University of Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study program/Specialization: Petroleum Engineering/Reservoir Engineering	Spring Semester, 2019 Open
Author: Juan Manuel Cadena Zetina	(Signature of author)
Programme coordinator: Supervisor(s): Anton Shchipanov	
Title of master's thesis: Well monitoring from om-the-fly analysis of data from Permanent Downhole Gauges (PDGs)	
Credits: 30	
Keywords: Pressure Transient Analysis (PTA) Permanent Downhole Gauges (PDGs) Automation	Pages: 40 +enclosure: 30 Stavanger, 15.06.2019

Well monitoring from on-the-Fly analysis of data from Permanent Downhole Gauges (PDGs)

Juan Manuel Cadena Zetina

Summary

The thesis focuses on time-lapse Pressure Transient Analysis (PTA) of data acquire with Permanent Downhole Gauges (PDGs). The objective is to develop and test approaches in automation of time-lapse PTA. Helping in providing well-reservoir parameters such as reservoir flow capacity (Kh) and well skin (S) is analyzed.

A short review of modern PDGs with description of their specifications (i.e. accuracy, resolution) is followed by literature review of recent paper focused on automating time-lapse PTA. This includes machine learning and multi-well interpretation. The main part of the thesis begins with the description of the code developed in combination with analytical solutions used for time-lapse PTA interpretations. The thesis continues with analysis of a synthetic injection well simulated in Saphir. And end up with testing the code with an actual history well production on the Norwegian Continental shelf.

The results of the test on the synthetic well example have shown that from two flow capacity calculation procedures using semi-log and log-log derivative analyses. The last one provides more accurate results on flowing transients. The estimated well skin depends on the flow capacity estimation above. Therefore, is also better estimated with the log-log (derivative) analysis. Both methodologies behave similar in shut-in cases. The inclusion of superposition time when estimating parameters was demonstrated to be a determinant factor.

The tests on the actual well showed that the code provides similar results with Saphir, where the automation routines gives values with an error of 12% and 6% for the semi-log and derivative analysis respectively.

Contents

Acknowledgement

Access to academic license of Saphir software from Kappa Engineering is gratefully acknowledged. The software was used for simulation of synthetic cases and comparison of the interpreted results obtained with the code developed in the thesis.

Free access to the Volve Field data provided by Equinor under a Creative Commons license is acknowledge. This data set was used for interpretation of pressure transient analysis and testing the code developed in the thesis.

1 Objectives and Scope

The thesis objectives are:

- Literature review of today's practice and recent developments in manual, semi- and fully automated PTA of PDG data.
- Development and coding of algorithms for interpretation of well flowing (constant rate) and shut-in periods with estimation of reservoir flow capacity (*kh*) and skin (*S*) accounting for time superposition effects.
- Analysis of possibilities to automate routine procedures of pressure / rate interpretations (time-lapse PTA).
- Development and coding of algorithms for automated time-lapse PTA with possibility of on-the-fly analysis and alarming on well performance changes in real-time.
- Analysis of real well data pressure and rate from a North Sea field.

2 Introduction

The increase in the amount of data provided during well measurements has been receiving a boost from the new technologies and the digitalization, leading to big data sets to be handle by reservoir engineers. This requires from engineers not only knowledge in reservoir engineering, but also the programing skills in helping and developing fast and efficient solutions. To deal with such big data sets.

Being able to perform simple tasks such as filtering, synchronizing data, or structuring before input into commercial software for such big sets of data can be a complicated task without a proper computational approach.

Here, one of the possibilities is automating certain tasks by using complementary software as MATLAB, Python or MS Excel as in the case of the thesis. The reasons for using macros in MS Excel was that most of the computers have access to MS Excel. The second reason is simple input and output via spreadsheets, while the amount of computational power to run simple models in MS

Excel is quite low. The input data can be updated easily via coping the data in the predefined MS Excel sheet, along with the coding based on VBA.

Well test analysis has the objectives to provide information about the well and the reservoir. Which in combination with geological, geophysical and petrophysical data enables to build a reservoir model that simulates the reservoir and can predict the field behavior and fluid recovery. Well tests provide the description of the dynamic conditions, in measurements of the flow (rates) and physical properties (like pressure, temperature, saturations, etc.) when fluids are flowing from the reservoir to the well and vice versa **[1]**.

Well test analysis was renamed in posterior years into **Pressure Transient Analysis (PTA)**. This includes all methodologies and tools developed to analyze well shut-in periods, or pressure build-ups and falloffs, with analyzing the whole well production life with time-lapse analysis.

In the past PTA was traditionally used to characterize well and reservoir parameters from well tests based on shut-in periods and was mainly used in the reservoir management and decision making before reservoir simulation **[14]**.

The development of Permanent Downhole Gauges (PDGs) has brought a massive number of pressure and temperature measurements, providing basis for significant improvement for well and reservoir monitoring.

PDG's provides high quality/high frequency pressure data for the whole well story, but the interpretation of the data becomes challenging related to the lack of interpretation techniques for on-the-fly data analysis dealing with short time periods, between receiving the data and taking of decisions **[14]**.

The comparison of different pressure transients is traditionally done by plotting all the transients and derivatives on the same log-log plot. In practice the comparison between the transients is normally done based on a reference transient chosen, conventionally the first one. The transients are normalized based on the rate **[14]**. This will further elaborated in Chapter 3 of the thesis. Analysis of time-lapse pressure transient can provide descriptions of long-term changes in well reservoir parameters.

The objective of the thesis is to use PTA as an interpretation tool for fast analytical solutions for data interpretation that can be easily updated with the new information provided by the PDG's, classifying in flowing or shut-in periods. Two interpretation methods that use the semi-log and log-log (pressure derivative) analysis to estimate the reservoir flow capacity (kh), permeability (k) and skin factor (s) from time-lapse PTA, are implemented and testes.

The analysis will focus on the effect of duration of a transient period. Estimation of the well-reservoir parameters described above related to the radial flow regime. The results are further compared with the "Saphir" software used for simulating a synthetic well case or an actual field case.

The following questions were addressed:

- How good does the filter of the macro work in classifying the flowing and shut in periods?

- Under which circumstances is better to use the semi-log analysis. Which uses less complex and simple equations than the derivative method.
- In which cases flowing periods can offer similar results to shut-in periods. How different are the calculated well-reservoir parameters for these two types of transients?
- What is the minimum tolerance accepted for determining the radial flow regime for flowing and shut-in periods to provide reliable and accurate information?

The questions also include the importance of superposition time as input data for the flow equations, which accounts for the whole well history before the analyzed transient.

Finally, a discussion of the results obtained and recommendations on using interpretation models for certain circumstances. And keeping in mind the possibility of providing a quick analysis of PDG data for making decisions.

3 Theoretical Background

This chapter will cover the basic theory related to automation, big data analysis and some applications. It will also cover general description, classification of Permanent Downhole Gauges, advantages and disadvantages of the different types.

In reservoir engineering the objective of Pressure Transient Analysis (PTA) is to obtain information about the rock, fluid and well properties including permeability, heterogeneity, reservoir pressure, reserves, wellbore damage, boundaries, fluid contacts, etc. Buildup, drawdown, injectivity, falloff and interference tests are used for this purpose [1] [2] [3].

A pressure disturbance (e.g. with short term production of injection for a well test) followed by pressure monitoring is required to get the information about a well or reservoir using PTA. Pressure transient response may be also created by a temporary change in the production rate. The well must be monitoring during a certain period depending on the well test objectives. It can last a few hours or days for well evaluation, up to month(s) for evaluating distant reservoir areas or well interferences. The pressure is measured in the well where the flow rate has been changed or in another well (interference). In most of the cases the flow rate is measured at surface while the pressure is recorded downhole [1] [2] [3].

In practice, PTA applications are often limited by [3]:

1) Insufficient data collection.
2) Inappropriate application of analysis techniques.
3) Failure to integrate other available or potentially available information.

These limitations make the most complex reservoir harder to analyze, making the acquisition of reliable data an important and valuable task in reservoir engineering.

For a long time, pressure and temperature measurement in wells have been obtained through surveys carried out using wireline interventions. Today, the Permanent Downhole Gauges offer an alternative to measure pressure and temperature in real-time [4].

3.1 Permanent Downhole Gauges

A Permanent downhole Gauge (PDG) is a device installed permanently in a well, to provide a continuous record of pressure, temperature and sometimes also flow rate during production well production. The continuous record provides us rich information about the reservoir and makes PDG data a valuable source for the reservoir analysis [5] [6].

Based on the measuring principle or sensor there are four main categories [7]:

1) Piezoelectric crystal gauges.
2) Optical sensors gauges.

3) Electronic silicon-on-insulator gauges.
4) Capillary tube gauges.

Piezoelectric crystal gauges: these gauges use piezoelectric substances as sensors, being quartz and sapphire as main crystal. These gauges generate a current when pressure is applied, this current is proportional to the pressure applied to it [7].

- Quartz gauges: has been the main primary sensor technology utilized in PDGs, due to providing a very high accuracy (order of 0.002%, full scale) and resolution (0.000006%, full scale) of pressure measurement, they can obtain continuous or intermittent data. Typical maximum temperature of 150 °C (302 °F) and pressures up to 1103.16 bar (16 000 psi). They have an approximate ten years of life expectancy, before they need to be replaced. Additionally, they can operate with other downhole equipment such as Electric submersible Pumps (ESP). However, they are not very suitable above 200 °C and require power from a battery cell to detect and relay measurements [7].
- Sapphire gauges: are similar to quartz features, applications and limitations, but they cannot operate as optimal as quartz in higher temperatures [7].

Optical gauges: these are non-electronic systems that use optical fiber as the primary sensing element or intrinsic sensor, which sends the information via electronic signals. The sensors are made of glass and can withstand high temperatures (175 °C), pressures and vibrations. The glass also prevents interference and noise pollution of the signal. Optical sensors require no power supply for measurement taking and relay, they don't cause additional wellbore restriction and are easy to maintain, plus they offer various configurations to allow a wide range of applications like: downhole and multiphase flow monitoring (rates of oil, gas and water are get distinctly, in zonal completions commingling and individual productivity are easily determined), distributed temperature sensing (identify leaking in casing or tubing, obstructions, and thief zones), and real time seismic imaging. However, its main limitation is the cost [7].

Electronic Silicon-On-Insulator (SOI) gauges are piezo-resistive transducers that convert pressure into a change in resistance. The strain of the applied pressure is measured across an active resistive bridge while the temperature is measured from a secondary of the main bridge. SOI are flexible and relatively economical. The measure is in moderate temperature and pressure ranges (up to 125 °C). Additionally, they can be used in the vibration and artificial lift monitoring (good measurement in vibration caused by natural flow or artificial lift systems), zonal monitoring array and multipoint sensing (several gauges can be arranged on the same array), coal bed methane. It is limited by temperature and pressure, since it is not suitable for these conditions [7].

Capillary tube or permanent pressure gauges are robust mechanical systems that acts on a piston or sleeve that in turns acts on hydraulic fluid to control the line, transmitting the downhole pressure to surface via a standard hydraulic control-line. As there are no electronic or electrical components, this pressure gauge system has great applications in harsher environments and is significantly cheaper. A floating piston mechanism ensures the systems compensates or temperatures effects.

They are suitable for HPHT wells, gas lift and chemical injection capability, but it does not allow simultaneous measurements besides temperature or pressure [7].

In the **Table 3-1** is shown a summary of all the different PDGs sensors and their performance depending on which characteristic, we are planning to monitor. The scale goes from 1 (not suitable) to 5 (most suitable). In **Table 3-1** it is easy to see that the optical sensors cover more of the necessities, follow by the piezoelectric crystals, the only problem for optical sensors is their high cost.

Table 3-1 *Comparative table of PDGs applications [7].*

Reservoir Monitoring	**Optical sensors**	**Piezoelectric crystals**	**Electronic silicon-On-Insulator (SOI)**	**Capillary tube**
Zonal isolation monitoring	5	5	5	3
Pressure Transient Analysis	5	5	3	1
Distributed temperature sensing	5	1	1	1
Multipoint sensing	5	3	4	1
Seismic imaging and monitoring	5	1	1	1
Water and steam breakthrough	5	1	1	1
Other Measurements				
HPHT	5 (175*)	4 (150*)	1 (125*)	5
Artificial Lift Monitoring	2	2	5	3
Production profiling	5	1	1	1
Well startup monitoring	5	1	1	1
Downhole flow measurement	5	1	1	1

* Temperature in °C.

As seen in these categories each of them as they can be used for additional measurements, and for different conditions. This increases the benefits for installing PDGs in wells by collecting the exact reservoir pressure with representative long shut-in period. Since the first day of completion, saving additionally rig time and money as it will not interrupt any on-going drilling activities. Another benefit is identification of completion failures such as leaking. PDGs also provides a real time well

response or monitor the downhole flowing conditions [4]. However, the characteristics of PDG data make the interpretation challenging.

First unlike conventional well testing where flow rate is always carefully controlled, the flow rate recorded by PDGs is subject to operational variations during production. The continuously variable flow rate history makes the pressure and temperature signal more difficult to de-convolve [6].

Secondly, the PDG data are often very noisy. The noise comes from the operational variations that occur in the well and should be treated as an inherent property of PDG data. The noise may hide the true reservoir response and makes it harder for us to recover the true reservoir model. Specifically, the noise in flow rate data brings difficulty for breakpoint identification, which is needed to divide the whole set of PDG data into separate transients. With noise in the flow rate data, it can be challenging to detect which is an actual rate change event and which is noise [6].

The large volume of data is another problem. Modern PDGs can record data at a frequency as high as once per second. This means that millions of data are stored in a PDG record after months of production. People would never want to attempt manually poring over data of such high volume [6].

PDGs were initially deployed for well monitoring and pressure transient analysis. But as explain before the advances in technologies, materials, multipoint tools have allowed the measurement of different properties. This also allowing new uses for the PDG data obtained, well communication, position of wells (injectors/producers) in waterflooding, flow rate reconstruction, for naming some of them.

3.1.1 Applications of Permanent Downhole Gauge data and automated analysis

In Waterflood performance, the most important data in any injection project are production and injection rates, the PDGs offer a continuous source of information, which can be used for optimizing the oil recovery by changing the injection patterns, location of injectors, well priorities in operations, recompletions of wells and targeting infill drilling [7].

Jahangiri et al (2014) proposed a method called Top-Down Waterflood (TDWF), which was applied in one filed in the North Sea in late 2012 and early 2013. This method evaluates the effectiveness of water injection efficiency in the reservoir, the value of injection water (VoiW) and the quantity of the relative connectivity of the injector/producer in the early life of the waterflood prior to significant water breakthrough. The process relies on good estimates of daily production an injection rate data [8].

In their work it was show that the two parameters that have a significant impact in water flooding are the maximum number of injectors that can be connected to a producer, and the distance radius around a producer within which an injector will be allowed to influence that producer [8].

Jahangiri et al (2014) also address the connectivity of the well in their work, which main idea was to identify the important connections through the time, and compared the results in the model TDWF with surveillance techniques, such as tracers, streamline models, and 4D seismic [8].

Figure 3-1 shows an example of their work. Well A04 is the injector well, while the other wells are producers. The figures contain the frequency of occurrence of statistically important connection, from seven periods of time. An important connection was defined as connectivity between the injector (A04) and the producers greater than 20% of the injector flow. Each spoke represents the connection between the injector and one of the producers. Finally, the colored line shows the magnitude of the connection [8]. In Figure 2.1 it is observed that wells A01 and A02 where the most important connections in 2006 and 2007, followed by A03 and A09. However, in 2008 when the well A10 became online, this importance change, being now well A10 the most important connection until the end of the experiment.

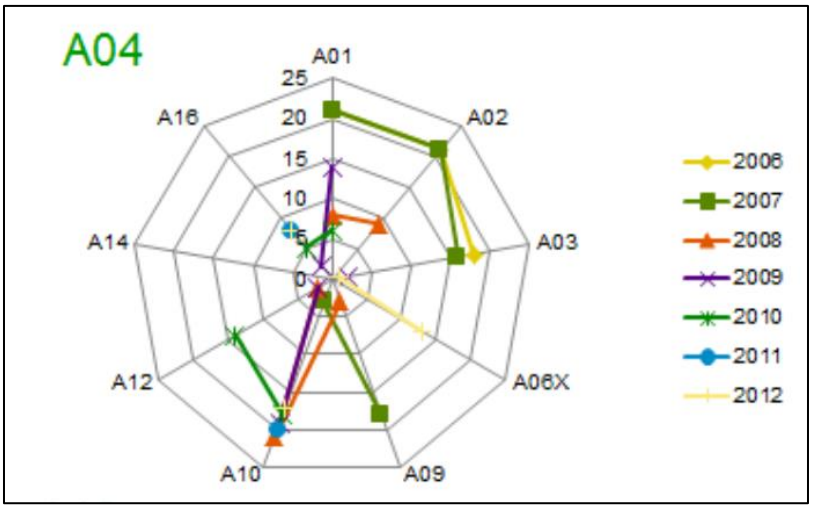

Figure 3-1 *Important connections for injector A04 for different years in the reservoir history [8].*

Tian and Horne (2016) also address the connectivity in waterflooding. In their work, data from PDGs was used to build a reservoir scale network based on the connectivity and perform reservoir analysis without referring to a reservoir simulation model that obliged to make assumptions about geology [9].

In this work they analyzed different scenarios of connectivity, while refining their model, between the different scenarios, they tested their connectivity model with synthetic and real field cases. The results showed consistency with the tracer test and the reservoir geology, but also works as a rough model of the reservoir [9].

Conventionally the estimation of reservoir pressure and some other dynamic reservoir properties are obtained through Pressure Transient Analysis. Pressure management is fundamental element of reservoir performance and is one of the variables to consider in field development strategy. This challenge has been addressed by using conventional techniques of PTA but doing it in real time (automatically), using the live data from PDGs. Automation and real time monitoring tools enable

proactive identification of problems, fewer interventions required, improved well integrity and maximized production for ultimate recovery.

Transient pressure responses to flow rate changes are modeled by solving the relevant partial differential equations analytically. These analytical models characterize the well and reservoir in terms of parameters such as permeability, skin, wellbore storage, type and distance to reservoir boundaries, initial pressure [10].

Pressure Transient Analysis has two parts [10]:

- Model identification: In this step, the reservoir flow model is identified using diagnostics plots and prior information about reservoir and well.
- Parameter estimation: the identified analytical model is matched to the measured pressure and flow rate data, through estimation of parameters. Conventional parameter estimation techniques use regression methods to match the analytical models to the field data.

Recently there have been some attempts to apply machine-learning techniques for PDG data analysis. Machine learning is an important tool for analyzing large sets of data as the one provided by PDGs. Fundamentally the goal of machine learning is to learn the patterns behind PDG data (variables), where the patterns contain the relation of implicitly of the reservoir [6]. Some applications of machine learning in the Pressure Transient Analysis are pressure history reconstruction, flow rate/temperature substitution as some examples.

As stated, before incomplete flow rate history is a common phenomenon in PDG measurements, Tian and Horne (2015) proposed that the missing flow rates could be estimated from the available pressure data. Their model was tested with synthetic and real data and showed promising performance. In comparison with analytical solutions the machine learning provides an effective alternative, this is due to machine learning doesn't require geological assumptions of the reservoir model [5].

Figure 3-2 shows an example of this work. After calibrating the machine-learning model, they give a partial information of the pressure and the flow rate of a new well, after the model prediction (red line) was done, they compared the result with the complete (true) data. In this figure, it is observed that the prediction offers an accurate reconstruction of the flow rate.

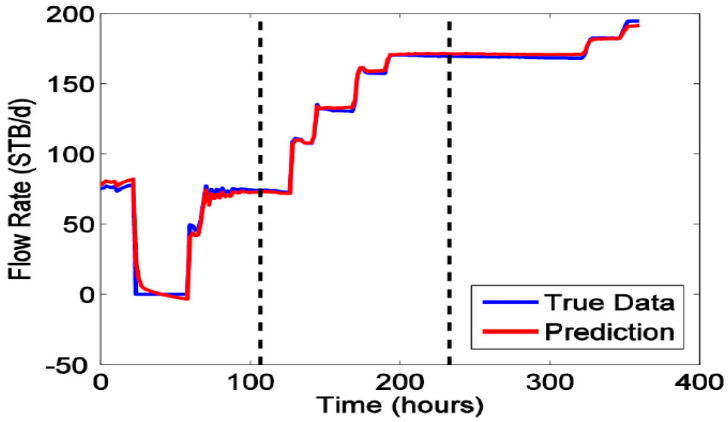

Figure 3-2 *Machine learning result of the reconstructed flow rate using the pressure [5].*

Tian and Horne (2015) in their second work, since machine learning contains the patterns between variables implicitly, can be used as a transformation between forward model and inverse model is easier than in conventional ways, which allows to model pressure from flow rate [6].

In **Figure 3-3**, there are some examples of their results. The graph on the left shows a comparison between the reconstruction of the temperature curve obtained from the model of machine learning (red line) and the pressure data, the curve calculated has the same form than the original data, and is very accurate at the beginning, but presents some inaccuracies after the 600 hours mark. The graph from the right shows the inverse case a reconstruction of the pressure data from the machine-learning model (red line) inverted and the temperature data, as in the case before the curve present the form of the data, with high accuracy at the beginning and some inaccuracies after the 650 hours.

Even if this offers a good alternative for reconstructing both sets of data when flow rate is not available or to have a second opinion to compared with the flow rate results. The only requirement needed for machine learning method is to have at least one proper set of data complete during the training process.

