Figure 6.4 where there are five known flow rates and six unknown flow rates. The equations relating the flow rates and the areas of the transients are as follows:

$$Q_0, Q_1, q_2: \qquad \frac{A_1}{A_2} q_2 = -Q_0 + \left(1 + \frac{A_1}{A_2}\right) Q_2 \qquad (6.15)$$

$$Q_1, q_2, q_3: \qquad -\left(1 + \frac{A_2}{A_3}\right) q_2 + \frac{A_2}{A_3} q_3 = -Q_1 \qquad (6.16)$$

$$q_2, q_3, q_4: \qquad q_2 - \left(1 + \frac{A_3}{A_4}\right) q_3 - \frac{A_3}{A_4} q_4 = 0 \qquad (6.17)$$

$$q_3, q_4, Q_5: \qquad q_3 - \left(1 + \frac{A_4}{A_5}\right) q_4 = -\frac{A_4}{A_5} Q_5 \qquad (6.18)$$

$$q_4, Q_5, Q_6: \qquad q_4 = \left(1 + \frac{A_5}{A_6}\right) Q_5 - \frac{A_5}{A_6} Q_6 \qquad (6.19)$$

$$Q_6, Q_7, q_8: \qquad \frac{A_7}{A_8} q_8 = -Q_6 + \left(1 + \frac{A_7}{A_8}\right) Q_7 \qquad (6.20)$$

$$Q_7, q_8, q_9: \qquad -\left(1 + \frac{A_8}{A_9}\right) q_8 + \frac{A_8}{A_9} q_9 = Q_7 \qquad (6.21)$$

$$q_8, q_9, q_{10}: \qquad q_8 - \left(1 + \frac{A_9}{A_{10}}\right) q_9 + q_{10} = 0 \qquad (6.22)$$

The system of equations from Equations (6.15) to (6.22) can be represented in matrix form as Equation (6.23) where M_{8x6}, q_{6x1} and b_{8x1} are the matrices and the solution is $q_{6x1} = [q_2\ q_3\ q_4\ q_8\ q_9\ q_{10}]^T$.

$$M_{8x6} q_{6x1} = b_{8x1}, \qquad q_{6x1} = inv\left(M_{6X8}^T\ M_{8x6}\right)\left(M_{6x8}^T b_{8x1}\right) \qquad (6.23)$$

One other useful piece of information besides the known flow rates is whether there is any fluid injection into the well. A producer well is normally not injected with fluid. If the well is a producer and it is known that fluid is not injected, the flow rate cannot be negative. If the initial unknown flow rate estimate is negative, these negative flow rates will be changed to zero and set as known by the algorithm. Nonzero unknown flow rates are estimated a second time by the algorithm. Table 6.1 summarizes the estimated unknown flow rates for the pressure transient example in Figure 6.4.

Time (hour)	Actual flow rate (STB/D)	Known flow rate (STB/D) (input to flow rate estimate algorithm)	Estimate of unknown flow rate (STB/D)	Estimate of unknown flow rate (STB/D) (no negative flow rate)
0	0	0	-	-
100	1000	1000	-	-
200	0	-	9.94	9.94
300	2000	-	1982.93	1982.93
400	0	-	16.60	16.60
500	1000	1000	-	-
600	2000	2000	-	-
700	1000	1000	-	-
800	0	-	-17.11	0
900	4000	-	4046.25	3995.00
1000	0	-	16.89	33.43
Total flow (STB)	45,833.33		46,064.58	45,991.25

Table 6.1: Initial estimate of unknown flow rates for transient in Figure (6.4).

It was found that the initial estimates of unknown flow rates are close to the actual flow rate values. Setting negative flow rates to zero improves the estimation of flow rate at 900 hour from 4046.25 STB/D to 3995.00 STB/D but degrades the estimation of flow rate at 1000 hour from 16.89 STB/D to 33.43 STB/D.

The effect of setting negative estimated flow rates to zero might improve some estimates while degrading others. The total flow estimated is also very close to the actual value. The total flow estimated when negative estimated flow rates were set to zero is better then the total flow rate estimated when negative estimated flow rates are not changed.

A minimum of two known flow rates (at least one to be nonzero) are needed by the initial rate estimation algorithm. Table 6.2 shows the estimated unknown flow rates when only two flow rates are known. Table 6.3 shows flow rate estimates when three flow rates are known.

Time (hour)	Actual flow rate (STB/D)	Known flow rate (STB/D) (input to flow rate estimate algorithm)	Estimate of unknown flow rate (STB/D)	Estimate of unknown flow rate (STB/D) (no negative flow rate)
0	0	0	-	-
100	1000	-	984.31	984.31
200	0	-	12.95	12.95
300	2000	-	1979.31	1979.31
400	0	-	30.54	30.54
500	1000	-	1008.89	1008.89
600	2000	2000	-	-
700	1000	-	1033.72	1033.72
800	0	-	50.92	50.92
900	4000	-	3977.24	3977.24
1000	0	-	83.77	83.77
Total flow (STB)	45,833.33		46,506.88	46,506.88

Table 6.2: Initial estimate of unknown flow rates when only two flow rates are known.

Time (hour)	Actual flow rate (STB/D)	Known flow rate (STB/D) (input to flow rate estimate algorithm)	Estimate of unknown flow rate (STB/D)	Estimate of unknown flow rate (STB/D) (no negative flow rate)
0	0	0	-	-
100	1000	-	974.43	977.59
200	0	-	3.51	6.53
300	2000	-	1969.22	1972.44
400	0	-	11.69	17.71
500	1000	-	994.52	999.11
600	2000	2000	-	-
700	1000	-	1001.16	1011.57
800	0	-	-24.60	0
900	4000	-	4072.03	4036.63
1000	0	0	-	-
Total flow (STB)	45,833.33		45,841.50	45,923.25

Table 6.3: Initial estimate of unknown flow rates when three flow rates are known.

Time (hour)	Actual flow rate (STB/D)	Known flow rate (STB/D) (input to flow rate estimate algorithm)	Estimate of unknown flow rate (STB/D)	Estimate of unknown flow rate (STB/D) (no negative flow rate)
0	0	0	-	-
100	1000	-	975.44	975.44
200	0	0	-	-
300	2000	-	1974.95	1974.95
400	0	0	-	-
500	1000	-	991.62	999.62
600	2000	2000	-	-
700	1000	-	1010.03	1010.03
800	0	0	-	-
900	4000	-	4030.48	4030.48
1000	0	0	-	-
Total flow (STB)	45,833.33		45,760.50	45,760.50

Table 6.4: Initial estimate of unknown flow rates when one nonzero flow rate and all zero flow rates are known.

These examples show that the algorithm could estimate the unknown flow rates quite well even when only two known flow rates were available. This procedure assumes the reservoir properties such as permeability and skin factor do not change. Even though this procedure needs only two known flow rates, the larger the number of flow rates that are known, the better will be the estimate of the unknown flow rates. Known zero flow rates provide valuable information for the algorithm in estimating unknown flow rates so the time when a well was shut-in should be recorded. The total flow estimated is also very close to the actual total flow. This method could also be used to estimate the production from a well when flow rate data are not available for the whole duration of flow.

Chapter 7

7. *Window* Algorithm Investigation

7.1. Moving Window Analysis Requirement

The moving window analysis approach was proposed by Athichanagorn (1999) to interpret permanent downhole gauge pressure transient data where reservoir properties or environment may change over time and also when flow rate data are not available for the whole duration of available pressure data. Sections of data are analyzed locally to determine local values of reservoir properties.

The pressure transient data required by the *Window* algorithm should be denoised and both spike and step outliers removed by the *Wavelet* algorithm. A reduced data set sampled at a lower sampling rate could be used. Only adjusted true break points should be provided in addition to good initial estimates of unknown flow rates. If available, cumulative flow if available is used to constrain the regression match in addition to the known flow rates. Correct fluid properties should be available as input parameters to the *Window* algorithm. Abnormal transients are removed by behavioral filtering during *Window* processing.

The reservoir model used to interpret the pressure transient data in the *Window* algorithm has to be identified using conventional well test interpretation method and geological information. Good initial estimates of reservoir parameters for the reservoir model selected is essential for the *Window* algorithm to converge to the correct estimates.

7.2. Effect of Window Step and Window Width

Figure 7.1 shows a field pressure data set from a permanent downhole gauge. The pressure data was processed using the moving window algorithm using a 399 hour window width and 99 hour window step. The estimates of reservoir permeability as a function of time are shown in Figure 7.2. The actual permeability of a reservoir is not likely to fluctuate as in Figure 7.2.

The effect of using different window step size in the *Window* algorithm was investigated in this research. It was found that varying the window step does not improve the fluctuation of the estimated permeability. This is shown in Figure 7.3 where all the three sets of estimated permeability were estimated using a 399 hour window with width window steps of 99, 199 and 299 hours were used. The effect of window size on the fluctuation of the estimated reservoir parameters was also looked into. It was found that a wider window width gave smoother variation of estimated reservoir parameters as illustrated in Figure 7.4.

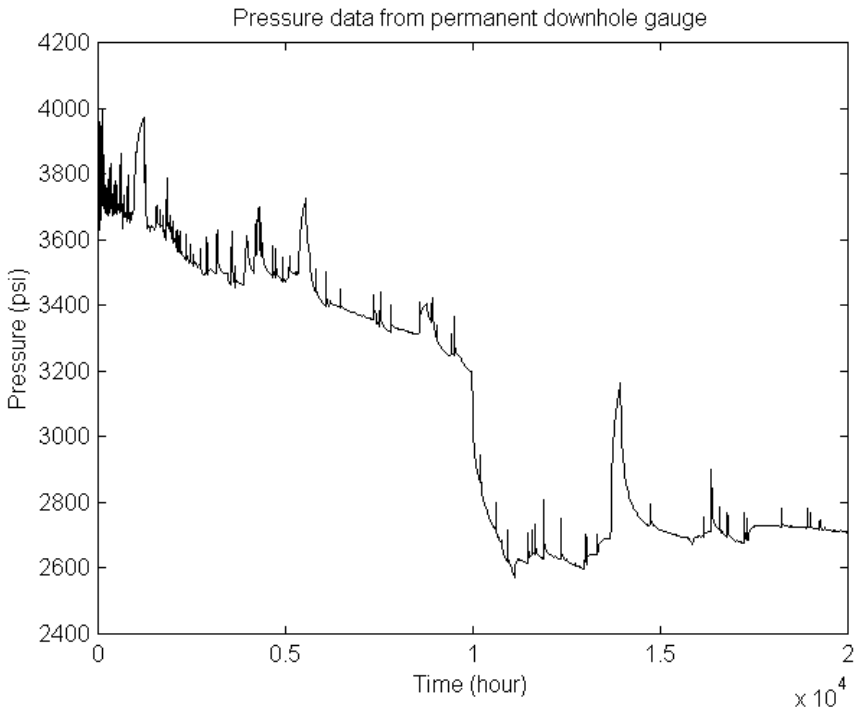


Figure 7.1: Field permanent downhole gauge data.

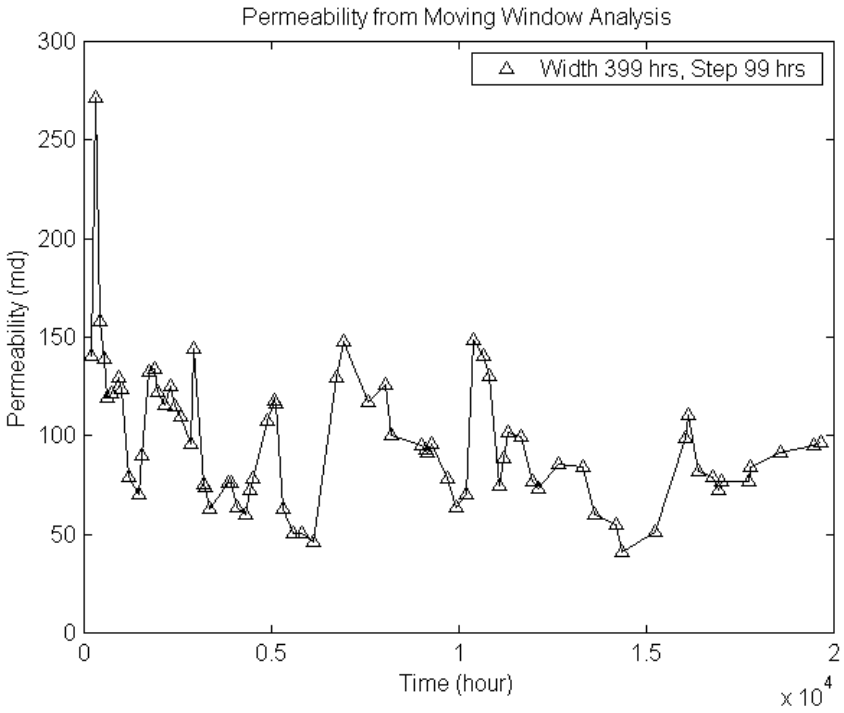


Figure 7.2: Permeability values estimated from pressure transient in Figure 7.1.

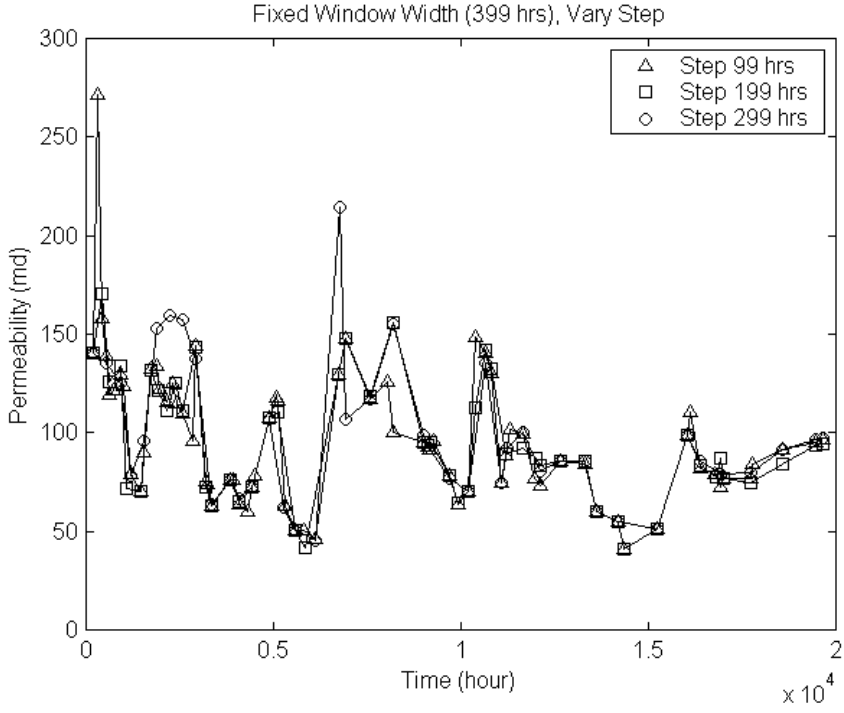


Figure 7.3: Varying window step does not reduce fluctuations of estimated permeability.

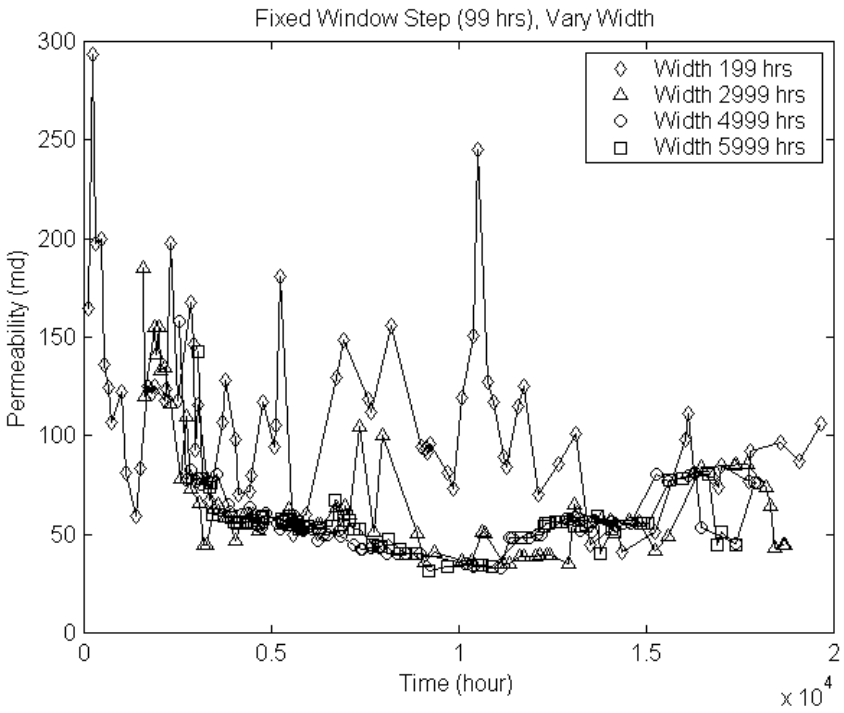


Figure 7.4: Increasing the window width smoothes the estimated reservoir properties.

Figure 7.5 shows the estimated permeability values using a window width of 5999 hour and a window step of 199 hour. The estimated permeability using the wide window width is much smoother. The trade off of using large window width is that parameters at the beginning and at the end of the transient could not be estimated. In this case the permeability does not appear to be varying much with time.

It was also found that the estimated unknown flow rates using two wide window widths of 4999 hour and 5999 hour were almost the same. The estimated flow rates for both window widths are plotted onto the same diagram in Figure 7.6. The triangles (window width of 4999 hour) and the circles (window width of 5999 hour) overlay one another for most of the flow rates.

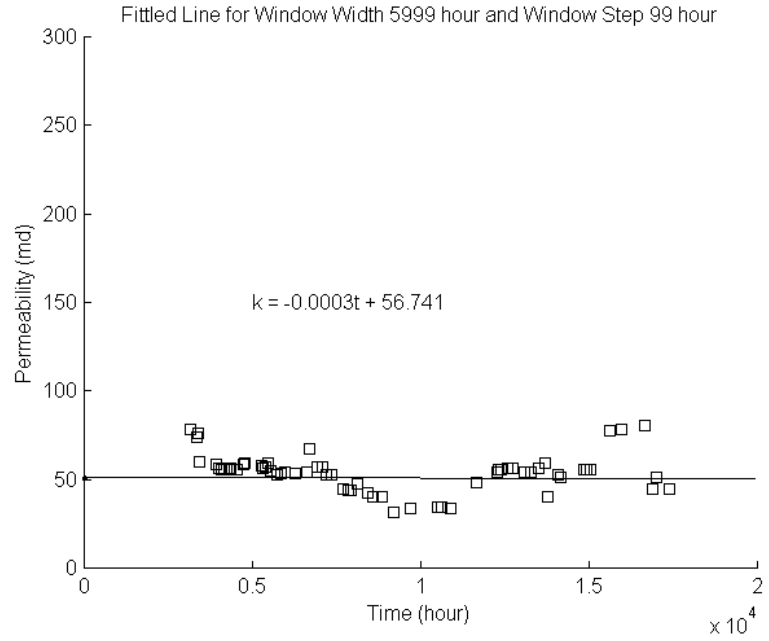


Figure 7.5: Straight line fitted to estimate permeability.

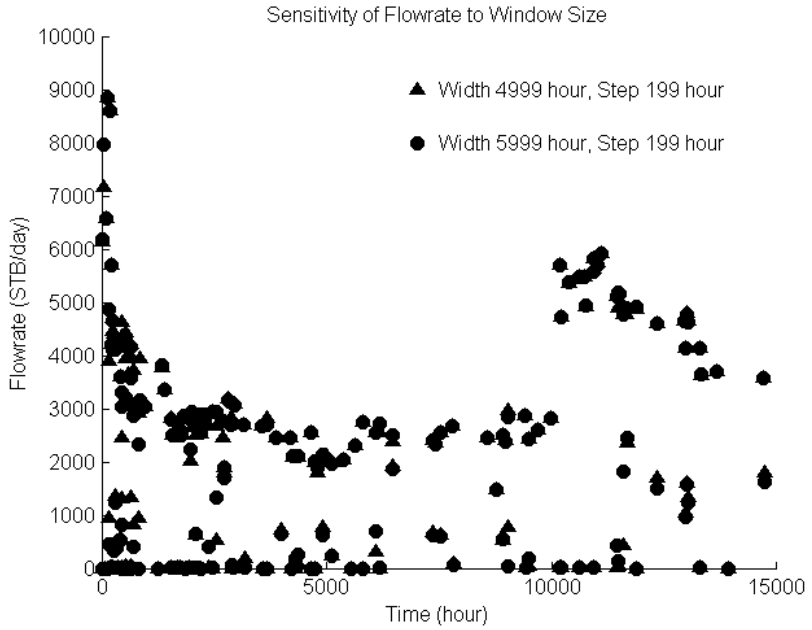


Figure 7.6: Estimated unknown flow rates are almost the same for two wide window widths of 4999 hour and 5999 hour.

7.3. Comparison of Estimated Flow Rate to Actual Flow Rate

In this research, the estimated flow rates from the window algorithm for two simulated pressure transients were compared to the actual flow rates.

Figure 7.7 shows the first simulated pressure transient data with the actual flow rates used to simulate the pressure data. Table 7.1 tabulates the actual flow rates together with the initial estimates of unknown flow rates and the estimated flow rates from the *Window* algorithm.

Figure 7.8 shows the first simulated pressure transient data with the actual flow rates used to simulate the pressure data. Table 7.2 tabulates the actual flow rates together with the initial estimates of unknown flow rates and the estimated flow rates from the *Window* algorithm.

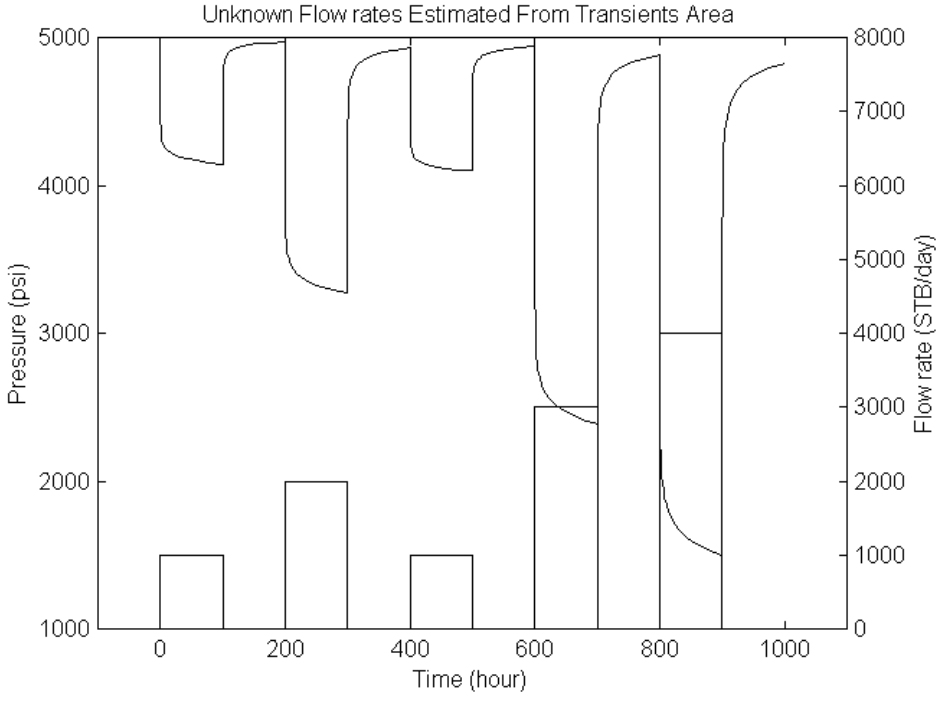


Figure 7.7: First simulated pressure transient with its actual flow rates.

Time (hour)	Actual flow rate (STB/D)	Known flow rate (STB/D) (input to flow rate estimate algorithm)	Estimate of unknown flow rate (STB/D) (no negative flow rate)	*Window* algorithm estimate of unknown flow rate (STB/D)
0	0	0	-	-
100	1000	-	983.68	999.99
200	0	-	0	0.05
300	2000	-	1991.63	1999.95
400	0	-	0	-0.03
500	1000	1000	-	-
600	0	-	0	-0.02
700	3000	-	3008.74	3000.03
800	0	-	0	-0.03
900	4000	-	4046.97	3999.97
1000	0	0	-	-
Total flow (STB)	45,833.33		45,962.58	45,832.96

Table 7.1: Actual and estimated flow rates for the first simulated pressure transient.

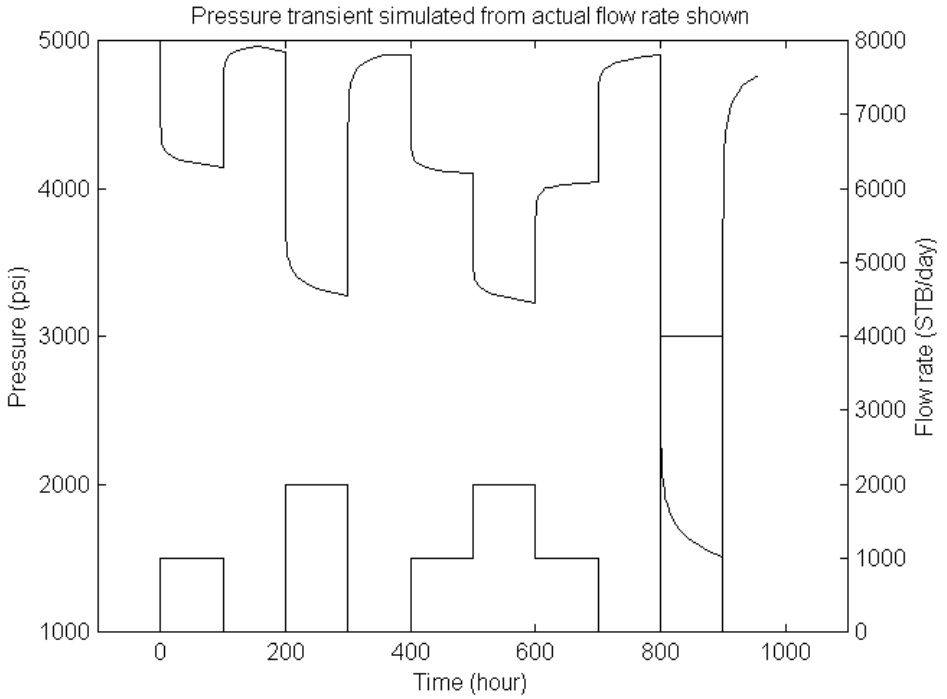


Figure 7.8: Second simulated pressure transient with its actual flow rates.

Time (hour)	Actual flow rate (STB/D)	Known flow rate (STB/D) (input to flow rate estimate algorithm)	Estimate of unknown flow rate (STB/D) (no negative flow rate)	*Window* algorithm estimate of unknown flow rate (STB/D)
0	0	0	-	-
100	1000	-	977.59	999.12
200	0	-	6.53	-2.18
300	2000	-	1972.44	1999.76
400	0	-	17.71	5.07
500	1000	-	999.11	1003.13
600	2000	2000	-	-
700	1000	-	1011.57	999.53
800	0	-	0	-0.05
900	4000	-	4036.63	3999.01
1000	0	0	-	-
Total flow (STB)	45,833.33		45,923.25	45,847.46

Table 7.2: Actual and estimated flow rates for the second simulated pressure transient.

Chapter 8

8. Interpretation Example

8.1. Programs and Input Files Used in Interpretation

The programs and the associated input data files used to interpret permanent downhole gauge data will be discussed here briefly. Two additional processing steps, pre-wavelet and post-*Wavelet* processing were added to complement Athichanagorn's *Wavelet* and *Window* processing algorithms.

Phase 1: Pre-*Wavelet* processing

Program Files: Overlap.m - Matlab program to remove data overlap.

Destep.m - Matlab program to remove step outlier.

Noise.m - Matlab program to estimate noise level.

Input File: Filename.tpr - Time and pressure data.

Output File: Filename.tpr - Overlap and step outlier removed pressure.

Phase 2: *Wavelet* processing

Program File: Wavelet.exe - C executable file from Athichanagorn (1999)

Input Files: P3.1 - Spline wavelet filter input file.

P3.2 - Spline wavelet filter input file.

Filename.tpr - After removing overlap and step outlier.

Output Files: Filename.dat - Data denoised and resampled at lower frequency

Filename#.break - Break time detected file.

# Denotes file number as each file includes break points detected for a combination of wavelet sample spacing and slope detection threshold.

Phase 3: Post-*Wavelet* processing

Program File: Preinrate.m - Matlab program to screen and adjust break points.

Input Files: Filename.dat - *Wavelet* processing output file.

Filename#.break - *Wavelet* processing break point output files.

Output Files: Filename.postbrk - File of screened break points.

Filename.preinrate - Initial rate estimates with known rates included.

Program File: Inrate.exe - C executable file that estimates unknown flow rates

Input Files: Filename.dat - *Wavelet* processing output file.

Filename.postbrk - Output file from Preinrate.m.

Filename.preinrate - Output file from Preinrate.m.

Output File: Filename.inrate - Inrate.exe output file for processing by *Window*.

Phase 4: *Window* processing

Program File: Window.exe - Fortran executable file from Athichanagorn (1999).

Input Files: Filename.dat - Time and pressure from *Wavelet* program.

Filename.inrate - Break time, known and initial rate estimates.

Filename.est - Initial estimates of unknown reservoir parameters.

Filename.prop - Known reservoir and fluid properties.

- *Window* step and *Window* width processing setting.

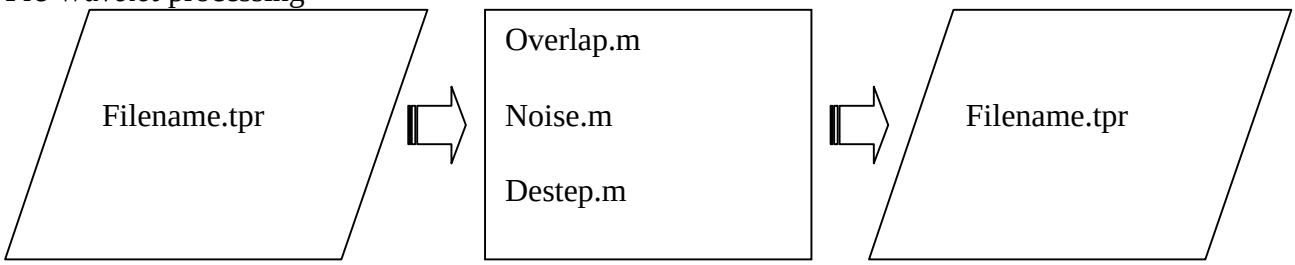
Filename. prod - Total production data (empty file if not available).

Output Files: Filename.filpara - Estimate of reservoir parameters.

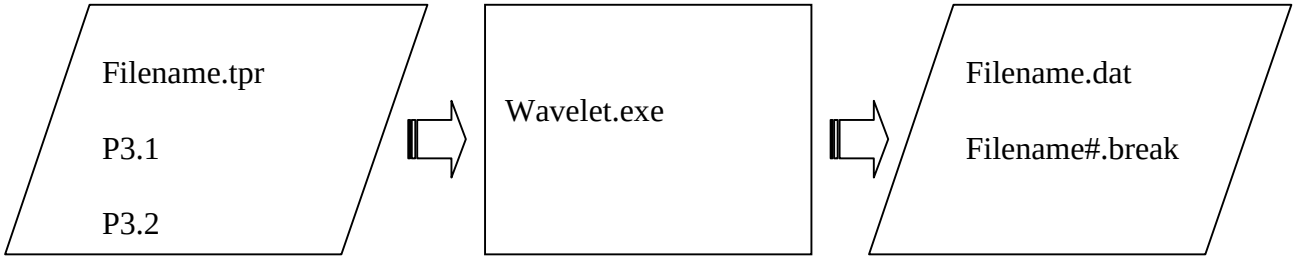
Filename.filrate - Estimate of flow rates.

Filename.outrate - Summary of flow rates estimates.

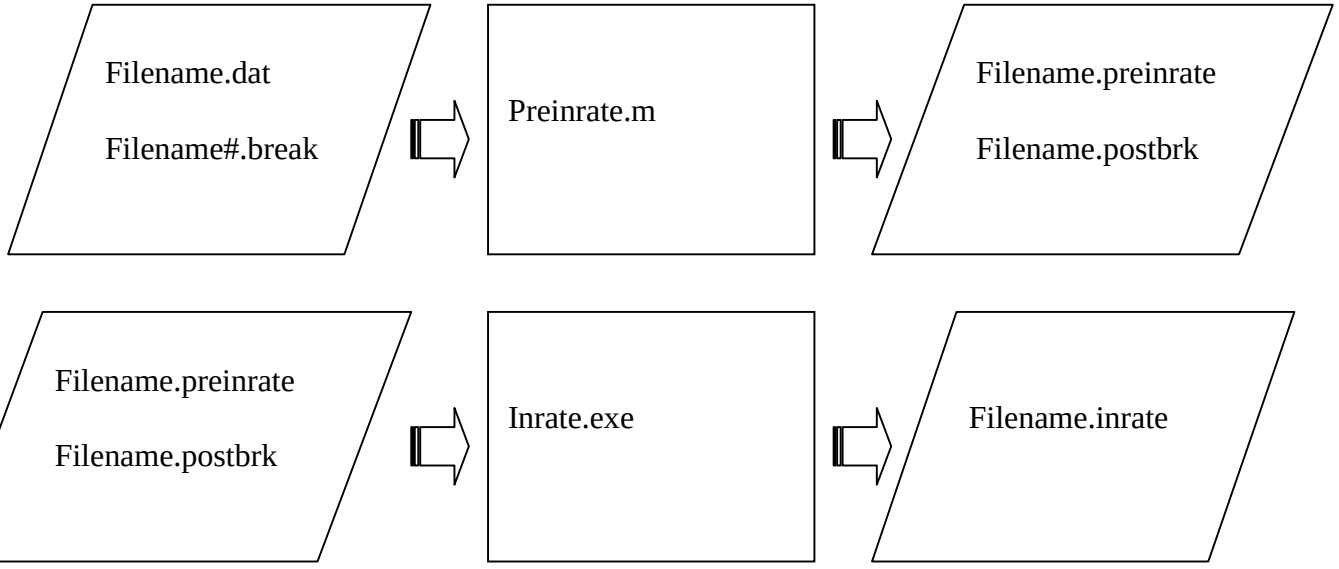
Pre-*Wavelet* processing



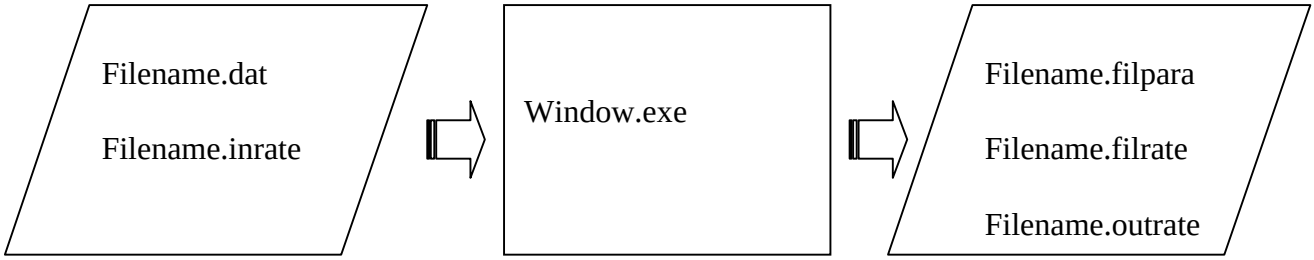
Wavelet processing



Post-*Wavelet* processing



Window processing



8.2. Case 1

In this case, a simulated set of pressure transient data was used to test the *Wavelet* and *Window* processing algorithms. Simulated pressure data were used in the first case so that estimated reservoir parameters could be compared with the actual reservoir parameters used to generate the data. The structure of the input output of the files will also be described for this case as a reference.

The pressure data before denoising, outlier removal and resampling are included in file filename.tpr. The data overlap and step outliers need to be removed and noise level estimated from filename.tpr before processing the data using the *Wavelet* algorithm. The *Wavelet* algorithm computes and writes the file filename.dat that is denoised, with spike outliers removed and the data resampled at a lower rate. The *Wavelet* algorithm uses two wavelet files P3.1 and P3.2, which are the wavelet coefficients used in the wavelet transform routine. filename.tpr and filename.dat are data files consisting of two columns, the first column is time and the second column is pressure.

As output from the *Wavelet* program, the break points are given in file filename.break when only one combination of wavelet sample spacing and slope detection threshold is selected. When more than one combination of wavelet sample spacing and slope detection threshold is selected in *Wavelet* processing, the break points for each combination are saved in different files named filename#.break (where # is 1,2,3, … number of combinations).

Program *Preinrate* generates files filename.postbrk and filename.preinrate, which include corrected break points after screening false breaks and with known flow rates given by the user. The *Inrate* algorithm uses files filename.postbrk, filename.preinrate and filename.dat and creates file filename.inrate that has the initial estimate of unknown flow rates. The *Window* algorithm uses filename.dat, filename.inrate, filename.prop, filename.est and filename.prop and provides output results in files filename.filpara, filename.filrate and filename.outrate.

In the Case 1 example here using simulated data, the *Wavelet* algorithm detected all the true break points with no false break points. The detected break points did not fall exactly at the beginning of their respective transients. Figure 8.1 shows the simulated pressure data and the break points detected. Listing 8.1, taken directly from the data file filename.break, shows the break points detected by the *Wavelet* algorithm. The screened and adjusted break times are listed in Listing 8.2 and the input file with known flow rates is listed in Listing 8.3. The flow rate associated with a break time is the rate that flowed before that break time. In Listing 8.3 there are three known flow rates, 0, 1000 and 0. An additional break point had been added to this file at the end point of the last transient.

```
%
 0.004050 0
 0.004050 10000
%
 100.004053 0
 100.004053 10000
%
 200.007109 0
 200.007109 10000
%
 300.004053 0
 300.004053 10000
%
 400.007109 0
 400.007109 10000
%
 500.004053 0
 500.004053 10000
%
 600.004053 0
 600.004053 10000
%
 700.004053 0
 700.004053 10000
%
 800.007109 0
 800.007109 10000
%
 900.004053 0
 900.004053 10000
```

Listing 8.1: (Filename.break) Break times detected by the *Wavelet* algorithm.

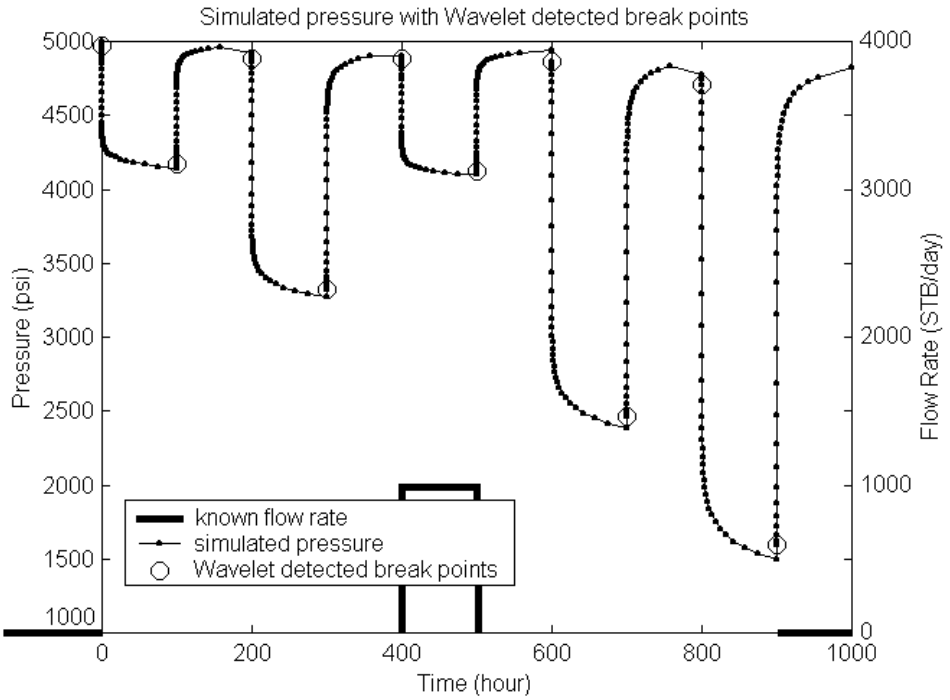


Figure 8.1: Detected break points are not exactly at the beginning of transients.

```
 0.000001        5000.00
 99.998915       4136.72
199.998222       4975.99
299.998967       3256.87
399.993879       4939.97
499.999135       4094.21
599.999056       4944.30
699.998960       2368.32
799.997487       4894.95
899.998999       1469.80
```

Listing 8.2: (Filename.postbrk) Screened and adjusted break times and pressures.

```
 0.000001      0.0000000    1
 99.998915   -999.000000    0
199.998222   -999.000000    0
299.998967   -999.000000    0
399.993879   -999.000000    0
499.999135   1000.000000    1
599.999056   -999.000000    0
699.998960   -999.000000    0
799.997487   -999.000000    0
899.998999   -999.000000    0
1000.00000      0.000000    1
```

Listing 8.3: (Filename.preinrate) Known rates are marked with 1 in the third column.

Figure 8.2 shows the adjusted break points with the initial estimate of unknown flow rates after running the Matlab program *Preinrate*. Listing 8.4 shows the initial estimates of unknown flow rates in column two. Listing 8.5 shows fluid, reservoir and completion parameters and the window width and window step processing parameters. Listing 8.6 shows the initial estimates of unknown reservoir parameters for an infinite-acting reservoir model. The listing for filename.prod is not included as there is no cumulative production data and filename.prod is just an empty file.

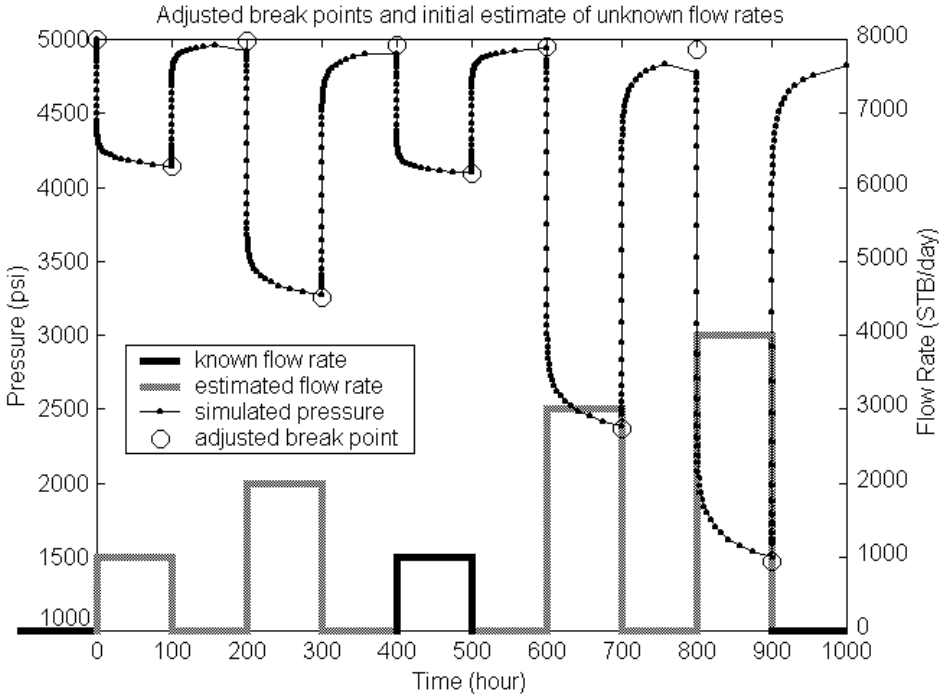


Figure 8.2: Adjusted break points and initial estimates of unknown flow rates.

1e-006	0	1	0	0
99.99891542	959.1519265	0	204.9282409	1713.375612
199.99822	0	0	-610.3508908	610.3508908
299.9989685	1946.24501	0	1043.957356	2848.532664
399.9938827	0	0	-610.3508908	610.3508908
499.9991393	1000	1	849.999994	1150.000006
599.999051	0	0	-610.3508908	610.3508908
699.9989573	3011.552073	0	1949.468353	4073.635792
799.9974644	0	0	-610.3508908	610.3508908
899.9989994	4069.005777	0	2848.303996	5289.707559
1000	0	1	0	0

Listing 8.4: (Filename.inrate) Initial estimates of unknown flow rates. (Data in columns: time, flow rates, known/unknown = 1/0, minimum, maximum).

```
porosity
 0.2
height
 10.00
well radius
 0.3
viscosity
 4.0
formation volume factor
 1.5
compressibility
 10.0e-6
window length (hours)
 249
translation length
 49
starting time
 0
```

Listing 8.5: (Filename.prop) Fluid, reservoir and completion properties and window width and window size, as input to program *Window*.

```
model
   1
initial pressure
   4500     3500.0     5500.0    0   (0 denotes not known, 1 denotes known)
permeability
   500.0    100   3000              (initial estimate     minimum     maximum)
skin
    5.0    -10    50
wellbore storage
    0.5   0.001    10.0
```

Listing 8.6: (Filename.est) Initial estimates of unknown reservoir properties, as input to program *Window*.

Listing 8.7 shows the estimated reservoir parameters from the *Window* algorithm, where the unknown parameters are as in filename.est in Listing 8.6. Although the window width specified is 249 hour and the window step specified is 49 hour, the actual window width used in the algorithm varies from 300 hour to 500 hour and the window step used by the *Window* algorithm is 100 hour.

The window width and step were adjusted during data processing by the *Window* algorithm. Listing 8.8 shows the estimated flow rates from the *Window* algorithm. Listing 8.9 shows the summary of estimated of unknown flow rates.

1	0.00	500.00	250.00	1002.45	1.0040	0.0102	4998.82	0.33	0.33
2	100.00	500.00	300.00	999.65	1.0084	0.0101	5000.83	0.36	0.36
3	200.00	500.00	350.00	929.43	1.0550	0.0094	5010.18	0.36	0.36
4	300.00	600.00	450.00	849.91	1.0881	0.0084	5154.70	0.34	0.34
5	399.99	700.00	550.00	934.30	1.0441	0.0094	5065.65	0.32	0.32
6	500.00	1000.00	750.00	1005.01	1.0081	0.0102	5000.09	0.45	0.45
7	600.00	1000.00	800.00	1017.42	1.0030	0.0103	4999.83	0.49	0.49
8	700.00	1000.00	850.00	1016.08	1.0619	0.0101	4999.80	0.07	0.07

Listing 8.7: (Filename.filpara) *Window* algorithm estimated reservoir parameters. (Data in columns: window number, begin of window, end of window, middle of window, permeability, skin, wellbore storage, initial reservoir pressure, error term 1, error term 2).

1	0.00	500.00	250.00	1000.69	-1.3249	2002.9914	-1.21
2	100.00	500.00	300.00	0.95	1998.9984	1.0456	
3	200.00	500.00	350.00	1865.33	9.9278		
4	300.00	600.00	450.00	147.32	142.2285		
5	399.99	700.00	550.00	65.89	2866.2189		
6	500.00	1000.00	750.00	-1.12	3012.0650	0.1276	4021.35
7	600.00	1000.00	800.00	3049.41	0.4032	4071.5579	
8	700.00	1000.00	850.00	-0.03	4065.5218		

Listing 8.8: (Filname.filrate) Estimated flow rates from *Window* algorithm. (Data in columns of a row: window number, begin of window, end of window, estimated flow rates)

0.0000	0.0000
99.9989	1000.6890
199.9982	0.9470
299.9990	1865.3293
399.9939	147.3203
499.9991	1000.0000
599.9991	-1.1179
699.9990	3049.4107

Listing 8.9: (Filename.outrate) Summary of *Window* estimated flow rates. (Data in columns: break times and estimated flow rates).

Some unknown flow rates are not listed by the *Window* algorithm in the file filename.outrate as in Listing 8.9. In Listing 8.9 flow rates for break times of 800 and 900 hour are not listed and have to be inferred from Listing 8.8 as –0.03 and 4065.5218 STB/day (actual flow rates are 0 and 4000 STB/day). Known flow rates are not listed in Listing 8.8. The flow rate for break time 1000 hour is known at 0 STB/day but is not listed in either Listing 8.8 or Listing 8.9. In Listing 8.8, the estimated flow rates for one window lies in a row. The first window has four estimated flow rates for break times in the first window (99.9989, 199.9982, 299.9990 and 399.9939 hour). filename.filrate and filename.outrate could be presented in a more convenient format because it is difficult to read the estimated flow rates from these two output files.

Figure 8.3 shows the estimated reservoir properties from the *Window* algorithm. The estimated values are very close to the actual values and this confirms that the *Window* algorithm can estimate reservoir properties well.

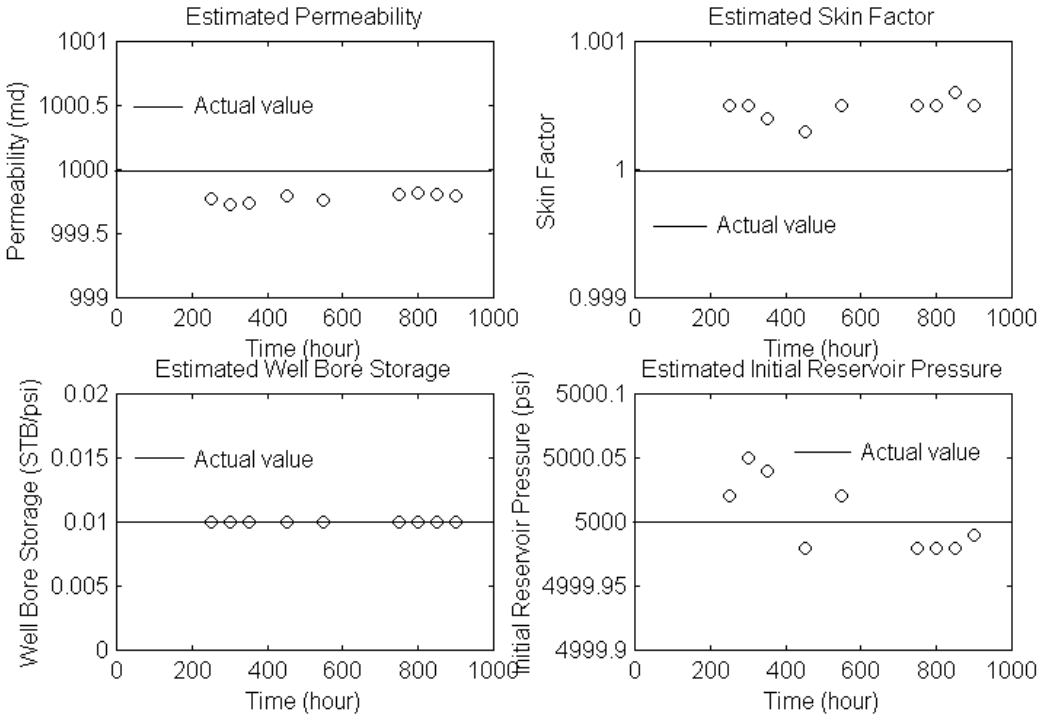


Figure 8.3: Estimated reservoir properties, Case 1.

8.3. Case 2

The denoised pressure data from Figure 3.14 were used in this study. The duration of this data is about 80 hours and was acquired at an operating oil field. Break points were detected using the *Wavelet* algorithm and false break points were screened using the histogram method as in Figure 4.19. In this case a relatively simple pressure transient was used to understand how to use the algorithms to process actual field data. The estimated reservoir parameters from the *Window* algorithm were compared to the interpretation result of one single transient using conventional well testing interpretation methods. Data points prior to the first break point had to be removed before processing using the *Window* algorithm.

8.3.1. Case 2 and Scenario 1 Known Flow Rates

As no flow rate data were available for this data set, two flow rates were assumed (one zero and one non-zero) and the other unknown flow rates were estimated using the area under the transient method discussed in Chapter 6. The known and estimated flow rates are shown Figure 8.4. These initial estimates of flow rate were also used as the flow rate history when interpreting the selected single transient using conventional well test methods. The result of the interpretation of this single transient is shown in Figure 8.5. Listing 8.10 shows the result of the interpretation of one transient using conventional well test interpretation methods.

Two processing runs were performed using the *Window* algorithm for Case 2 Scenario 1 known flow rates. In the first run, the initial reservoir pressure, p_i was set as known as in Listing 8.13 in the file filename.est. The initial reservoir pressure estimated from the first run highlighted in bold in Listing 8.14 was 2065.07 psia is far from the estimated value of 2986.7 psia from interpretation of one transient. The skin factor, permeability and distances from boundary were also quite different from the interpreted values from one transient. A second run was performed with the initial pressure set as known at 3000 psia to compare for any differences.