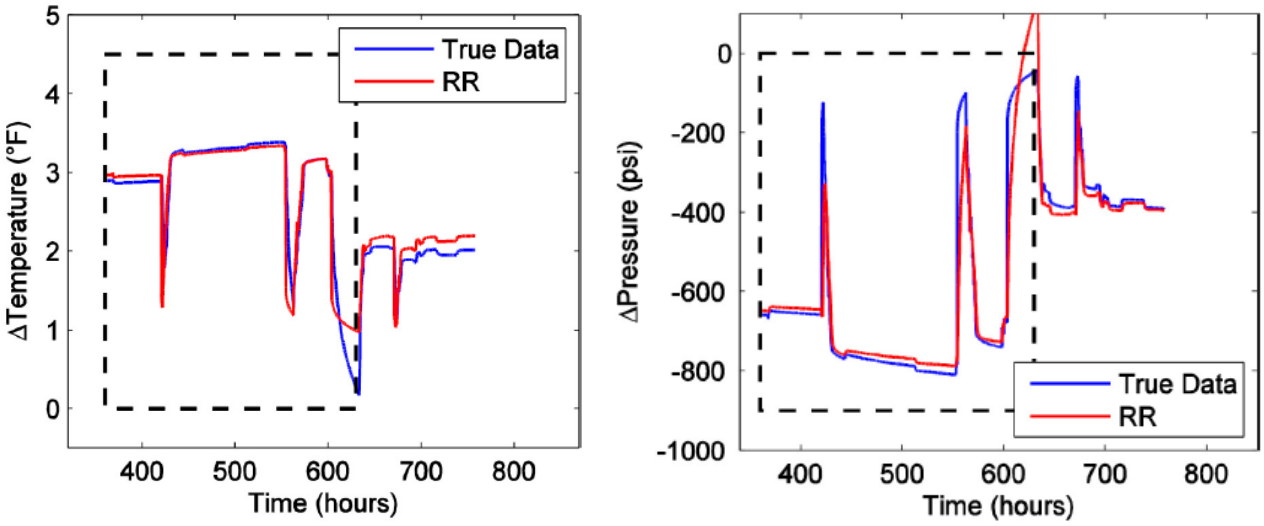

Figure 3-3 *Left graph shows machine learning using ridge regression (RR) to model temperature from pressure data. Right graph shows machine learning using the ridge regression inverted to model pressure from temperature data[6].*

A second way to approach the pressure problem was proposed in the same work by using the temperature as a substitute for flow rate to model the pressure. Since, temperature have been measured by PDGs since their initial installation, in this case the machine learning was trained to find a pattern between the temperature and pressure for predictions, this model was tested by comparing the results obtained with the flow rate and pressure model. The results obtained were fair and presented some limitations. This can be due to the pressure and the temperature having physical independent properties that couldn't be modeled property [6].

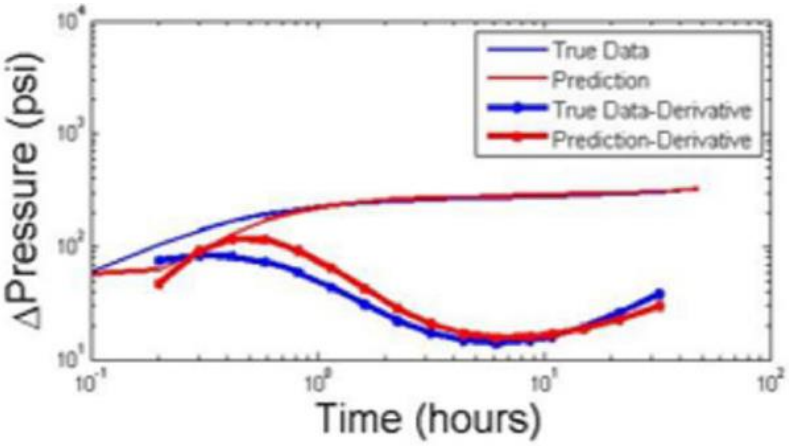

Figure 3-4 *Machine learning results using temperature as flow rate substitute [6].*

Figure 3-4 shows one of their machine learning results when using temperature as a substitute of the flow rate when contracting the pressure and derivative pressure curves, for realizing Pressure Transient Analysis. The pressure curve after 1-hour is very accurate, while before the 1-hour, mark has some differences, but this is not a problem since is related to storage effect. The pressure derivative by the other hand is more inaccurate than the true data derivative suing the flow rate data and the differential equations but offers an alternative when the flow rate data is missing.

Virtual flow metering was addressed in a paper by Bello (2014), multiphase technology solutions have enabled the petroleum industry to improve their production performance. However, a multiphase flow in wells is quite complex as reservoir types and fluid composition varies. Multiphase flow meter has been used for this purpose, of continuous metering of produced hydrocarbon.

Virtual metering has started to become an alternative to measure three phase flow rates by using machine learning to generate patterns, which are compared with the historical well flow rates data to evaluate their match and update the model parameters [11].

4 Methodology

According to the objectives of the thesis described above, algorithms for time-lapse Pressure Transient Analysis (further time-lapse PTA) were developed and implemented. These was done in macros in MS Excel based on literature review. This Chapter describes the background, equations and logic used in the analysis, as well as input data. The code implementing the macros is given in Appendix A. The Chapter ends up by testing the algorithms on a synthetic case of injection into a vertical well simulated with software Saphir from Kappa Eng.

4.1 Input data synthetic example

The Excel file contains three sheets. The first one is called “Main”, which has the command button for interacting with the macro. It contains the number of wells to work and the rock and fluid properties of the well. The second sheet is “Pressure” here is added the time and pressure data. Finally, the last sheet is called “Rate” in which the end time and rate test are added.

A synthetic case of one-well injection of water into an infinite saline aquifer (single-phase flow) was used for testing purposes. The simulated well was assumed to have an induced fracture causing a negative skin factor. The case contains simulated pressure data for more than 6000 hours (about 8 months) as response to a sequence of injection and shut-in periods specified. The well and reservoir properties used are:

a) Initial pressure: 2000 psia.
b) Formation volume factor: 1 RB/STB.
c) Viscosity: 1 cp.
d) Well radius: 0.2 ft.
e) Porosity: 30%
f) Compressibility of the rock: $1x10^{-5}$ $psia^{-1}$.
g) Thickness: 200 ft.

4.2 Explanation of the main parts of the code and formulas used

The main subroutine is “Workflow”, in **Figure 4-1** can be seen the order in which it called the rest of the subroutines. The first subroutine called is “Prepare”. This subroutine creates the layout for the input data depending on the number of wells the user wants to analyze. “Initialize” reads the initial properties of the rock and fluids on the well. “Generate” reads the input data of pressure, time, and rate and identifies the liquid rate segments, its’ initial and ending time, it proposed names to the different transient, main and secondary, i.e. “Injection 4-3”, is the fourth injection period, with its’

third different injection rate. The criteria used for establishing the transient is production (positive rate), Fall-off (rates equal to 0) and injection (negative rate).

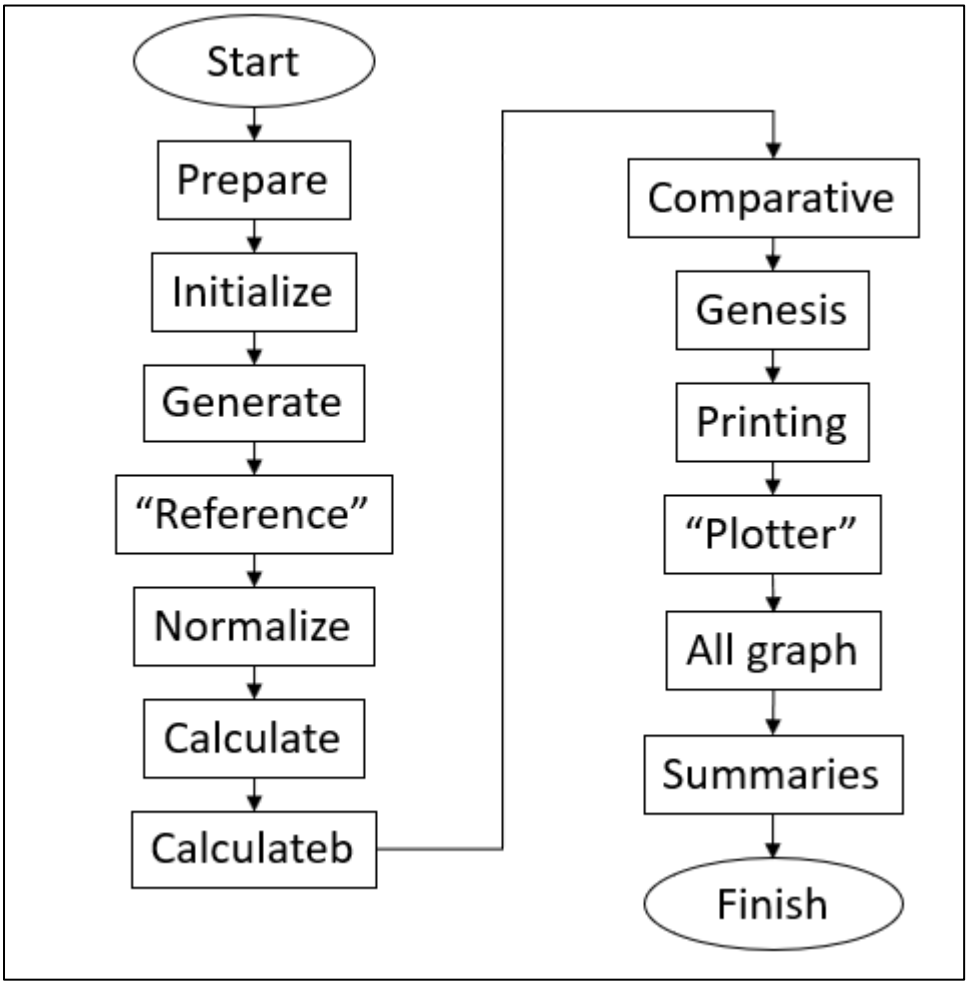

Figure 4-1 *Workflow of the main subroutines in the macro developed.*

"Generates" also perform some calculations. The **Equation 4-1** calculate the time for each segment *dt*, in which T_{is} is the time in which the segment starts and *T* is the time registered.

$$dt = \mathrm{T} - T_{\mathrm{is}} \ldots \quad \textit{Equation 4-1}$$

Then the pressure of the segment and pressure with initial pressure are calculated with **Equation 4-2**, **Equation 4-3** respectively, in which P_{is} is the initial pressure of the transient, and P_i is the initial pressure at the time 0 of the test, *P* is the pressure registered.

$$\mathrm{dps} = |\mathrm{P} - \mathrm{P_{is}}| \ldots \quad \textit{Equation 4-2}$$
$$\mathrm{dp_{@Pi}} = |\mathrm{P} - \mathrm{P_i}| \ldots \quad \textit{Equation 4-3}$$

The next step in **Figure 4-1** is showing the User-form "Reference", **Figure 4-2**. The User-Form allows to select the flowing or shut in periods. The reference transient to use for normalization and the transients you want to compare with can be chosen. Once this is done, the user-form is closed, and the workflow continues.

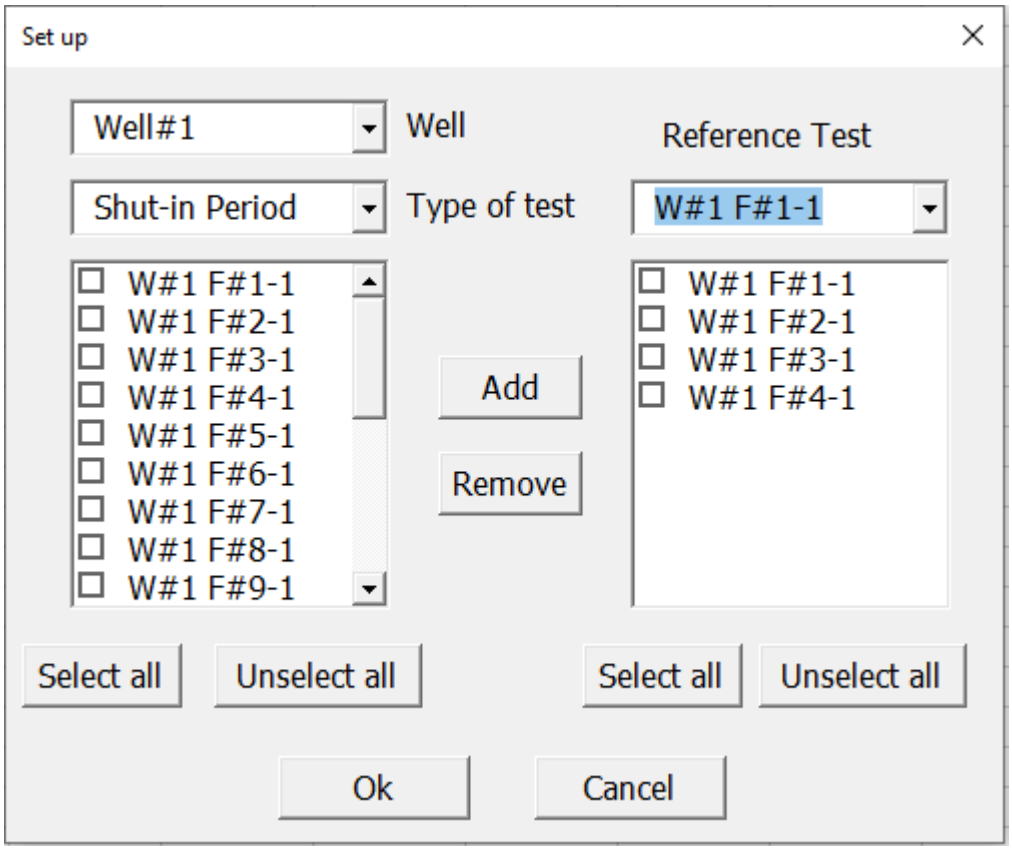

Figure 4-2 *Reference User-Form generated by Macros.*

The next subroutine to run is called "Normalize" this subroutine normalizes the pressure for superposition, derivative and adjusted for non-normalized calculations to mimic the results in the Kappa software "Saphir" the equations used for normalizing injection and production are **Equation 4-4**, **Equation 4-5** and **Equation 4-6**:

$$dp_{np} = \left|\frac{Q_{ref}}{Q_i} * \mathrm{dp}_{@\mathrm{Pi}}\right| \ldots \quad \textit{Equation 4-4}$$

$$dp_{nd} = \left|\frac{Q_{ref}}{Q_i - Q_{i-1}} * dps\right| \ldots \quad \textit{Equation 4-5}$$

$$dp_r = \left|\frac{Q_i}{Q_{ref}} * dp_{nd}\right| \ldots \quad \textit{Equation 4-6}$$

The variables dp_{np} is the pressure normalized for the superposition time reported in the transients, dp_{nd} is the pressure normalized for the derivative and dp_r is the pressure re-scale for the transient that is reported with the derivative results. Q_{ref} is the reference rate to normalize the rest of the values, by default is the first injection/production. Q_i is the rate of the transient analyzed. Q_{i-1} is the rate of the previous transient, all the calculations are obtained in absolute value due to injection having a negative rate.

The equation for normalization for the fall-off tests is similar to **Equation 4-4** with the difference that it should use the rates before the shut-in period and dps. By default, Q_{ref} is the previous transient before the first shut in period. Changing Equation 4-4 into **Equation 4-7**:

$$dp_{np} = \left|\frac{Q_{ref}}{Q_{i-1}} * \mathrm{dps}\right| \ldots \quad \textit{Equation 4-7}$$

In Fall-off transients, there is used only one equation for normalization. The next subroutine called is “Calculate”. “Calculate” has the cycles depending on the number of transients analyzed. For each transient, it will call the superposition function, **Equation 4-8**:

$$Ts = \frac{1}{Q_n}\sum_{i}^{n}(Q_i - Q_{i-1}) * \log(\mathrm{T} - T_i) \ldots \quad \textit{Equation 4-8}$$

In which Q_n is the rate of the segment analyzed. Q_i and Q_{i-1} are the rates for the previous transient before the transient analyzed. T is the time referred and T_i is the end time of the previous transient. For calculating the superposition for Fall-off, the rate should be the previous one prior to the shut-in period.

The next subroutine is “Calculateb”. This calculates the Pressure derivative, and establish its’ left and right side with **Equation 4-9, Equation 4-10** respectively. The first point for the left derivative is set up by default as the smallest of the logarithmic cycle, i.e. if the smaller value is 0.65 the smallest value of the cycle is 0.1.

$$\ln\left(\frac{Ts_i}{T_L}\right) > w \ldots \quad \textit{Equation 4-9}$$

$$\ln\left(\frac{T_R}{Ts_i}\right) > w \ldots \quad \textit{Equation 4-10}$$

TL and TR are not necessarily the points previous or subsequent to the analyzed point. This depends on the smoother, by default is the smoother is set in 0.1. The subroutine “Calculateb” will call the function derivative. The derivative, [13] is calculated with the **Equation 4-11** and then is normalized with **Equation 4-12**.

$$\left(\frac{d\Delta p}{d[\ln(\Delta t)]}\right)_j = \frac{(\Delta P_j - \Delta P_L)}{\ln\left(\frac{\Delta T_j}{\Delta T_L}\right)} \frac{\ln\left(\frac{\Delta T_R}{\Delta T_j}\right)}{\ln\left(\frac{\Delta T_R}{\Delta T_L}\right)} + \frac{(\Delta P_R - \Delta P_j)}{\ln\left(\frac{\Delta T_R}{\Delta T_j}\right)} \frac{\ln\left(\frac{\Delta T_j}{\Delta T_L}\right)}{\ln\left(\frac{\Delta T_R}{\Delta T_L}\right)} \ldots \quad \textit{Equation 4-11}$$

$$d\Delta p_n = \frac{Q_{ref}}{Q_{i-1}} * \left(\frac{d\Delta p}{d[\ln(\Delta t)]}\right)_j \ldots \quad \textit{Equation 4-12}$$

The next subroutine is "Comparative", this will manage the cycles for calculating the permeability and skin. First, it will call the subroutine "Limits", which oversees establishing the right and left limit of the radial flow (horizontal line). This function uses the slope value of the derivative to detect the beginning of the radial flow. "Limits" uses ***Equation 4-13***. Where, p'_i is the current pressure derivative point and p'_{i-1} is the previous pressure derivative point.

$$tol\ error(\%) = \left|\frac{p'_i - p'_{i-1}}{p'_{i-1}}\right| * 100 \ldots \quad \textit{Equation 4-13}$$

The condition for calculating the derivative is by finding a slope with less than 0.1% (value by default) tolerance. In case it doesn't find a point in the derivative slopes that fulfill the condition, the subroutine will automatically increase the error by 0.1, and start looking again until a solution is found. Once the value has been found, the program will establish it as the beginning of the radial flow and will check all the subsequent values that are within the tolerance range. This is to define the end of the radial flow.

The next step for "Comparative" is to call "Propertyks". This calculates the permeability and skin factor of the transient in semi-log and derivative method. The equations used for the semi-log analysis for injection and production are **Equation 4-14** and **Equation 4-15:**

$$m = \frac{162.5683 * q * Bo * \mu o}{kh} \ldots \quad \textit{Equation 4-14}$$

$$S = 1.151\left(\frac{P_i - P_{1hr}}{m} - \log\left(\frac{k}{\varphi * \mu o * Ct * rw2}\right) + 3.2275\right) \ldots \quad \textit{Equation 4-15}$$

For the Fall-off the **Equation 4-15** is changed for the **Equation 4-16**:

$$S = 1.151\left(\frac{P_{1hr} - P_{wf,s}}{m} - \log\left(\frac{t}{t+1}\right) - \log\left(\frac{k}{\varphi * \mu o * Ct * rw2}\right) + 3.2275\right) \ldots \quad \textit{Equation 4-16}$$

For calculating the permeability and the skin with the derivative method, the following equations are used:

$$m' = \frac{70.6 * q * Bo * \mu o}{kh} \ldots \quad \textit{Equation 4-17}$$

$$S = 1.151\left(\frac{dpd * k * h}{162.5683 * q * Bo * \mu o} - \log\left(\frac{t}{t+1}\right) - \log\left(\frac{k}{\varphi * \mu o * C_t * rw^2}\right) + 3.2275\right) \ldots \quad \textit{Equation 4-18}$$

In **Equation 4-16**, and **Equation 4-18** for the Fall-off, the term $\log\left(\frac{t}{t+1}\right)$ has been replaced for the superposition time 10^{Ts}, dpd is the value of the constant derivative calculated in log-log plot.

After the calculations are done the next step in **Figure 4-1** is calling the subroutine "Genesis". This only creates a new sheet in excel. Then the subroutine "Printing" is called, which prints the results for superposition, derivative, pressure, for normalized and non-normalized cases. Then it shows one final User-Form called "Plotter", **Figure 4-3**. This User-Form plots the pressure and pressure derivative of the transients previously selected.

Inside the "Plotter" User-form there is a matrix in charge of storing the limits of the radial flow. It includes one graphic subroutine called "Gengraph" which plots the Pressure and Pressure Derivative graph in the User-Form.

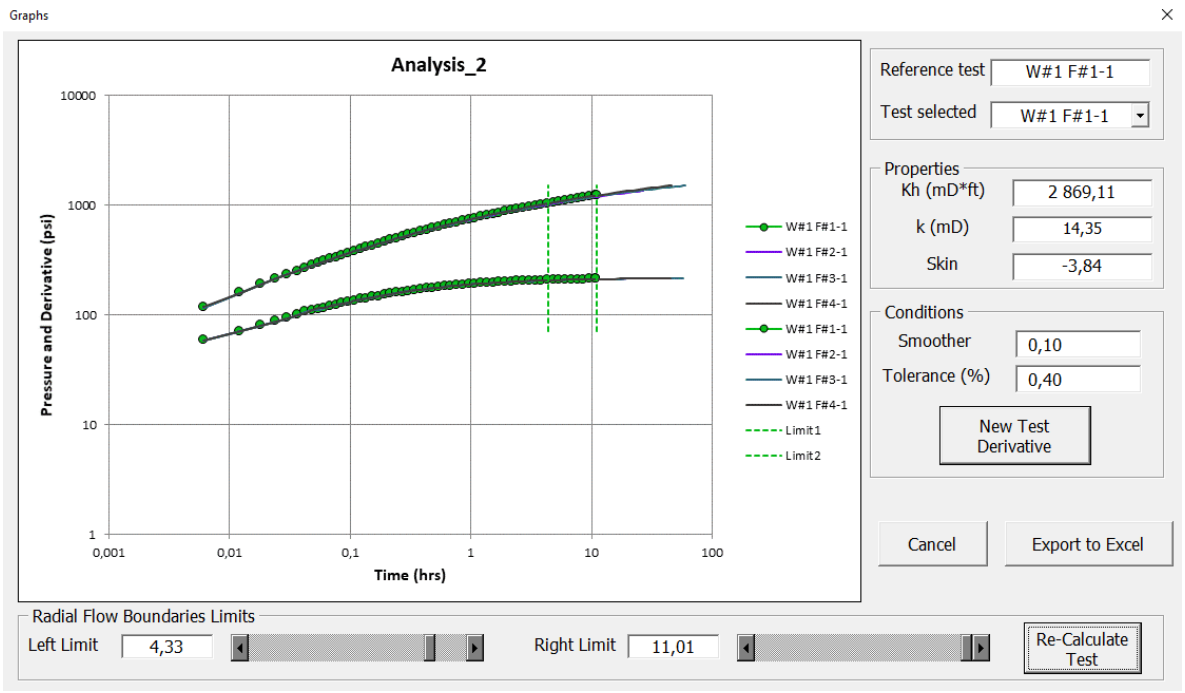

Figure 4-3 *Plotter User-form generated by macros.*

The user can interact with the left and right bar to readjust the boundaries of the radial flow and the click the button "Recalculate Test". This will run the subroutine "Propertyks" described above.