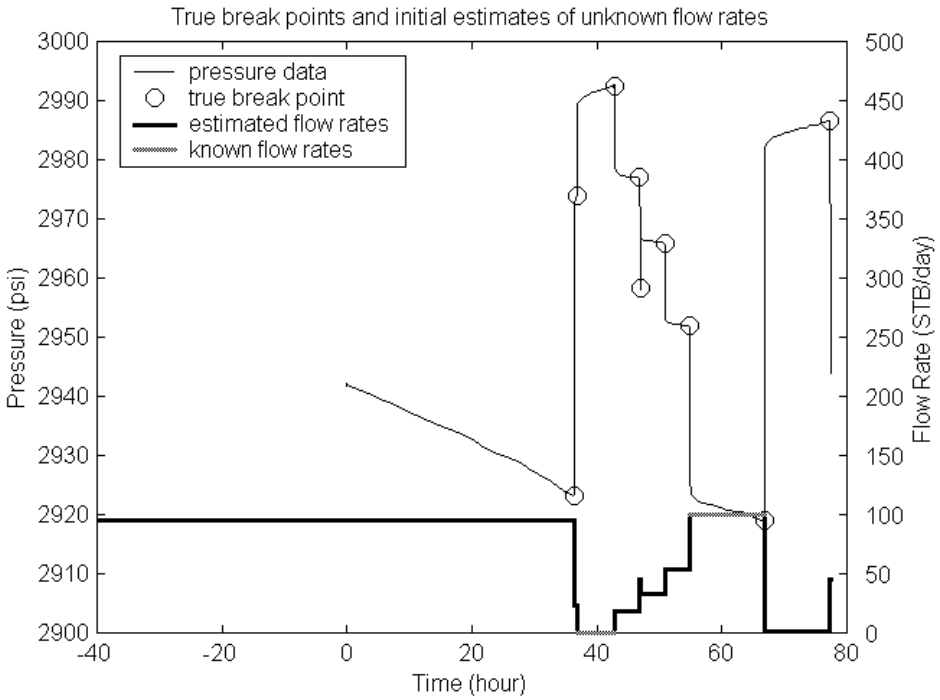


Figure 8.4: Case 2, Scenario 1 initial estimates of unknown flow rates.

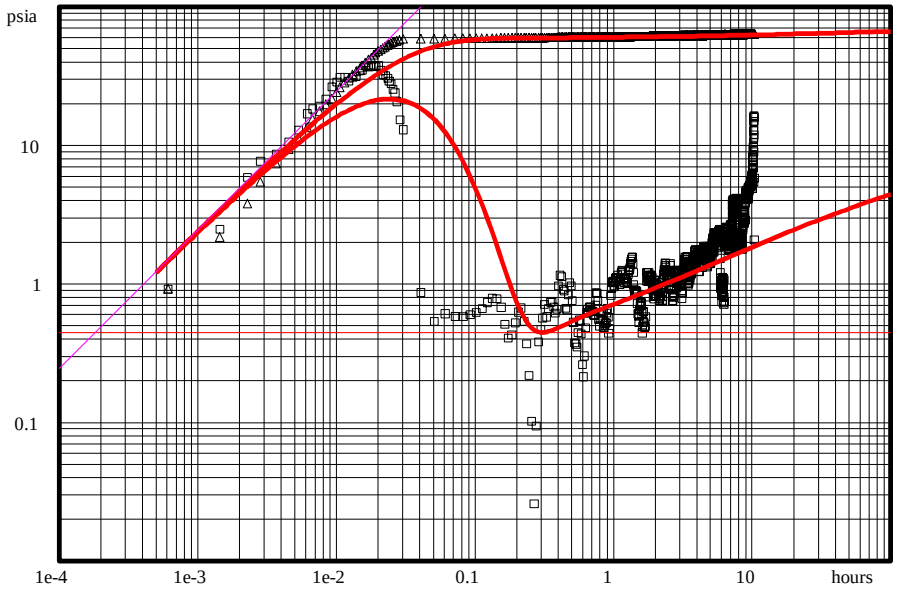


Figure 8.5: Interpretation of one transient using estimated flow rates history.

Interpretation Result		Fluid Properties	
Storage and skin reservoir model		Reservoir Fluid	Oil
Homogeneous reservoir		Viscosity	4 cp
Rectangular Boundary		Compressibility	10e-6/psi
Storage, STB/psi	2.47e-3	Formation Volume Factor	1.5 RB/STB
Skin	61.2	Porosity	0.2 frac
Permeability md	923	Well radius	0.3 ft
Init pressure, psia	3000	Formation thickness	100 ft
Boundary N	175 ft	Reservoir Temperature	200 degF
Boundary S	175 ft	Pressure Formation	2986.7 psia

Listing 8.10: Result of the interpretation from one transient.

```
36.39488046   94.74760686   0    75.27284588   114.2223678
36.75975613   23.35687828   0    11.02119027    35.6925663
42.85019762   0             1     0              0
46.87751772   18.56180972   0     6.705628568   30.41799086
46.99487182   45.03232234   0    30.52908989    59.53555479
50.85955882   32.51397529   0    19.26257757    45.76537302
54.81491249   53.94048985   0    38.54644063    69.33453906
66.79211969   100           1    89.99999985   110.0000001
77.30033512   0.6594481896  0    -9.406496779   10.72539316
77.57028 45.  47.59749      0   30.54283752    59.55235745
```

Listing 8.11: (Filename.inrate) Initial estimates of unknown flow rates, p_i unknown. (Data in columns: break time, initial estimate of flow rates, known/known flag, minimum flow rates and maximum flow rates)

```
porosity
 0.2
height
 100.00
well radius
 0.3
viscosity
 4.0
formation volume factor
 1.5
compressibility
 10.0e-6
window length (hours)
 49
translation length
 9
starting time
 0
```

Listing 8.12: (Filename.prop) Fluid, reservoir and completion properties, p_i unknown.

```
model
    9
initial pressure
   3000     0.0     5000.0  0
permeability
   1000.0    10   10000
skin
   50    -10    200
wellbore storage
   0.001   0.00001    1.0
distance to boundary
   200       50      1000
distance to boundary
   200      50      1000
```

Listing 8.13: (Filename.est) Initial estimates of unknown reservoir properties, p_i unknown.

```
 1       36.39       77.57       56.98     1181.78     84.8008      0.0028      983.28      983.27
2065.07       0.16       0.16
```

Listing 8.14: (Filename.filpara) Estimated reservoir parameters from *Window* algorithm, p_i unknown.

(Data in columns: window number, begin of window, end of window, middle of window, permeability, skin, wellbore storage, distance to boundary 1, distance to boundary 2, initial reservoir pressure, error term 1, error term 2).

```
 1       36.39       77.57       56.98       12.17      6.9783     37.6782       25.28       49.26
8.74       56.02
```

Listing 8.15: (Filename.filrate) Estimated flow rates from *Window* algorithm, p_i unknown. (Data in colums of a row: window number, begin of window, end of window, mid of window, rate for 36.76, 46.88, 47.99, 50.86, 54.81, 77.30 and 77.57 hour) Known flow rates are not listed in Filename.filrate.

```
   36.3949    94.7476
   36.7598    12.1664
   42.8502     0.0000
```

Listing 8.16: (Filename.outrate) Summary of *Window* estimated flow rates, p_i unknown. (Data in columns: break time and estimated flow rates).

In the second run when the initial reservoir pressure, p_i was set as known, the estimated reservoir parameters are much closer to the reservoir parameters found from interpreting one transient as in Figure 8.5. The results of the second run are shown in Listings 8.17, 8.18, 8.19 and 8.20.

```
model
    9
initial pressure
   3000     0.0     5000.0  1       (notice that the known/ unknown flag is set to 1)
permeability
   1000.0    10   10000
skin
   50    -10    200
wellbore storage
   0.001   0.00001    1.0
distance to boundary
   200       50      1000
distance to boundary
   200      50      1000
```

Listing 8.17: (Filename.est) Initial estimates of unknown reservoir properties, p_i known.

```
 1       36.39       77.57       56.98     1143.87     48.3201      0.0011      258.25      258.25
9.20       9.20
```

Listing 8.18: (Filename.filpara) Estimated reservoir parameters from *Window* algorithm, p_i known. (Data in columns: window number, begin of window, end of window, middle of window, permeability, skin, wellbore storage, distance to boundary 1, distance to boundary 2, initial reservoir pressure, error term 1, error term 2).

```
 1       36.39       77.57       56.98       23.43     18.1009     45.1003       33.13       56.56
-3.74      45.12
```

Listing 8.19: (Filename.filrate) Estimated flow rates from *Window* algorithm, p_i known.

```
   36.3949   94.7476
   36.7598   23.4347
   42.8502    0.0000
```

Listing 8.20: (Filename.outrate) Summary of *Window* estimated flow rates, p_i known.

8.3.2. Case 2 and Scenario 2 Known Flow Rates

The Case 2 set of data was interpreted assuming a second scenario where the known flow rates were different from the known flow rates from Scenario 1. The known and estimated flow rates are shown in Figure 8.6. The first run of the *Window* algorithm was performed using an unknown initial reservoir pressure while the second run was performed using a known initial reservoir pressure.

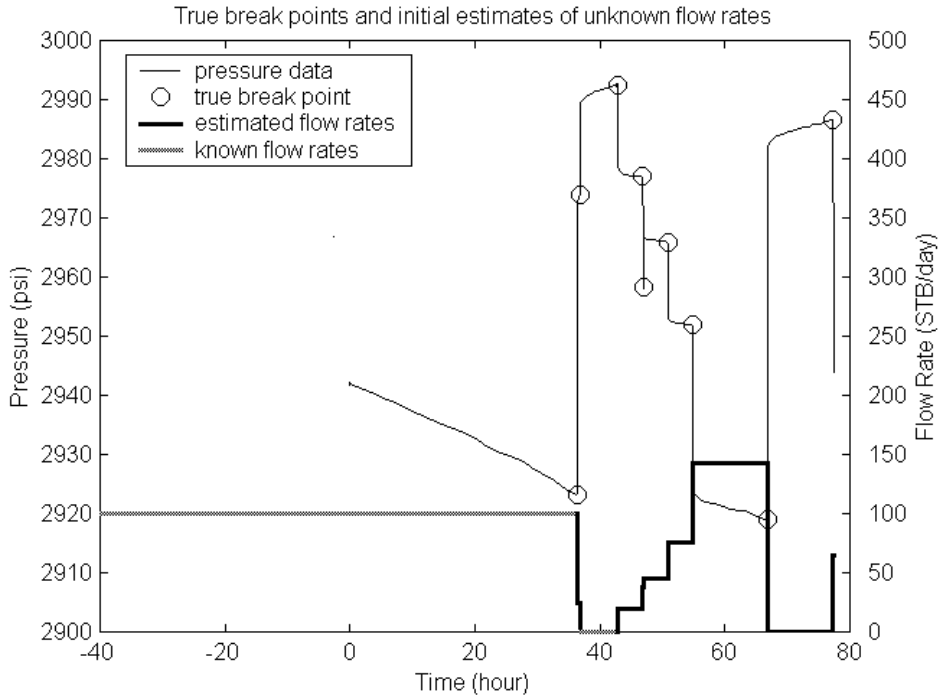


Figure 8.6: Case2, Scenario 2, initial estimates of unknown flow rates.

```
36.39494385   100          1   84.9999994     115.0000006
36.7597271    24.5352931   0   -0.5289195483   49.59950574
42.84734363   0            1   0                0
46.87758543   18.91572131  0   -5.30555553     43.13699816
46.89528      37.79338374  0   10.74045742     64.84631006
50.85954854   45.04988553  0   16.9084839      73.19128716
54.81488621   76.13225555  0   43.32849823    108.9360129
66.79216813   142.5594512  0   99.79161416    185.3272883
77.2999145    0            0  -21.38391853     21.38391853
77.57028      64.1227595   0   33.12042666     95.12509235
```

Listing 8.21: (Filename.inrate) Case 2, Scenario 2, initial estimates of unknown flow rates, p_i unknown. (Data in columns: time, flow rates, known/unknown = 1/0, minimum, maximum).

The initial estimate of unknown flow rates input file used for Case 2, Scenario 2 known flow rates with unknown reservoir pressure is as listed in Listing 8.21. The filename.prop and filename.est input files for this example is the same as the input files in Listing 8.12 and 8.13. The interpretation results are listed in Listings 8.24, 8.25 and 8.26.

```
 1      36.39      77.57      56.98     1073.43    70.6926     0.0025     147.70     147.70
2928.49      0.09      0.09
```

Listing 8.22: (Filename.filpara) Case 2, Scenario 2, estimated reservoir parameters from *Window* algorithm, p_i unknown. (Data in columns: window number, begin of window, end of window, middle of window, permeability, skin, wellbore storage, distance to boundary 1, distance to boundary 2, initial reservoir pressure, error term 1, error term 2).

```
 1      36.39      77.57      56.98      25.02    26.0255    62.2546      44.91      67.59
116.36      13.68      63.30
```

Listing 8.23: (Filename.filrate) Case 2, Scenario 2, estimated flow rates from *Window* algorithm, p_i unknown. (Data in colums of a row: window number, begin of window, end of window, mid of window, rate for 36.76, 46.88, 47.99, 50.86, 54.81, 77.30 and 77.57 hour) Known flow rates are not listed in Filename.filrate.

```
36.3949  100.0000
36.7597   25.0162
42.8473    0.0000
```

Listing 8.24: (Filename.outrate) Case 2, Scenario 2, summary of *Window* estimated flow rates, p_i unknown.

The estimated reservoir parameters when the initial reservoir pressure was set as unknown are shown in Listing 8.22. The estimated initial reservoir pressure in this case was 2828.49 psia, is better than 2065.07 found in Scenario 1 (Listing 8.14) but this estimation is still not valid because the maximum value in the pressure data is clearly higher than 2828.49 psia. The estimate of initial reservoir pressure is a good check of the validity of the interpretation as the estimated initial reservoir pressure should be higher than the recorded pressure transient data.

As an additional comparison, in a second run of the *Window* algorithm of Scenario 2 of with known flow rates, the initial reservoir pressure was set as known. The modified initial estimates of reservoir parameters input file is listed in Listing 8.25. The results of Scenario 2 with known flow rates and with the initial reservoir pressure known are listed in Listings 8.26, 8,27 and 8.28.

```
model
    9
initial pressure
   3000     0.0     5000.0  1  (notice that the known/ unknown flag is set to 1)
permeability
   1000.0    10   10000
skin
   50    -10    200
wellbore storage
   0.001   0.00001    1.0
distance to boundary
   200       50      1000
distance to boundary
   200      50      1000
```

Listing 8.25: (Filename.est) Case 2, Scenario 2, initial estimates of unknown reservoir properties, p_i known.

```
1       36.39       77.57       56.98     8813.00     -6.5000      0.0160      832.81      832.81
1.54       1.54
```

Listing 8.26: (Filename.filpara) Case 2, Scenario 2, estimated reservoir parameters from *Window* algorithm, p_i known.

```
 1       36.39       77.57       56.98       20.61     34.0776     37.9176       66.48      104.94
185.08      14.73      67.59
```

Listing 8.27: (Filename.filrate) Case 2, Scenario 2, estimated flow rates from *Window* algorithm, p_i known.

```
   36.3949   100.0000
   36.7597    20.6112
   42.8473     0.0000
```

Listing 8.28: (Filename.outrate) Case 2, Scenario 2, summary of *Window* estimated flow rates, p_i known.

The results in Listing 8.26 do not look valid even though the *Window* algorithm reported that it had converged. The estimated permeability is 8813 md and the skin factor is now –6.5. These do not compare well with the values from other interpretations. The distances to the boundaries are also higher by a factor of about five. The error term is higher in comparison to the interpretation results in Listing 8.14 and Listing 8.22.

The results from Scenario 2 with known flow rates and known initial reservoir pressure should be better but it is defies logic by giving results that were not valid at all. One indication that the result may not be valid is by checking the error terms. Both error term 1 and error term 2 were higher when the initial reservoir pressure was set as known. The *Window* algorithm seems to have relaxed the error terms when the initial reservoir pressure is set as known and this leads to acceptance of wrong estimated values.

8.3.3. Case 2 Scenario 3 (Sensitivity to Initial Guess of Reservoir Parameters)

In this third example, the initial rates are as in Listing 8.11 and the fluid properties as in Listing 8.12. The initial reservoir pressure was set as known but initial estimates of unknown reservoir parameters were different from Scenario 2 (Listing 8.17). This example was run to find out the sensitivity of the *Window* algorithm to the initial estimate of unknown reservoir parameters provided to the program. The input files, filename.inrate, filename.prop and filename.est are listed in Listings 8.29, 8.30 and 8.31.

```
36.39488046   94.74760686   0    75.27284588   114.2223678
36.75975613   23.35687828   0    11.02119027    35.6925663
42.85019762   0             1     0              0
46.87751772   18.56180972   0     6.705628568   30.41799086
46.99487182   45.03232234   0    30.52908989    59.53555479
50.85955882   32.51397529   0    19.26257757    45.76537302
54.81491249   53.94048985   0    38.54644063    69.33453906
66.79211969   100           1    89.99999985   110.0000001
77.30033512   0.6594481896  0   -9.406496779    10.72539316
77.57028 45.  47.59749      0   30.54283752     59.55235745
```

Listing 8.29: (Filename.inrate) Initial estimates of unknown flow rates, p_i known.

```
porosity
 0.2
height
 100.00
well radius
 0.3
viscosity
 4.0
formation volume factor
 1.5
compressibility
 10.0e-6
window length (hours)
 49
translation length
 9
starting time
 0
```

Listing 8.30: (Filename.prop) Fluid, reservoir and completion properties, p_i known.

```
model
    9
initial pressure
   3000     0.0     5000.0  1
permeability
   500.0    10   10000
skin
   25    -10    200
wellbore storage
   0.01   0.00001    1.0
distance to boundary
   100       50      1000
distance to boundary
   100      50      1000
```

Listing 8.31: (Filename.est) Initial estimates of unknown reservoir properties, p_i known.

Listings 8.32, 8.33 and 8.34 show the result of moving window analysis for this example. The estimates are different from the results in Listing 8.18, 8.19 and 8.20. Case 2 Scenario 3 results should not be accepted because the estimated wellbore storage does not agree with wellbore storage estimated from the interpretation of a single transient.

```
   1      36.39      77.57      56.98      569.43     28.3601     0.0104     109.05     109.05
5.45       5.45
```

Listing 8.32: (Filename.filpara) Estimated reservoir parameters from *Window* algorithm, p_i known.

```
 1      36.39      77.57      56.98      24.43     22.9184     45.4191      37.56      60.36
4.83
```

Listing 8.33: (Filename.filrate) Estimated flow rates from *Window* algorithm, p_i known.

```
   36.3949    94.7476
   36.7598    24.4271
   42.8502     0.0000
```

Filename 8.34: (Filename.outrate) Summary of *Window* estimated flow rates, p_i known.

The example runs in Case 2 show that the interpretation result from the *Window* algorithm should be checked with the interpretation result of a single transient. Estimated values of initial reservoir pressure are also a good check of the validity of the interpretation because the estimated initial reservoir pressure from the *Window* algorithm should be higher than the pressure transient data. There is also uncertainty in the choice of the reservoir model used to interpret the data. Even if the choice of reservoir model is correct, the estimated parameters may be sensitive to initial values of the unknown reservoir parameters. Good initial estimates of unknown reservoir parameters are required. Case 2 interpretations are summarized in Table 8.1.

	Initial pressure (psia)	Permeability (md)	Skin factor	Wellbore Storage (STB/psi)	Boundary distance (ft)	Boundary distance (ft)
Single transient	3000	923	61.2	2.47e-3	175	175
Scenario 1	2065.07	1181.78	84.8	2.8e-3	983.28	983.27
Scenario 1	3000 (known)	1143.87	48.3	1.1e-3	258.25	258.25
Scenario 2	2928.49	1073.43	70.7	2.5e-3	147.7	147.7
Scenario 2	3000 (known)	8813	-6.5	1.6e-2	832.8	832.8
Scenario 3	3000 (known)	569.43	28.36	1.04e-2	109.05	109.05

Table 8.1: Case 2 results.

8.4. Case 3

8.4.1 Incorrect Adjustment When Adjustment Window is Too Wide

In this example, the robustness of the straight line break point adjustment algorithm was investigated. The pressure data from Case 2 were used here. The break points left after screening using the histogram and Fourier screening method are shown in Figure 8.7. The *Wavelet* detected break points are not exactly at the beginning of a transient.

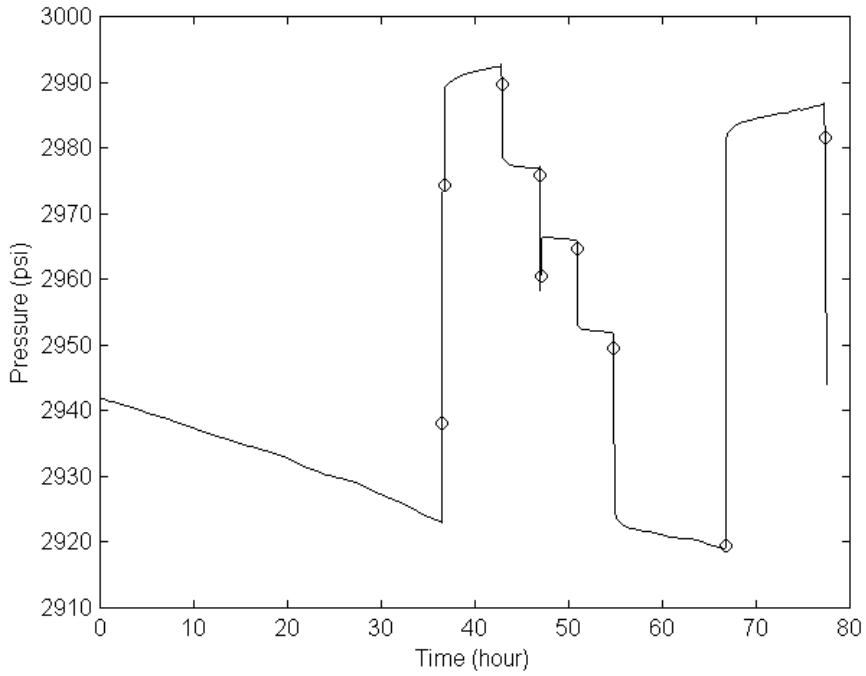


Figure 8.7: Break points after histogram and Fourier screening. Break points do not fall exactly at the beginning of transients.

The straight line break point adjustment algorithm was applied to correct the *Wavelet* detected break points. In the present straight line adjustment algorithm, an adjustment window was defined around the break point. The algorithm searches for the steepest straight line in the window. Next, the least steep straight line is searched to the right of the steepest straight line found earlier. This procedure will work if all break points are close to the actual beginning of their transient and each small adjustment window opened encloses only one break point. This procedure will most likely fail if the adjustment window includes more than one beginning of a transient.

When the adjustment window is so wide that the left edge may include the region just after the beginning of an earlier transient, the procedure may fail because the steepest line found may be at the beginning of the earlier transient. This limitation causes the procedure to be fragile because the adjustment window has to be defined wide enough so that it can correct the biggest error in the detected break point. When the adjustment window is declared is too wide the algorithm may break down. On the other hand, when the adjustment window is not wide enough some break points may not be corrected properly. This issue will be demonstrated further in this section.

In Figure 8.7, the break points were adjusted using a window width of 0.1 hour to the left of the break point and 0.1 hour to the right of the break point. A second adjustment window definition of a minimum of 10 points to the left and 10 points to the right of the break point was specified. The straight lines were defined using four data points. The result shows the break points were adjusted properly as near as possible to their respective transient.

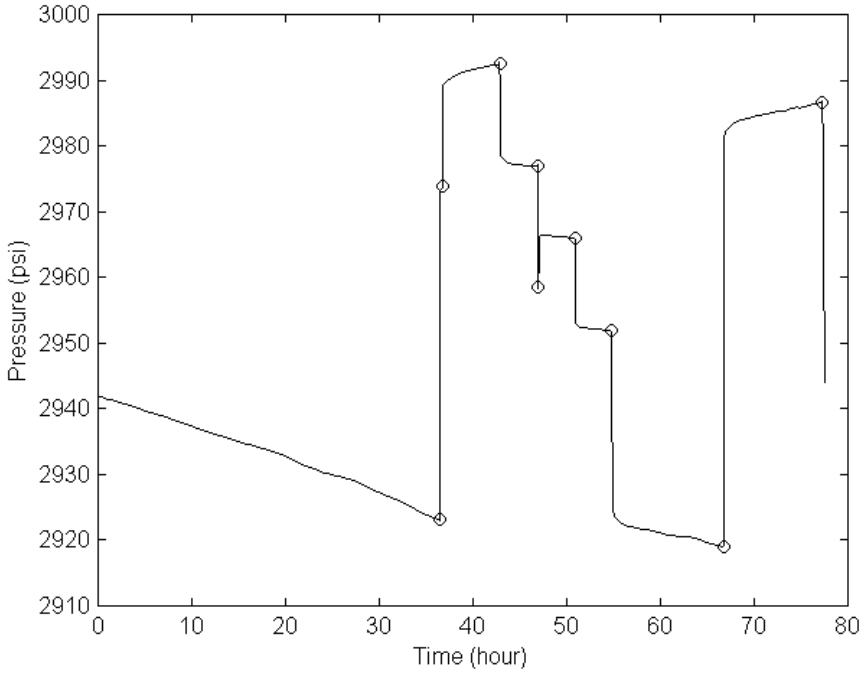


Figure 8.8: Adjustment using a window width 0.1 hour left of break point and 0.1 hour right of break point.

In Figure 8.9, the break points are adjusted using a window width of 0.2 hour to the left and to the right of the break point. A second adjustment window criterion of 20 points to the left and 20 points to the right of the break point was specified. The straight lines were defined using four data points. The result shows that the fifth break point could not be adjusted properly. This is because the 0.2 hour left window width extended to the beginning of the fourth transient and the beginning of the fourth transient is steeper than the beginning of the fifth transient.