Here the smoother and the tolerance can be changed. The code will call again "Calulateb", "Limits", "Propertyks" and "Printing", to update the results with the new conditions proposed by the user.

Once the analysis is performed, the next subroutine is "Allgraph". This subroutine is used for creating the pressure and pressure derivative graph and the superposition time in separate graph

sheets. Finally, it will run the "Summaries" subroutine which creates a new sheet in which it prints the permeability, skin factor for the semi-log and derivative method, as well as the smoother used.

Some results are shown in **Figure 4-4**, **Figure 4-5**, **Figure 4-6** and **Figure 4-7**. These figures compare the values obtained with Kappa software "Saphir" (solid lines) and the program in excel (dots), for pressure, pressure derivative and superposition time, for some injection and fall-off transients.

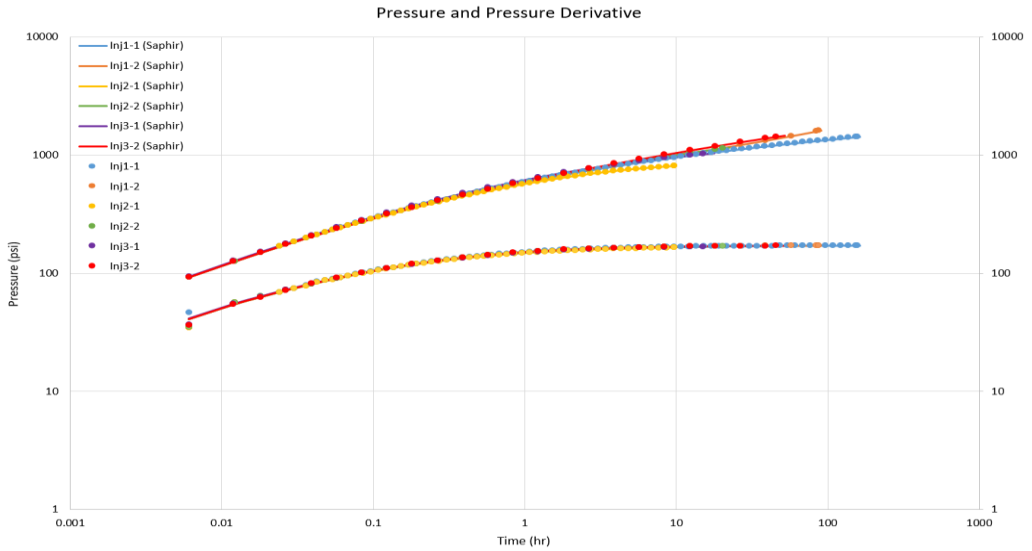

Figure 4-4 *Comparison between Saphir (solid line) and the program (dots) for Pressure and Pressure Derivative for some injection transients.*

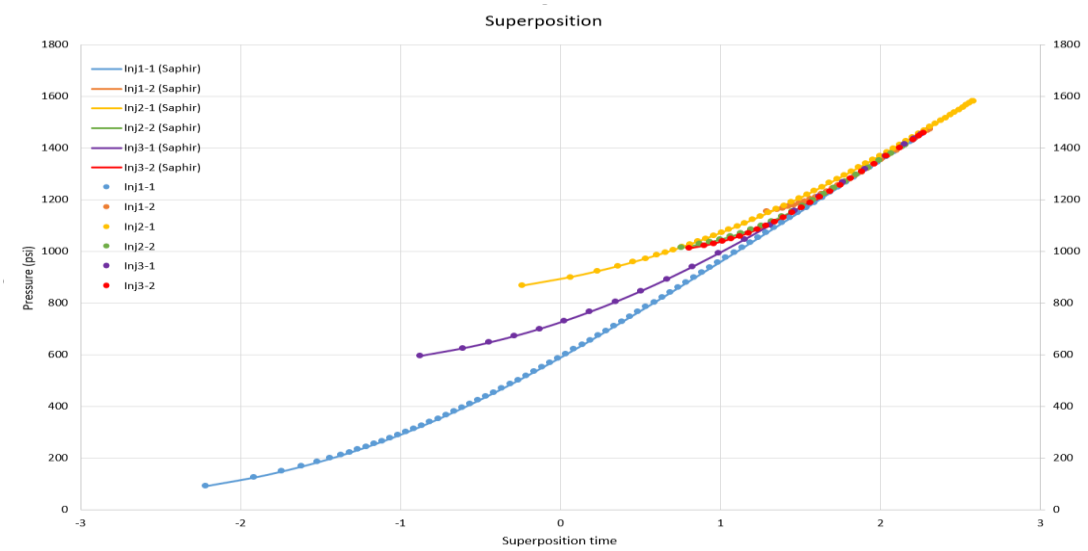

Figure 4-5 *Comparison between Saphir (solid line) and the program (dots) for superposition time for some injection transients..*

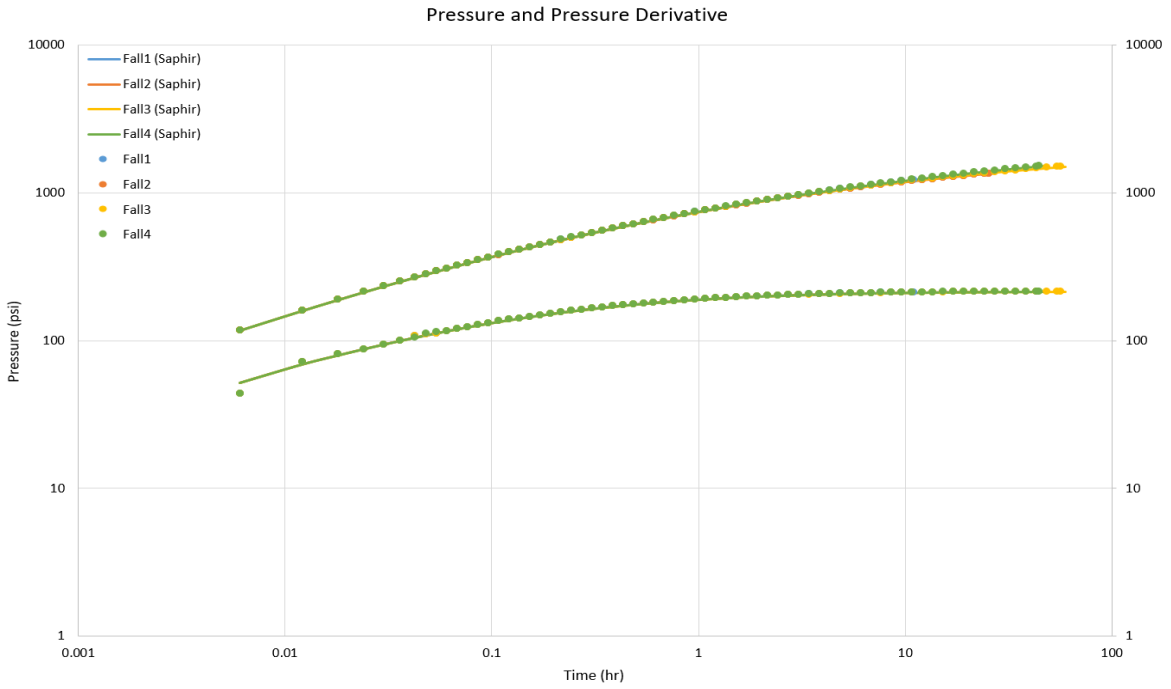

Figure 4-6 *Comparison between Saphir (solid line) and the program (dots) for pressure and Pressure Derivative for some Fall-off transients.*

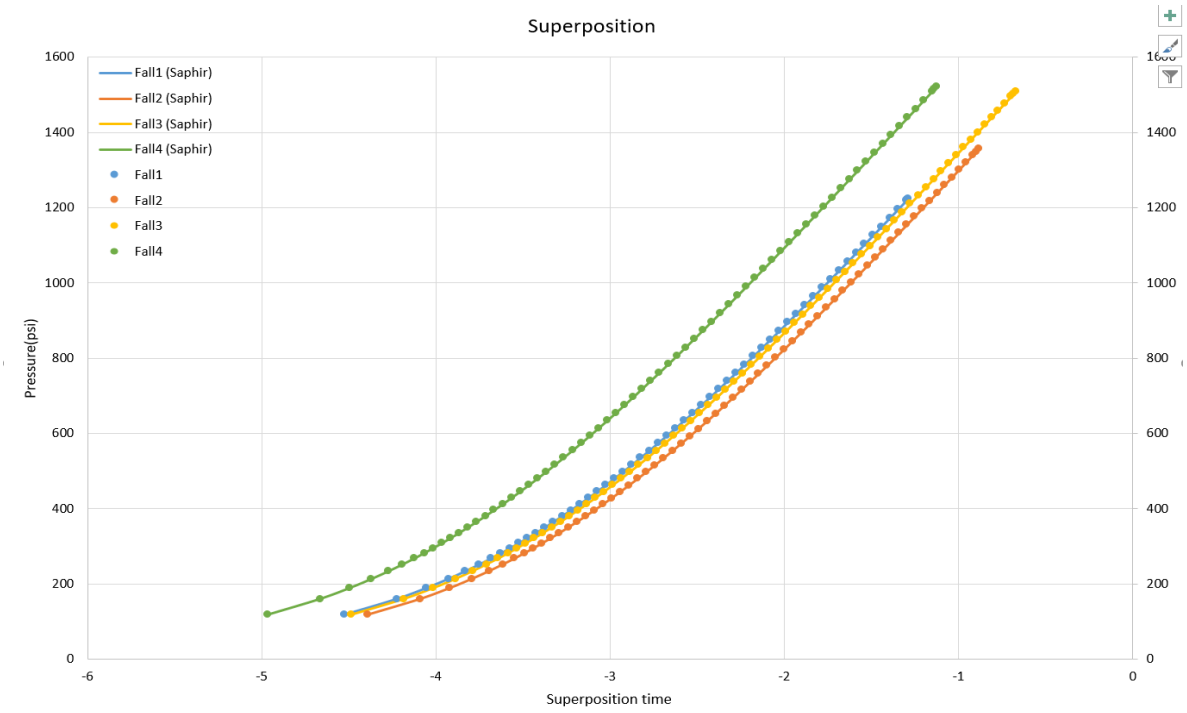

Figure 4-7 *Comparison between Saphir (solid line) and the program (dots) for superposition time for some Fall-off transients.*

Has seen in the **Figure 4-4**, **Figure 4-5**, **Figure 4-6** and **Figure 4-7** the values calculated are slightly higher than the Kappa software "Saphir". For pressure derivative figures, the first value calculated is the one presenting the highest error. This is due to "Saphir" choosing a different arbitrary number for starting the derivative calculation than the proposed code. Due to the first point value will normally be inside the wellbore storage effect, this point is not considered in the area of interest (the radial flow).

The **Table 4-1** shows the results obtained in the macro the Kappa software "Saphir". It is seen that the results obtained are very similar. The derivative method offers better approximation than the semi-log analysis.

Table 4-1 *Comparison of Permeability and Skin Factor*

	Saphir	**Semi-log**	**Derivative**
Flow capacity (mD*ft)	2780	2796	2783
Permeability (mD)	13.90	14.04	14.03
Skin Factor	-3.93	-3.89	-3.90

5 Synthetic Case further analysis

This chapter will analyze some transients from the synthetic case to test the algorithm. The transients were evaluated for comparison with the "Saphir" software. The semi-log and log-log (derivative pressure) methods are compared to know under which conditions is better to use one or another. The injection and Fall-off periods were analyzed separately.

5.1 Injection analysis

The **Figure 5-1** shows the history plot simulated. Here, can be seen the injection periods used in the analysis, covering different durations and times in the history.

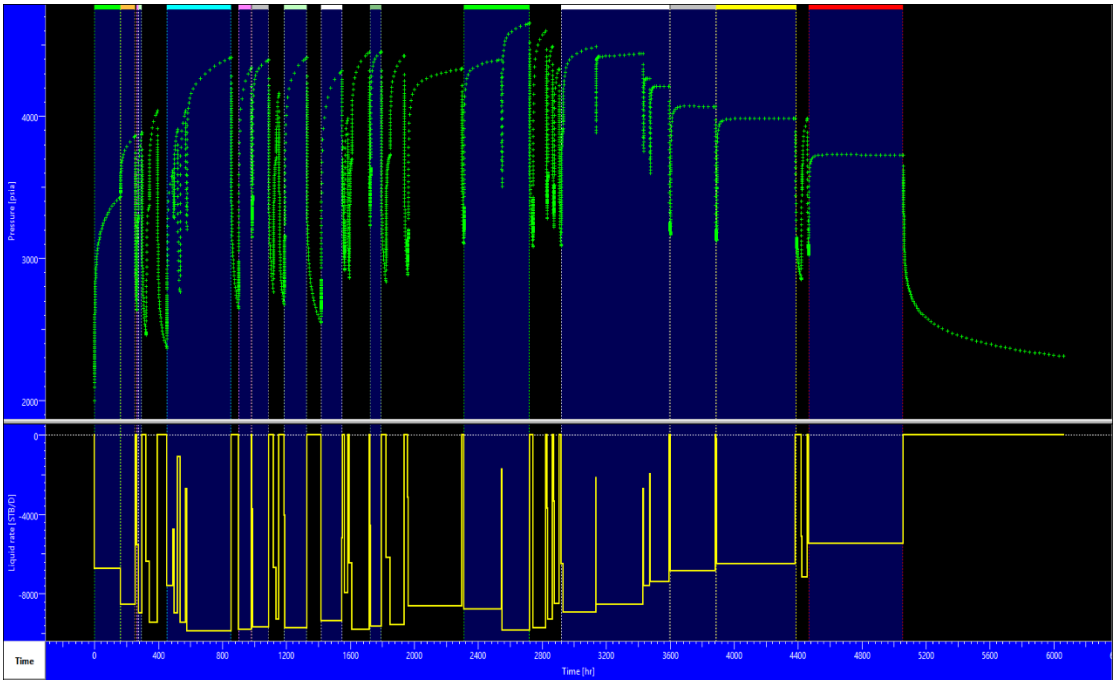

Figure 5-1 *History plot (pressure and rate) for the whole simulation period.*

The **Figure 5-2** shows the pressure and pressure derivative for the 2 to 10 hours transients chosen. The reason for not choosing transients below 2 hours is due to not having an established radial flow, and due to be related to wellbore storage effect.

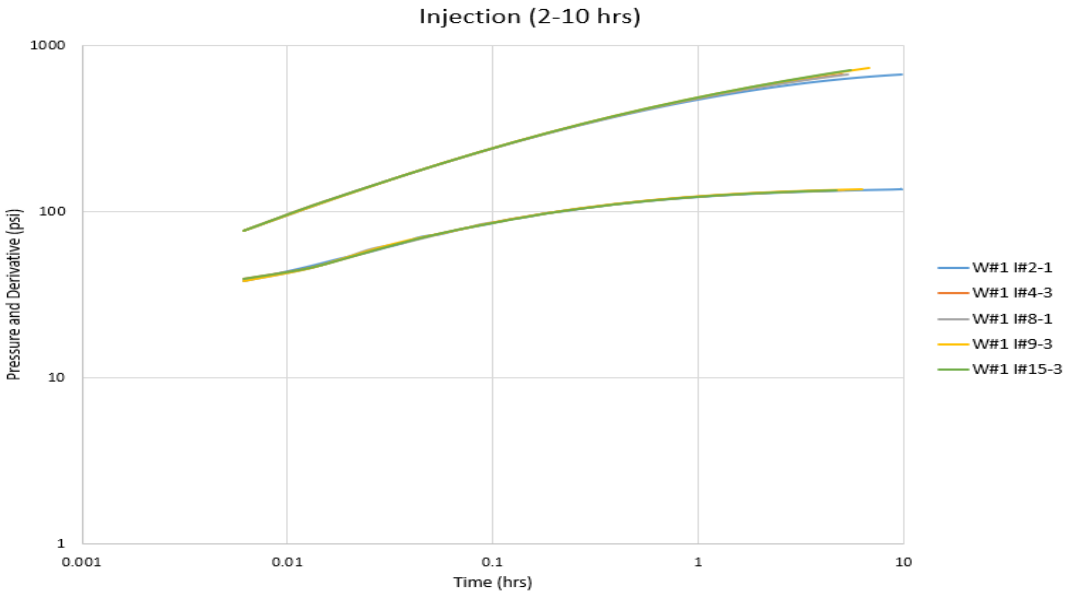

Figure 5-2 *Pressure and Pressure derivative for five injection transients with time duration between 2 and 10 hours.*

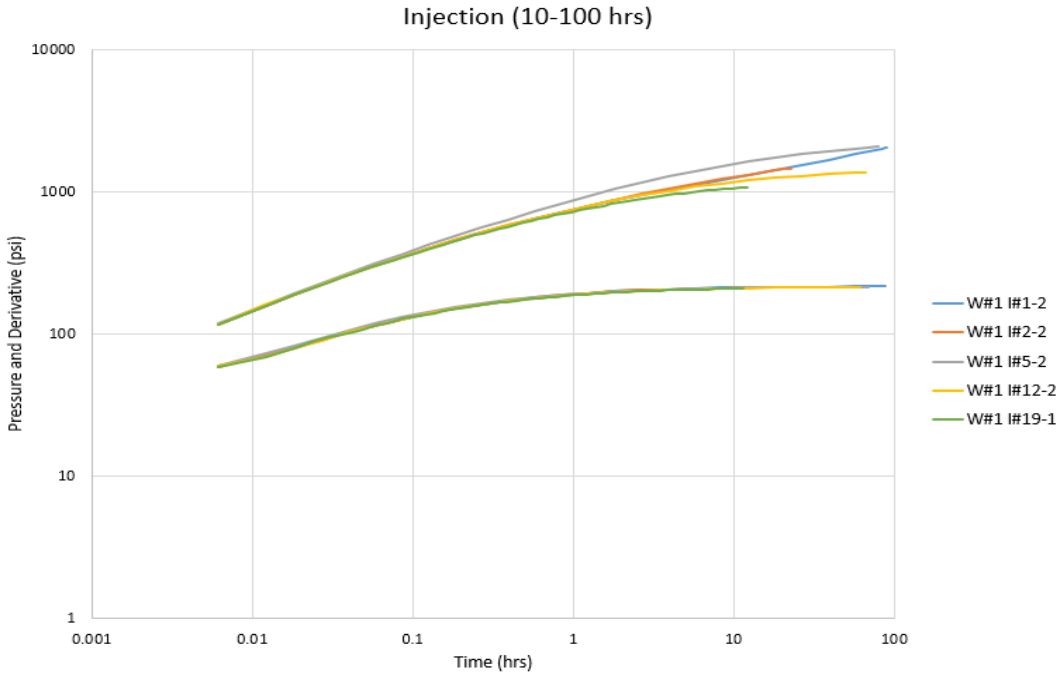

Figure 5-3 *Pressure and Pressure derivative for five injection tests with time duration between 10 and 100 hours.*

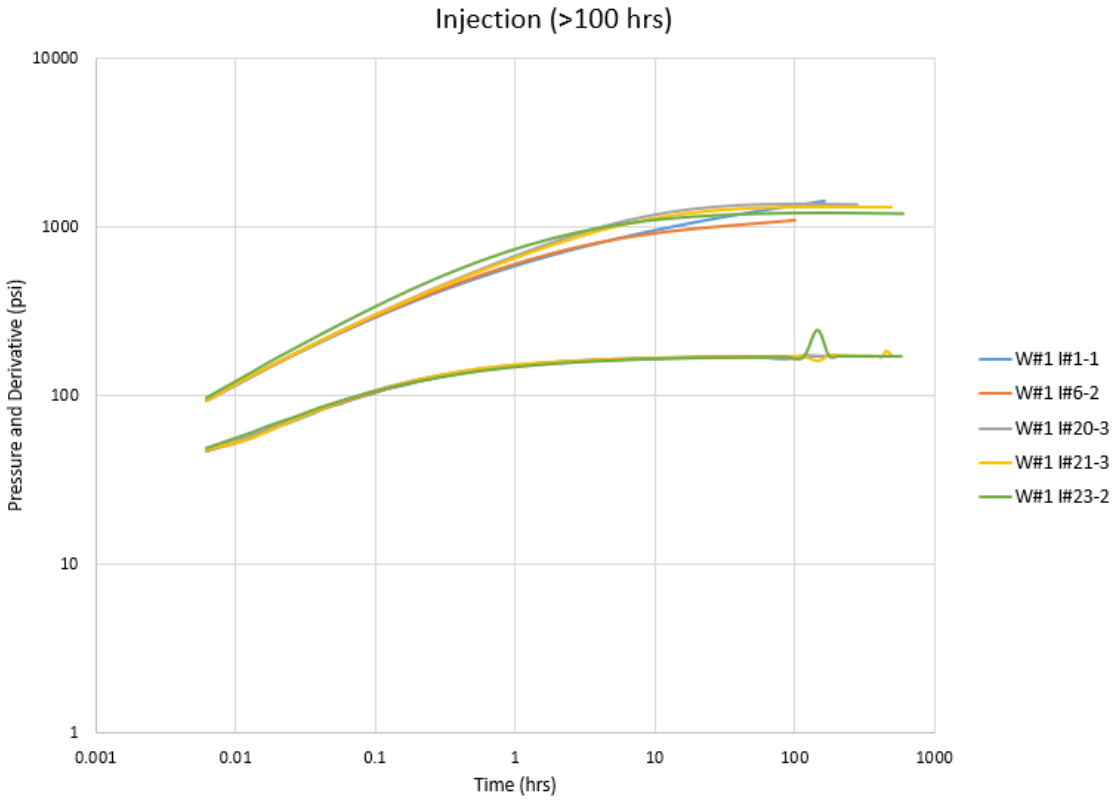

Figure 5-4 *Pressure and pressure derivative for five injection transients with time duration higher than 100 hours.*

The **Figure 5-3** and **Figure 5-4** shows the pressure and pressure derivative of injections transients of 10 to 100 hours and higher than 100 hours respectively. For each one 5 different injection transients were selected.

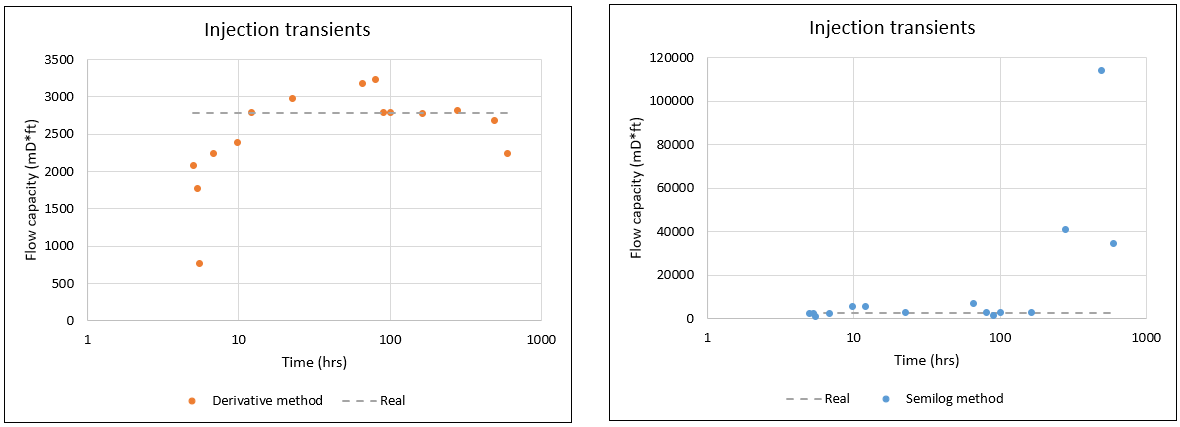

Figure 5-5 *Flow capacity comparison between derivative and semi-log analysis for 15 injections transients with different durations at 0.1% tolerance. The left graph shows the derivative method. The right graph shows the semi-log analysis.*

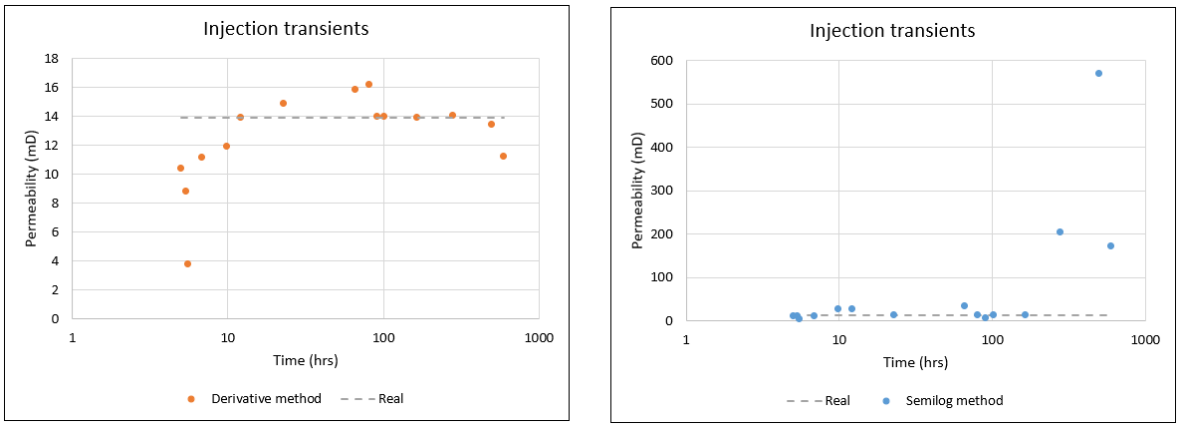

Figure 5-6 *Permeability comparison between derivative and semi-log analysis for 15 injection transients with different durations at 0.1% tolerance. The left graph shows the derivative method. The right graph shows the semi-log analysis.*

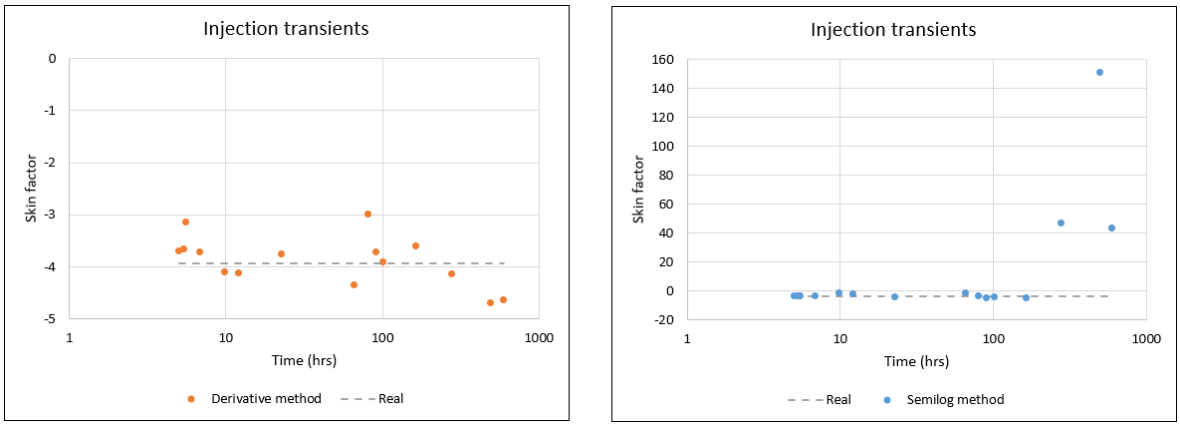

Figure 5-7 *Skin factor comparison between derivative and semi-log analysis for 15 injection transients with different durations at 0.1% tolerance. The left graph shows the derivative method. The right graph shows the semi-log analysis.*

The **Figure 5-5**, **Figure 5-6** and **Figure 5-7** shows 15 injection transients that were evaluated with a tolerance of 0.1% when estimating the radial flow regime.

The **Figure 5-5** and **Figure 5-6** show a comparison between semi-log and derivative analysis for Flow capacity and permeability respectively. Both graphs show that the semi-log analysis has some trouble when calculating the properties by obtaining higher values than the derivative case. This is due to the semi-log analysis doesn't consider the superposition time. In the derivative case it's clear that longer duration periods estimate values closer to the true value. In comparison with the shorter duration ones (less than 10 hours).

The short duration ones estimate lower values. Curiously the transient with the longer time duration has values more deviated to the true value than the rest of transients higher than 100 hours. The **Figure 5-7** shows the skin factor for semi-log and derivative analysis. Again, the semi-log analysis

has transients with higher values than the derivative case. The skins calculated with the derivative show more homogeneous results with all the transients analyzed.

The **Figure 5-8**, **Figure 5-9** and **Figure 5-10** show 15 injection transients that were evaluated with a tolerance of 1% when estimating the radial flow regime.

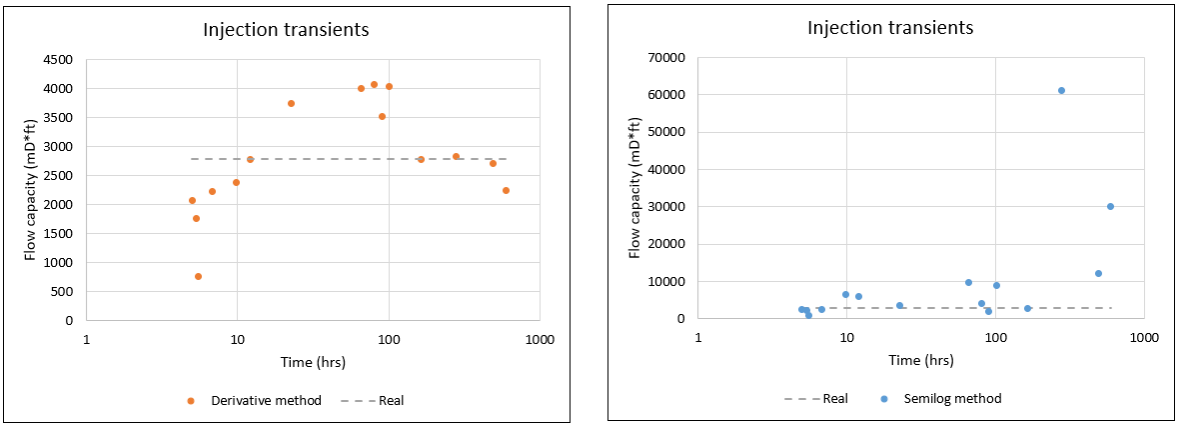

Figure 5-8 *Flow capacity comparison between derivative and semi-log analysis for 15 injection transients with different durations at 1% tolerance. The left graph shows the derivative analysis. The right graph shows the semi-log analysis.*

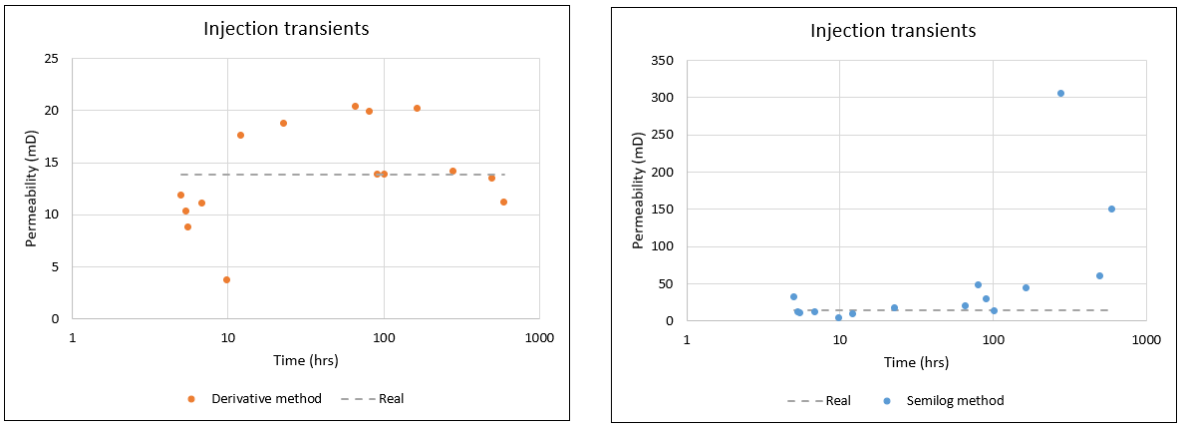

Figure 5-9 *Permeability comparison between derivative and semi-log analysis for 15 injection transients with different durations at 1% tolerance. The left graph shows the derivative analysis. The right graph shows the semi-log analysis.*

The **Figure 5-8** and **Figure 5-9** when increasing the tolerance the values obtained diverged more from the true value of the synthetic case. This is appreciated in both methodologies. The transients with the longer durations were the ones that suffer the less changed values. These can be due to having more stabilized points in comparison with the short duration transients.

By the other hand, **Figure 5-10** for skin factor shows less modification of their values in comparison with the flow capacity and the permeability. The semi-log analysis is still giving higher values than the derivative ones.

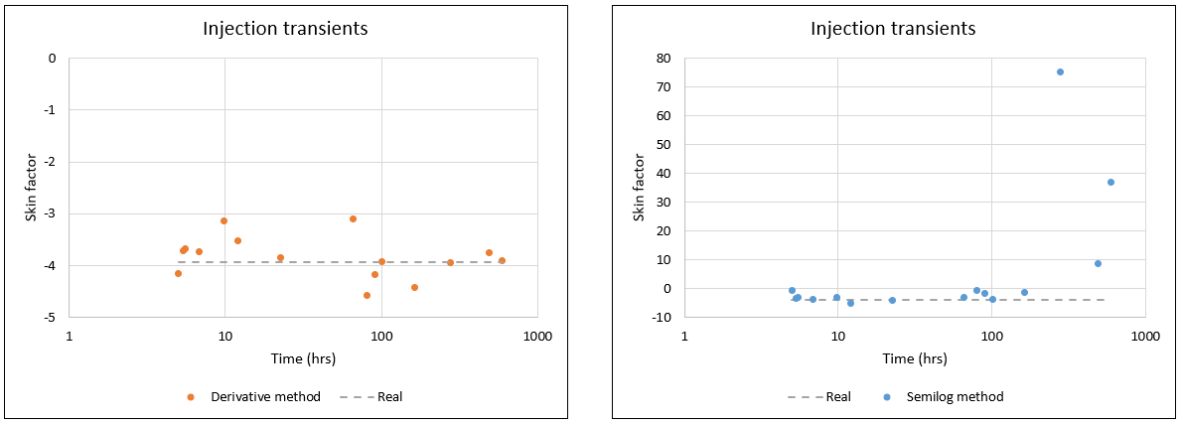

Figure 5-10 *Skin factor comparison between derivative and semi-log analysis for 15 injection transients with different duration at 1% tolerance. The left graph shows the derivative analysis. The right graph shows the semi-log analysis.*

5.2 Fall-off analysis

The **Figure 5-11** shows the history plot simulated. Here, can be seen the Fall-off periods used in the analysis, covering different durations and time in the history. There were analyzed 9 transients. Four between 2 and 10 hours and 5 with more than 10 hours duration. The **Figure 5-12** and **Figure 5-13** show the pressure and pressure derivative of Fall-off transients between 2 and 10 hours and higher than 10 hours respectively.

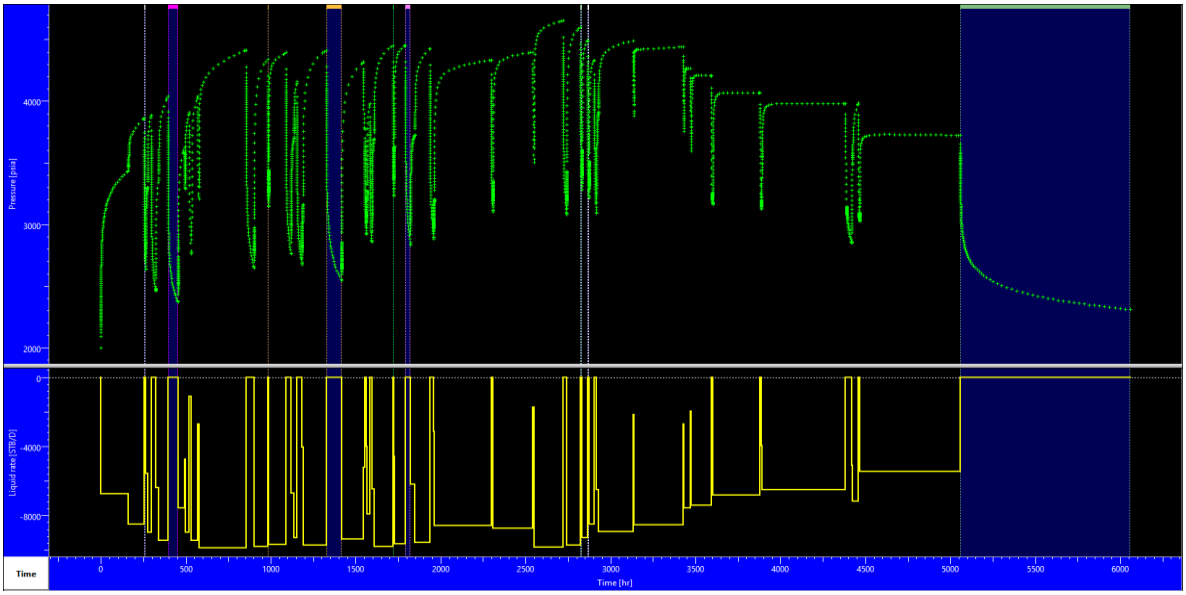

Figure 5-11 *History plot (pressure and rate) for the whole simulation period.*

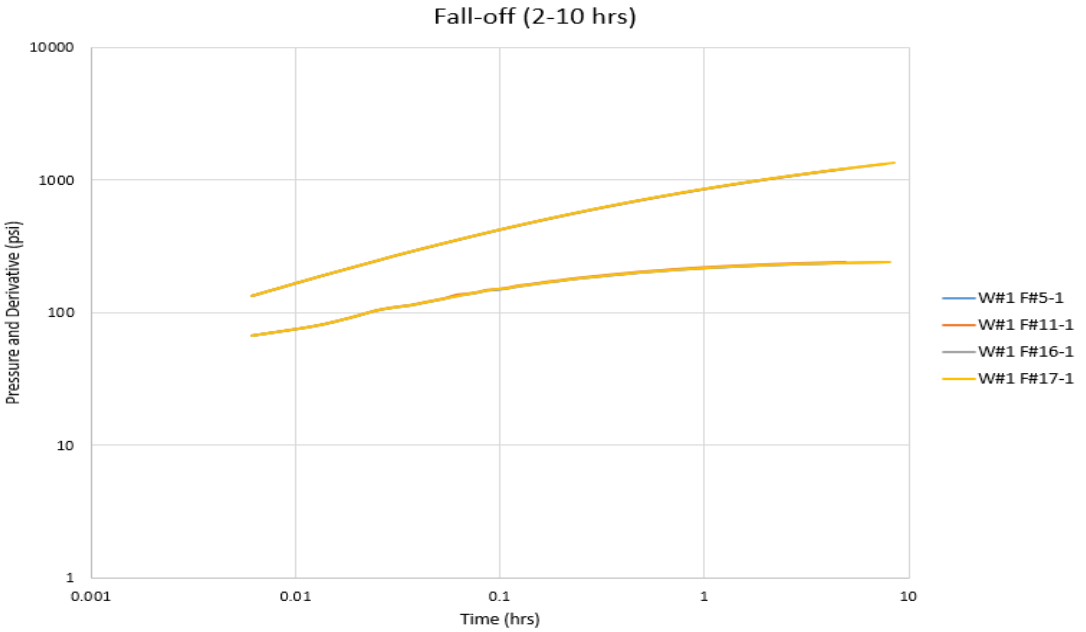

Figure 5-12 *Pressure and pressure derivative from four Fall-off transients with time duration between 2 and 10 hours.*

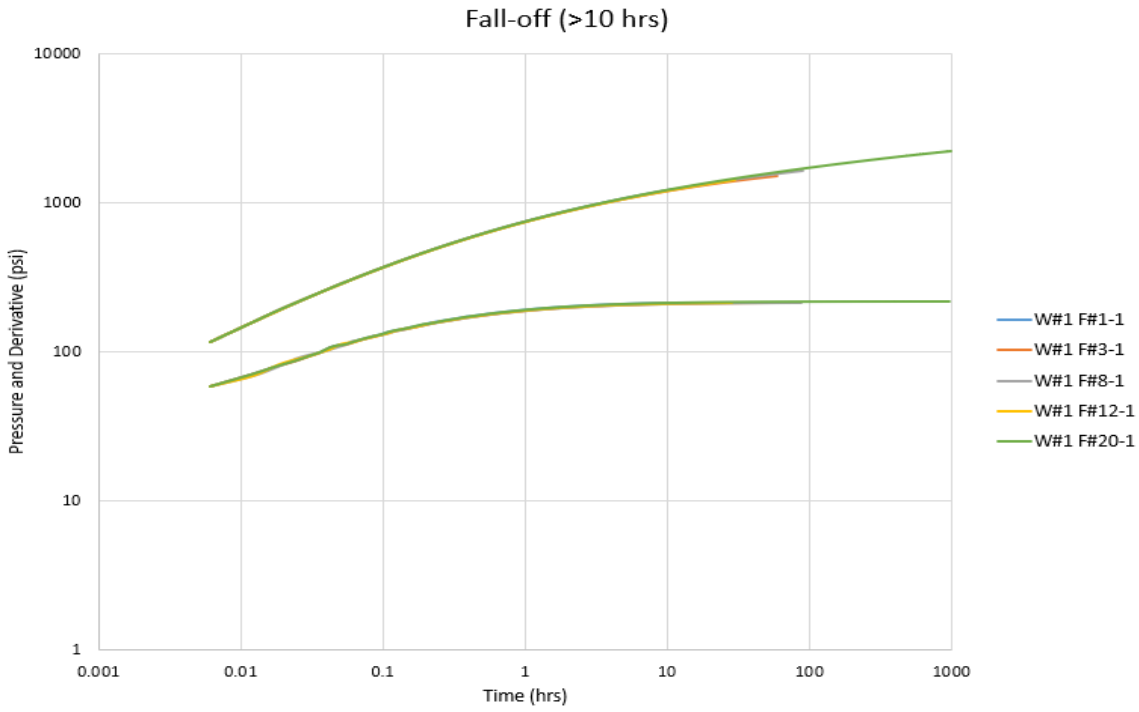

Figure 5-13 *Pressure and pressure derivative from five Fall-off transients with time intervals higher than 10 hours.*

In **Figure 5-14** and **Figure 5-15** were consider a tolerance of 0.1% for estimating the radial flow regime. In both figures the values estimated with the methodologies are more alike. This is due to the equations for shut-in periods consider the superposition time. For the flow capacity and permeability, the semi-log analysis gives closer values than the derivative with the true values. While the skin factor derivative is more accurate than the semi-log estimation.