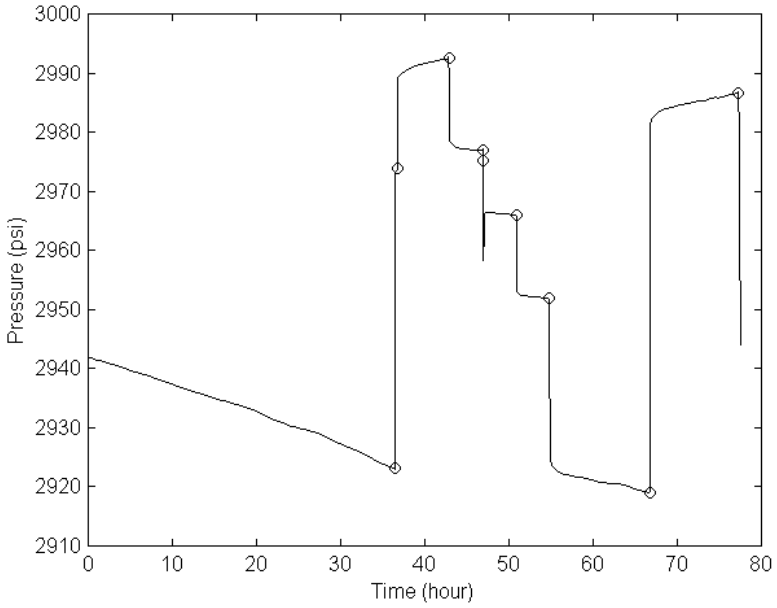


Figure 8.9: Incorrect adjustment when the defined adjustment window is too wide.

8.4.2. Insufficient Adjustment When Adjustment Window is Too Narrow

It was shown in Section 8.4.1 that the adjustment algorithm is not sufficiently robust when a wide adjustment window is used. The effectiveness of the adjustment algorithm using a narrower adjustment window was investigated. It was found that some break points could be under-adjusted as the adjustment step is limited by the window width. Figure 8.10 shows detected break points before adjustment and Figure 8.11 shows the adjusted break points using adjustment windows of 0.05 hours to the left and right of the break point. The second adjustment window constraint was seven data points to the left and to the right of the break points. The straight lines are fitted using four data points. In several cases the adjusted break points missed the true breaks.

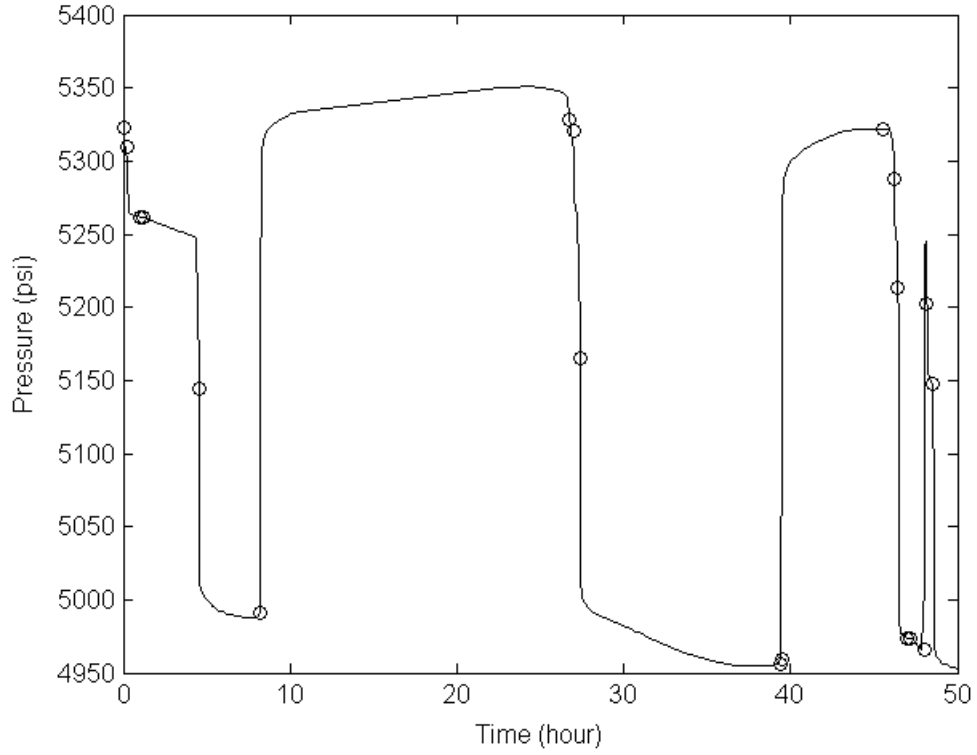


Figure 8.10: Break points before straight line adjustment.

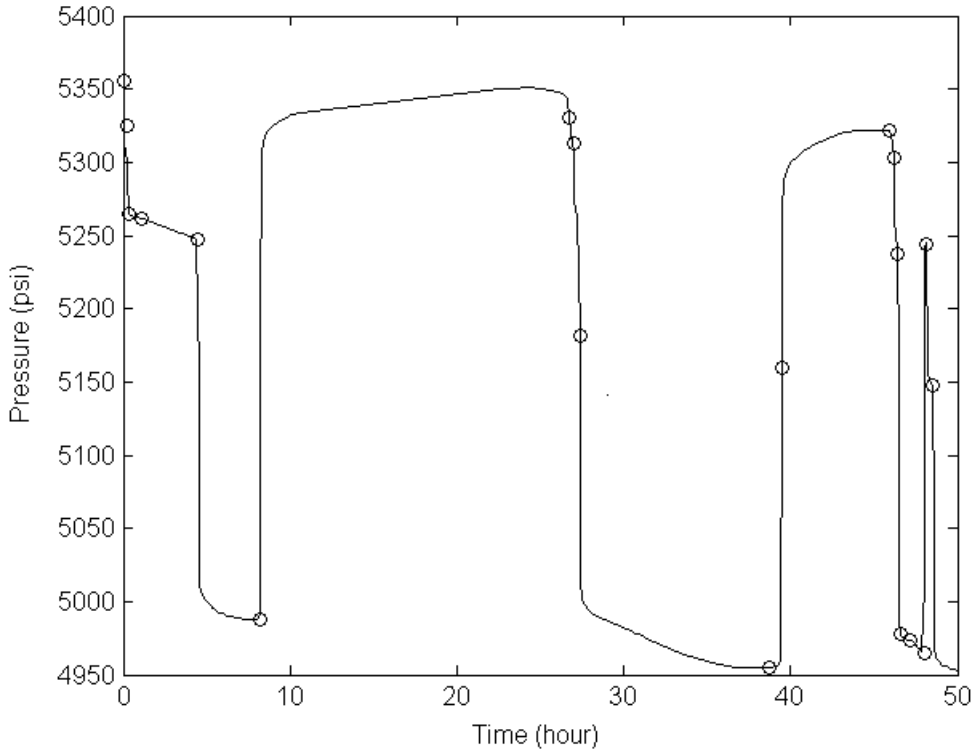


Figure 8.11: Under-adjustment of break point for narrow adjustment window.

8.4.3. Iterative Adjustment Using Narrow Adjustment Window

The iterative adjustment of break points using narrow adjustment windows was investigated as a possible solution to the limitation encountered when using a wide adjustment windows. Figure 8.12 shows the break points before adjustment with four break points (test points) adjusted manually to test the robustness of the iterative adjustment approach. The adjustment window was 0.05 hour to the left and 0.05 hour to the right of a break point. The second adjustment window constraint was seven points to the left and seven points to the right of a break point. The straight line was fitted using four data points.

Figure 8.13 shows the adjusted break points after four iterations. It was found that test points 1, 3 and 4 could not be adjusted. Test points 1 and 4 were stuck at small events that prevented them from being corrected further.

An additional adjustment using a window width of 0.2 hour to the left and 0.2 hour to the right of the break points was performed. The second adjustment window constraint was 15 points to the left and 15 points to the right of a break point. The straight line was fitted using four data points. The result of this adjustment is shown in Figure 8.14. Test points 1 and 4 could now be adjusted but test point 3 still could not be adjusted. Break point 5 was also not adjusted properly when the wide adjustment window was used.

This example also showed that detected break points to the left of the beginning of their transients could not be adjusted using the present algorithm. However, it was noticed that almost all detected break points were to the right of the beginning of their transients.

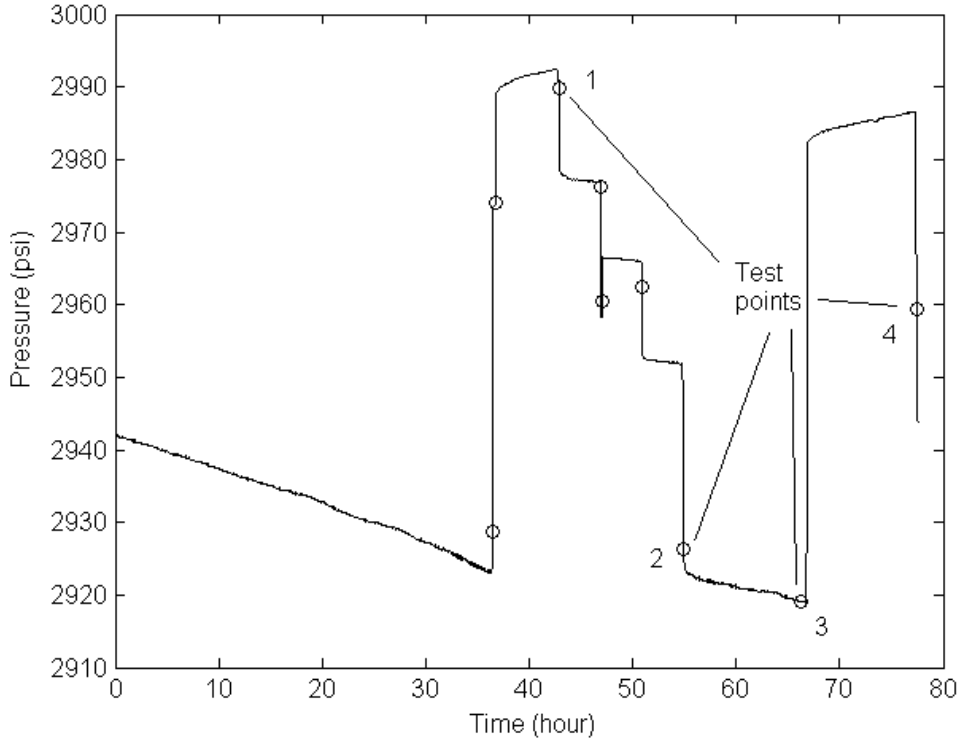


Figure 8.12: Break points before adjustment with four test points defined.

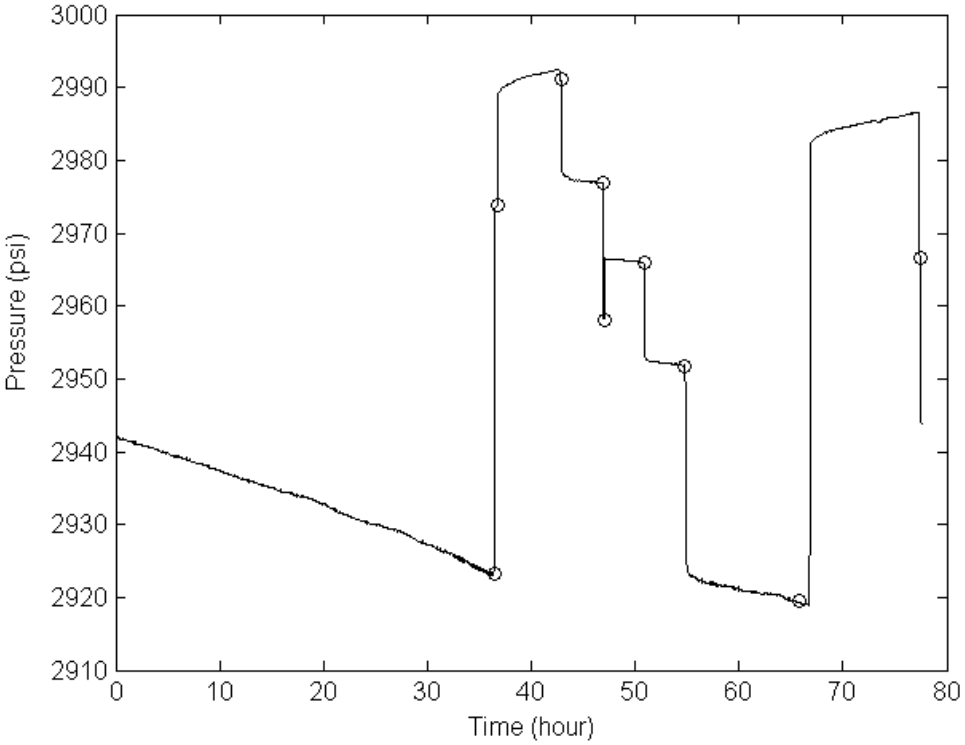


Figure 8.13: Adjusted break points after four iterative adjustments using narrow window.

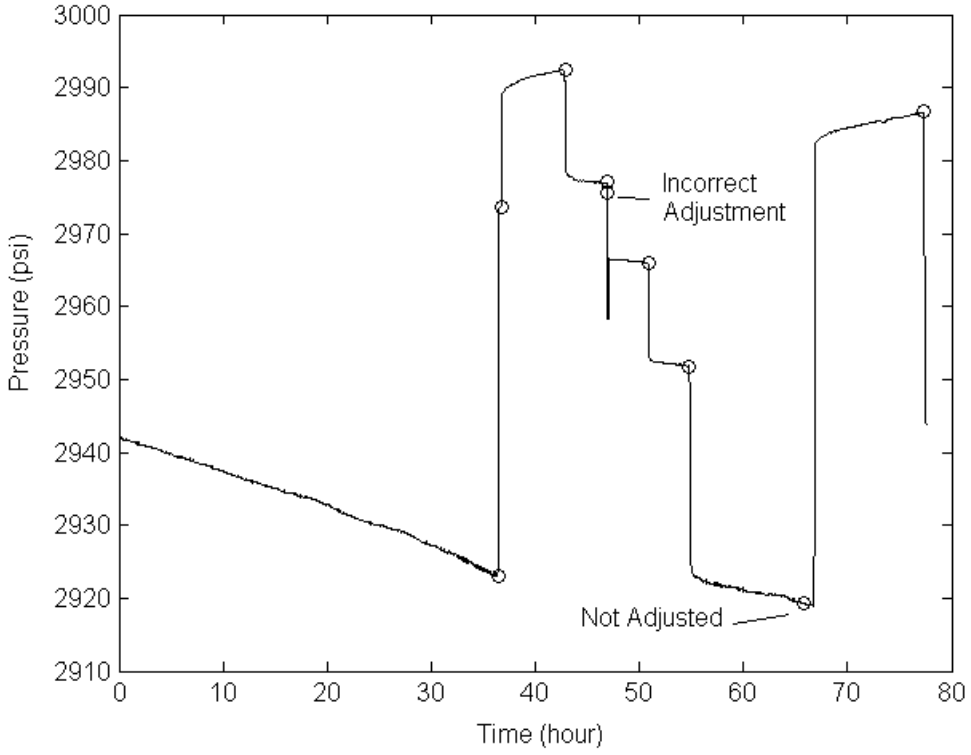


Figure 8.14: Break points after adjustment using wide window (0.2 to the left and right).

8.4.4. Possible Improvement to Straight Line Adjustment Algorithm

As the present straight line break point adjustment algorithm is not sufficiently robust, additional information is needed to help the algorithm to pick the proper break point when a wide adjustment window may enclose a few beginnings of transients. Break points that need additional heuristics have to be identified first. A reverse line search order can be implemented where a least steep line is sought starting from the left edge of the adjustment window, a steepest line right of the least steep line is searched for next. Break points that adjust to the same point using the normal and reverse search do not need additional heuristics.

A break point should be corrected to the nearest beginning of a transient. The gradient pattern could be used to help guide the algorithm. Figure 8.15 shows a pressure transient and its pressure derivative (gradient). Consider two scenarios, in which the detected break point is either to the left of or the right of the beginning of the transient.

For both the left and right break points, the gradient increases while moving left until reaching the earlier transient where the gradient decreases. The gradient decreases while moving right until at the beginning of the next transient gradient at which point it increases again. The break point should be corrected to the nearest beginning of a transient. Figure 8.16 shows absolute gradient values for all four scenarios of pair of transients. Consideration of the sequence of gradient changes in each of the four scenarios could be used to help locate the breaks.

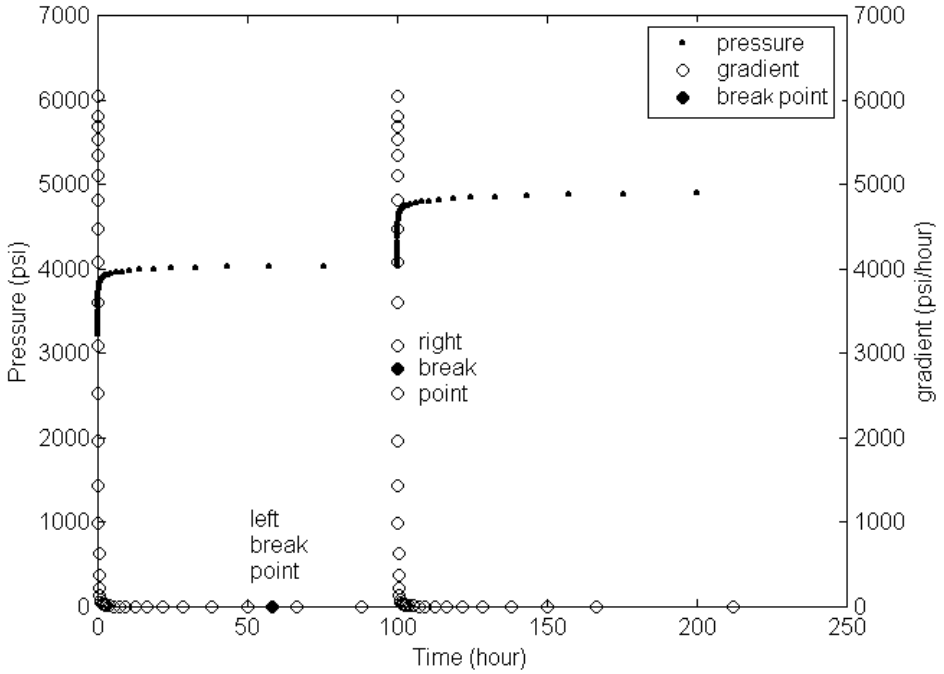


Figure 8.15: Pressure transient and its derivative.

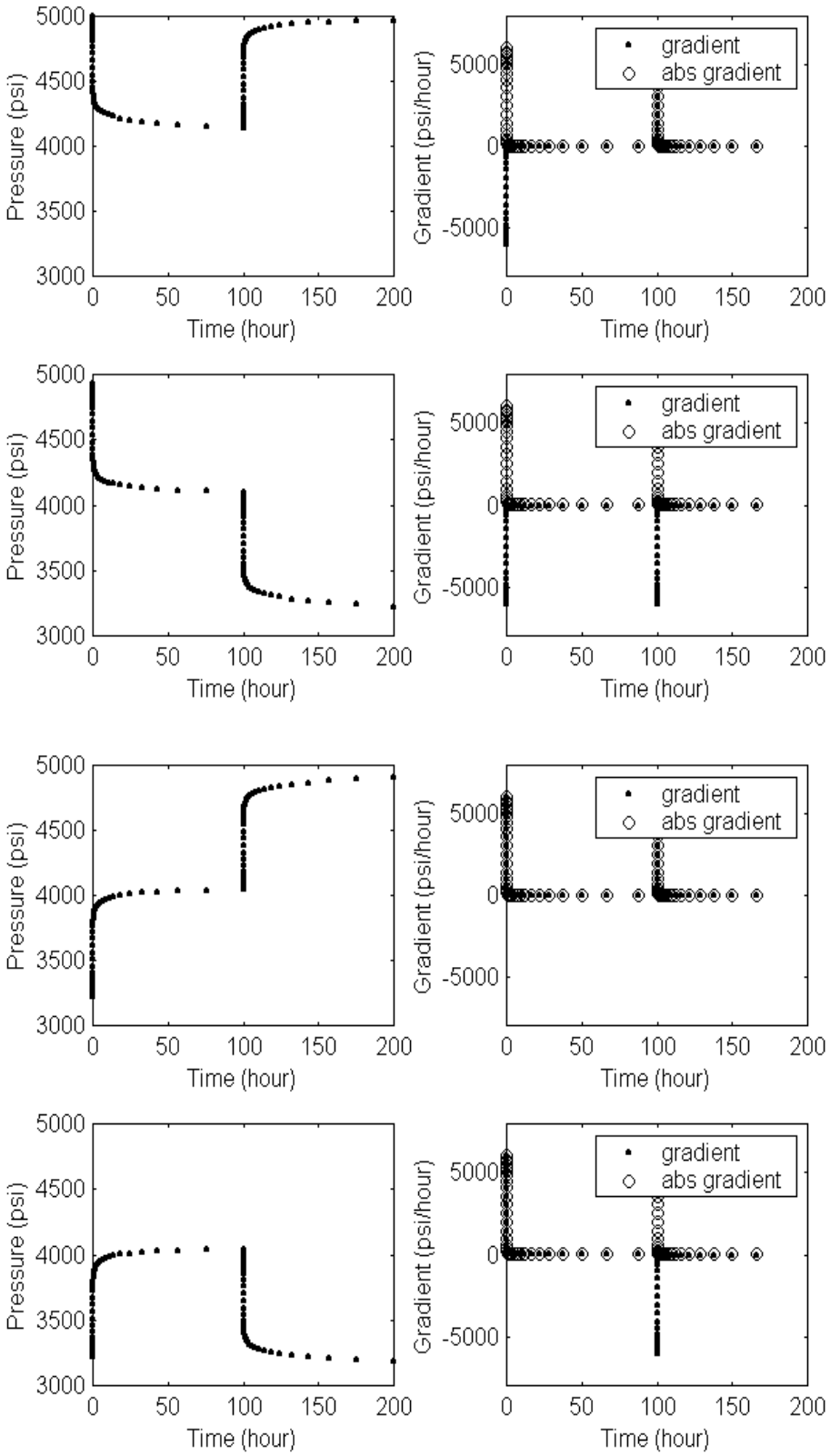


Figure 8.16: Absolute gradient values for all four transient pair scenarios.

8.5. Case 4

A real field record of about 375 hours duration was used in this example as shown in Figure 8.17. The data was denoised, spike outliers were removed and the data resampled at a regular sampling interval using the *Wavelet* algorithm. Step outliers were not present in the data. There were gaps in the record where data were not available so data were resampled at regular sampling interval only in the regions where data were present.

The data was processed using the *Wavelet* algorithm for 0.0005, 0.001, 0.005, 0.01 and 0.05 hour wavelet sample spacing. Slope detection values of 5 to 50 psi/hour (in 1 hour steps) were used for each of the wavelet sample spacing values. The detected break points for all the wavelet sample spacing and slope detection threshold values combinations were binned into a histogram with 0.05 hour bin width and break points within 0.05 hour were combined. Fourier break point screening was performed after histogram screening.

The histogram screening cutoff level was set low so that all true break points would be selected since false break points still selected were screened further using the Fourier approach. The histogram break point selection graph is shown in Figure 8.18. The distribution of the number of detection of break points is plotted in Figure 8.19. This distribution was used to select the histogram cutoff level of 15. Figure 8.20 shows the Fourier break point selection graph for a Fourier window width of 2 hour. The distribution of the Fourier level in Figure 8.21, was used to select the Fourier cutoff of 5.

The detected break points for histogram cutoff of 15 and Fourier cutoff of 5 are shown in Figure 8.22. These break points were then adjusted using the straight line adjustment algorithm using the wider of 0.1 hour or 10 points left of the break point and the wider of 0.1 hour or 10 points to the right. The straight lines were fitted using four data points. The adjusted break points are shown in Figure 8.23. It was found that the histogram and Fourier screening method is still not perfect and further research will be needed on screening false break points. This example also shows that the straight line break point adjustment algorithm needs improvement to make it more robust.

The effect of applying a higher cutoff level during histogram screening was investigated by using histogram cutoff levels of 25, 40, 50 and 100, the detected break points are shown Figures 8.24, 8.25, 8.26 and 8.27. At higher cutoff values, some break points were missed. Additional research is needed to find the proper cutoff where all true break points are included. Fourier screening can screen false break points in the flat region of a transient but will not be able to screen false break points in the steep region just after the beginning of a transient.

In order to proceed, the break points from the *Wavelet* algorithm were screened manually and adjusted using the straight line break point adjustment algorithm as shown in Figure 8.28. The break point at about 94.2 hour cannot be adjusted properly. There are no data between 78 and 94 hour and the straight line algorithm requires five data points to the left of the break point, these five data points would lie between 77 and 78 hour. The gap in the data at the beginning of the transient prevents the adjustment of the break point. The beginning of transients are rounded in their appearance due to the denoising algorithm and this makes the correction using the straight line algorithm less accurate.

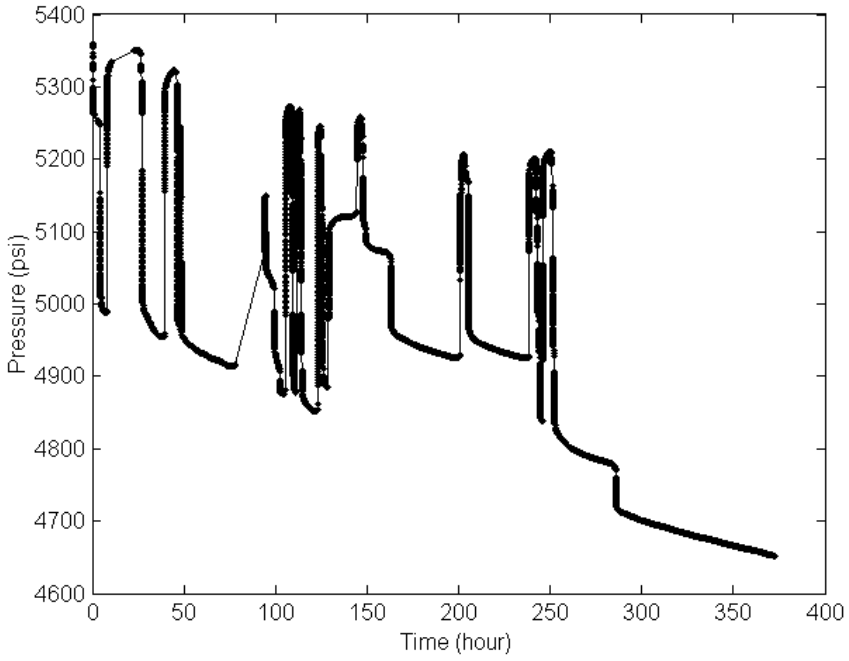


Figure 8.17: Real field data used in Case 4.