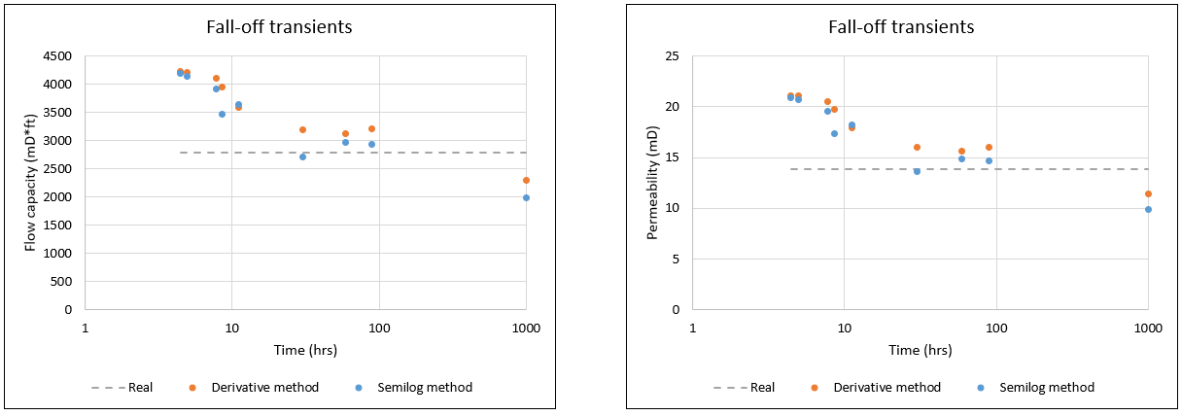

Figure 5-14 *Comparison between the semi-log and derivative analysis for 9 transients at 0.1% tolerance. The left graph shows the flow capacity. The right graphs shows the permeability.*

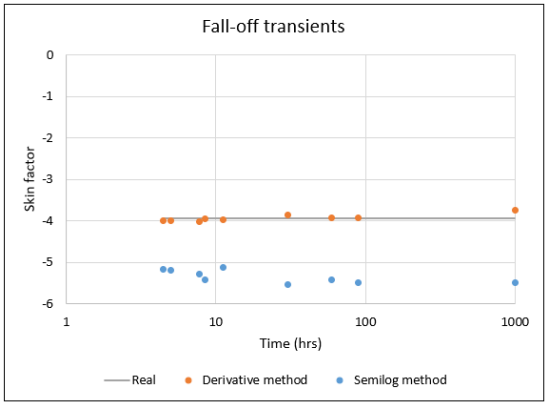

Figure 5-15 *Skin factor comparison between the semi-log and derivative analysis for 9 transients at 0.1% tolerance.*

In **Figure 5-16** and **Figure 5-17** were consider a tolerance of 1% for estimating the radial flow regime. As in the previous case the values between the semi-log and derivative analysis are very similar. Also, when increasing the tolerance for estimating the radial flow regime the values calculated in flow capacity and permeability divert more from the true value. The skin factor is less affected when the tolerance increases. The most affected transients were the ones between 10 and 100 hours. This can be explained due to adding points that are not stable enough to the estimated radial flow.

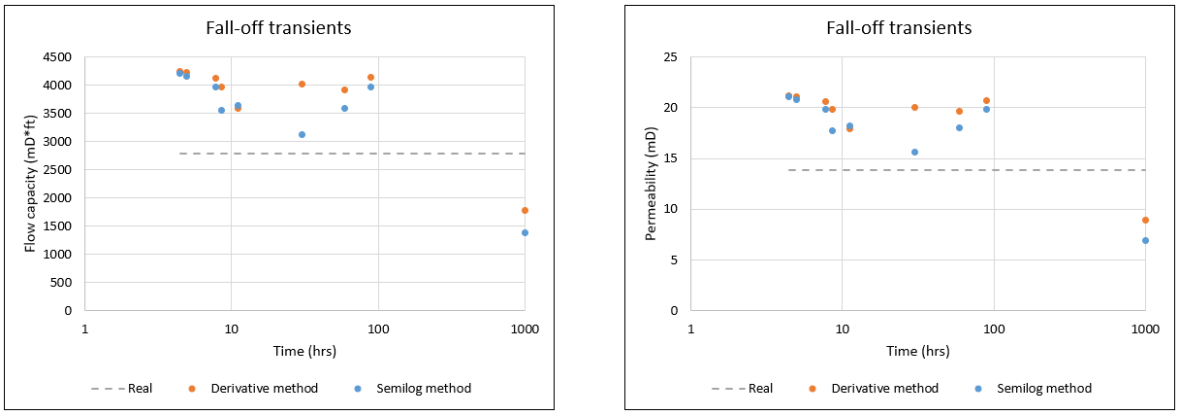

Figure 5-16 *Comparison between the semi-log and derivative analysis for 9 transients at 1% tolerance. The left graph shows the flow capacity. The right graph shows the permeability.*

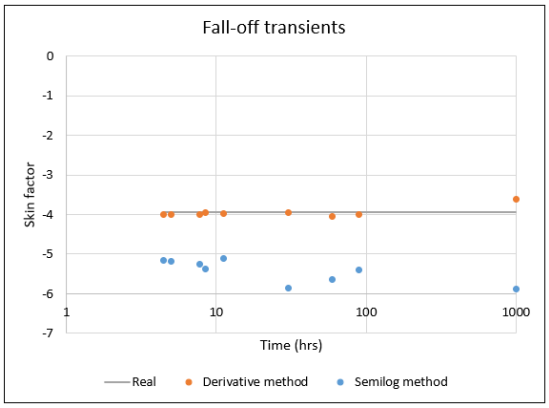

Figure 5-17 *Skin factor comparison between the semi-log and derivative analysis for 9 transients at 1% tolerance.*

5.3 Synthetic analysis discussion

In all cases increasing the tolerance value for estimating the radial flow diverge more for both methodologies and injection and fall-off transients. The transients between 2 and 10 hours tends to give less accurate values than the ones with higher duration. Also are more susceptible to tolerance.

For the injection case is recommended to perform a derivative analysis than a semi-log analysis due to the superposition time. For the fall-off both methodologies show similar results. The semi-log performs better for flow capacity and permeability, but the derivative estimates a skin factor more accurately to the true value. For the fall-off case can be considered to calculate an average between the semi-log and derivative analysis.

6 Real Case Analysis

From the free access data set provided by Equinor for Volve field an analysis was performed. The transients analyzed were chosen after visualizing the data in Kappa software "Saphir". The best sets of data for Volve were Build-up transients from well NO 15/9F-15D. The properties used where the ones reported in the Volve data set, after conversions to be used in the macro, as follow:

a) Initial pressure: 4670 psia.
b) Formation volume factor: 1.44 RB/STB.
c) Viscosity: 0.32 cp.
d) Well radius: 0.35 ft
e) Porosity: 22%
f) Compressibility of the rock: 2.33×10^{-5} $psia^{-1}$.
g) Pay zone (Thickness): 72 ft.

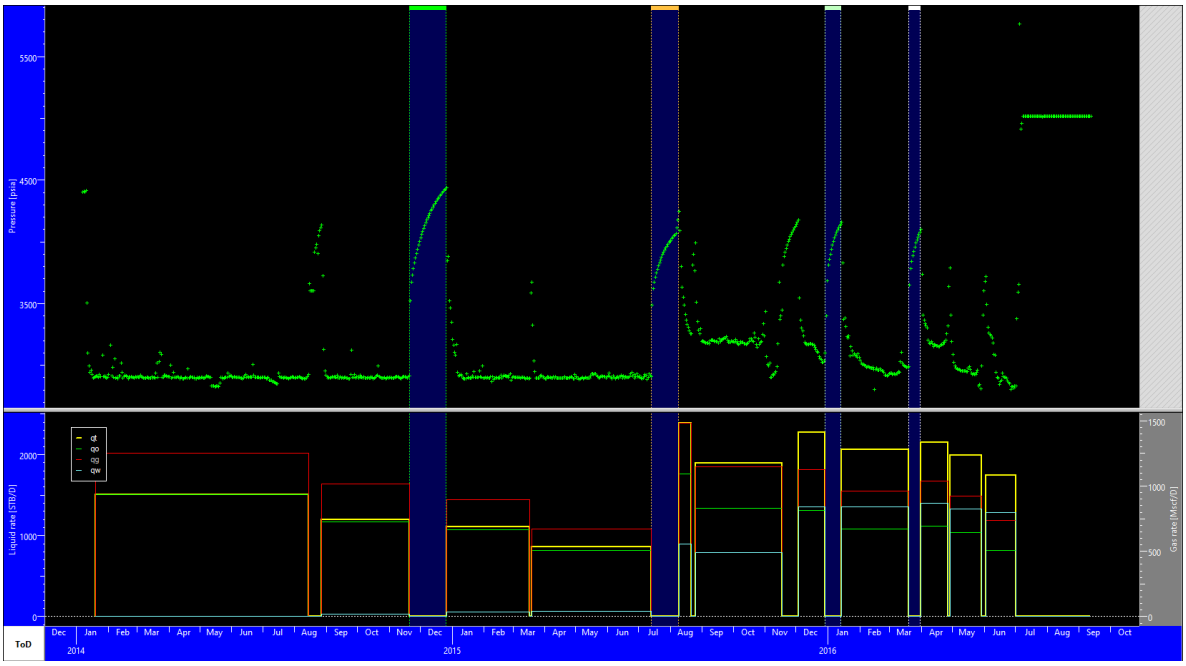

Figure 6-1 *History plot (rate and pressure) for the whole simulation period.*

The **Figure 6-1** shows the history plot of the well, the transients analyzed are highlighted. The Macro used the total rate calculated by "Saphir", instead of the separated phases.

The derivative and superposition time from the code was compared with the "Saphir" software and are shown in **Figure 6-2** and **Figure 6-3**. The calculation from the algorithm showed good performance in pressure and pressure derivative in the real case data **Figure 6-2**. Only the build-up transient 5-1 has the bigger difference with the trend calculated.

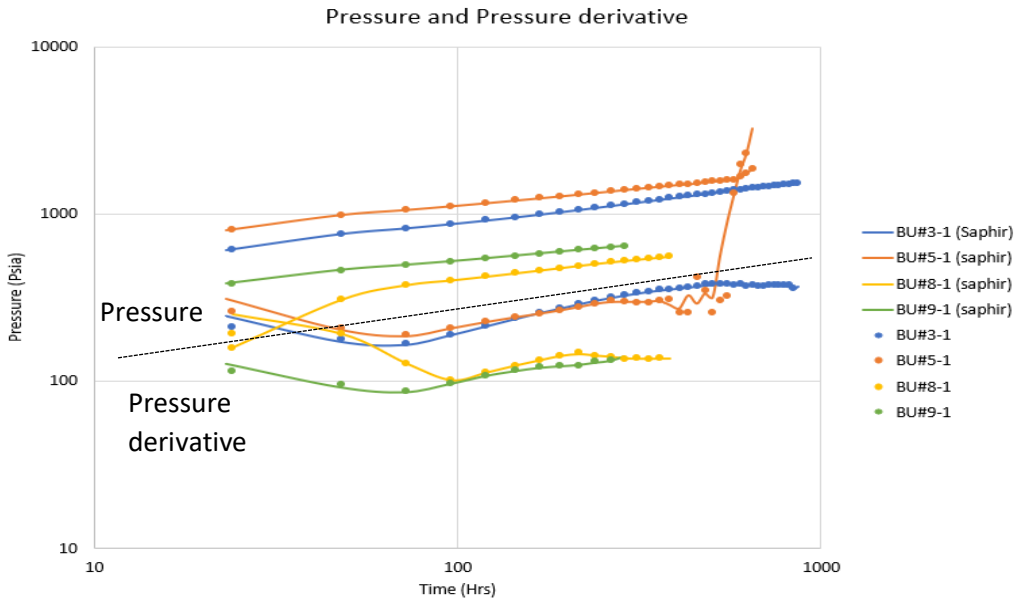

Figure 6-2 *Pressure and Pressure derivative comparison between Macro and Saphir.*

The **Figure 6-3** in superposition time showed really good results for the real case. These gives the possibility to perform a quick analysis with the smoother for the derivative. There were performed two different smoother runs. The first one with the default value 0.1 and the second one with 0.3 for all cases, this is shown in **Figure 6-4**. There were no changes in build-ups 3-1 and 9-1. For the Build-up 8-1 the curve became more smoothed but 5-1 got more spiked. This means that there can be a possibility for more smoother analysis in further works with other sets of data.

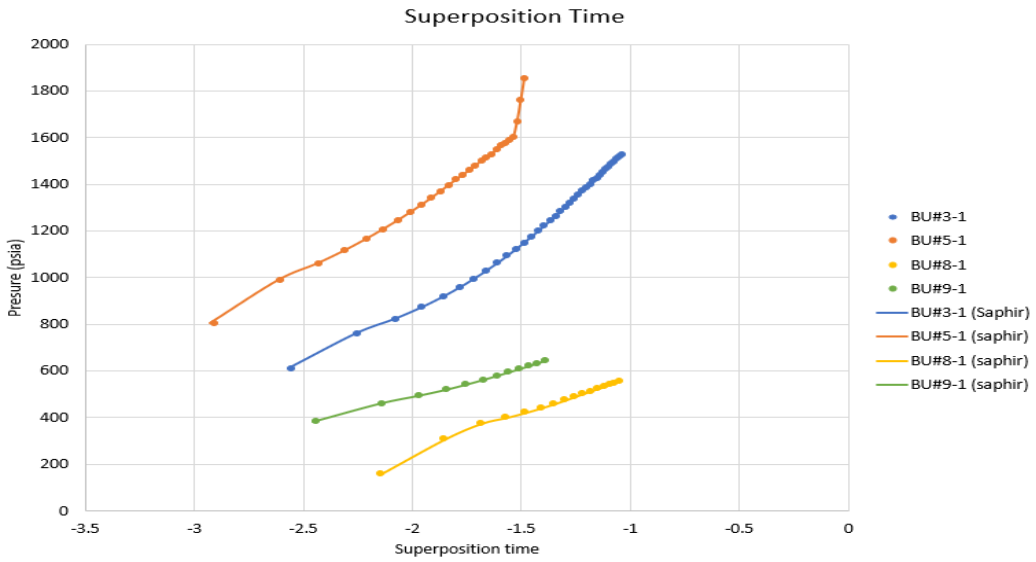

Figure 6-3 *Superposition time comparison between the Macro and Saphir.*

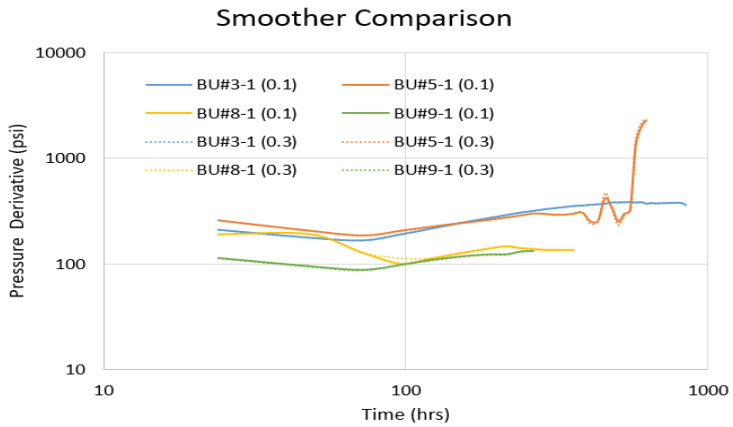

Figure 6-4 *Smoother comparison of the pressure derivative.*

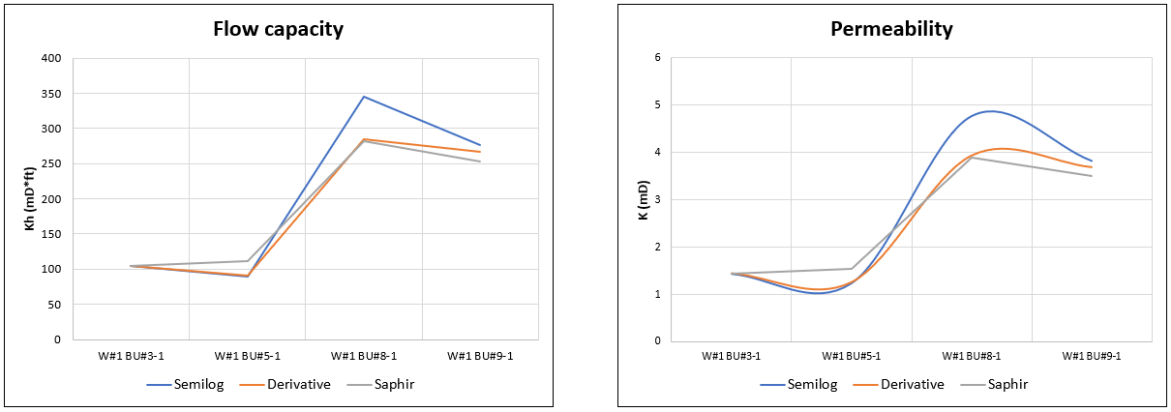

Figure 6-5 *Comparison graphs between semi-log, derivative analysis and Saphir from four build-up transients. Left graph shows the flow capacity. Right graph shows the permeability.*

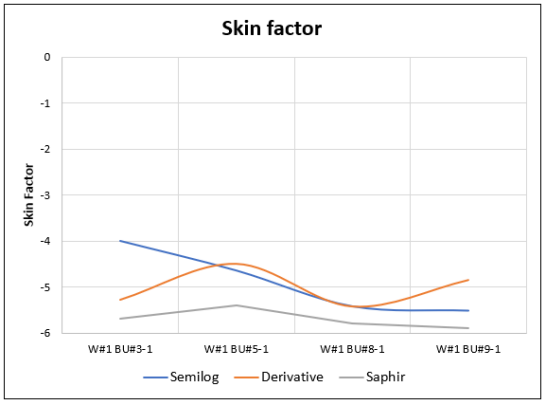

Figure 6-6 *Comparison of skin factor between semi-log, derivative and Saphir analysis from four build-up transients.*

From the analysis done in the synthetic case related to shut-in transients, it's possible to make both analyses. These are shown in **Figure 6-5** and **Figure 6-6**. For flow capacity and permeability, the semi-log analysis gives slightly higher values than the derivative.

Table 6-1 *Duration of four Build-up transients.*

Transient	Duration (hours)
W#1 BU#3-1	864
W#1 BU#5-1	648
W#1 BU#8-1	384
W#1 BU#9-1	288

The **Table 6-1** shows that the transients analyzed lasted more than 100 hours. Here, it can see that the transients with the best results are the Build-up with the longest duration. The derivative analysis worked better at estimating flow capacity and permeability values than the semi-log analysis. For the skin factor both methodologies worked better for some transients than the other one and vice versa.

Table 6-2 *Percentage error of the average parameters estimated with the code in comparison with Saphir.*

	Semi-log	Derivative
Kh (mD*ft)	13%	6%
K (mD)	13%	6%
Skin	14%	12%

The **Table 6-2** shows the average error estimated with both methodologies in comparison with saphir results. The derivative as seen in the graphs estimates better values than the semi-log. The derivative calculates values 6% error in comparison with the 12% for the semi-log for flow capacity and permeability. Both methodologies offer similar error for the skin factor. The transients that increases the average error was the build-up 5-1, due to having a not so stable radial flow regime.

7 Conclusions

Using the MS Excel allows an easy way to add new data (pressure and rate) acquire from the PDGs into the defined input sheets. Suggest a first filter of the different transients included in the data.

From what was discussed in the chapters 5 and 6 there can be listed the following statements when using the semi-log analysis or the derivative analysis.

1. Not considering superposition in injection periods leads to higher error in the results for the semi-log analysis in all properties estimated.
2. Considering superposition time in shut-in equations, for both methodologies gives very similar results. In the synthetic case the semi-log analysis performed slightly better than the derivative analysis. However, in the real case the derivative showed better results. It can be considered to perform an average between both methodologies for a better estimation.
3. Derivative analysis is recommended for flowing periods. Due to derivative using values corrected with the superposition time. This means that the period response analyzed is a function of the previous periods.

When analyzing the impact of the tolerance, it can be mentioned that a higher tolerance leads to higher error values. This is mainly notice in periods that last less than 10 hours. The periods that last longer are less affected, due to having more points stabilized.

In further work for the thesis it can be consider to:

1. Work in a filter of noise for detecting the problematic points. A different solution could be allowing the user to manually specify the exclusion of specific points.
2. Other improvement could be the inclusion of an extra step after the proposed data has been filtered and classified in injection or Fall-off periods. In here, it can be highlighted the transients that can be considered problematic or too short (less than 2 hours) to perform an analysis. This will leave the engineer to decide whether to continue with these periods or filtered them.
3. Would be good to analyze the impact of the smoother in the derivative analysis. Since, the smoother mainly fixes late times when calculating the derivative.
4. Consider using different methodologies than the slope to estimate the “window” of the radial flow regime. And compare both estimations to define the best approach.

The Inclusion of three phases systems and calculation of the total rate without the necessity of using “Saphir” software should be considered. Also, a comparison between different sandstones and carbonates reservoir and the behavior on the macro when suggesting the radial flow and properties calculated.

8 References

[1] Bourdet D., Well Test analysis: The Use of Advanced Interpretation Models, Handbook of Petroleum Exploration and Production 3, Elsevier Science, 2002.

[2] Bourdarot G., Well Testing: Interpretation Methods, Institut Francais du Petrole Publications, Editions Technip, 1998.

[3] Earlougher R., Advances in Well Test Analysis, Society of petroleum engineers of AIME, second printing, 1977.

[4] Jiung C., Salmi N., Permanent Downhole Gauge a Need or a Luxury, Paper SPE 122604 was presented in Jakarta, Indonesia, 2009

[5] Tian C., Horne R., Machine Learning Applied to Multiwall Test Analysis and Flow Rate Reconstruction, Paper SPE 175059-MS was presented in Houston, USA, 2015.

[6] Tian C., Horne R., Applying Machine Learning Techniques to Interpret Flow Rate, Pressure and Temperature Data from Permanent Downhole Gauges, Paper SPE 174034-MS was presented in California, USA, 2015.

[7] Enyekwe A, Ajienka J., Comparative Analysis of Permanent Downhole Gauges and Their Applications, Paper SPE-172435-MS was presented in Lags, Nigeria, 2014.

[8] Jahangiri H., Adler C. et al, A Data-Driven Approach Enhances Conventional Reservoir Surveillance Methods for Waterflood Performance Management in the North Sea, Paper SPE-167849-MS was presented in Utrecht, Netherlands, 2014.

[9] Tian C., Horne R., Inferring Interwell Connectivity using Production Data, Paper SPE-181556-MS was presented in Dubai, UAE, 2016.

[10] Bandyopadhyay P., Horne R., Development of General Likelihood Maximization Method for Robust Parameter Estimation from Pressure Transient Data, Paper SPE 175003-MS was presented in Houston, USA, 2015.

[11] Bello O., Development of Hybrid Intelligent System for Virtual Flow Metering in Production Wells, Paper SPE 167880-MS was presented in Utrecht, Netherlands, 2014.

[12] Bourdet, Dominique, Use of Pressure Derivative in Well-Test Interpretation, Paper SPE 12777-PA was presented in California, USA, 1984.

[13] Hosseinpour-Zonoozi, N., Blasingame, A., The Pressure Derivative Revisited – Improved Formulations and applications, Paper SPE 103204 was presented in San Antonio, USA, 2006.

[14] Shchipanov A., Berenblyum R., Kollboth L., Pressure Transient Analysis as an Element of Permanent Reservoir Monitoring, Paper SPE 170740-MS was presented in Amsterdam, Netherlands, 2014.

9 Nomenclature

dt	Difference in time between the time analyzed and the start of the segment.
T	Time from the data point analyzed.
T_{is}	Initial time at the start of the segment.
dps	Difference pressure between the pressure analyzed and the pressure in the segment.
P	Pressure from the data point analyzed.
P_{is}	Initial pressure at the start of the segment.
$dp_{@Pi}$	Difference pressure between the pressure analyzed and the initial pressure at time 0.
P_i	Pressure at the initial time zero.
d_{pnp}	Pressure normalized for the superposition time in the point analyzed.
Q_{ref}	Reference rate used for the normalization.
Q_i	Rate from the test analyzed.
d_{pnd}	Pressure normalized for the derivative in the point analyzed.
Q_{i-1}	Rate from the previous test analyzed.
dpr	Pressure re-scale for the derivative pressure.
T_s	Superposition time.
Q_n	Most recent rate for the superposition time analyzed.
T_i	End time from the previous test.
Tsi	Time of the point analyzed for the derivative calculation.
TL	Left point to the analyzed point for the derivative calculation.
TR	Right point to the analyzed point for the derivative calculation.
w	Smoother value.
dΔp	Derivative from the delta pressure.
dΔt	derivative from the delta time.
j	Point analyzed.
ΔP_j	Pressure difference at point j.
ΔP_L	Pressure difference to the left point of the point j.

ΔP_R	Pressure difference to the right point of the point j.
ΔT_R	Time difference to the right point of the point j.
ΔT_L	Time difference to the left point of the point j.
ΔT_j	Time difference at point j.
$d\Delta_{pn}$	Pressure derivative normalized.
p'_i	Pressure derivative at point i.
p'_{i-1}	Previous pressure derivative point to the point i.
n	Number of data points in the MSE calculation.
Yi	Value of the derivative at the point i during MSE calculation.
Y	Value of the constant derivative in the MSE calculation.
m	Slope for the semi-log analysis.
q	Rate of the test analyzed in stb/d.
Bo	Oil volume factor in RB/STB.
µo	Oil viscosity in centipoise.
kh	Flow capacity in mD*ft.
k	Permeability in mD.
S	Skin factor.
P_{1hr}	Pressure at 1 hour in psia.
φ	Porosity (dimensionless).
C_t	Total compressibility in $psia^{-1}$.
rw	Well radius in ft.
P_{wfs}	Last flowing pressure before the shut-in.
t	Production time before the shut-in.
m'	Slope for the derivative analysis.
dpd	Constant pressure in the derivative analysis.
h	Formation thickness.

10 Appendix A

The appendix A contains the subroutines written in the module. The Global variables were defined as follow

Global ww As Byte, sapphire(), topaz(), emerald(), ruby(), summat, pini(), bo(), muo(), rw(), hgt()
Global poro(), ct(), tt As Integer, quartz(), derev, interp, ws As Worksheet, wellww As Byte

10.1 Workflow

This is the main subroutine that calls the rest of the subroutines.

Public Sub Workflow()

```
Dim condb As String

condb = Worksheets(1).Shapes(1).TextFrame.Characters.Text
If condb = "Generate" Then
   Prepare
ElseIf condb = "Calculate" Then
   Initialize
   Generate
   Reference.Show 'Calls the first userform
   If quartz(1) = 1 Then
      Normalize
      Calculate
      Calculateb
      Comparative
      Genesis
      Printing
      Plotter.Show 'Calls the second userform
   End If
End If
```

End Sub

10.2 Prepare

This subroutine prepares the layout of the main excel sheet for input the properties of the well and pressure/rate data.

Public Sub Prepare()

```
Dim aa As Byte

Application.ScreenUpdating = False 'Faster calculations

ww = Worksheets(1).Cells(3, 4)
Worksheets(1).Unprotect Password:="crystal"
Worksheets(1).Cells(3, 4).Interior.Color = RGB(0, 204, 0)
Worksheets(1).Cells(3, 4).Locked = True
Worksheets(2).Unprotect Password:="crystal"
Worksheets(3).Unprotect Password:="crystal"

aa = 0
For i = 1 To ww
   Worksheets(2).Columns(1 + 4 * aa).Resize(, 2).Locked = False
   Worksheets(3).Columns(1 + 4 * aa).Resize(, 2).Locked = False
   aa = aa + 1
Next i

aa = 0
For i = 1 To ww
   Worksheets(2).Cells(2, 1 + 4 * aa) = "Well " & aa + 1
   Worksheets(2).Cells(2, 1 + 4 * aa).Locked = True
   Worksheets(2).Cells(3, 1 + 4 * aa) = "Date"
   Worksheets(2).Cells(3, 1 + 4 * aa).Locked = True
   Worksheets(2).Cells(3, 2 + 4 * aa) = "Pressure (psia)"
   Worksheets(2).Cells(3, 2 + 4 * aa).Locked = True
   Worksheets(2).Cells(3, 2 + 4 * aa).WrapText = True
   Worksheets(3).Cells(2, 1 + 4 * aa) = "Well " & aa + 1
   Worksheets(3).Cells(2, 1 + 4 * aa).Locked = True
   Worksheets(3).Cells(3, 1 + 4 * aa) = "Time@end"
   Worksheets(3).Cells(3, 1 + 4 * aa).Locked = True
   Worksheets(3).Cells(3, 2 + 4 * aa) = "Liquid rate (STB/D)"
   Worksheets(3).Cells(3, 2 + 4 * aa).WrapText = True
   Worksheets(3).Cells(3, 2 + 4 * aa).Locked = True
   Worksheets(1).Cells(8, 2 + 4 * aa) = "Well " & aa + 1
   Worksheets(1).Cells(9, 2 + 4 * aa) = "Pi(psia)"
   Worksheets(1).Cells(9, 3 + 4 * aa).Interior.ColorIndex = 6
   Worksheets(1).Cells(9, 3 + 4 * aa).Locked = False
   Worksheets(1).Cells(10, 2 + 4 * aa) = "Bo(RB/STB)"
   Worksheets(1).Cells(10, 3 + 4 * aa).Interior.ColorIndex = 6
   Worksheets(1).Cells(10, 3 + 4 * aa).Locked = False
   Worksheets(1).Cells(11, 2 + 4 * aa) = "Viscosity(cp)"
   Worksheets(1).Cells(11, 3 + 4 * aa).Interior.ColorIndex = 6
   Worksheets(1).Cells(11, 3 + 4 * aa).Locked = False
   Worksheets(1).Cells(12, 2 + 4 * aa) = "rw(ft)"
```

```
   Worksheets(1).Cells(12, 3 + 4 * aa).Interior.ColorIndex = 6
   Worksheets(1).Cells(12, 3 + 4 * aa).Locked = False
   Worksheets(1).Cells(13, 2 + 4 * aa) = "h(ft)"
   Worksheets(1).Cells(13, 3 + 4 * aa).Interior.ColorIndex = 6
   Worksheets(1).Cells(13, 3 + 4 * aa).Locked = False
   Worksheets(1).Cells(14, 2 + 4 * aa) = "Porosity"
   Worksheets(1).Cells(14, 3 + 4 * aa).Interior.ColorIndex = 6
   Worksheets(1).Cells(14, 3 + 4 * aa).Locked = False
   Worksheets(1).Cells(15, 2 + 4 * aa) = "Ct(psia-1)"
   Worksheets(1).Cells(15, 3 + 4 * aa).Interior.ColorIndex = 6
   Worksheets(1).Cells(15, 3 + 4 * aa).Locked = False
   aa = aa + 1
Next i

Worksheets(2).Protect Password:="crystal", AllowFormattingCells:=True,
AllowFormattingColumns:=True, AllowFormattingRows:=True
Worksheets(3).Protect Password:="crystal", AllowFormattingCells:=True,
AllowFormattingColumns:=True, AllowFormattingRows:=True
Worksheets(1).Shapes(1).TextFrame.Characters.Text = "Calculate"
Worksheets(1).Protect Password:="crystal", AllowFormattingCells:=True,
AllowFormattingColumns:=True, AllowFormattingRows:=True
Application.ScreenUpdating = True 'Faster calculations
Worksheets(1).Activate 'To return to the main sheet

End Sub
```

10.3 Initialize

This subroutine saves the input data from the well properties.

```
Public Sub Initialize()

Dim aa As Byte, bb As Integer, mm As Integer

Application.ScreenUpdating = False 'Faster calculations
Worksheets(1).Unprotect Password:="crystal"
Worksheets(3).Unprotect Password:="crystal"
ww = Worksheets(1).Cells(3, 4)

ReDim pini(ww), bo(ww), muo(ww), rw(ww), hgt(ww), poro(ww), ct(ww)

aa = 0
For i = 1 To ww
   'Initial data of wells
   pini(i) = Worksheets(1).Cells(9, 3 + 4 * aa) 'Initial pressure
   Worksheets(1).Cells(9, 3 + 4 * aa).Interior.Color = RGB(0, 204, 0)
```

```
   Worksheets(1).Cells(9, 3 + 4 * aa).Locked = True
   bo(i) = Worksheets(1).Cells(10, 3 + 4 * aa) 'Oil formation volume factor
   Worksheets(1).Cells(10, 3 + 4 * aa).Interior.Color = RGB(0, 204, 0)
   Worksheets(1).Cells(10, 3 + 4 * aa).Locked = True
   muo(i) = Worksheets(1).Cells(11, 3 + 4 * aa) 'Oil viscosity
   Worksheets(1).Cells(11, 3 + 4 * aa).Interior.Color = RGB(0, 204, 0)
   Worksheets(1).Cells(11, 3 + 4 * aa).Locked = True
   rw(i) = Worksheets(1).Cells(12, 3 + 4 * aa) 'Wellbore radius
   Worksheets(1).Cells(12, 3 + 4 * aa).Interior.Color = RGB(0, 204, 0)
   Worksheets(1).Cells(12, 3 + 4 * aa).Locked = True
   hgt(i) = Worksheets(1).Cells(13, 3 + 4 * aa) 'Height reservoir
   Worksheets(1).Cells(13, 3 + 4 * aa).Interior.Color = RGB(0, 204, 0)
   Worksheets(1).Cells(13, 3 + 4 * aa).Locked = True
   poro(i) = Worksheets(1).Cells(14, 3 + 4 * aa) 'Porosity
   Worksheets(1).Cells(14, 3 + 4 * aa).Interior.Color = RGB(0, 204, 0)
   Worksheets(1).Cells(14, 3 + 4 * aa).Locked = True
   ct(i) = Worksheets(1).Cells(15, 3 + 4 * aa) 'Compressibility of the rock
   Worksheets(1).Cells(15, 3 + 4 * aa).Interior.Color = RGB(0, 204, 0)
   Worksheets(1).Cells(15, 3 + 4 * aa).Locked = True
   aa = aa + 1
Next i

Application.ScreenUpdating = True 'Faster calculations
Worksheets(1).Activate 'To return to the main sheet

End Sub
```

10.4 Generate

This subroutine reads the pressure and rate data, set up the working matrixes and filter the transients by type.

```
Public Sub Generate()

Dim aa As Byte, bb As Integer, mm As Integer, cc As Byte, dd As Byte, ff As Byte, gg As Byte
Dim nn as Byte

Application.ScreenUpdating = False 'Faster calculations
ww = Worksheets(1).Cells(3, 4)

'Counts the number rows to use (the longest well data) in production matrix
aa = 0: mm = 0
For i = 1 To ww
   bb = Worksheets(3).Cells(Rows.Count, 1 + 4 * aa).End(xlUp).row 'Last active cell in the column
   bb = bb - 3
```

```
   If bb > mm Then
      mm = bb
   End If
   aa = aa + 1
Next i

ReDim topaz(ww, mm, 10)

'Save data in the production matrix
aa = 0
For i = 1 To ww 'Number of Wells
   topaz(i, 0, 1) = 0
   For j = 1 To mm 'Number of row in rate sheet
      If IsEmpty(Worksheets(3).Cells(j + 3, 1 + 4 * aa)) Then 'Check time data as void
         topaz(i, j, 1) = 0
      Else
         topaz(i, j, 1) = Worksheets(3).Cells(j + 3, 1 + 4 * aa) 'Time data
         topaz(i, 0, 1) = topaz(i, 0, 1) + 1 'starts counting number of data of each well
      End If
      If IsEmpty(Worksheets(3).Cells(j + 3, 2 + 4 * aa)) Then
         topaz(i, j, 2) = 0
      Else
         topaz(i, j, 2) = Worksheets(3).Cells(j + 3, 2 + 4 * aa) 'Rate data
      End If
      If topaz(i, j, 2) < 0 Then
         topaz(i, j, 3) = 1 'Injection test
      ElseIf topaz(i, j, 2) = 0 Then
         topaz(i, j, 3) = 2 ' Fall-off test
      Else
         topaz(i, j, 3) = 3 'Production test
      End If
   Next j
   Worksheets(1).Cells(17, 2 + 4 * aa) = "No. test"
   Worksheets(1).Cells(17, 3 + 4 * aa) = topaz(i, 0, 1)
   aa = aa + 1
Next i

'Establishing segments dd counter for injection secondary, ff counter for fall-off, gg counter for
' production cc counter for secondary
aa = 0
For i = 1 To ww
   cc = 0: dd = 0: ff = 0: gg = 0
   nn = topaz(i, 1, 3)
   'For the first value
   Select Case nn
      Case 1
         dd = dd + 1
```

```
        topaz(i, 1, 4) = dd
        topaz(i, 1, 6) = "Well#" & i & " Injection#" & dd
        topaz(i, 1, 7) = "W#" & i & " I#" & dd & "-"
    Case 2
        ff = ff + 1
        topaz(i, 1, 4) = ff
        topaz(i, 1, 6) = "Well#" & i & " Fall-off#" & ff
        topaz(i, 1, 7) = "W#" & i & " F#" & ff & "-"
    Case 3
        gg = gg + 1
        topaz(i, 1, 4) = gg
        topaz(i, 1, 6) = "Well#" & i & " Production#" & gg
        topaz(i, 1, 7) = "W#" & i & " P#" & gg & "-"
End Select
cc = cc + 1
topaz(i, 1, 5) = cc
topaz(i, 1, 7) = topaz(i, 1, 7) & cc
Worksheets(3).Cells(4, 3 + 4 * aa) = topaz(i, 1, 7) 'Show the name of the test detected
For j = 2 To topaz(i, 0, 1)  'For the rest of the data
    If nn = topaz(i, j, 3) Then 'Check the type of test
        cc = cc + 1
        Select Case nn
            Case 1
                topaz(i, j, 4) = dd
                topaz(i, j, 6) = "Well#" & i & " Injection#" & dd
                topaz(i, j, 7) = "W#" & i & " I#" & dd & "-"
            Case 2
                topaz(i, j, 4) = ff
                topaz(i, j, 6) = "Well#" & i & " Fall-off#" & ff
                topaz(i, j, 7) = "W#" & i & " F#" & ff & "-"
            Case 3
                topaz(i, j, 4) = gg
                topaz(i, j, 6) = "Well#" & i & " Production#" & gg
                topaz(i, j, 7) = "W#" & i & " P#" & gg & "-"
        End Select
    Else
        cc = 1
        nn = topaz(i, j, 3)
        Select Case nn
            Case 1
                dd = dd + 1
                topaz(i, j, 4) = dd
                topaz(i, j, 6) = "Well#" & i & " Injection#" & dd
                topaz(i, j, 7) = "W#" & i & " I#" & dd & "-"
            Case 2
                ff = ff + 1
                topaz(i, j, 4) = ff
```

```
            topaz(i, j, 6) = "Well#" & i & " Fall-off#" & ff
            topaz(i, j, 7) = "W#" & i & " F#" & ff & "-"
         Case 3
            gg = gg + 1
            topaz(i, j, 4) = gg
            topaz(i, j, 6) = "Well#" & i & " Production#" & gg
            topaz(i, j, 7) = "W#" & i & " P#" & gg & "-"
      End Select
   End If
   topaz(i, j, 5) = cc 'Secondary name
   topaz(i, j, 7) = topaz(i, j, 7) & cc
   Worksheets(3).Cells(j + 3, 3 + 4 * aa) = topaz(i, j, 7) 'Show the name of the test detected
 Next j
 aa = aa + 1
Next i
'Counts the number rows to use (the longest well data) in temporal matrix
aa = 0: mm = 0
For i = 1 To ww
 bb = Worksheets(2).Cells(Rows.Count, 1 + 4 * aa).End(xlUp).row 'Last active cell in the column
 bb = bb - 3
 If bb > mm Then
   mm = bb
 End If
 aa = aa + 1
Next i

ReDim quartz(ww, mm, 3)

'Save data in the temporal matrix
aa = 0
For i = 1 To ww 'Number of Wells
 quartz(i, 0, 1) = 0
 For j = 1 To mm 'Number of row
   If IsEmpty(Worksheets(2).Cells(j + 3, 1 + 4 * aa)) Then 'Check time data as void
     quartz(i, j, 1) = 0
   Else
     quartz(i, j, 1) = Worksheets(2).Cells(j + 3, 1 + 4 * aa) 'Time data
     quartz(i, 0, 1) = quartz(i, 0, 1) + 1 'starts counting number of data of each well
       'Calculating accumulative time data
     quartz(i, j, 2) = DateDiff("s", quartz(i, 1, 1), quartz(i, j, 1)) / 3600
   End If
   If IsEmpty(Worksheets(2).Cells(j + 3, 2 + 4 * aa)) Then
     quartz(i, j, 3) = 0
   Else
     quartz(i, j, 3) = Worksheets(2).Cells(j + 3, 2 + 4 * aa) 'Pressure data
   End If
 Next j
```

```
   aa = aa + 1
Next i

'Calculate position of the data start time in the segment
For i = 1 To ww
   bb = 1
   For j = 1 To quartz(i, 0, 1)
      If quartz(i, j, 1) > topaz(i, bb, 1) Then
         topaz(i, bb, 8) = j - 1 'Saves the end of the segment
         topaz(i, bb, 9) = topaz(i, bb, 8) - topaz(i, bb - 1, 8) 'Saves the number of max data for the test
         bb = bb + 1
      End If
   Next j
   topaz(i, topaz(i, 0, 1), 8) = quartz(i, 0, 1) 'Saves the last end point in the last segment
   topaz(i, topaz(i, 0, 1), 9) = topaz(i, topaz(i, 0, 1), 8) - topaz(i, topaz(i, 0, 1) - 1, 8)
Next i
'Detects larger set of data in test and test
mm = 0: bb = 0
For i = 1 To ww
   If topaz(i, 0, 1) > bb Then
      bb = topaz(i, 0, 1) 'Larger set of test
   End If
   For j = 1 To topaz(i, 0, 1)
      If topaz(i, j, 9) > mm Then
         mm = topaz(i, j, 9) 'Larger set of data test
      End If
   Next j
Next i

ReDim sapphire(ww, bb, mm, 6)

'Preparing pressure matrix
For i = 1 To ww
   bb = 1: mm = 0
   For j = 1 To topaz(i, 0, 1)
      sapphire(i, j, 0, 1) = topaz(i, j, 3) 'Saves type of test
      k = 1
      Do
      'For k = 1 To topaz(i, j, 9)
         sapphire(i, j, k, 1) = quartz(i, bb, 1) 'Saving time
         sapphire(i, j, k, 2) = quartz(i, bb, 2) 'Saving acumulative time
         sapphire(i, j, k, 3) = quartz(i, bb, 3) 'Saving pressure
           'Calculating time of segments dt
         sapphire(i, j, k, 4) = quartz(i, bb, 2) - quartz(i, topaz(i, j - 1, 8), 2)
         If j = 1 Then
            sapphire(i, j, k, 5) = Abs(quartz(i, bb, 3) - pini(ww)) 'Calculating dp for the first segment
         Else
```

```
            'Calculating dp segments
              sapphire(i, j, k, 5) = Abs(quartz(i, bb, 3) - quartz(i, topaz(i, j - 1, 8), 3))
          End If
          sapphire(i, j, k, 6) = Abs(quartz(i, bb, 3) - pini(ww)) 'Calculating dp with initial P(t=0)
          If quartz(i, bb, 2) <> 0 Then
             k = k + 1
          Else
             topaz(i, j, 9) = topaz(i, j, 9) - 1
          End If
          bb = bb + 1
       'Next k
       Loop While k <= topaz(i, j, 9)
       If topaz(i, j, 3) = 2 Then
          topaz(i, j, 10) = topaz(i, j - 1, 10) 'Saves production time
          mm = j
       Else
          topaz(i, j, 10) = sapphire(i, j, topaz(i, j, 9), 2) - sapphire(i, mm, topaz(i, j, 9), 2)
       End If
    Next j
Next i

Worksheets(1).Shapes(1).TextFrame.Characters.Text = "Calculate"
Worksheets(1).Protect Password:="crystal", AllowFormattingCells:=True,
AllowFormattingColumns:=True, AllowFormattingRows:=True
Worksheets(3).Protect Password:="crystal", AllowFormattingCells:=True,
AllowFormattingColumns:=True, AllowFormattingRows:=True
Application.ScreenUpdating = True 'Faster calculations
End Sub
```