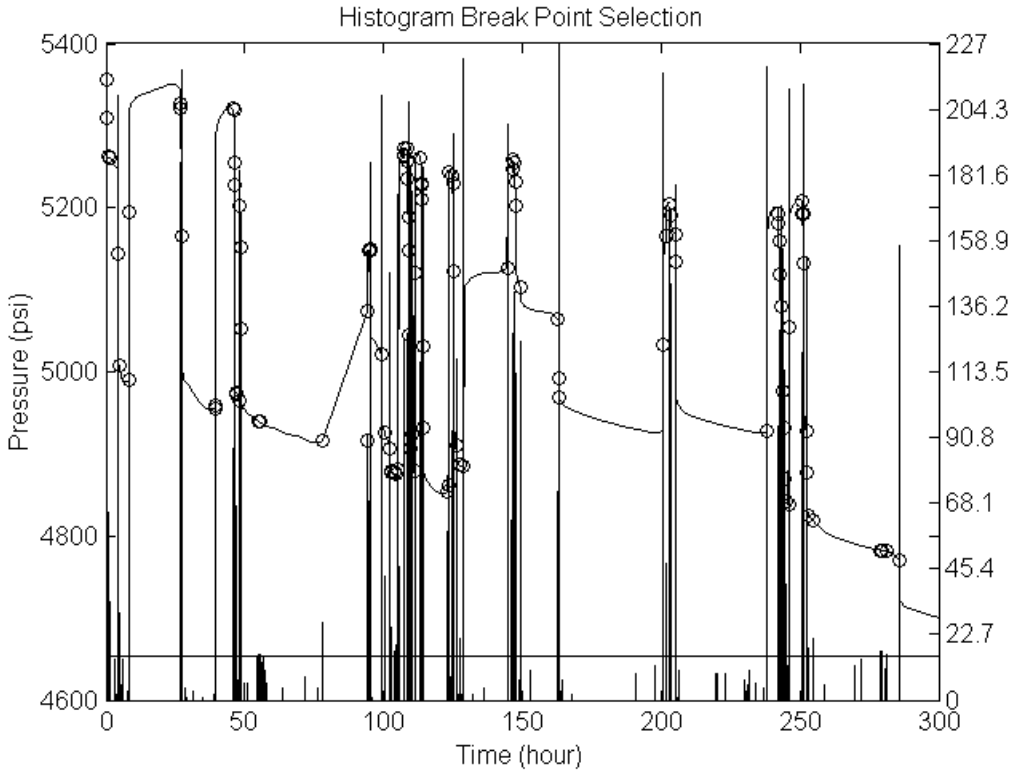


Figure 8.18: Histogram screening, 0.05 hour bin width, break points within 0.05 hour combined.

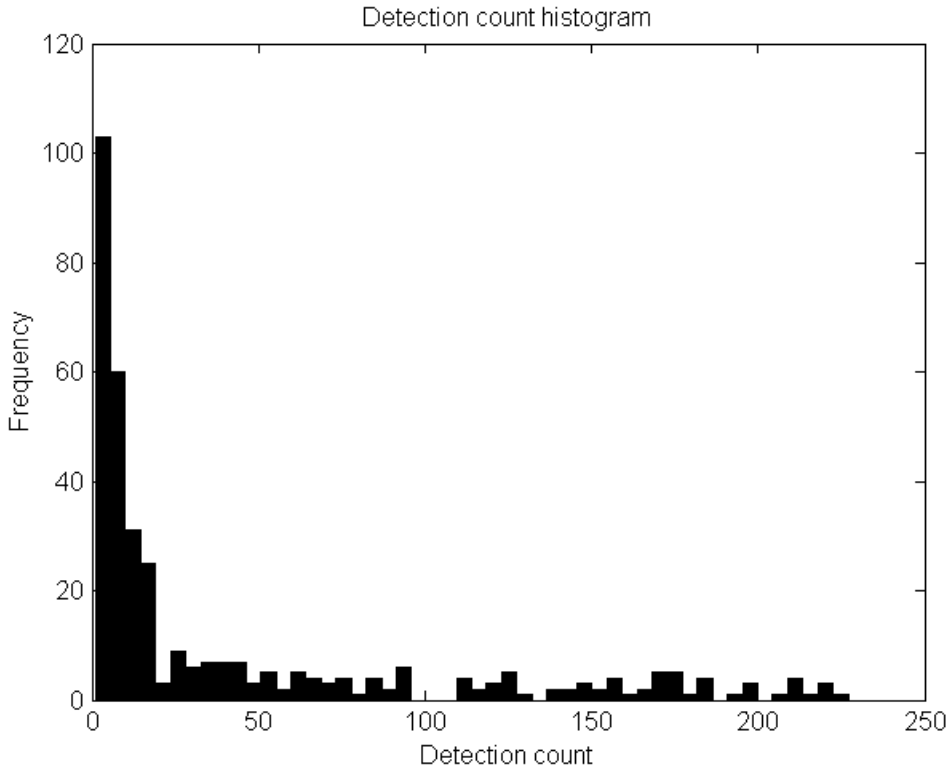


Figure 8.19: Distribution of the number of times break points were detected.

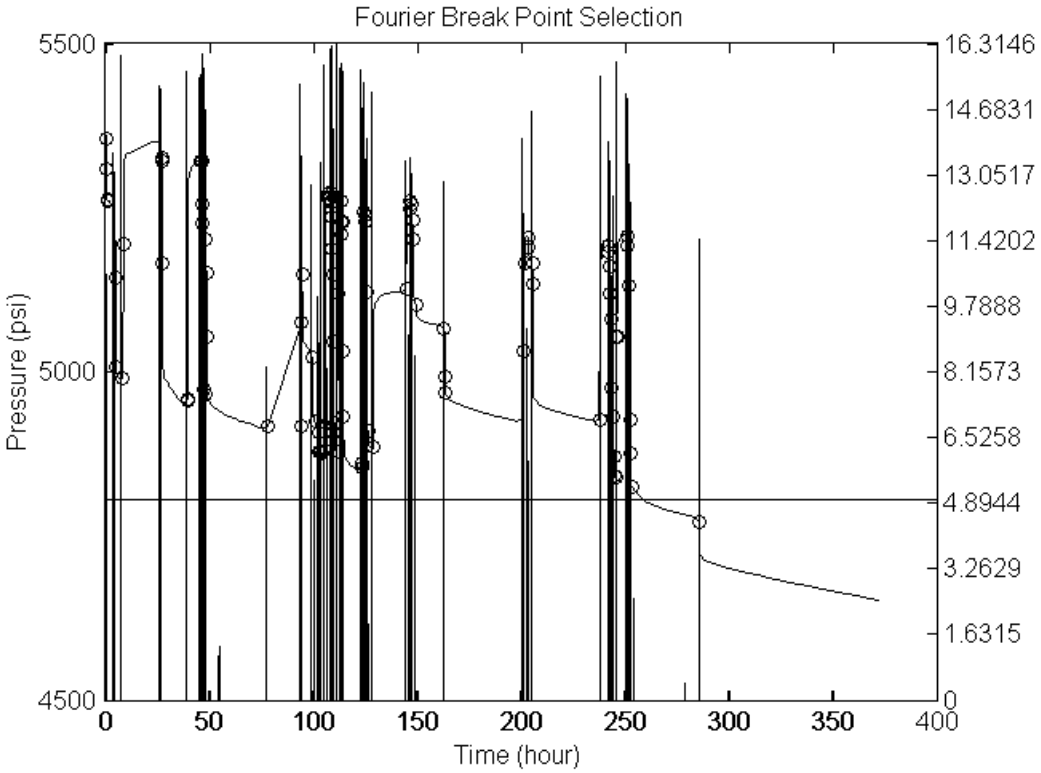


Figure 8.20: Fourier screening, 2 hour Fourier transform window, Fourier cutoff of 5.

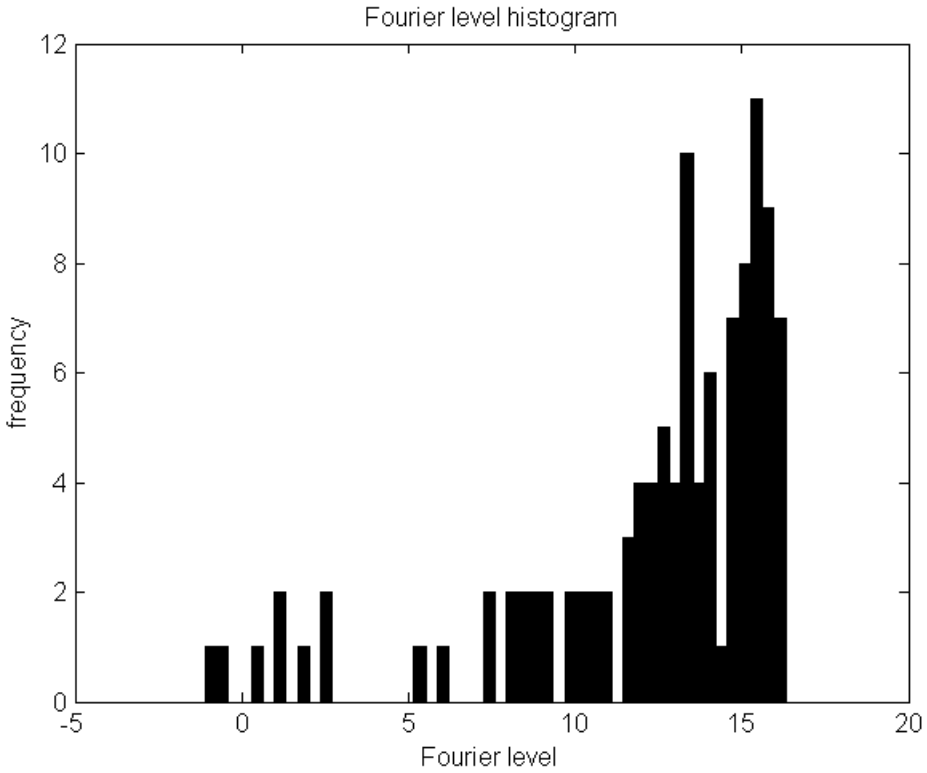


Figure 8.21: Distribution of Fourier level of break points.

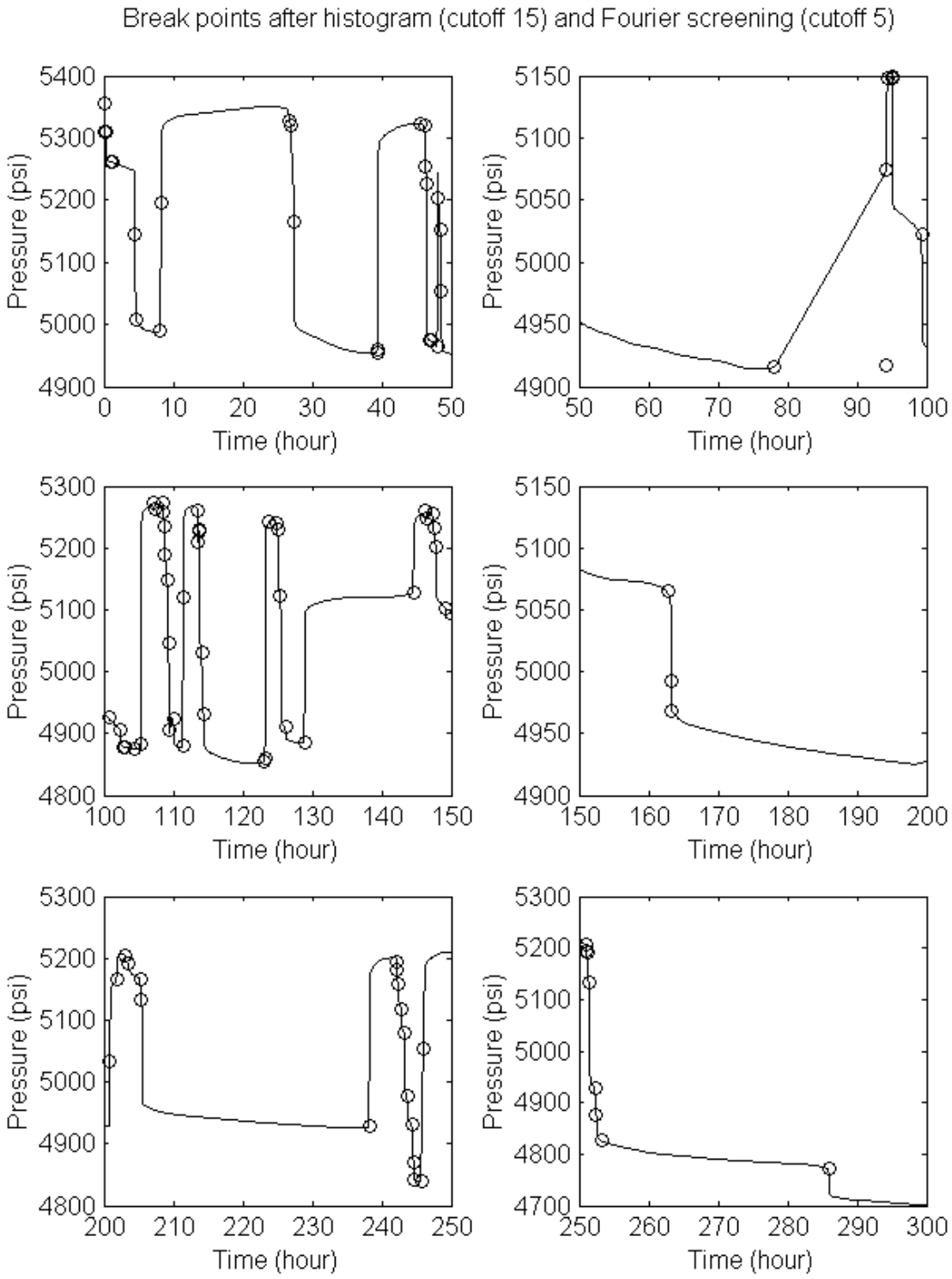


Figure 8.22: Break points after histogram (cutoff 15) and Fourier screening (cutoff 5).

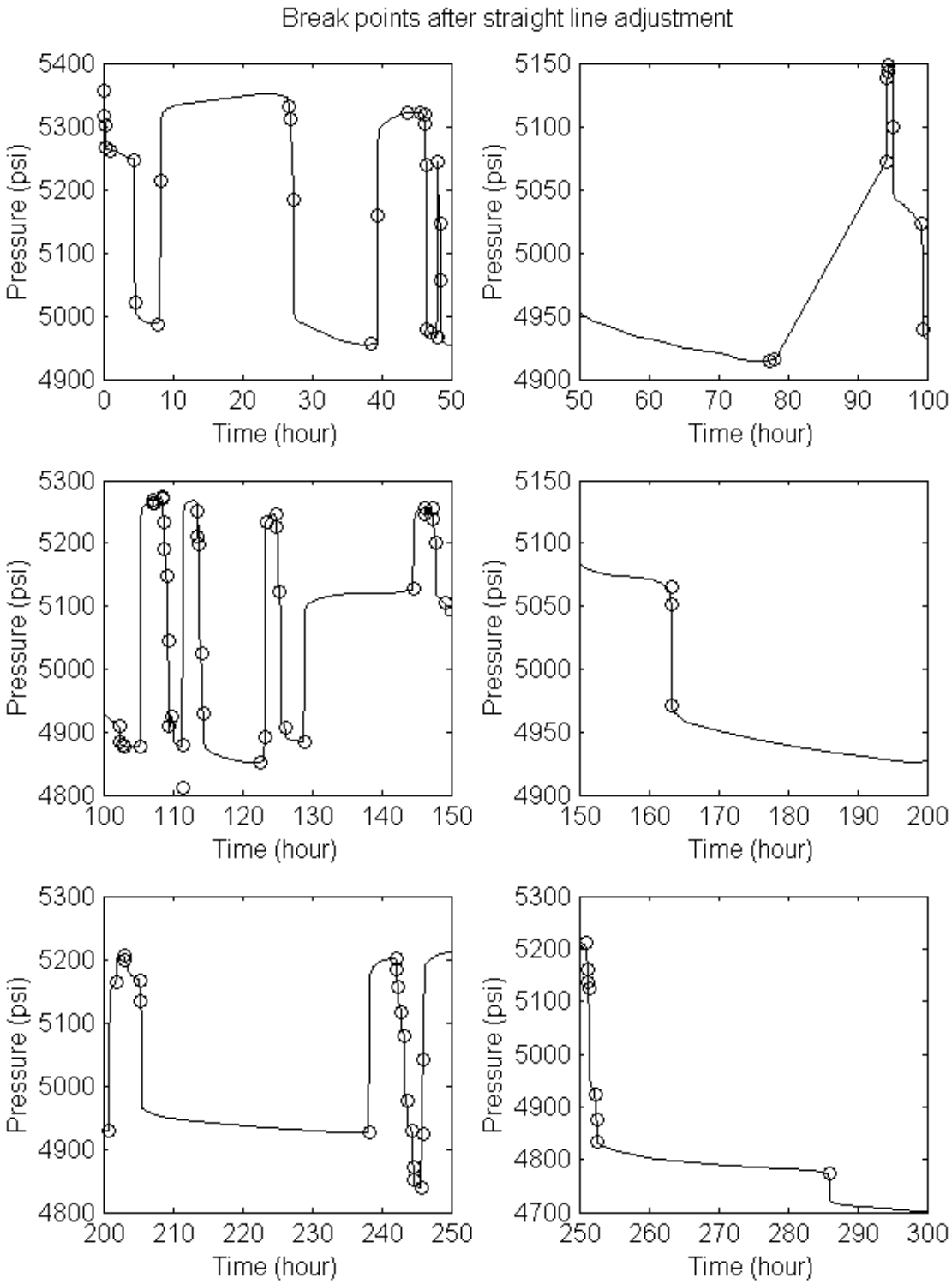


Figure 8.23: Break points after straight line adjustment for histogram (cutoff 15) and Fourier screening (cutoff 5).

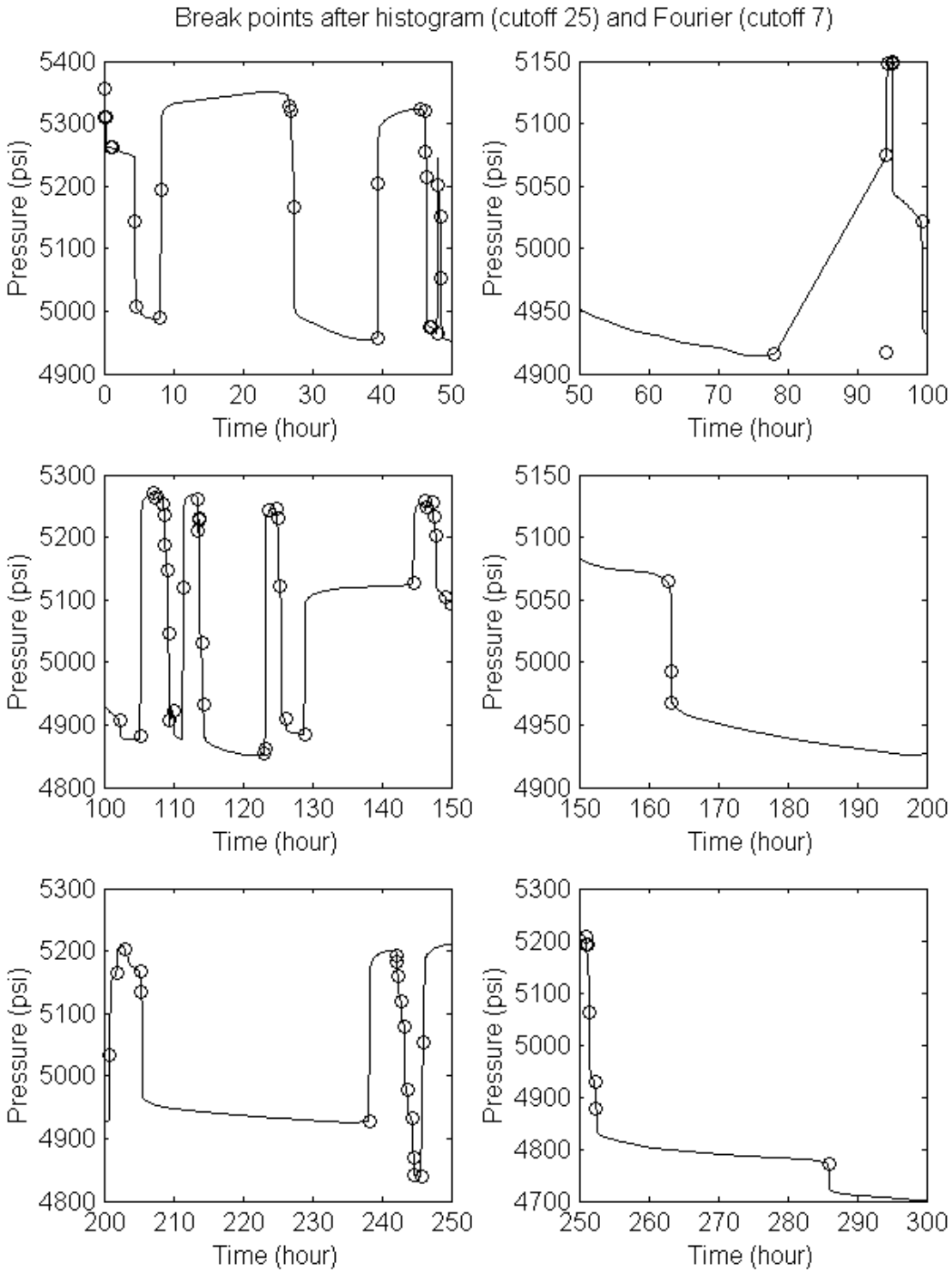


Figure 8.24: Break points after histogram (cutoff 25) and Fourier screening (cutoff 7).

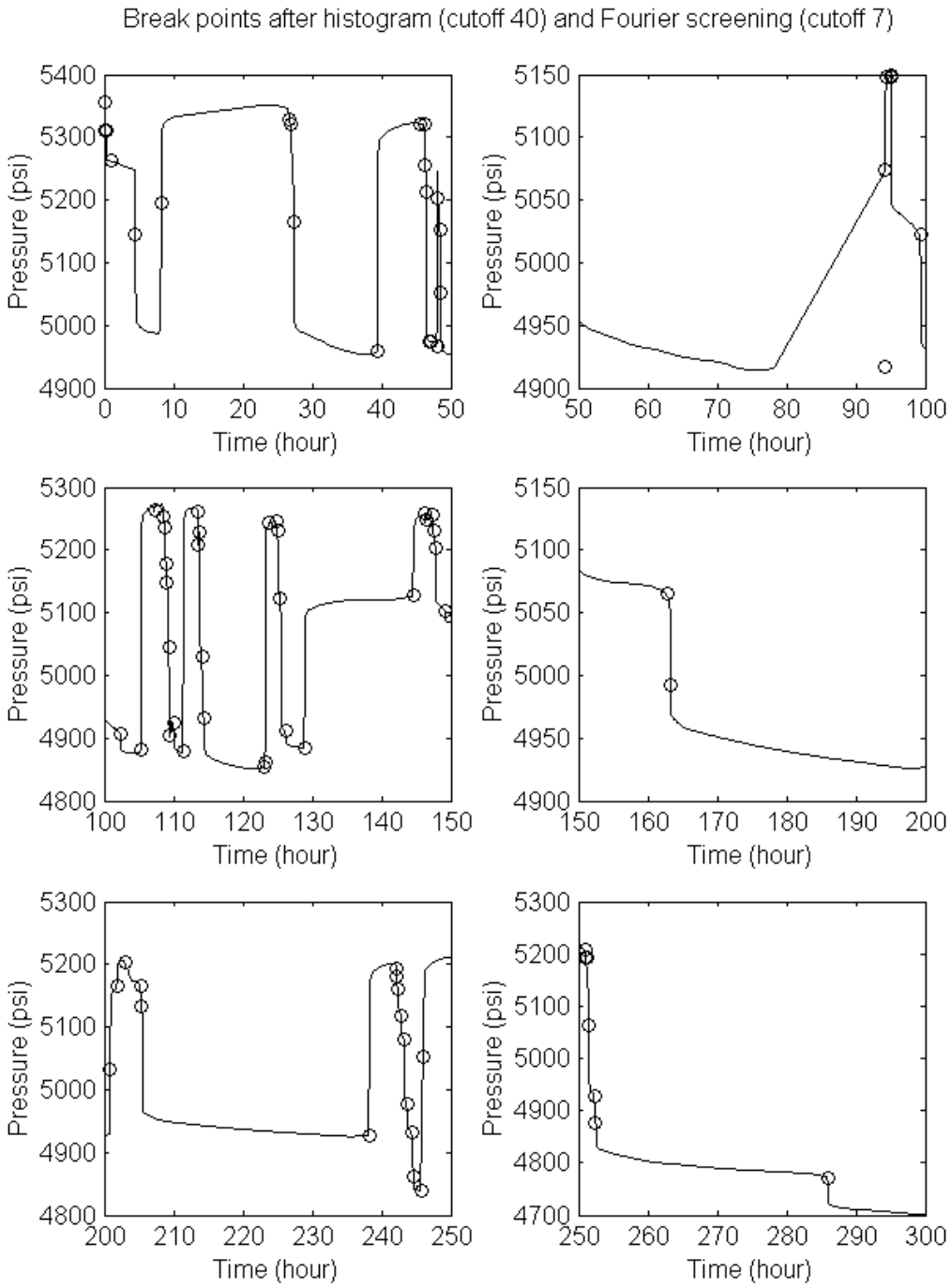


Figure 8.25: Break points after histogram (cutoff 40) and Fourier screening (cutoff 7).

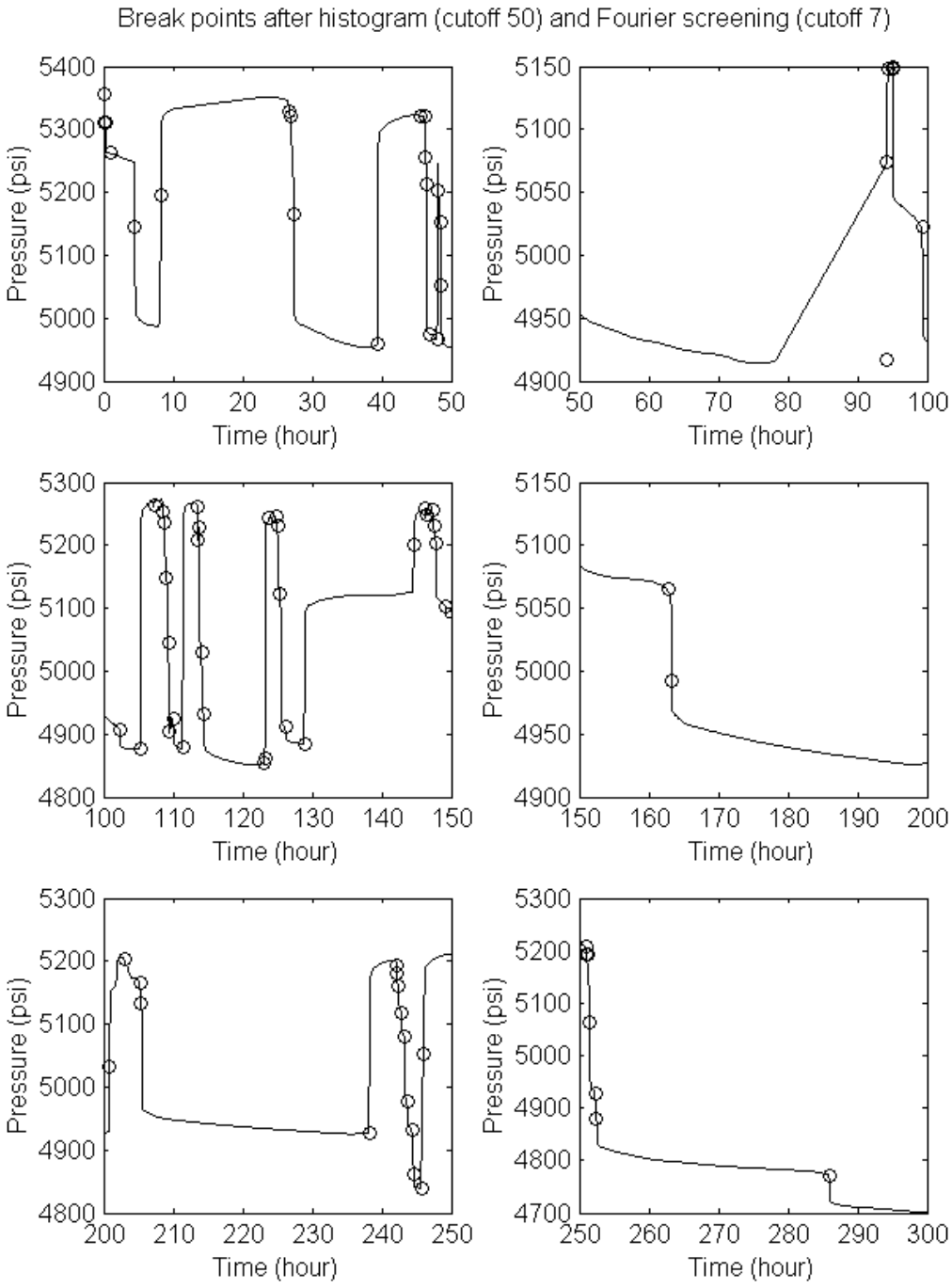


Figure 8.26: Break points after histogram (cutoff 50) and Fourier screening (cutoff 7).

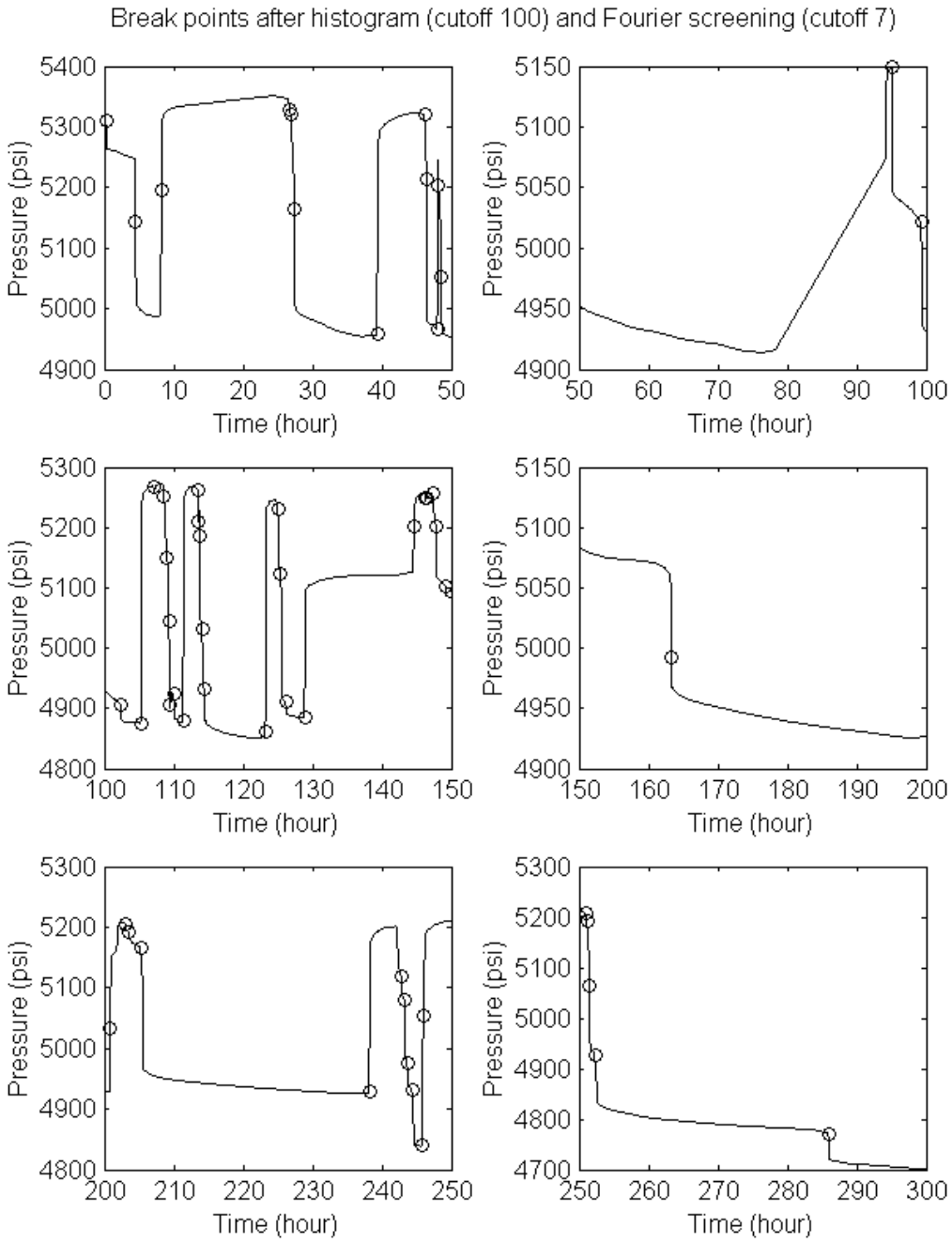


Figure 8.27: Break points after histogram (cutoff 100) and Fourier screening (cutoff 7).

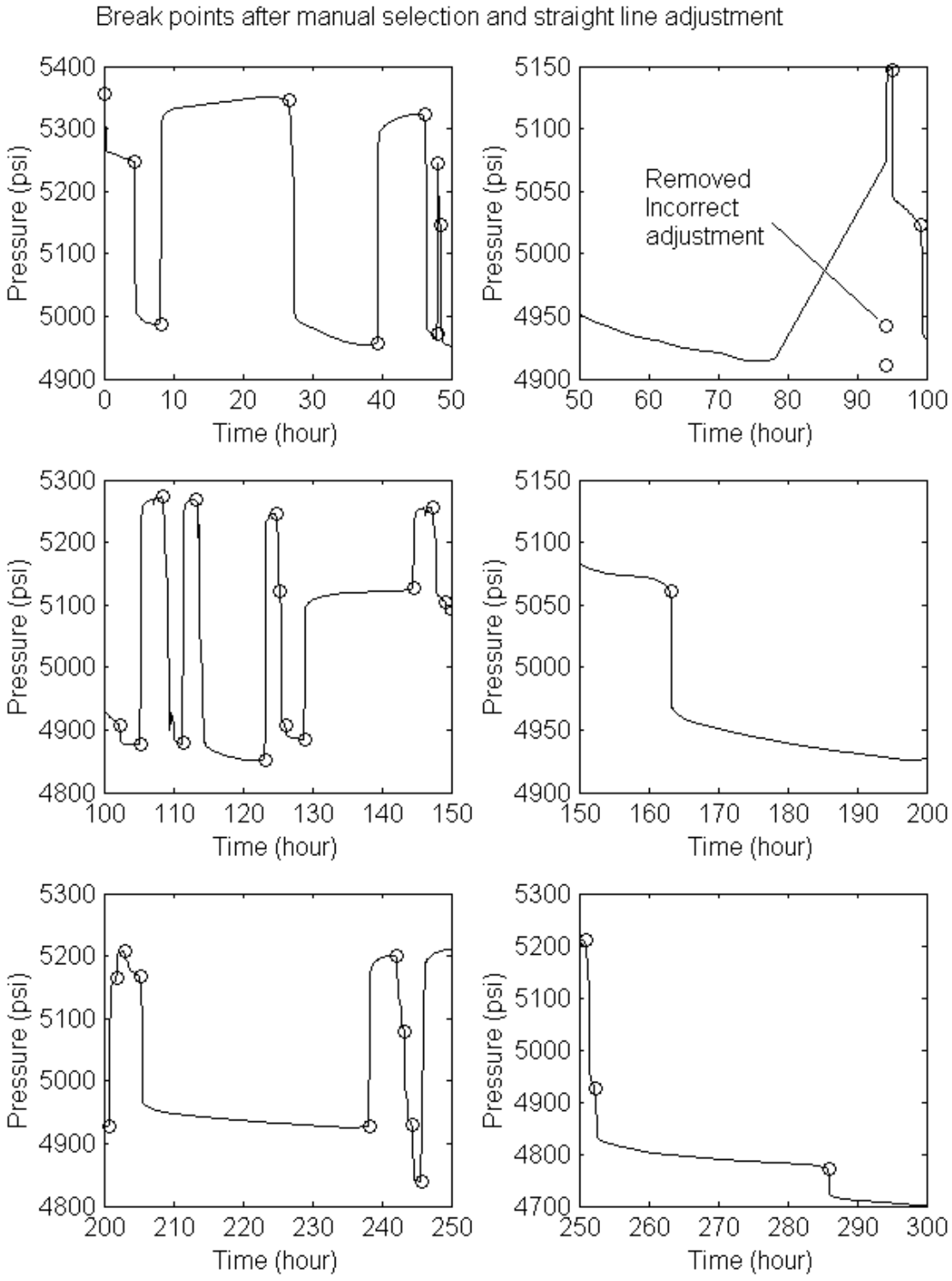


Figure 8.28: Break points after manual selection and straight line adjustment.

Only two flow rates and break times 0 and 285.99 hour were assumed to be known (one zero and another nonzero). The result from the *Inrate* algorithm is shown in Listing 8.35. The first break time was corrected manually to coincide with an actual data point since the *Window* algorithm requires the first break point to be at the same time or earlier than the first data point.

0.00000	**0**	**1**	0	0
4.49966	0	0	-251.8176641	251.8176641
8.198874	0	0	-251.8176641	251.8176641
26.652562	0	0	-251.8176641	251.8176641
39.44814	0	0	-251.8176641	251.8176641
46.18087	0	0	-251.8176641	251.8176641
48.001192	0	0	-251.8176641	251.8176641
48.150382	0	0	-251.8176641	251.8176641
48.549519	0	0	-251.8176641	251.8176641
94.210967	0	0	-251.8176641	251.8176641
95.102588	0	0	-251.8176641	251.8176641
99.296186	0	0	-251.8176641	251.8176641
102.393386	0	0	-251.8176641	251.8176641
105.349102	0	0	-251.8176641	251.8176641
108.546558	463.097886	0	118.6606433	807.5351287
111.449474	95.10106145	0	-175.7368153	365.9389382
113.396894	629.879486	0	252.0859228	1007.673049
123.246188	231.2500324	0	-66.81763892	529.3177037
124.785598	839.34016	0	419.6544614	1259.025859
125.449533	708.8645865	0	315.2740029	1102.45517
126.148601	386.0534611	0	57.02510359	715.0818186
128.948556	358.2334782	0	34.76911735	681.697839
144.69997	889.7537979	0	459.9853715	1319.522224
147.292628	1193.495127	0	702.9784341	1684.01182
149.192759	918.7844743	0	483.2099126	1354.359036
163.196734	868.6642629	0	443.1137436	1294.214782
200.849087	683.8705918	0	295.2788072	1072.462376
201.845099	1125.216701	0	648.3556934	1602.077709
203.105392	1204.431506	0	711.7275367	1697.135474
205.292318	1161.46066	0	677.3508605	1645.57046
238.24901	964.9694767	0	520.1579144	1409.781039
242.052682	1223.209762	0	726.7501414	1719.669382
243.248577	1147.984916	0	666.5702651	1629.399566
244.451027	1027.774401	0	570.4018535	1485.146948
245.848937	948.8677888	0	507.2765641	1390.459014
250.938172	1259.088302	0	755.4529736	1762.72363
252.346225	1085.175097	0	616.3224102	1554.027784
285.996233	**1000**	**1**	799.999997	1200.000003
372.52771	952.8514448	0	510.4634889	1395.239401

Listing 8.35: (Filename.inrate) Output of *Inrate* algorithm. (Data in columns: time, flow rates, known/unknown = 1/0, minimum, maximum).

The input files filename.est and filename.prop are shown in Listings 8.36 and 8.37. An infinite-acting reservoir model was assumed. The initial reservoir pressure was set based on the highest pressure in the data file (the first data point). No cumulative production data were available so the file filename.prod was included as an empty file. The result of *Window* processing from files filename.filpara and filename.outrate are listed in Listings 8.38 and 8.39.

```
model
    1
initial pressure
   5360     5000     6500  1
permeability
   500.0    10   10000
skin
   0    -10    100
wellbore storage
   0.01   0.00001    1.0
```

Listing 8.36: (Filename.est) Initial estimates of unknown reservoir parameters.

```
porosity
 0.2
height
 10.00
well radius
 0.3
viscosity
 4.0
formation volume factor
 1.5
compressibility
 10.0e-6
window length (hours)
 49
translation length
 9
starting time
 0
```

Listing 8.37: (Filename.prop) Fluid, completion and processing parameters.

1	0.00	286.00	143.00	1116.11	-1.6185	0.0092	1.45	1.45
2	26.65	286.00	156.32	**8828.51**	**48.4057**	0.0070	**3.91**	**3.91**
3	39.45	286.00	162.72	1113.18	-1.6429	0.0081	1.63	1.63
4	48.55	286.00	167.27	1020.96	-2.2615	0.0062	2.38	2.38
5	94.21	286.00	190.10	1026.01	-2.2296	0.0061	2.38	2.38
6	105.35	286.00	195.67	2269.20	5.6126	0.0061	3.69	3.69
7	123.25	286.00	204.62	**8192.66**	**44.2796**	0.0064	**3.91**	**3.91**
8	144.70	286.00	215.35	1094.24	-1.7534	0.0040	1.08	1.08
9	163.20	286.00	224.60	1100.81	-1.7109	0.0038	0.97	0.97
10	200.85	286.00	243.42	986.12	-2.4241	0.0053	1.35	1.35
11	238.25	372.53	305.39	812.22	-3.4910	0.0035	1.21	1.21
12	250.94	372.53	311.73	811.29	-3.4962	0.0037	1.25	1.25

Listing 8.38: (Filename.filpara) Estimated unknown reservoir parameters. (Data in columns: window number, begin of window, end of window, middle of window, permeability, skin, wellbore storage, error term 1, error term 2).

0.0000	0.0000
4.4997	243.1347
8.1989	209.8433
26.6526	19.3168
39.4481	239.0652
46.1809	59.0229
48.0012	253.8073
48.1504	233.6565
48.5495	241.3058
94.2110	236.5565
95.1026	243.8297
99.2962	230.6979
102.3934	239.4941
105.3491	237.9757
108.5466	156.3006
111.4495	356.7360
113.3969	258.1390
123.2462	519.9466
124.7856	429.8547
125.4495	363.5774
126.1486	705.6456
128.9486	662.5278
144.7000	463.8307
147.2926	713.9675
149.1928	483.6722
163.1967	508.5632
200.8491	766.6552
201.8451	657.3939
203.1054	727.5730
205.2923	692.8951
238.2490	728.3018
242.0527	736.1094
243.2486	674.2141
244.4510	668.1961
245.8489	1023.1819
250.9382	768.8365
252.3462	628.6244

Listing 8.39: (Filename.outrate) Estimates of unknown flow rates. (time, flow rate)

The estimated reservoir parameters in window 2 and 7 in Listing 8.38 should not be accepted as they differ from the estimated parameters from other windows and their error terms are also higher. The error terms as well as the values of estimated parameters should be compared to each other to identify unacceptable estimates.

The initial estimates of reservoir parameter input file was changed slightly by changing the low and high bounds of the permeability estimates to 100 and 3000 as in Listing 8.40. The results of the estimated reservoir parameters as in Listing 8.41 were better after adjusting the permeability bounds.

```
model
    1
initial pressure
   5360     5000     6500  1
permeability
   500.0    100   3000
skin
   0    -10    100
wellbore storage
   0.01   0.00001    1.0
```

Listing 8.40: (Filename.est) Adjusted initial estimates of unknown reservoir parameters.

```
1      0.00   286.00   143.00   1139.64   -1.4727   0.0136   1.49   1.49
2     26.65   286.00   156.32   1044.33   -2.1103   0.0160   2.36   2.36
3     39.45   286.00   162.72   1055.30   -2.0586   0.0198   2.93   2.93
4     48.55   286.00   167.27   1038.62   -2.1612   0.0191   2.99   2.99
5     94.21   286.00   190.10   1041.46   -2.1425   0.0191   2.98   2.98
6    105.35   286.00   195.67   1030.50   -2.2063   0.0192   2.96   2.96
7    123.25   286.00   204.62   1040.25   -2.1461   0.0182   3.01   3.01
8    144.70   286.00   215.35   1115.89   -1.6183   0.0078   1.13   1.13
9    163.20   286.00   224.60   1121.40   -1.5801   0.0070   1.01   1.01
10   200.85   286.00   243.42   1014.87   -2.2446   0.0102   1.39   1.39
11   238.25   372.53   305.39    818.49   -3.4511   0.0096   1.28   1.28
12   250.94   372.53   311.73    815.88   -3.4663   0.0102   1.32   1.32
```

Listing 8.41: (Filename.outrate) Estimated unknown reservoir parameters. . (Data in columns: window number, begin of window, end of window, middle of window, permeability, skin, wellbore storage, error term 1, error term 2).

An additional experiment was performed where the initial reservoir pressure was set as unknown in the filename.est, with this setting the *Window* algorithm performed processing for two windows of data only and then failed.

Setting the initial reservoir pressure constraint as known will most likely help the *Windows* algorithm to regress to better results. The lower and upper bounds of the estimated reservoir parameters also help the regression routine in the *Window* algorithm to reach more consistent results.

8.5. Summary

The test cases described in this chapter reveal that the algorithms are able to interpret simulated data without difficulty, but real field data present greater difficulties. Adjusting the break points remains a problem in real data, and real permanent gauges records are likely to need manual screening after break point detection. Parameter estimation is sensitive to the initial estimates of the reservoir parameters, particularly the initial reservoir pressure.

Chapter 9

9. Conclusions and Recommendations

9.1. *Wavelet* Algorithm

The *Wavelet* program successfully denoises noisy pressure data. A minimum denoising threshold of three times the noise standard deviation should be used. The noise standard deviation can be estimated automatically using the noise estimation algorithm noise.m. The denoising threshold calculated based on Donoho and Johnstone (1994) should be used when this value is higher than three times the noise standard deviation. The hybrid denoising method defined by Athichanagorn (1999) applies the soft thresholding approach in the flat regions of the pressure data and applies the hard thresholding approach in the vicinity of discontinuities in the signal but the algorithm is not fully robust because the *Wavelet* program may halt execution for certain combinations of wavelet sample spacing and slope detection threshold. The soft thresholding method is more robust and should be used when using the *Wavelet* program is used for multiple combinations of wavelet sample spacing and slope detection threshold.

The *Wavelet* program removes spike outliers successfully but could not remove step outliers. Step outliers have to be removed before running the *Wavelet* program. At present, step outliers need to be selected manually in the program destep.m. Further research effort should be invested to develop algorithm to identify step outliers. Step outliers can be a positive or negative step. Step outliers that stay exactly at the same value can be discriminated automatically but step outliers that are off by even one decimal point can be encountered and are difficult to recognize for automatic removal.

It was found the *Wavelet* program could not accept data with data overlap. Data overlap can be removed using the data overlap removal algorithm overlap.m before feeding the data to the *Wavelet* algorithm.

The *Wavelet* program resamples data at a lower sampling rate. The data can be resampled at an even sampling rate by setting the pressure threshold for data reduction to a very high value so that the data are resampled at the specified time threshold. An evenly sampled data set is needed to apply other signal processing methods such as the Fourier transform.

The number of break points detected by the *Wavelet* program depends on the wavelet sample spacing and the slope detection threshold used. The *Wavelet* program tends to detect false break points when low wavelet sample spacing and low slope detection threshold are used. True break points can be left out by the *Wavelet* program when a high value of wavelet sample spacing and high value of slope detection threshold are used. It is not easy to find the combination of wavelet sample spacing and slope detection threshold setting that detects all the true break points without detecting any false break points. Different pressure transient signals will most likely require a different combination of wavelet sample spacing and slope detection threshold to detect all the true breaks.

It is possible to gather all the detected break points by using a range of combinations of wavelet sample spacing and slope detection threshold and binning them in a histogram. The *Wavelet* program was modified to be able to perform break point detection for multiple combinations of wavelet sample spacing and slope detection threshold. It was found that false break points tend to be detected less frequently than true break points. Further research work could be performed to find out the optimal histogram cutoff level automatically where all the true break points are included with minimum false break points.

False break points may also be screened out with other signal processing methods. The Fourier transform method to screen false break points was investigated and was found to be useful in the flat regions of a transient but not near the beginning of a transient. The proper Fourier discrimination cutoff level should be investigated further.

Further research work could also be performed to determine the combinations of wavelet sample spacing and slope detection thresholds that should be used to generate the break point population that is suitable for the histogram discrimination method.

It was also found that the *Wavelet* program works best within a window of wavelet sample spacing values. At present, the low and high value of this window are found by manual binary search, an automatic search algorithm could be included in the future.

The break points detected by the *Wavelet* program usually did not fall exactly at the beginning of a transient. Adjustment can be made to reduce this error using the straight line approximation method. The error after adjustment will be higher if the data is from a sparse data set or the number of points used to fit the line is higher. The break point adjustment error cannot be reduced further by overlaying two transients and attempting to adjust the datum pressure and time.

Further research effort is needed to improve the present straight line break point correction algorithm as the present algorithm can only correct break points very near the true beginning of a transient.

9.2. *Window* Algorithm

The *Window* program developed by Athichanagorn (1999) requires all the true break points to be specified with no false break points. The program also requires initial estimates of the unknown flow rates. Bad initial estimates of the unknown flow rates may cause the regression algorithm to fail to converge. Unknown flow rates can be estimated using the area under a transient method. A minimum of two known flow rates will be required with at least one of them nonzero. Periods of zero flow provide valuable flow rate information as these help to provide better initial estimates of unknown flow rates and there is no uncertainty in the value of zero.

The ability of the *Window* algorithm to replace abnormal transient improves the quality of the estimates of the reservoir parameters. Missing sections of data can also be reconstructed from the regression fit.

The reservoir model has to be determined before interpreting the transient data using the *Window* program. The initial estimates of the reservoir parameters associated with the specified reservoir model have to be provided.

If available, cumulative production and initial reservoir pressure data can help the *Window* algorithm to constrain the estimated parameters. The estimated parameters from the moving window analysis should be checked and compared to single transient interpretation results to confirm the validity of the estimated values.

A wider window width gives a smoother time series of estimated parameters. Varying the window step for the same window width does not affect the smoothness of the estimates. The estimated flow rates are not sensitive to the window size.

At present, only the initial reservoir pressure or the cumulative production can be used to constrain the regression algorithm. It would be useful to be able to set other parameters as fixed in order to constrain the regression algorithm. For example, the distance to a fault could be known and set as fixed or if it is known that permeability does not vary with pressure for a particular reservoir then the value could be constrained. The ability to constrain certain parameters could be useful for example in analyzing permanent down hole gauge data to check for sudden changes in skin factor.

Currently, the estimated flow rates and results of pressure data replaced by regression fit are not saved in a convenient format. A lot of effort and time are needed to decipher the flow rates and pressure output files from the regression. An algorithm to extract these results into more convenient format would be useful. At present, there are ten reservoir models incorporated into the moving window analysis. Other reservoir models could be incorporated as needed. An automated model recognition algorithm could also be useful.

9.3. Recommendations

Athichanagorn (1999) introduced methods to interpret permanent gauge data. This research investigated additional processing steps such as the initial estimation of unknown flow rates and methods to adjust the break points. Also, procedures to select the true break points using the histogram and Fourier discrimination methods were investigated. After identifying a number of issues, some improvements were implemented in Athichanagorn's (1999) *Wavelet* and *Window* algorithms to make them more robust. Future research effort could be applied usefully as follows:

1) Improve the straight line break time adjustment algorithm.

2) Implement an automatic step outlier removal algorithm.

3) Define the appropriate combinations of wavelet sample spacing and slope detection threshold to obtain a population of break points that includes all the true break points.

4) Define the optimal cutoff frequency for the histogram discrimination method.

5) Define the optimal the cutoff level for the Fourier discrimination method.

6) Investigate other signal processing methods to select true break points and screen false break points.

7) Investigate the constraint of other estimated reservoir parameters such as permeability rather than just the initial reservoir pressure and total production now. Also investigate whether the *Window* algorithm will relax the error criteria when the initial reservoir pressure is set as known.

8) Design an algorithm to extract and present estimated flow rates and computed pressure in a more convenient format.

9) Allow for the possibility to change the reservoir model or fluid properties as the moving window analysis moves forward and extend the *Window* algorithm to be able to process one data file after another.

10) Allow known flow rates to be read from a file rather than via the console keyboard.

11) Develop algorithms to interpret permanent down hole gauge data from multiple wells.

Appendix 1

A1. Program Guide

A1.1. Wavelet, Preinrate, Inrate, Window, Overlap, Noise and Destep Algorithms

This section is intended as a quick guide for the permanent down hole gauge data interpretation programs. Permanent down hole gauge data processing is performed in two main steps, using the *Wavelet* and *Window* programs. An intermediate step between the *Wavelet* and *Window* algorithms is the rate estimation step, performed by the *Inrate* algorithm.

Summaries of input files needed and output files produced during processing are provided here. A list and brief description of the source codes for the *Wavelet*, *Inrate* and *Window* programs are also included.