10.5 Normalize

This subroutine normalizes the pressure with a reference test previously selected in the userform "Reference".

```
Public Sub Normalize()

Dim aa As Byte, bb As Integer, agata As Integer

For i = 1 To ruby(0, 1)
   aa = ruby(i, 1) 'Number of well
   bb = ruby(i, 2) 'Position of the test data
   agata = topaz(aa, bb, 3) 'Type of test
   emerald(i, 0, 1) = 0.1 'Saves the smoother by default
   For j = 1 To topaz(aa, bb, 9) 'Data calculations
      emerald(i, j, 1) = sapphire(aa, bb, j, 4) 'Copying dt segment
```

```
    Select Case agata
      Case 1 'Injection
        emerald(i, j, 2) = sapphire(aa, bb, j, 6) 'Copying pressure unormalized sp
        'Normalization of pressure for superposition
        emerald(i, j, 4) = Abs(ruby(0, 2) / topaz(aa, bb, 2) * emerald(i, j, 2))
        'Normalization of pressure for derivative
        emerald(i, j, 7) = Abs(ruby(0,2)/(topaz(aa,bb,2)-topaz(aa, bb-1,2))*sapphire(aa,bb,j, 5))
        'dp re-scale
        emerald(i, j, 6) = Abs((topaz(aa, bb, 2) / ruby(0, 2)) * emerald(i, j, 7))
      Case 2 'Fall-off
        emerald(i, j, 2) = sapphire(aa, bb, j, 5) 'Copying pressure unormalized sp
        'Normalization of pressure for superposition
        emerald(i, j, 4) = Abs(ruby(0, 2) / topaz(aa, bb - 1, 2) * emerald(i, j, 2))
        'Normalized of pressure for derivative
        emerald(i, j, 7) = emerald(i, j, 4)
        'dp re-scale
        emerald(i, j, 6) = emerald(i, j, 7)
      Case 3 'Production
        emerald(i, j, 2) = sapphire(aa, bb, j, 6) 'Copying pressure unormalized sp
        'Normalization pressure for superposition
        emerald(i, j, 5) = Abs(ruby(0, 2) / topaz(aa, bb, 2) * emerald(i, j, 2))
        'Normalization of pressure for derivative
        emerald(i, j, 7) = Abs(ruby(0,2)/(topaz(aa,bb,2)-topaz(aa,bb-1,2))*sapphire(aa,bb,j,5))
        'dp re-scale
        emerald(i, j, 6) = Abs((topaz(aa, bb, 2) / ruby(0, 2)) * emerald(i, j, 7))
    End Select
  Next j
Next i
```

End Sub

10.6 Calculate

This subroutine is the one in charge of calculating the superposition time and a multiplier factor, which is used for faster calculations or to avoid mathematical errors due to number being too big or small.

Public Sub Calculate()

```
Dim aa As Byte, bb As Integer

For i = 1 To ruby(0, 1)
  aa = ruby(i, 1) 'Number of well
  bb = ruby(i, 2) 'No. test
```

```
   For j = 1 To topaz(aa, bb, 9)
      Call Superposition(bb, sapphire(aa, bb, j, 2), aa)
      emerald(i, j, 3) = summat 'To save value obtained in superposition
   Next j
   'Calculating factor
   ruby(i, 3) = 1 'Saves adjust factor by default
   If Abs(emerald(i, topaz(aa, bb, 9), 3)) > Abs(emerald(i, 1, 3)) Then
      diff = Abs(emerald(i, topaz(aa, bb, 9), 3))
   Else
      diff = Abs(emerald(i, 1, 3))
   End If
   dtmax = diff
   Do While diff > 9
      ruby(i, 3) = ruby(i, 3) * 10
      diff = dtmax / ruby(i, 3)
   Loop
Next i
ReDim quartz(ruby(0, 1), 6)

End Sub
```

10.7 Calculateb

This subroutine calculates the derivative with the smoother assigned by default or by the user. The derivative is calculated with the superposition time and the pressure difference with the time 0.

```
Public Sub Calculateb()

Dim aa As Byte, bb As Integer, cc As Integer, diff, dd As Integer, dtmin, dpmin
Dim dtmax, dpmax, agata As Byte

For i = 1 To ruby(0, 1)
   aa = ruby(i, 1) 'Number of well
   bb = ruby(i, 2) ' No. test
   'Defining left and right smoother
   For j = 1 To topaz(aa, bb, 9) - 1
      cc = j - 1 'Initial guess left side
      dd = j + 1 'Initial guess right side
      'Calculating left side
      If j = 1 Then
         agata = 0
         diff = emerald(i, j, 1)
         dtmin = 10 ^ (-agata)
         Do While dtmin > diff
            agata = agata + 1
```

```
            dtmin = 10 ^ (-agata) 'Calculating lowest value
        Loop
        Do
            'Simple interpolation with dt segment and dp(dt=0)
            Call Interpolation(sapphire(aa, bb, j, 6), sapphire(aa, bb - 1, topaz(aa, bb - 1, 9), 6), _
emerald(i, j, 1), 0, dtmin)
            dpmin = interp
            'dpmin = sapphire(aa, bb, j, 6) - (emerald(i, j, 1) - dtmin) * ((sapphire(aa, bb, j, 6) -__
sapphire(aa, bb - 1, topaz(aa, bb - 1, 9), 6)) / (emerald(i, j, 1) - emerald(i, j - 1, 1)))
            dtmin = dtmin + sapphire(aa, bb - 1, topaz(aa, bb - 1, 9), 2)
            'Calculate superposition
            Call Superposition(bb, dtmin, aa)
            dtmin = (sapphire(aa, bb, j, 2) - dtmin) / 2
            diff = Abs(summat - emerald(i, topaz(aa, bb, 9), 3)) / ruby(i, 3)
        Loop While diff > 99
        dtmin = summat / ruby(i, 3)
    Else
        Do
            diff = Log(10 ^ ((emerald(i, j, 3) - emerald(i, cc, 3)) / ruby(i, 3)))
            If Abs(diff) > emerald(i, 0, 1) Then
                Exit Do
            Else
                cc = cc - 1
            End If
        Loop While cc > 1
        'Uses superposition time and dp(dt=0) for calculations
        dpmin = sapphire(aa, bb, cc, 6)
        dtmin = emerald(i, cc, 3) / ruby(i, 3) 'Adjust with factor
    End If
    'Calculating right side
    Do
        diff = Log(10 ^ ((emerald(i, dd, 3) / emerald(i, j, 3)) / ruby(i, 3)))
        If Abs(diff) > emerald(i, 0, 1) Then
            Exit Do
        Else
            dd = dd + 1
        End If
    Loop While dd < topaz(aa, bb, 9) - 1
    dpmax = sapphire(aa, bb, dd, 6)
    dtmax = emerald(i, dd, 3) / ruby(i, 3) 'Adjust with factor
    emerald(i, j, 3) = emerald(i, j, 3) / ruby(i, 3) 'Adjust with factor
    Call Derivative(dpmin, dtmin, sapphire(aa, bb, j, 6), emerald(i, j, 3), dpmax, dtmax)
    emerald(i, j, 3) = emerald(i, j, 3) * ruby(i, 3) 'Original superposition time before factor
    emerald(i, j, 5) = derev / ruby(i, 3) 'Derivative corrected by factor
    'Calculate derivative normalized
    agata = topaz(aa, bb, 3)
    'Normalizing derivative
```

```
    Select Case agata
      Case 1
        'Injection case
        emerald(i, j, 8) = Abs(ruby(0, 2) / topaz(aa, bb, 2)) * emerald(i, j, 5)
      Case 2
        'Fall-off case
        emerald(i, j, 8) = Abs(ruby(0, 2) / topaz(aa, bb - 1, 2)) * emerald(i, j, 5)
      Case 3
        'Production case
        emerald(i, j, 8) = Abs(ruby(0, 2) / topaz(aa, bb, 2)) * emerald(i, j, 5)
    End Select
  Next j
Next i
End Sub
```

10.8 Superposition

This subroutine calculates the sum used in the superposition time.

```
Public Sub Superposition(maxs As Integer, timew, wellno As Byte)

Dim deltaq, deltat

summat = 0: ctn = 1
Do
  deltaq = topaz(wellno, ctn, 2) - topaz(wellno, ctn - 1, 2)
  deltat = Log(timew - sapphire(wellno, ctn - 1, topaz(wellno, ctn - 1, 9), 2)) / Log(10)
  summat = summat + deltaq * deltat
  ctn = ctn + 1
Loop While ctn <= maxs
If topaz(wellno, maxs, 3) <> 2 Then
  summat = summat / topaz(wellno, maxs, 2) 'Superposition for injection/production
Else
  'superposition for Fall-off
  summat = summat / Abs(topaz(wellno, maxs, 2) - topaz(wellno, maxs - 1, 2))
End If

End Sub
```

10.9 Derivative

This subroutine calculates the 3-point derivative with smoother and normalize it with the reference test set up before in the Reference userform.

```
Public Sub Derivative(dpl, dtl, dpj, dtj, dpr, dtr)
```

```
Dim term1, term2, term3, term4

term1 = (dpj - dpl) / Log(10 ^ (dtj - dtl))
term2 = Log(10 ^ (dtr - dtj)) / Log(10 ^ (dtr - dtl))
term3 = (dpr - dpj) / Log(10 ^ (dtr - dtj))
term4 = Log(10 ^ (dtj - dtl)) / Log(10 ^ (dtr - dtl))
derev = Abs(term1 * term2 + term3 * term4)

End Sub
```

10.10 Interpolation

This subroutine is used for interpolating values when need it.

```
Public Sub Interpolation(vv1, vv2, uu1, uu2, uu)
interp = vv2 - (vv2 - vv1) * (uu2 - uu) / (uu2 - uu1)

End Sub
```

10.11 Comparative

This subroutine calls the subroutines that calculate permeabilities and skin factor by semi-log and derivative analysis.

```
Public Sub Comparative()

For i = 1 To ruby(0, 1)
   'Calculate limits based on tolerance
   Call Limits(i, 1)
   'Calculate permeability, skin factor
   Call Propertyks(ruby(i, 1), i, ruby(i, 2), quartz(i, 1), quartz(i, 2))
Next i

End Sub
```

10.12 Limits

This subroutine calculates the limits of the radial flow to use for calculating the flow capacity, permeability and skin factors.

```
Public Sub Limits(testno, tole)
```

```
Dim dref, bb As Integer, pfixed, j As Integer, diff

'Initial point
dref = emerald(testno, ruby(testno, 5), 1) - emerald(testno, 1, 1)
j = 1
If emerald(testno, j, 1) = 0 Then
   j = j + 1 'To avoid zero from the initial value
End If
Do
   If dref > 2 Then
      'Calculate slope every 15 min, for faster calculations and jump wellbore storage
      bb = j
      Do
         j = j + 1
         diff = emerald(testno, j, 1) - emerald(testno, bb, 1)
         If j >= ruby(testno, 5) - 1 Then
            j = ruby(testno, 5) - 1
            Exit Do 'To avoid infinite loop
         End If
      Loop While diff < 0.25 'Every 15 min
   Else
      bb = j
      j = j + 1
   End If
   pfixed = Abs((emerald(testno, j, 8) - emerald(testno, bb, 8)) / emerald(testno, bb, 8))
   pfixed = pfixed * 100
   If pfixed < ruby(testno, 4) Then
      Exit Do
   End If
   If j >= ruby(testno, 5) - 2 Then
      If tole <> 0 Then
         'For recalculating tolerance
         j = 1
         ruby(testno, 4) = ruby(testno, 4) + 0.1 'Increases error
      Else
         'For using fix tolerance
         j = ruby(testno, 5) - 2
         Exit Do
      End If
   End If
Loop While j < ruby(testno, 5) - 1
quartz(testno, 1) = j 'Initial point

'End point
Do
   j = j + 1
   pfixed = Abs((emerald(testno, j, 8) - emerald(testno, quartz(testno, 1), 8)) / emerald(testno,_
```

```
quartz(testno, 1), 8)) * 100
   If pfixed > 4 Then
      Exit Do
   End If
Loop While j < ruby(testno, 5) - 1
quartz(testno, 2) = j 'End point

End Sub
```

10.13 Propertyks

This subroutine calculates the flow capacity, permeability and skin factor with semi-log and derivative analysis, with the limits calculated in “Limits” or the ones assigned by the user.

```
Public Sub Propertyks(aa, testno, bb, lptn, rptn)

Dim agl, agr, hyddif, slope, p1hr, st1hr, pfixed, agata As Integer, cc As Integer, dd As Integer

cc = lptn: dd = rptn
'Calculate constant value for derivative
pfixed = 0: agata = 0
For k = cc To dd
   pfixed = pfixed + emerald(testno, k, 8) 'summatory
   agata = agata + 1
Next k
pfixed = pfixed / agata 'Average value for derivative constant

'Calculating permeabilities
If topaz(aa, bb, 3) <> 2 Then
   slope = Abs((emerald(testno, dd, 7) - emerald(testno, cc, 7)) / (Log(emerald(testno, dd, 1) /_
emerald(testno, cc, 1)) / Log(10))) 'Semilog
   ruby(testno, 6) = Abs((162.5683 * topaz(aa, bb, 2) * bo(aa) * muo(aa)) / slope) 'kh for semilog
   ruby(testno, 7) = ruby(testno, 6) / hgt(aa) 'k for semilog
   ruby(testno, 9) = Abs((70.6 * topaz(aa, bb, 2) * bo(aa) * muo(aa)) / pfixed) 'kh for derivative
   ruby(testno, 10) = ruby(testno, 9) / hgt(aa) 'k for derivative
Else
   agl = emerald(testno, cc, 1) * topaz(aa, bb, 10) / (emerald(testno, cc, 1) + topaz(aa, bb, 10))
   agr = emerald(testno, dd, 1) * topaz(aa, bb, 10) / (emerald(testno, dd, 1) + topaz(aa, bb, 10))
   slope = Abs((emerald(testno, dd, 7) - emerald(testno, cc, 7)) / (Log(agr / agl) / Log(10))) 'Semilog
   ruby(testno, 6) = Abs((162.5683 * topaz(aa, bb - 1, 2) * bo(aa) * muo(aa)) / slope) 'kh for semilog
   ruby(testno, 7) = ruby(testno, 6) / hgt(aa) 'k for semilog
   ruby(testno, 9) = Abs((70.6 * topaz(aa, bb - 1, 2) * bo(aa) * muo(aa)) / pfixed) 'kh for derivative
   ruby(testno, 10) = ruby(testno, 9) / hgt(aa) 'k for derivative
End If
```

```
'Calculating skin for semilog
p1hr = Abs(emerald(testno, dd, 7) - slope * Log(emerald(testno, dd, 1)) / Log(10))
hyddif = ruby(testno, 7) / (poro(aa) * muo(aa) * ct(aa) * rw(aa) * rw(aa))
hyddif = Log(hyddif) / Log(10)
If topaz(aa, bb, 3) <> 2 Then
   ruby(testno, 8) = 1.151 * (Abs((p1hr - 0) / slope) - hyddif + 3.2275) 'skin for semilog
Else
   st1hr = 1 * topaz(aa, bb, 10) / (1 + topaz(aa, bb, 10)) 'Agarwal time
   sthr1 = Log(st1hr) / Log(10)
   ruby(testno, 8) = 1.151 * (Abs((p1hr - 0) / slope) - st1hr - hyddif + 3.2275) 'skin for semilog
End If

'Calculating skin for derivative
dd = cc
Do
   If dd >= ruby(testno, 5) - 1 Then
      dd = ruby(testno, 5) - 1
      Exit Do
   End If
   dd = dd + 1 'Right side of the derivative average
Loop While emerald(testno, dd, 8) < pfixed

cc = dd - 1 'Left side of the derivative average
Call Interpolation(emerald(testno, cc, 7), emerald(testno, dd, 7), emerald(testno, cc, 8),_
emerald(testno, dd, 8), pfixed)
p1hr = interp
hyddif = ruby(testno, 10) / (poro(aa) * muo(aa) * bo(aa) * ct(aa) * rw(aa) * rw(aa))
hyddif = Log(hyddif) / Log(10)
If topaz(aa, bb, 3) <> 2 Then
   Call Interpolation(emerald(testno, cc, 1), emerald(testno, dd, 1), emerald(testno, cc, 8),_
emerald(testno, dd, 8), pfixed)
   st1hr = Log(interp) / Log(10)
   slope = Abs((p1hr * ruby(testno, 9)) / (162.5683 * topaz(aa, bb, 2) * bo(aa) * muo(aa)))
   ruby(testno, 11) = 1.151 * (slope - st1hr - hyddif + 3.2275) 'skin factor for derivative
Else
   agr = emerald(testno, dd, 1) * topaz(aa, bb, 10) / (emerald(testno, dd, 1) + topaz(aa, bb, 10))
   agl = emerald(testno, cc, 1) * topaz(aa, bb, 10) / (emerald(testno, cc, 1) + topaz(aa, bb, 10))
   Call Interpolation(agl, agr, emerald(testno, cc, 8), emerald(testno, dd, 8), pfixed)
   st1hr = Log(interp) / Log(10)
   slope = Abs((p1hr * ruby(testno, 9)) / (162.5683 * topaz(aa, bb - 1, 2) * bo(aa) * muo(aa)))
   ruby(testno, 11) = 1.151 * (slope - st1hr - hyddif + 3.2275) 'skin factor for derivative
End If
```

End Sub

10.14 Genesis

This subroutine creates anew sheet in excel, which will be containing the results from the derivative, and renamed as “Analysis”

Public Sub Genesis()

```
Dim aa As Byte, tempo As String, maxy As Byte, bb As Byte

Application.ScreenUpdating = False 'Faster calculations

With ThisWorkbook
   Set ws = .Sheets.Add(After:=.Sheets(.Sheets.Count))
   aa = 1
   For i = 1 To Sheets.Count
      If Left(Sheets(i).Name, 8) = "Analysis" Then
         bb = Len(Sheets(i).Name)
         maxy = Val(Right(Sheets(i).Name, bb - 9))
         If maxy >= aa Then
            aa = maxy + 1
         End If
      End If
   Next i
   tempo = "Analysis_" & aa
   .Unprotect
   ws.Name = tempo
End With

ws.Visible = xlSheetVeryHidden '-------------------------Important to work in background
Application.ScreenUpdating = True 'Faster calculations
```
End Sub

10.15 Printing

This subroutine prints the results in the sheet Analysis, that are going to be used for generating the graph in the Plotter userform, see Appendix C for its’ code. It also prints the unnormalized data in case one wants to check it.

Public Sub Printing()

```
Dim bb As Integer, miny, maxy

Application.ScreenUpdating = False 'Faster calculations
ThisWorkbook.Unprotect
ws.Activate 'To change activation for new sheet
```

```
bb = 0
For i = 1 To ruby(0, 1)
   'For the Normalized data headers
   ws.Cells(2 + bb, 1) = topaz(ruby(i, 1), ruby(i, 2), 7) 'Print name of the test
   ws.Cells(2 + bb, 2) = "Normalized"
   ws.Cells(3 + bb, 1) = "dTime (hrs)"
   ws.Cells(3 + bb, 1).WrapText = True
   ws.Cells(3 + bb, 2) = "p-p@dt=0 (psia)"
   ws.Cells(3 + bb, 2).WrapText = True
   ws.Cells(3 + bb, 3) = "Derivative (psia)"
   ws.Cells(3 + bb, 3).WrapText = True
   ws.Cells(3 + bb, 4) = "Superposition time()"
   ws.Cells(3 + bb, 4).WrapText = True
   ws.Cells(3 + bb, 5) = "p (psia)"
   ws.Cells(3 + bb, 5).WrapText = True
   'For the unnormalized headers
   ws.Cells(2 + bb, 8) = topaz(ruby(i, 1), ruby(i, 2), 7) 'Print name of the test
   ws.Cells(2 + bb, 9) = "Unnormalized"
   ws.Cells(3 + bb, 8) = "dTime (hrs)"
   ws.Cells(3 + bb, 8).WrapText = True
   ws.Cells(3 + bb, 9) = "p-p@dt=0 (psia)"
   ws.Cells(3 + bb, 9).WrapText = True
   ws.Cells(3 + bb, 10) = "Derivative (psia)"
   ws.Cells(3 + bb, 10).WrapText = True
   ws.Cells(3 + bb, 11) = "Superposition time()"
   ws.Cells(3 + bb, 11).WrapText = True
   ws.Cells(3 + bb, 12) = "p (psia)"
   ws.Cells(3 + bb, 12).WrapText = True
   'To print results for emerald matrix
   For j = 1 To ruby(i, 5)
      ws.Cells(3 + bb + j, 1) = emerald(i, j, 1) 'dt segment
      ws.Cells(3 + bb + j, 2) = emerald(i, j, 7) 'Pressure derivative normalized
      ws.Cells(3 + bb + j, 3) = emerald(i, j, 8) 'Derivative normalized
      ws.Cells(3 + bb + j, 4) = emerald(i, j, 3) 'Superposition time
      ws.Cells(3 + bb + j, 5) = emerald(i, j, 4) 'Pressure superposition normalized
      ws.Cells(3 + bb + j, 8) = emerald(i, j, 1) 'dt segment
      ws.Cells(3 + bb + j, 9) = emerald(i, j, 6) 'Pressure derivative
      ws.Cells(3 + bb + j, 10) = emerald(i, j, 5) 'Derivative
      ws.Cells(3 + bb + j, 11) = emerald(i, j, 3) 'Superposition time
      ws.Cells(3 + bb + j, 12) = emerald(i, j, 2) 'Pressure superposition
   Next j
   Columns("A:E").EntireColumn.AutoFit
   Columns("H:L").EntireColumn.AutoFit
   bb = bb + ruby(i, 5) + 5
Next i
```

```
'Calculate max and min in pressure and derivative
miny = 10000000: maxy = 0
For i = 1 To ruby(0, 1)
   For j = 2 To ruby(i, 5) - 1
      If emerald(i, j, 7) > maxy Then
         maxy = emerald(i, j, 7)
      End If
      If emerald(i, j, 8) > maxy Then
         maxy = emerald(i, j, 8)
      End If
      If miny > emerald(i, j, 7) Then
         miny = emerald(i, j, 7)
      End If
      If miny > emerald(i, j, 8) Then
         miny = emerald(i, j, 8)
      End If
   Next j
Next i

bb = 0
For i = 1 To ruby(0, 1)
   ws.Cells(10 + bb, 15) = "Xleft"
   ws.Cells(11 + bb, 15) = emerald(i, quartz(i, 1), 1) 'Left point
   ws.Cells(12 + bb, 15) = emerald(i, quartz(i, 1), 1) 'Left point
   ws.Cells(10 + bb, 16) = "Y"
   ws.Cells(11 + bb, 16) = miny
   ws.Cells(12 + bb, 16) = maxy
   ws.Cells(10 + bb, 17) = "Xright"
   ws.Cells(11 + bb, 17) = emerald(i, quartz(i, 2), 1) 'Right point
   ws.Cells(12 + bb, 17) = emerald(i, quartz(i, 2), 1) 'Right point
   'Generating random colors
   quartz(i, 3) = WorksheetFunction.RandBetween(0, 255) 'Red color
   quartz(i, 4) = WorksheetFunction.RandBetween(0, 255) 'Green color
   quartz(i, 5) = WorksheetFunction.RandBetween(0, 255) 'Blue color
   quartz(i, 6) = 11 + bb 'Position of the lines
   bb = bb + ruby(i, 5) + 5
Next i
Application.ScreenUpdating = True 'Faster calculations

End Sub
```

10.16 Gengraph

This subroutine generates the graph in excel and export it to the image box in the Plotter userform. It also assigned random colors to the lines, the format and names shown in the legend.