Matlab programs that remove data overlap (overlap.m, estimate the noise level (noise.m) and remove step outliers (destep.m) are also discussed. These are run before using the *Wavelet* algorithm.

The Preinrate.m Matlab program prepares the output files required by the *Inrate* algorithm. True break points are selected using the histogram and Fourier screening method and adjusted using the straight line intersection method. Known flow rates are input via keyboard console input in this program. The Preinrate.m program allows the user to manually add or delete break points and also manually correct unsatisfactory adjustments made by the straight line intersection break point adjustment algorithm.

A1.2. Overlap.m

The Matlab program *Overlap* requests the user for the filename (without filename extension). A file of type filename.tpr, the data file before *Wavelet* processing is expected. Overlaps in time are removed and the screened data are saved as filename.tpr again. The user console input/output is as follows:

Filename (without filename extension): Testdata

A1.3. Noise.m

The Matlab program *Noise* estimates the noise standard deviation and the denoising threshold suggested by Donoho and Johnstone (1994). The user can either select automatic or manual processing mode. In manual processing mode, the user picks the zone of the data where the standard deviation of the noise will be estimated. In automatic processing mode, the program will break the data into 200 sections of equal width. If these sections are wider than 2 hours, a maximum section width of 2 hours will be used.

First, the noise standard deviation and denoising threshold will be calculated for the section which is the least steep. Second, the noise for each section is extracted and combined and a noise standard deviation value for the combined noise of all the sections is calculated. A second value of average denoising threshold value is calculated from this average noise standard deviation.

Examples of the Matlab console input/output for manual and automatic processing modes are presented as follows:

Filename (without filename extension): Testdata
Automatic or manual zone selection (1=auto, 2=manual): 2
Begining Time window (hours): 73
End Time window (hours): 74

Values for selected section if manual or the least steep section if auto selected
---
selected_sigma = 0.1952
selected_denoising_threshold = 0.6171
Filename (without filename extension): Testdata
Automatic or manual zone selection (1=auto, 2=manual): 1

Values for selected section if manual or the least steep section if auto selected

---

least_steep_section_sigma = 0.1562

least_steep_section_denoising_t = 0.4938

Values are average of all data sections for auto selected

---

average_sigma = 0.1561

average_denoising_threshold = 0.4933

A1.4. Destep.m

The Matlab program *Destep* allows the user to remove step outliers selected manually. There are two options of step outlier removal, global or local. In the global option, a global positive cutoff and a global negative cutoff are requested. These cutoff values are entered using mouse clicks on the plotted data. Data above the positive cutoff value and below the negative cutoff value are considered outliers.

In the local option, two mouse clicks are requested to bracket the data over the time interval where the step outliers have to be removed. Another user input is required to specify if the step outlier in the bounded region is either a step-up outlier or a step-down outlier. The lower pressure value of the two end points of the bounded region will be used as the local positive cutoff if step-up outlier removal is specified. The higher pressure value of the two end points of the bounded region will be used as the local positive cutoff if step-down outlier removal is specified. Matlab console input/output for destep.m are shown as follow:

Filename (with filename extension: .tpr .dat .denoise): stat.tpr

Global or local outliers? (1=global , 2=local): 1

High pressure cutoff (Use mouse to select):

Low pressure cutoff (Use mouse to select):

Filename (with filename extension: .tpr .dat .denoise) :stat.tpr

Global or local outliers? (1=global , 2=local): 2

Left edge of outlier step removal window (Use mouse to select) :

Right edge of outlier step removal window (Use mouse toselect) :

Up Step Outlier

Left edge of outlier step removal window (Use mouse to select) :

Right edge of outlier step removal window (Use mouse toselect) :

Down Step Outlier

Figure A1.1 shows the Matlab graphical user interface for step outlier selection and removal.

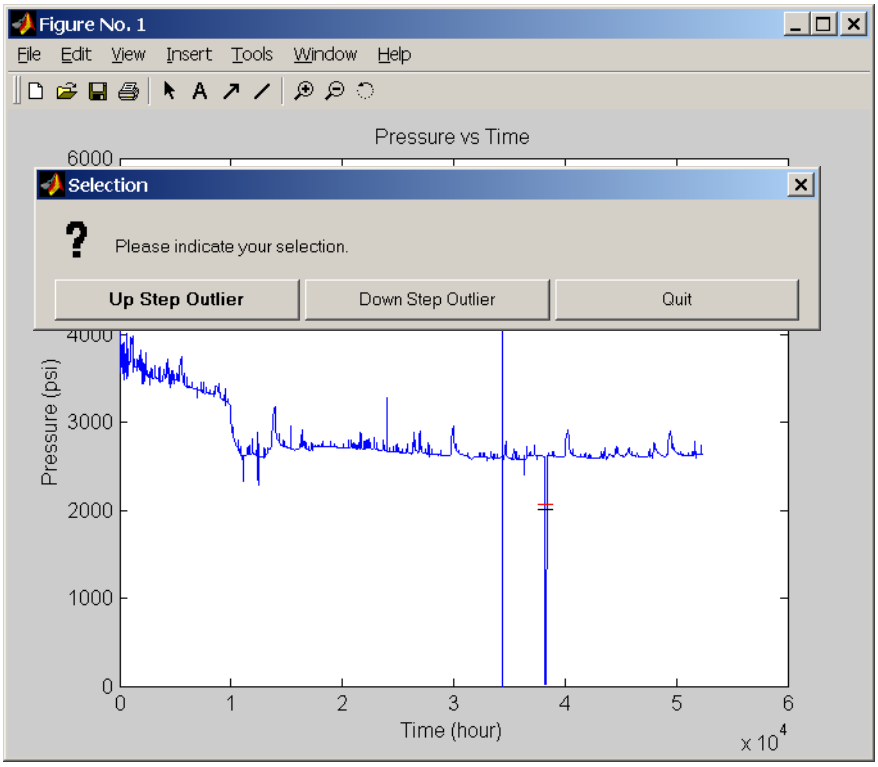


Figure A1.1: Step outlier removal.

A1.5. Wavelet Processing

The compiled executable file and input files have to be in the same directory. The output files will be written to the current directory of the executable and input files. The raw pressure data (from file filename.tpr) is outlier removed, denoised, transient identified and size reduced. Two default spline wavelet filters (from files p3.1 and p3.2) are needed and used during *Wavelet* processing. The resulting pressure data (output file filename.dat) is used for the next (rate estimate) processing step. Break times detected are written to file filename.break if just one run of *Wavelet* processing is selected. The detected break times are written to files filename#.break where # is 1,2,3,… when multiple combinations of wavelet sample spacings and slope detection thresholds are selected. Filename.break or filename#.break are input files to the Preinrate.m program.

A1.5.1. Wavelet Processing (Wavelet.exe) User Interface, Single Run

Single run or multiple runs? (1) Single (2) Multiple
1 ← user keyboard input
Enter input filename
filename ← user keyboard input (omit filename extension)
reading data ...

Sample spacing
Min = 0.000277 hours
Avg = 1.040964 hours
Max = 9.826667 hours (Excluding gaps)

Enter Sampling spacing (hours).
0.000277 ← user keyboard input
Enter slope threshold for flow rate detection (5-30)
10 ← user keyboard input
Denoise? (1) Yes (2) No
1 ← user keyboard input
Thresholding methods (1) hard (2) soft (3) hybrid
3 ← user keyboard input
Enter threshold for denoising (1.5)
1.5 ← user keyboard input
Enter pressure threshold for data reduction (0.5)
0.5 ← user keyboard input
Enter time threshold for data reduction (1.0)
1.0 ← user keyboard input

A1.5.2. Wavelet Processing (Wavelet.exe) User Interface, Multiple Runs

Single run or multiple runs? (1) Single (2) Multiple
2
Enter input filename
filename ← user keyboard input (omit filename extension)
Enter threshold for denoising (1.5)
1.5 ← user keyboard input
Enter low sampling spacing (hours).
0.000277 ← user keyboard input
Enter high sampling spacing (hours).
0.000277 ← user keyboard input
Enter total number of sampling spacings to run.
1 ← user keyboard input
Enter low slope threshold for flow rate detection. (5)
5 ← user keyboard input
Enter high slope threshold for flow rate detection. (50)
50 ← user keyboard input
Enter total number of slope thresholds to run.
46 ← user keyboard input
Enter the starting file number for the runs.
1 ← user keyboard input

A1.5.3. Guideline in Setting Wavelet Processing Parameters

The data time sampling spacing is the time between linear interpolations used to obtain a uniform set of data at regular time intervals. The lowest time sampling value is constrained by the wavelet sample spacing used. Keep to the highest sampling rate (lowest sampling spacing) when preparing a dense data set. A reduced data set can be generated by increasing the sampling spacing or increasing the time threshold parameter.

The slope threshold value is used by the *Wavelet* algorithm to detect break points. If the rate of pressure change per unit time between two data points exceeds the slope threshold, the beginning of a new transient is detected. The detected break points depend on the wavelet sample spacing and slope detection threshold values used. It can be difficult to find the correct combination of wavelet sample spacing and slope detection threshold that detects all the true break points with no false break points. Perform multiple runs of *Wavelet* processing and use the histogram and Fourier screening methods (using program Preinrate.m) to screen out false break points.

Denoising is recommended for pressure data. Hybrid thresholding is recommended. The hard thresholding function is generally used when the data signal contains peak heights and discontinuities but hard thresholding may lead to occasional artifacts that roughen the appearance of the denoised signal. Soft thresholding is recommended when smoothness of the estimates is sought since the soft thresholding function shrinks all the detail signals towards zero.

Hybrid thresholding keeps the benefit of hard thresholding in preserving signal sharpness and the benefit of soft thresholding in smoothing noisy features. Hybrid thresholding applies hard thresholding to the data located at the vicinities of the discontinuities and soft thresholding in the more or less continous data regions.

The following denoising threshold (Donoho and Johnstone, 1994) is suggested as a first pass. If the data looks noisy, increase the denoising threshold. Noise can be represented a as Gaussian signal having the distribution $\boldsymbol{Normal(0,\sigma)}$. A minimum denoising threshold of three times the standard deviation of the noise is suggested.

denoising threshold = $\boldsymbol{\sigma\sqrt{2\,log_{10}(n)}}$

n = sample size of the complete data, $\boldsymbol{\sigma}$ = noise standard deviation

The pressure and time threshold settings control how the data are reduced. A sample will only be saved if the pressure or time changes exceed the pressure or time threshold. It is recommended to generate two data sets: dense and reduced. The original (dense) data set used in the break point detection algorithm gives more accurate results. The reduced data set can be used in the *Window* algorithm to lower the processing time.

If a multiple run is selected, the denoising threshold is set in the same way. The denoising method is set to “soft” by the program because it is more robust. The *Inrate* program requires low and high sampling spacing values and will run the total number of sampling combinations spaced equally between them. Also low and high slope detection threshold values are needed and the *Inrate* program will run the total number of slope detection combinations spaced equally between them.

A1.5.4. Wavelet Program Console Input/Output

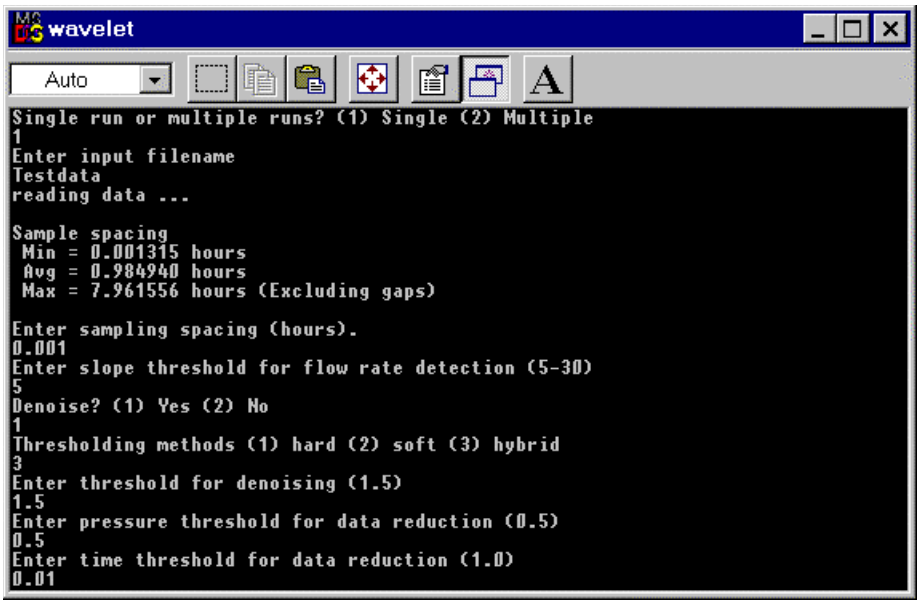


Figure A1.2: *Wavelet* single run console input/output.

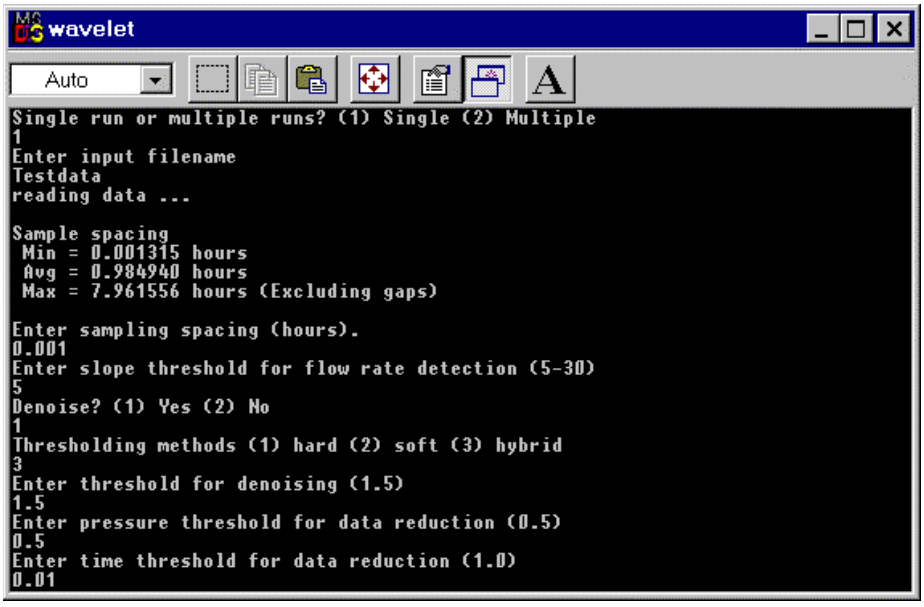


Figure A1.3: *Wavelet* multiple run console input/ouput.

A1.5.5. Wavelet Source Files

The *Wavelet* code is written in C for the DEC UNIX platform.

```
main.c          = main program
init.c          = initialize processing
outlier.c       = outlier removal
transient.c     = transient identification
filter.c        = functions to deal with wavelet transform filters
fast_ddecomp.c  = fast dyadic decomposition and reconstruction
util.c          = utility subroutine
wave1.h         = header file of original Mallat wavelet
                  singularity detection programs
```

Below is the makefile for the DEC UNIX platform.

```
prog: main.o init.o outlier.o transient.o filter.o fast_ddecomp.o
      util.o
      cc -o prog main.o init.o outlier.o transient.o filter.o
           fast_ddecomp.o util.o
main.o: main.c
      cc -c -g main.c
init.o: init.c
      cc -c -g init.c
outlier.o: outlier.c
      cc -c -g outlier.c
transient.o: transient.c
      cc -c -g transient.c
filter.o: filter.c
      cc -c -g filter.c
fast_ddecomp.o: fast_ddecomp.c
      cc -c -g fast_ddecomp.c
util.o: util.c
      cc -c -g util.c
```

********** NOTE : HEADER FILE **********

wave1.h and the source codes above will be needed for compilation.
wave1.h is a header file for the UNIX C.

********** NOTE **********

NOTE: Original Mallat's source code can be obtained via ftp to cs.nyu.edu
The program is located at /pub/wave/software/wave1.tar.Z

********** NOTE **********

A1.5.6. Wavelet Input and Output Source Files

executable: wavelet.exe

input files:	filename.tpr	initial time vs. pressure raw data
	p3.1	spline wavelet filter
	p3.2	spline wavelet filter

output files:	filename.good	good data
	filename.outlier	removed outlier
	* filename.break	breakpoint if single run selected
or	* filename#.break	breakpoint if multiple runs selected
	* filename.dat	reduced version of the good data

* important files that are useful to interpreter

********* WAVELET INPUT FILE FORMAT **********

filename.tpr = pressure data
$time $pressure

p3.1 = default spline wavelet filters

p3.2 = default spline wavelet filters

********* WAVELET OUTPUT FILE FORMAT **********

filename.good = pressure data after eliminating outliers and spikes
$time $pressure

filename.dat = pressure data after data reduction
$time $pressure

filename.outlier = oulier in pressure signal
$time $pressure

filename.break = transient detection for single run of wavelet processing
$time $dummy values

filename#.break = transient detection for multiple runs of wavelet processing
$time $dummy values

A1.6. Preinrate.m

The Matlab program *Preinrate* performs histogram and Fourier screening. It is possible to add or delete break points manually after Fourier screening. The screened break points are then adjusted using the straight line intersection method. As the straight line intersection method is not sufficiently robust, the adjusted break point can be further modified manually. Finally, the known flow rates are entered so that *Preinrate* will produce two output files, filename.postbrk and filename.preinrate. The Matlab console input/output for *Preinrate* is included below as a reference. The six parts of the *Preinrate* program are:

1) Histogram break point screening.

2) Fourier break point screening.

3) After Fourier screening, manual addition or deletion of break points.

4) Break point adjustment using straight line intersection.

5) After break point adjustment manual addition or deletion of break points.

6) Input known flow rates.

A1.6.1. Histogram Break Point Screening

The number of break point files is the number of different files saved when using the *Wavelet* program to detect break points using various combinations of wavelet sample spacing and slope detection threshold. Specify the window of data to display by specifying the start and end time. The detected break points will be binned at the bin width specified. Break points below the histogram cutoff frequency specified will be screened out. The remaining break points can be combined if they are within the specified combination interval.

A1.6.2. Fourier Break Point Screening

The Fourier window width and Fourier cutoff level have to be specified.

A1.6.3. After Fourier Manual Addition or Deletion of Break Point

Break points can be deleted or added by selecting the break points using the mouse.

A1.6.4. Break Point Adjustment Using Straight Line Intersection

Specify the window left and right of the current break point where the adjusted break point can be. Specify both in terms of the number of data points and time, the higher of the two values will be selected by the program. Specify the number of data points used to fit the straight lines.

A1.6.5. After Break Point Adjustment Manual Addition or Deletion of Break Point

Break points can be deleted or added by selecting the break points using the mouse.

A1.6.6. Input Known Flow Rates

Use the mouse to select the transient where rates are known. The known flow rate is specified before a break point. Enter the known flow rate via Matlab console input/output.

```
================================================

1) Histogram Break Point Screening

================================================
Filename (without filename extension) :all
Number of breaktime files :230
Perform or continue break point selection? (Y/N):y
Start time display window :0
End time display window :80
Bin width in hour :0.05
Cutoff frequency:30

tbreak =

   36.3941
   36.4080
   36.7604
   42.8548
```

```
   46.8790
   46.9962
   47.0062
   50.8712
   54.8171
   66.7887
   66.8052
   77.3224

ans = 12

Combine break points within (hour):0.1

ans =

   36.4010
   36.7604
   42.8548
   46.8790
   47.0012
   50.8712
   54.8171
   66.7969
   77.3224

ans =     9

Perform or continue break point selection? (Y/N):n

================================================
2) Fourier Break Point Screening
================================================
Perform Fourier Break Point Screening? (Y/N):y
Fourier window width (hour):2

ans =

  1.0e+012 *

    0.0000    1.6296    0.0000
   -0.0000    0.1762    0.0000
   -0.0000    0.0020    0.0000
   -0.0000    0.0055    0.0000
    0.0000    0.0542    0.0000
   -0.0000    0.0014    0.0000
   -0.0000    0.0924    0.0000
```

```
    0.0000    0.9157    0.0000
   -0.0000    0.0124    0.0000

Fourier cutoff level:3

ans =

   36.4010
   36.7604
   42.8548
   46.8790
   47.0012
   50.8712
   54.8171
   66.7969
   77.3224

ans =  9

=====================================================
3) After Fourier Manual Addition or Deletion of Break Point
=====================================================
Use the mouse to select break point you want to delete or add.

Use the mouse to select other than data points to quit or change scale.

Change scale using MATLAB figure/Tools/Axes Properties.

=====================================================
4) Break Point Adjustment Using Straight Line Intersection
=====================================================
Average time between sample is   0.010089 hours.

Left break point window width (min hour, 0.1) :0.1
Left break point window width (min point, 10) :10
Right break point window width (min hour, 0.1) :0.1
Right break point window width (min point, 10) :10
Number of points defining left break point line (min 3) :4
Number of points defining right break point line (min 3):4

=====================================================
Manual Addition or Deletion of Break Point
While Performing Break Point Adjustment
=====================================================
Use the mouse to select break point you want to delete or add.
```

Use the mouse to select other than data points to quit or change scale.

Change scale using MATLAB figure/Tools/Axes Properties.

Perform break point adjustment again? (Y/N): n
==========
5) After Break Point Adjustment
   Manual Addition or Deletion of Break Point
==========
Use the mouse to select break point you want to delete or add.

Use the mouse to select other than data points to quit or change scale.

Change scale using MATLAB figure/Tools/Axes Properties.

==========
6) Input Known Flow Rates
==========

Mark the interval of known rate using the mouse. :

Change scale if needed, Key in the known rate and press <CR>.

Known flow rate for break time 42.851356 hours :
Known rate (stb/d) :0

Known flow rate for break time 66.793711 hours :
Known rate (stb/d) :1000

Break Points
---

newtbreak =

36.40	2923.29
36.76	2973.90
42.85	2992.45
46.88	2976.99
47.00	2958.10
50.86	2965.96
54.82	2951.82
66.79	2919.07
77.32	2986.55

Known Flow Rates
---

tbreakpreinrate =

36.40	-999.00	0
36.76	-999.00	0
42.85	0	1.00
46.88	-999.00	0
47.00	-999.00	0
50.86	-999.00	0
54.82	-999.00	0
66.79	1000.00	1.00
77.32	-999.00	0
77.57	-999.00	0

Figure A1.4 shows the Matlab graphical user interface for histogram break point selection. Figure A1.5 shows the Matlab graphical user interface for Fourier break point selection. Figure A1.6 shows the Matlab graphical user interface to manually add or delete break points. Figure A1.7 shows the Matlab graphical user interface to input known flow rates.

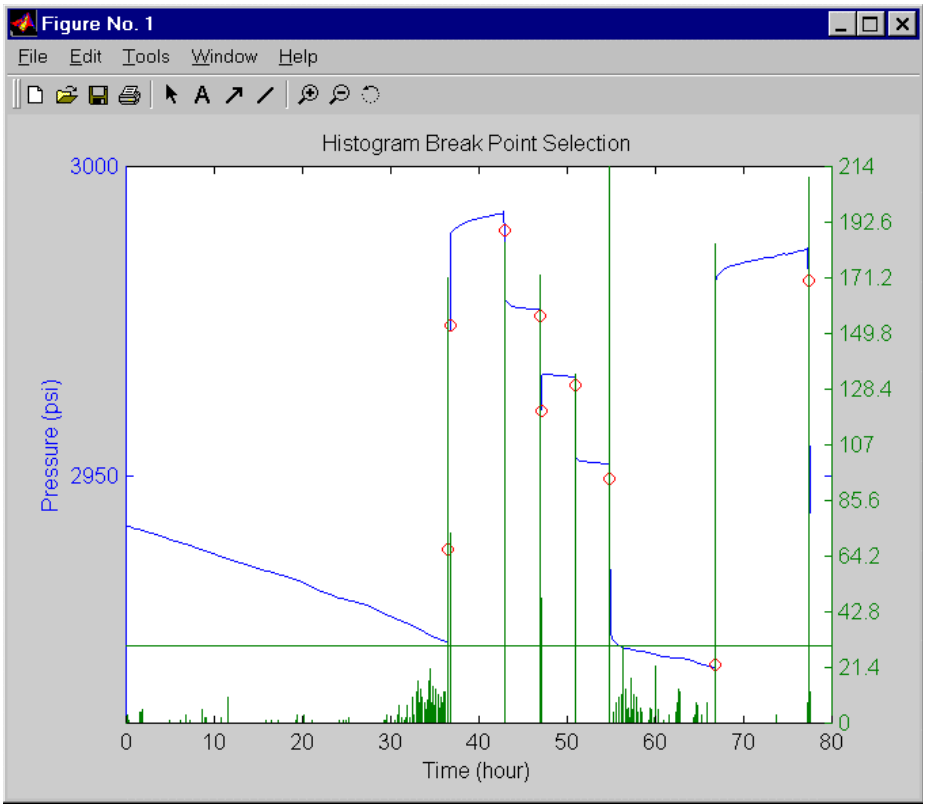


Figure A1.4: Histogram break point selection.

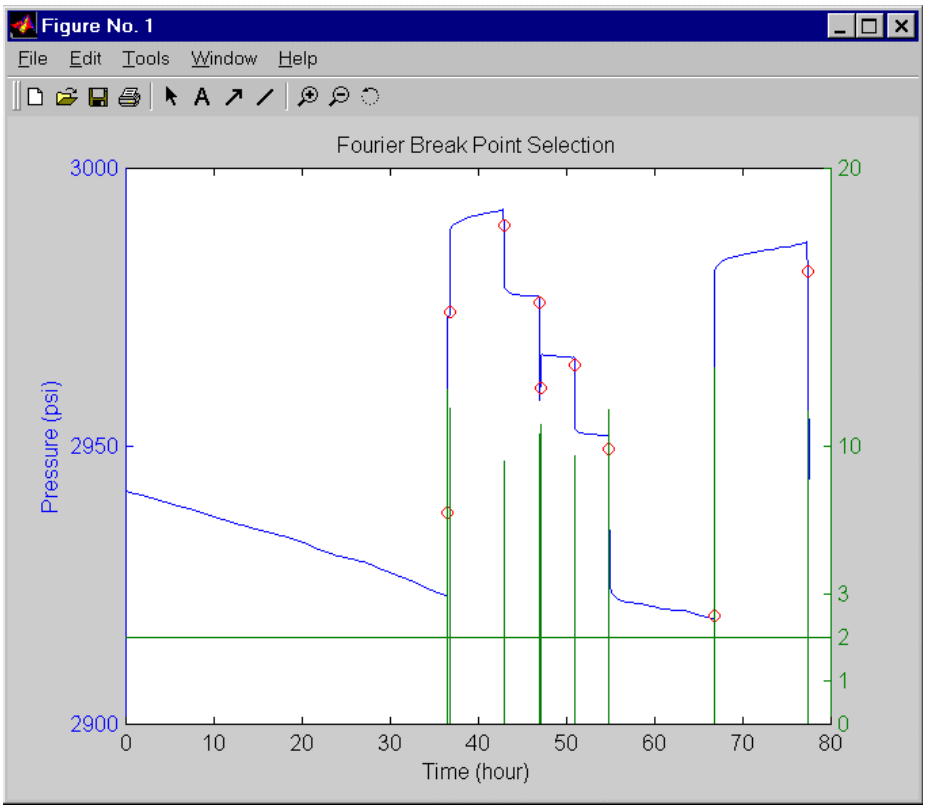


Figure A1.5: Fourier break point selection.

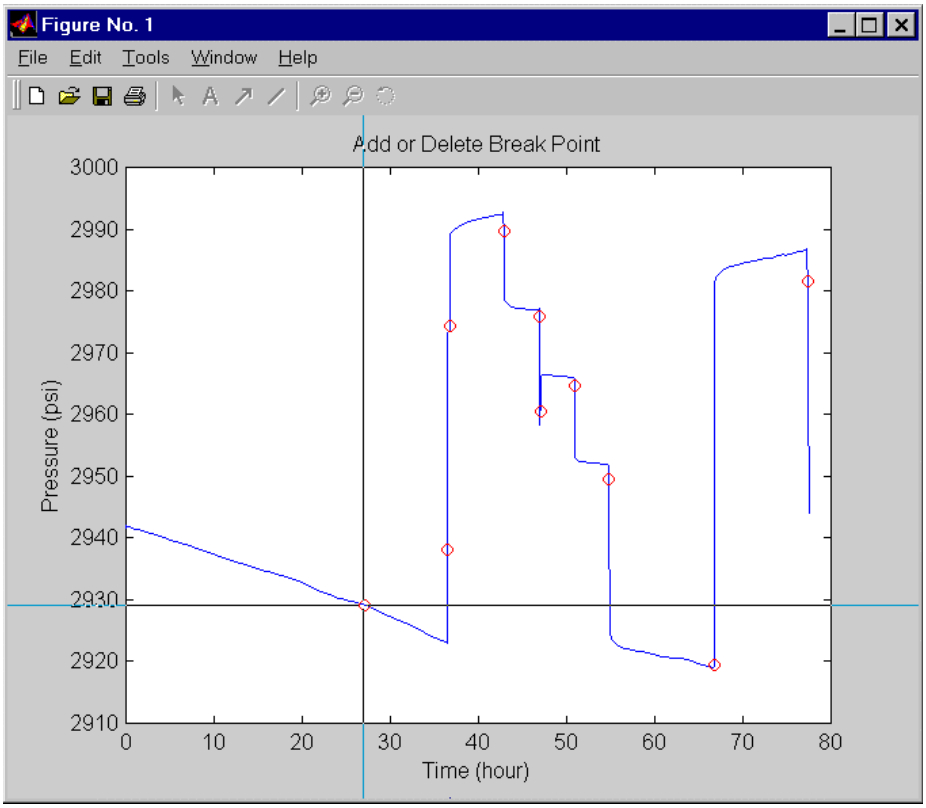


Figure A1.6: Manually add or delete break point.

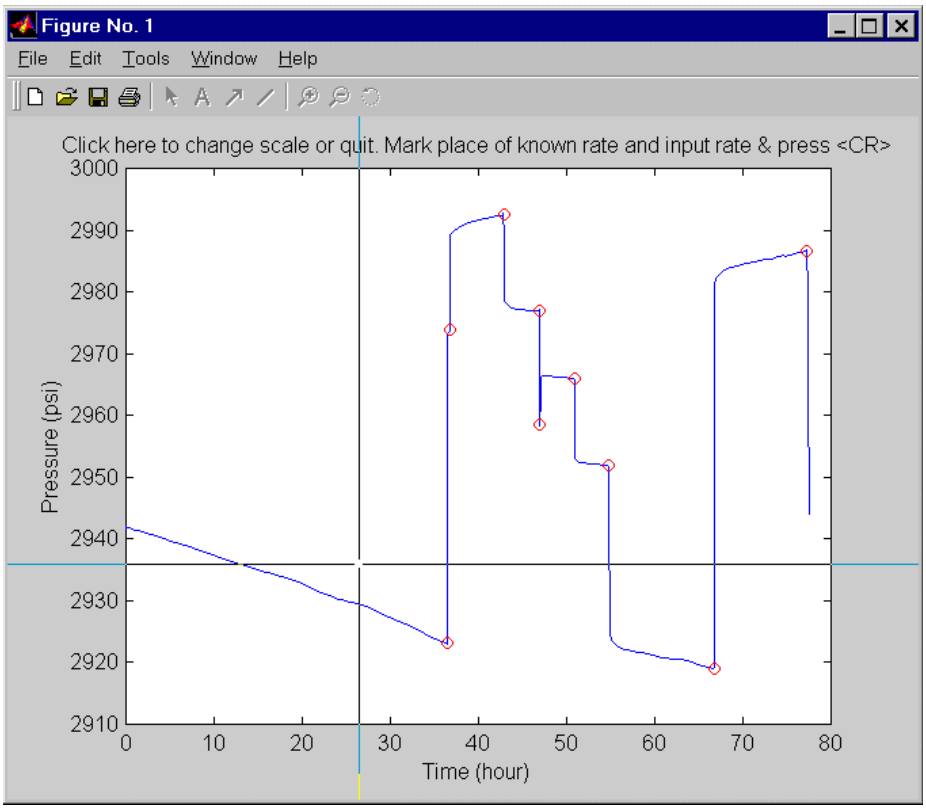


Figure A1.7: Input known flow rate.

A1.7. *Inrate* Estimates Initial Unknown Rate

The compiled executable file and input files have to be in the same directory. The output files will be written to the same (current) directory as the executable and input files. The resulting reduced version of the pressure data (filename.dat) from *Wavelet* processing, the true break points (filename.postbrk) from the break points detected from various combinations of wavelet sample spacings and slope detection thresholds and the known flow rates (filename.preinrate) are needed as input files.

A minimum of two known flow rates are needed, at least one being nonzero. The rate is the constant flow rate before the break time specified. The first break time in filename.preinrate is the break time of the first transient with the rate value specifying the flow rate before the first transient. The last break time in filename.preinrate is the end time value of filename.dat with the rate value being the rate during the last transient.

The file filename.postbrk consists of two columns. The first column is break time and the second column is break pressure. The *Wavelet* algorithm just gives break times that do not fall exactly at the beginning of a transient. The break time in filename.postbrk should be adjusted and also its corresponding break pressure provided.

If filename.preinrate is prepared manually from filename.postbrk, edit the file filename.postbrk by adding the last $break time and adding known/unknown flow rates value in the $rate column and a 1 or 0 flag in the $flag column. The file is then saved as filename.preinrate. Unknown rates can be any value but need to be present for the file reading routine. The values will be ignored in the algorithm. The files filename.postbrk and filename.preinrate can be prepared using the Matlab program *Preinrate.m*.

Unknown rates are estimated and the combined known flow rates and estimated flow rates are written to the output flow rate file (filename.inrate) that will be needed for the *Window* processing step. Upper and lower limits of both known and unknown flow rates are needed to limit the search region of the regression subroutine in *Window* processing. If it is known that there is no fluid injection into the well, enter 2 (for No) so that any estimates of negative rate are set to zero.

A1.7.1 Initial Rate Estimate User Interface (Inrate)

The name of the rate estimate algorithm executable file is Inrate.exe. The user keyboard input in this example are {filename, 2, .15}. The user console input/output as follows:

Filename (exclude filename extension) : filename ← user keyboard input

Any period of fluid injection? (1) Yes (2) No : 2 ← user keyboard input

Accuracy of known flow rate. (i.e. +/-15% enter as .15) : 0.15 ← user keyboard input

A1.7.2. Inrate Program Console Input/Output

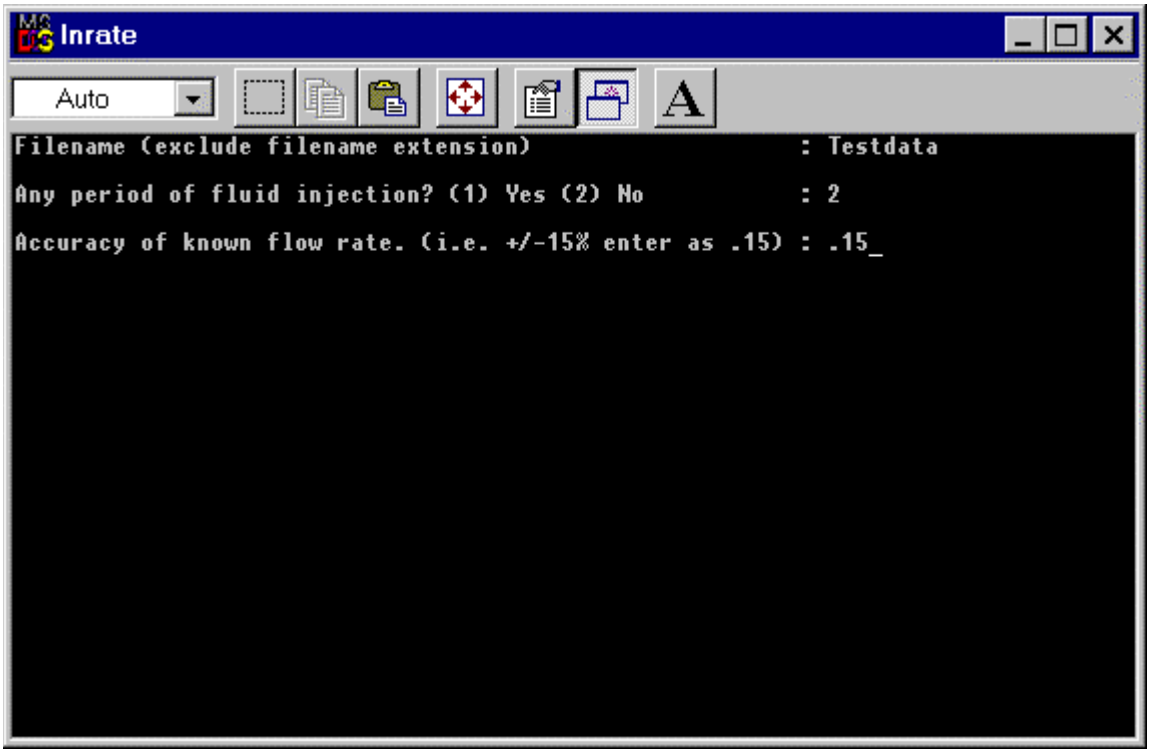


Figure A1.8: *Inrate* console input/output.

A1.7.3 Initial Rate Estimate Source Files

Inrate.cpp is written and compiled using Microsoft Visual C++.
The program has been compiled successfully in both Windows 2000 and Windows 98.

Inrate.cpp = Inrate program

A1.7.4 Initial Rate Estimate Input and Output Files

executable: Inrate.exe

input files: # filename.dat time vs. pressure from *Wavelet*
& filename.postbrk adjusted true break time and break pressure
% filename.preinrate known flow rate data

output file of *Wavelet* processing
& output file of Matlab program *Preinrate.m*

% output file of Matlab program *Preinrate.m*

output files: * filename.inrate known and estimated flow rate

* needed for *Window* processing as input file

******* RATE ESTIMATE INPUT FILE FORMAT *******

filename.dat = pressure data
$time $pressure

filename.postbrk = adjusted true break time and break pressure
$break time $break pressure

filename.preinrate = known flow rate data
$break time $rate $flag (1 known, 0 unknown)

******* RATE ESTIMATE OUTPUT FILE FORMAT *******

filename.inrate = known and estimated flow rate data
$break time $rate $flag (1 known, 0 unknown) $lower limit $upper limit

A1.8. *Window* Processing

The compiled excutable file and input files should be in the same directory. Five input files are needed for the window processing routine.

The pressure data file (filename.dat) from *Wavelet* processing can be used directly. The field combined known and estimated flow rates from the rate estimate algorithm *Inrate* (filename.inrate) will also be needed.

The reservoir model has to be decided before hand and used as an input parameter. Ten different reservoir models are supported. The initial guess of parameters specific to the reservoir model is specified in the input file filename.est

Reservoir properties common to all reservoir models are specified in the input file filename.prop. The window size, window step and starting time are also specified here. The starting time parameter allows a large file to be made into smaller files with different starting times.

Total cumulative flow data, if available, can be included using the input file (filename.prod). If cumulative flow data is not available, this file still needs to be included but without any data.

A1.8.1. Window Processing (Window.exe) User Interface

The name of window processing executable file is Window.exe.

Enter name of data file:
filename ← user keyboard input (omit filename extension)
Select writing mode
(1) = write all computed pressure
(2) = do not write computed pressure
1 or 2 ← user keyboard input

A1.8.2. Window Program Console Input/Output

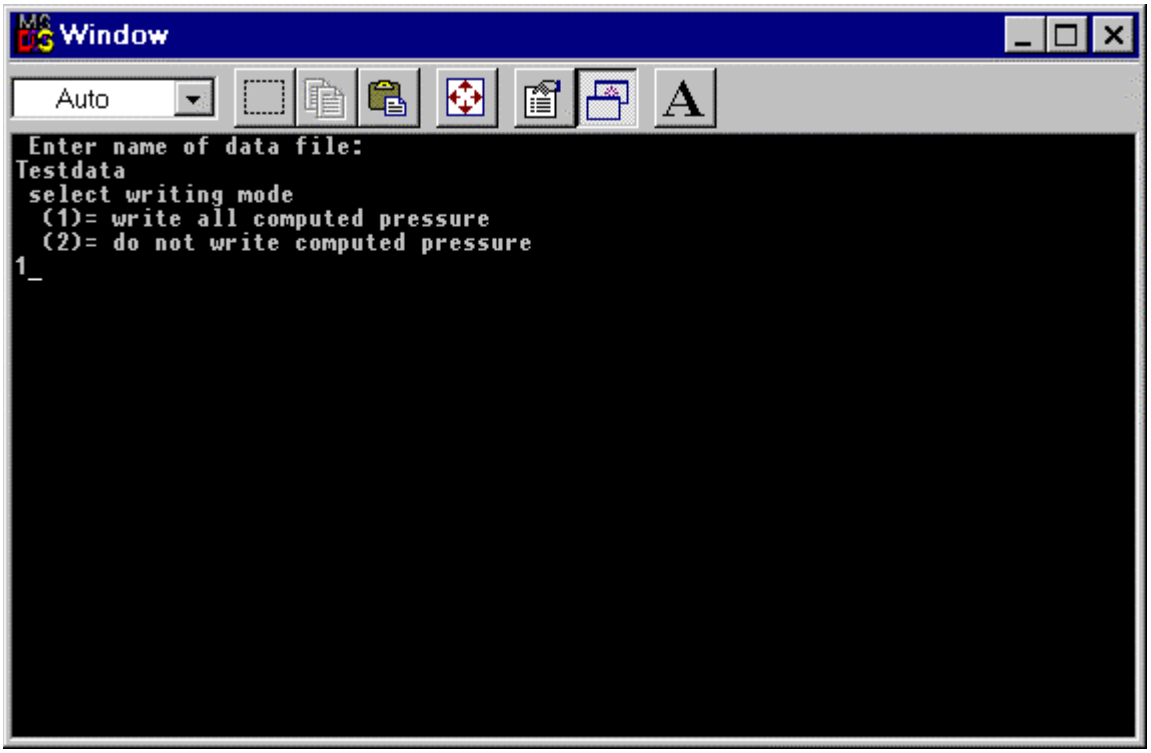


Figure A1.9: *Window* console input/output.

A1.8.3. Guideline for Setting Window Processing Parameters

Window algorithm parameters are set in the input files rather than by user console interface. The following parameters of filename.prop are explained further:

$window length	(hours)
$translation length	(hours)
$starting time	(hours)

Window length is the width of the window of data on which regression is performed. Computed outputs will be assigned (time stamped) at the middle of the window. A longer window length will give a smoother result.

Translation length is the forward movement in time of the window to process the next section of data upon completion of the current window. Output values are spaced apart at translation length hours.

A1.8.4. Window Source Files

The *Window* code is written in Fortran for the DEC UNIX platform.

---

main.f = main program
read.f = read input data
regress.f = nonlinear regression subroutine
deriv.f = derivatives and objective functions needed for regression
stehfest.f = framework for the stehfest algorithm
model.f = compute pressure response and derivatives for nine models
10.f = rectangular boundary model
util.f = utility subroutines

Below are the makefile instructions for the DEC UNIX platform.

```
prog1: main.o read.o util.o regress.o deriv.o stehfest.o model.o 10.o
       f77 -o prog1 main.o read.o util.o regress.o deriv.o stehfest.o
              model.o 10.o
main.o: main.f
       f77 -c main.f
read.o: read.f
       f77 -c read.f
util.o: util.f
```

```
	f77 -c util.f
regress.o: regress.f
	f77 -c regress.f
deriv.o: deriv.f
	f77 -c deriv.f
stehfest.o: stehfest.f
	f77 -c stehfest.f
model.o: model.f
	f77 -c model.f
10.o: 10.f
	f77 -c 10.f
```

A1.8.5. Window Input and Output Files

executable: Window.exe

input files: # filename.dat time vs pressure from wavelet
% filename.inrate known and estimated flow rate data
filename.prop reservoir properties
filename.est initial guess of reservoir parameters
filename.prod cumulative production of oil
(can be an empty file if production is not available)

output file of wavelet processing
% output file of rate estimate processing

output files:* filename.para estimates of model parameter at
each iteration of the current window
filename.filpara final estimate of model parameter of each window
filename.rate estimate of unknown transient flow rate
for each iteration of current window
filename.filrate final estimate of unknown transient
flow rates in each window
* filename.outrate summary of flow history of
both known and unknown
filename.prob record of window processing
fort.1### computed pressure response
before behavioral filtering
fort.2### computed pressure response for section
removed during behavioral filtering
fort.3### computed pressure response that are
kept after behavioral filtering (good part of data)
l1 record of processing

The last three number (###) in these files are the window number.
* important ones which are useful to interpreter

**********WINDOW INPUT FILE FORMAT **********

filename.dat = pressure data
 $time $pressure

filename.inrate = known and estimated flow rate data
 $break time $rate $flag (1 known, 0 unknown) $lower limit $upper limit

filename.prop = reservoir properties
 $porosity
 $height (ft)
 $well radius (ft)
 $viscosity (cp)
 $formation volume factor (rb/stb)
 $compressibility (1/psi)
 $window length (hours)
 $translation length (hours)
 $starting time (hours)

filename.est = initial guess of reservoir parameters
 (the one below is for model 1, Infinite Acting Model)
 $model
 $initial guess pressure $min pressure $max pressure $flag
 $permeability $min $max
 $skin $min $max
 $wellbore storage

filename.prod = cumulative oil production data
 $time $total production

**********WINDOW OUTPUT FILE FORMAT **********

filename.para = estimate of parameter for each iteration
 (the one below is for model 1, Infinite Acting Model)
 $window number $start time $end time $middle time
 $permeability $skin $wellbore storage $pressure
 $error1/number of data $error2/number of data

filename.filpara = estimate of parameter for each window
 (the one below is for model 1, Infinite Acting Model)
 $window number $start time $end time $middle time
 $permeability $skin $wellbore storage $pressure
 $error1/number of data $error2/number of data

filname.rate = estimate of unknown transient flowrates
in current window for each iteration
$window $start time $end time $middle time
$estimate of unknown flowrate, $estimate of unknown flowrate, ...

filname.filrate = final estimate of unknown transient flowrates
in current window
$window $start time $end time $middle time
$estimate of unknown flowrate, $estimate of unknown flowrate, ...

filename.outrate = summary of flow history of both known and unknown
$time $rate

filename.prob = record of window processing
% $window begin time $window end time
$transient $ems1(l) $ems $transient begintime $transient endtime
$length of transient $i-iold $ipsec(1-iqbgin + 1)

NOTE : The following variables are not commented in the source code
and will be checked into. The output file filename.prob is used
by the programmer rather than the interpreter.

ems1(l) , ems , i-iold , ispec(1-igbgin + 1)

fort.1### = computed pressure before behavioral filtering
$time $computed pressure $abs(computed pressure - measured pressure)

fort.2### = computed pressure removed during behavioral filtering
$time $computed pressure $abs(computed pressure - measured pressure)

fort.3### = computed pressure kept after behavioral filtering
$time $computed pressure $abs(computed pressure - measured pressure)

l1 = record of processing if screen output piped to a file

A1.8.6. Reservoir Models Supported

The ten reservoir models supported by *Window* algorithm are listed below:

1. Infinite Acting Model
2. Sealing Fault Model
3. Closed Circular Model
4. Constant Pressure Circular Model
5. Double Porosity Infinite Acting Model
6. Double Porosity Sealing Fault Model
7. Double Porosity Closed Circular Model
8. Double Porosity Constant Pressure Circular Model
9. Parallel Fault Model
10. Rectangular Model

A1.8.7. Example of filename.prop

```
porosity
 0.2
height
 100.00
well radius
 0.35
viscosity
 1.5
formation volume factor
 1.2
compressibility
 5.0e-6
window length (hours)
 999
translation length
 499
starting time
```

Listing A1.1: Example of input file filename.prop.

A1.8.8. Example of filename.est for Supported Reservoir Models

The structure of the input file filename.est for each of the reservoir supported by *Window* algorithm is listed as follows:

model - Infinite Acting Model
    1
initial pressure
    initial guess   min max flag (1 known, 0 unknown)
permeability
    initial guess   min max
skin
    initial guess   min max
well bore storage
    initial guess min   max

---

model - Sealing Fault Model
    2
initial pressure
    initial guess min max flag (1 known, 0 unknown)
permeability
    initial guess min max
skin
    initial guess min max
well bore storage
    initial guess min max
distance to boundary
    initial guess min max

---

model - Closed Circular Model
    3
initial pressure
    initial guess min max flag (1 known, 0 unknown)
permeability
    initial guess min max
skin
    initial guess min max
well bore storage
    initial guess min max
distance to boundary
    initial guess min max

---

model - Constant Pressure Circular Model
    4
initial pressure

initial guess min max flag (1 known, 0 unknown)
permeability
initial guess min max
skin
initial guess min max
well bore storage
initial guess min max
distance to boundary
initial guess min max

---

model - Double Porosity Infinite Acting Model
5
initial pressure
initial guess min max flag (1 known, 0 unknown)
permeability
initial guess min max
skin
initial guess min max
well bore storage
initial guess min max
Omega
initial guess min max
Lambda
initial guess min max

---

model - Double Porosity Sealing Fault Model
6
initial pressure
initial guess min max flag (1 known, 0 unknown)
permeability
initial guess min max
skin
initial guess min max
well bore storage
initial guess min max
Omega
initial guess min max
Lambda
initial guess min max
distance to boundary
initial guess min max

---

model - Double Porosity Closed Circular Model
    7
initial pressure
    initial guess min max flag (1 known, 0 unknown)
permeability
    initial guess min max
skin
    initial guess min max
well bore storage
    initial guess min max
Omega
    initial guess min max
Lambda
    initial guess min max
distance to boundary
    initial guess min max

---

model - Double Porosity Constant Pressure Circular Model
    8
initial pressure
    initial guess min max   flag (1 known, 0 unknown)
permeability
    initial guess min max
skin
    initial guess min max
well bore storage
    initial guess min max
Omega
    initial guess min max
Lambda
    initial guess min max
distance to boundary
    initial guess min max

---

model - Parallel Fault Model 9
initial pressure
initial guess min max flag (1 known, 0 unknown)
permeability
initial guess min max
skin
initial guess min max
well bore storage
initial guess min max
distance to boundary
initial guess min max
distance to boundary
initial guess min max

---

model - Rectangular Model
10
initial pressure
initial guess min max flag (1 known, 0 unknown)
permeability
initial guess min max
skin
initial guess min max
well bore storage
initial guess min max
distance to boundary
initial guess min max
distance to boundary
initial guess min max
distance to boundary
initial guess min max
distance to boundary
initial guess min max

Nomenclature

A = Least square fit matrix

B = Formation volume factor, *RB/STB*

b = Least square fit pressure vector

c_t = Total compressibility, *1/psi*

f_c = Nyquist critical frequency

k = Permeability, *md*

h = Reservoir thickness, *ft*

n = Number of data points

N = Number of data points

p = Pressure, *psi*

p_i = Initial well pressure, *psi*

p_{wf} = Well flowing pressure, *psi*

q = Flow rate, *STB/day*

r_w = Well radius, *ft*

s = Skin factor

t = Time, *hour*

x_0 = Abscissa of least square linear fit

x_1 = Slope of least square linear fit

GREEK LETTERS

Δ = Sampling interval

ϕ = Porosity

λ = Denoising threshold

μ = Viscosity, *cp*

σ = Noise standard deviation

Bibliography

[1] Athichanagorn, S.: *Development of an Interpretation Methodology For Long-Term Pressure Data From Permanent Downhole Gauges*, PhD dissertation, Stanford University (June 1999).

[2] Athichanagorn, S., Horne, R. N., and Kikani, J.: "Processing and Interpretation of Long-term Data from Permanent Downhole Pressure Gauges," paper SPE 56419 presented at the 1999 SPE Annual Technical Conference and Exhibition, Houston, TX, October.

[2] Barua, J., Horne, R. N., Greenstadt, J. L. and Lopez, L.: "Improved Estimation Algorithms for Automated Type-Curve Analysis of Well Tests," SPEFE (March 1988) 186-196.

[4] Donoho, D. L. and Johnstone, I.M.: "Adapting to Unknown Smoothness Via Wavelet Shrinkage," Journal of the American Statistical Association (1994) 90, No. 432, 1200-1224.

[5] Horne, R. N.: *Modern Well Test Analysis, A Computer-Aided Approach*, Petroway, Inc., Palo Alto, 2nd edition (1995).

[6] Miller, R. E.: *Optimization Foundations and Applications*, John Wiley & Sons, Inc. (2000).

[7] Press, W., Teukolsky, S. A., Vetterling, W. T., and Flannery, B.: *Numerical Recipes in C*, Cambridge (1995).

[8] Redfern, D. and Campbell, C.: *The Matlab 5 Handbook*, Springer (1997).

[9] Strang, G.: *Linear Algebra And Its Application*, Harcourt Brace & Company (1986).