Public Sub Gengraph()

```
Dim Mychart As Object, aa As Integer, bb As Integer

Application.ScreenUpdating = False 'To create the graph but not generating, it makes it faster
'Detect the workbook with the data for graphs
With ThisWorkbook
   Set ws = .Sheets(.Sheets.Count)
End With
ws.Activate
Set Mychart = ActiveSheet.Shapes.AddChart(xlXYScatterLinesNoMarkers, width:=600,
height:=400).Chart

aa = 4
For i = 1 To 4
   Mychart.SeriesCollection(aa).Delete 'since is creating a graph with predefined series
   aa = aa - 1
Next i

'Adding the pressure part
aa = 4: bb = 0
For i = 1 To ruby(0, 1)
   bb = bb + 1
   'Adding the pressure part
   Mychart.SeriesCollection.NewSeries 'Add new series
   Mychart.SeriesCollection(bb).Name = topaz(ruby(i, 1), ruby(i, 2), 7) 'Add the name of the series
   'X values
   Mychart.SeriesCollection(bb).XValues = ws.Range(Cells(aa, 1), Cells(aa + ruby(i, 5) - 1, 1))
   Mychart.SeriesCollection(bb).Values = ws.Range(Cells(aa, 2), Cells(aa + ruby(i, 5) - 1, 2)) 'Y values
   'Choose color
   Mychart.SeriesCollection(bb).Border.Color = RGB(quartz(i, 3), quartz(i, 4), quartz(i, 5))
   'Choose color
   Mychart.SeriesCollection(bb).Interior.Color = RGB(quartz(i, 3), quartz(i, 4), quartz(i, 5))
   aa = aa + ruby(i, 5) + 5
Next i
aa = 4
For i = 1 To ruby(0, 1)
   bb = bb + 1
   'Adding derivative part
   Mychart.SeriesCollection.NewSeries 'Add new series
   Mychart.SeriesCollection(bb).Name = topaz(ruby(i, 1), ruby(i, 2), 7) 'Add the name of the series
    'X values
   Mychart.SeriesCollection(bb).XValues = ws.Range(Cells(aa, 1), Cells(aa + ruby(i, 5) - 1, 1))
   Mychart.SeriesCollection(bb).Values = ws.Range(Cells(aa, 3), Cells(aa + ruby(i, 5) - 1, 3)) 'Y values
   'Choose color
```

```
   Mychart.SeriesCollection(bb).Border.Color = RGB(quartz(i, 3), quartz(i, 4), quartz(i, 5))
   'Choose color
   Mychart.SeriesCollection(bb).Interior.Color = RGB(quartz(i, 3), quartz(i, 4), quartz(i, 5))
   aa = aa + ruby(i, 5) + 5
Next i
'For adding the boundaries, add two lines for the first test
For i = 1 To 2
   bb = bb + 1
   Mychart.SeriesCollection.NewSeries 'Limit line
   Mychart.SeriesCollection(bb).Name = "Limit" & i
   Mychart.SeriesCollection(bb).XValues = ws.Range(Cells(quartz(tt, 6), 13 + i * 2), _
  Cells(quartz(tt, 6) + 1, 13 + i * 2)) 'X values
   Mychart.SeriesCollection(bb).Values = ws.Range(Cells(quartz(tt, 6), 16), _
  Cells(quartz(tt, 6) + 1, 16)) 'Y values
   Mychart.SeriesCollection(bb).Border.LineStyle = xlDot
   'Choose color
   Mychart.SeriesCollection(bb).Border.Color = RGB(quartz(tt, 3), quartz(tt, 4), quartz(tt, 5))
   'Choose color
   Mychart.SeriesCollection(bb).Interior.Color = RGB(quartz(tt, 3), quartz(tt, 4), quartz(tt, 5))
Next i

'Mychart.SeriesCollection(tt).Format.Line.Weight = 2
Mychart.SeriesCollection(tt).MarkerStyle = xlMarkerStyleCircle
Mychart.SeriesCollection(tt).MarkerSize = 7
Mychart.SeriesCollection(tt).MarkerBackgroundColor = RGB(quartz(tt, 3), quartz(tt, 4),_
quartz(tt, 5))
Mychart.SeriesCollection(tt).MarkerForegroundColor = RGB(0, 0, 0)
Mychart.SeriesCollection(tt + ruby(0, 1)).MarkerStyle = xlMarkerStyleCircle
Mychart.SeriesCollection(tt + ruby(0, 1)).MarkerSize = 7
Mychart.SeriesCollection(tt + ruby(0, 1)).MarkerBackgroundColor = RGB(quartz(tt, 3),_
quartz(tt, 4), quartz(tt, 5))
Mychart.SeriesCollection(tt + ruby(0, 1)).MarkerForegroundColor = RGB(0, 0, 0)

'For editing graph
Mychart.HasTitle = True 'Add title
Mychart.ChartTitle.Text = ws.Name 'Title name
Mychart.ChartTitle.Font.Size = 16 'Font size
Mychart.Axes(xlCategory, xlPrimary).HasTitle = True 'Add X-axis title
Mychart.Axes(xlCategory, xlPrimary).AxisTitle.Text = "Time (hrs)" 'X name
'font size
Mychart.Axes(xlCategory, xlPrimary).AxisTitle.Format.TextFrame2.TextRange.Font.Size = 12
Mychart.Axes(xlCategory, xlPrimary).ScaleType = xlScaleLogarithmic 'Log scale
Mychart.Axes(xlCategory, xlPrimary).HasMajorGridlines = True
Mychart.Axes(xlValue, xlPrimary).HasTitle = True 'Add Y-axis title
Mychart.Axes(xlValue, xlPrimary).AxisTitle.Text = "Pressure and Derivative (psi)" 'Y name
Mychart.Axes(xlValue, xlPrimary).AxisTitle.Format.TextFrame2.TextRange.Font.Size = 12 'font size
Mychart.Axes(xlValue, xlPrimary).ScaleType = xlScaleLogarithmic 'Log scale
```

```
Mychart.Axes(xlValue, xlPrimary).HasMajorGridlines = True

'Small number so x axis crosses the y at 0.000000001
Mychart.Axes(xlValue, xlPrimary).CrossesAt = 0.0000000001
'Automatically changes the x axis cross to minimum value
Mychart.Axes(xlValue, xlPrimary).Crosses = xlCustom
'Small number so y axis crosses the x at 0.000000001
Mychart.Axes(xlCategory, xlPrimary).CrossesAt = 0.0000000001
'Automatically changes the y axis cross to minimum value
Mychart.Axes(xlCategory, xlPrimary).Crosses = xlCustom

'Now for generating the image
Dim imagename As String

imagename = Application.DefaultFilePath & Application.PathSeparator & "Tempchart.gif"
'MsgBox "The default path is" & Application.DefaultFilePath 'Just to know where image is save
Mychart.Export Filename:=imagename
ws.ChartObjects(1).Delete
Application.ScreenUpdating = True
Plotter.Image1.Picture = LoadPicture(imagename)

End Sub
```

10.17 Allgraph

This subroutine generates the final graphs for pressure and pressure derivative and the superposition graph, after the manual or automatic adjustments.

```
Public Sub Allgraph()

Dim aa As Integer, bb As Integer

Application.ScreenUpdating = False 'To create the graph but not generating, it makes it faster
'Detect the workbook with the data for graphs
With ThisWorkbook
   Set ws = .Sheets(.Sheets.Count)
End With
ws.Activate
'Creating pressure and pressure derivative graph
ActiveSheet.Shapes.AddChart2(240, xlXYScatterLinesNoMarkers).Select

aa = 4
For i = 1 To 4
   ActiveChart.FullSeriesCollection(aa).Delete 'since is creating a graph with predefined series
   aa = aa - 1
```

```
Next i

'Adding the pressure part
aa = 4: bb = 0
For i = 1 To ruby(0, 1)
   bb = bb + 1
   'Adding the pressure part
   ActiveChart.SeriesCollection.NewSeries 'Add new series
    'Add the name of the series
   ActiveChart.FullSeriesCollection(bb).Name = topaz(ruby(i, 1), ruby(i, 2), 7)
   'X values
   ActiveChart.FullSeriesCollection(bb).XValues = ws.Range(Cells(aa, 1), Cells(aa + ruby(i, 5) - 1, 1))
   'Y values
   ActiveChart.FullSeriesCollection(bb).Values = ws.Range(Cells(aa, 2), Cells(aa + ruby(i, 5) - 1, 2))
   'Choose color
   ActiveChart.FullSeriesCollection(bb).Border.Color = RGB(quartz(i, 3), quartz(i, 4), quartz(i, 5))
  'Choose color
   ActiveChart.FullSeriesCollection(bb).Interior.Color = RGB(quartz(i, 3), quartz(i, 4), quartz(i, 5))
   aa = aa + ruby(i, 5) + 5
Next i
aa = 4
For i = 1 To ruby(0, 1)
   bb = bb + 1
   'Adding derivative part
   ActiveChart.SeriesCollection.NewSeries 'Add new chart
   'Add the name of the series
   ActiveChart.FullSeriesCollection(bb).Name = topaz(ruby(i, 1), ruby(i, 2), 7)
   'X values
   ActiveChart.FullSeriesCollection(bb).XValues = ws.Range(Cells(aa, 1), Cells(aa + ruby(i, 5) - 1, 1))
  'Y values
   ActiveChart.FullSeriesCollection(bb).Values = ws.Range(Cells(aa, 3), Cells(aa + ruby(i, 5) - 1, 3))
  'Choose color
   ActiveChart.FullSeriesCollection(bb).Border.Color = RGB(quartz(i, 3), quartz(i, 4), quartz(i, 5))
   'Choose color
   ActiveChart.FullSeriesCollection(bb).Interior.Color = RGB(quartz(i, 3), quartz(i, 4), quartz(i, 5))
   aa = aa + ruby(i, 5) + 5
Next i

'For editing graph
ActiveChart.HasTitle = True 'Add title
ActiveChart.ChartTitle.Text = ws.Name 'Title name
ActiveChart.ChartTitle.Font.Size = 20 'Font size
ActiveChart.Axes(xlCategory, xlPrimary).HasTitle = True 'Add X-axis title
ActiveChart.Axes(xlCategory, xlPrimary).AxisTitle.Text = "Time (hrs)" 'X name
'font size
ActiveChart.Axes(xlCategory, xlPrimary).AxisTitle.Format.TextFrame2.TextRange.Font.Size = 14
ActiveChart.Axes(xlCategory, xlPrimary).ScaleType = xlScaleLogarithmic 'Log scale
```

```
ActiveChart.Axes(xlCategory, xlPrimary).HasMajorGridlines = True
ActiveChart.Axes(xlValue, xlPrimary).HasTitle = True 'Add Y-axis title
ActiveChart.Axes(xlValue, xlPrimary).AxisTitle.Text = "Pressure and Derivative (psi)" 'Y name
'font size
ActiveChart.Axes(xlValue, xlPrimary).AxisTitle.Format.TextFrame2.TextRange.Font.Size = 14
ActiveChart.Axes(xlValue, xlPrimary).ScaleType = xlScaleLogarithmic 'Log scale
ActiveChart.Axes(xlValue, xlPrimary).HasMajorGridlines = True
'Small number so y axis crosses the x at 0.000000001
ActiveChart.Axes(xlCategory, xlPrimary).CrossesAt = 0.0000000001
'Automatically changes the y axis cross to minimum value
ActiveChart.Axes(xlCategory, xlPrimary).Crosses = xlCustom
ActiveChart.SetElement (msoElementLegendRight)

ThisWorkbook.Unprotect
ActiveChart.Location Where:=xlLocationAsNewSheet, Name:="Der_" & ws.Name

'Creating superposition graph
ws.Activate
ActiveSheet.Shapes.AddChart2(240, xlXYScatterLinesNoMarkers).Select

aa = 4
For i = 1 To 4
   ActiveChart.FullSeriesCollection(aa).Delete 'since is creating a graph with predefined series
   aa = aa - 1
Next i

'Adding the superposition time
aa = 4: bb = 0
For i = 1 To ruby(0, 1)
   bb = bb + 1
   ActiveChart.SeriesCollection.NewSeries 'Add new series
  'Add the name of the series
   ActiveChart.FullSeriesCollection(bb).Name = topaz(ruby(i, 1), ruby(i, 2), 7)
  'X values
   ActiveChart.FullSeriesCollection(bb).XValues = ws.Range(Cells(aa, 4), Cells(aa + ruby(i, 5) - 1, 4))
  'Y values
   ActiveChart.FullSeriesCollection(bb).Values = ws.Range(Cells(aa, 5), Cells(aa + ruby(i, 5) - 1, 5))
    'Choose color
   ActiveChart.FullSeriesCollection(bb).Border.Color = RGB(quartz(i, 3), quartz(i, 4), quartz(i, 5))
  'Choose color
   ActiveChart.FullSeriesCollection(bb).Interior.Color = RGB(quartz(i, 3), quartz(i, 4), quartz(i, 5))
   aa = aa + ruby(i, 5) + 5
Next i

'For editing graph
ActiveChart.HasTitle = True 'Add title
ActiveChart.ChartTitle.Text = ws.Name 'Title name
```

```
ActiveChart.ChartTitle.Font.Size = 20 'Font size
ActiveChart.Axes(xlCategory, xlPrimary).HasTitle = True 'Add X-axis title
ActiveChart.Axes(xlCategory, xlPrimary).AxisTitle.Text = "Superposition time" 'X name
'font size
ActiveChart.Axes(xlCategory, xlPrimary).AxisTitle.Format.TextFrame2.TextRange.Font.Size = 14
ActiveChart.Axes(xlCategory, xlPrimary).HasMajorGridlines = True
ActiveChart.Axes(xlValue, xlPrimary).HasTitle = True 'Add Y-axis title
ActiveChart.Axes(xlValue, xlPrimary).AxisTitle.Text = "Pressure (psi)" 'Y name
'font size
ActiveChart.Axes(xlValue, xlPrimary).AxisTitle.Format.TextFrame2.TextRange.Font.Size = 14
ActiveChart.Axes(xlValue, xlPrimary).HasMajorGridlines = True
'Small number so y axis crosses the x at 0.000000001
ActiveChart.Axes(xlCategory, xlPrimary).CrossesAt = -100
'Automatically changes the y axis cross to minimum value
ActiveChart.Axes(xlCategory, xlPrimary).Crosses = xlCustom
ActiveChart.SetElement (msoElementLegendRight)

Application.ScreenUpdating = True 'For faster operations
ThisWorkbook.Unprotect
ActiveChart.Location Where:=xlLocationAsNewSheet, Name:="Spt_" & ws.Name

End Sub
```

10.18 Summaries

This subroutine creates a new sheet and named it “Summary” and will print on it the results from the flow capacity, permeability and skin factor calculated in the plotter userform.

```
Public Sub Summaries()

Dim aa As Byte, tempo As String, maxy As Byte, bb As Integer

Application.ScreenUpdating = False 'Faster calculations
tempo = ws.Name
bb = Len(tempo)
aa = Val(Right(tempo, bb - 9)) 'to extract the number
With ThisWorkbook
   Set ws = .Sheets.Add(After:=.Sheets(.Sheets.Count))
   tempo = "Summary_" & aa
   .Unprotect
   ws.Name = tempo
End With
ws.Activate

bb = 0
```

```
For i = 1 To ruby(0, 1)
   'To print results from ruby matrix
   ws.Cells(2 + bb, 2) = topaz(ruby(i, 1), ruby(i, 2), 7) 'Print name of the test
   ws.Cells(3 + bb, 2) = "Kh (mD*ft)"
   ws.Cells(4 + bb, 2) = "K (mD)"
   ws.Cells(5 + bb, 2) = "Skin"
   ws.Cells(2 + bb, 3) = "Semilog method"
   ws.Cells(3 + bb, 3) = ruby(i, 6) 'Kh semi
   ws.Cells(4 + bb, 3) = ruby(i, 7) 'K semi
   ws.Cells(5 + bb, 3) = ruby(i, 8) 'S semi
   ws.Cells(2 + bb, 4) = "Derivative method"
   ws.Cells(3 + bb, 4) = ruby(i, 9) 'Kh der
   ws.Cells(4 + bb, 4) = ruby(i, 10) 'K der
   ws.Cells(5 + bb, 4) = ruby(i, 11) 'S der
   ws.Cells(2 + bb, 5) = "Average"
   ws.Cells(3 + bb, 5) = (ruby(i, 6) + ruby(i, 9)) / 2 'Average kh
   ws.Cells(4 + bb, 5) = (ruby(i, 7) + ruby(i, 10)) / 2 'Average k
   ws.Cells(5 + bb, 5) = (ruby(i, 8) + ruby(i, 11)) / 2 'Average s
   ws.Cells(2 + bb, 6) = "Smoother"
   ws.Cells(3 + bb, 6) = emerald(i, 0, 1) 'Final smoother
   Columns("B:D").EntireColumn.AutoFit
   bb = bb + 6
Next i
Application.ScreenUpdating = True 'Faster calculations
End Sub
```

10.19 Cleaning

This subroutine will reset the main sheets in the excel file as a new project by deleting all the data contained but not the summaries, analysis and graphs from previous projects.

```
Public Sub Cleaning()

Dim aa As Byte, bb As Integer

Application.ScreenUpdating = False 'Faster calculations
Worksheets(1).Unprotect Password:="crystal"
Worksheets(1).Cells(3, 4).Locked = False
Worksheets(1).Cells(3, 4).Interior.ColorIndex = 6
Worksheets(1).Shapes(1).TextFrame.Characters.Text = "Generate"
Worksheets(2).Unprotect Password:="crystal"
Worksheets(3).Unprotect Password:="crystal"

ww = Worksheets(1).Cells(3, 4)
```

```
aa = 0
For i = 1 To ww
   Worksheets(2).Cells(2, 1 + 4 * aa).Clear
   Worksheets(2).Cells(3, 1 + 4 * aa).Clear
   Worksheets(2).Cells(3, 2 + 4 * aa).Clear
   Worksheets(3).Cells(2, 1 + 4 * aa).Clear
   Worksheets(3).Cells(3, 1 + 4 * aa).Clear
   Worksheets(3).Cells(3, 2 + 4 * aa).Clear
   Worksheets(1).Cells(8, 2 + 4 * aa).Clear
   Worksheets(1).Cells(9, 2 + 4 * aa).Clear
   Worksheets(1).Cells(9, 3 + 4 * aa).Clear
   Worksheets(1).Cells(10, 2 + 4 * aa).Clear
   Worksheets(1).Cells(10, 3 + 4 * aa).Clear
   Worksheets(1).Cells(11, 2 + 4 * aa).Clear
   Worksheets(1).Cells(11, 3 + 4 * aa).Clear
   Worksheets(1).Cells(12, 2 + 4 * aa).Clear
   Worksheets(1).Cells(12, 3 + 4 * aa).Clear
   Worksheets(1).Cells(13, 2 + 4 * aa).Clear
   Worksheets(1).Cells(13, 3 + 4 * aa).Clear
   Worksheets(1).Cells(14, 2 + 4 * aa).Clear
   Worksheets(1).Cells(14, 3 + 4 * aa).Clear
   Worksheets(1).Cells(15, 2 + 4 * aa).Clear
   Worksheets(1).Cells(15, 3 + 4 * aa).Clear
   Worksheets(1).Cells(17, 2 + 4 * aa).Clear
   Worksheets(1).Cells(17, 3 + 4 * aa).Clear
   bb = Worksheets(2).Cells(Rows.Count, 1 + 4 * aa).End(xlUp).row 'Last active cell in the column
   For j = 4 To bb
      Worksheets(2).Cells(j, 1 + 4 * aa).Clear 'Clear date
      Worksheets(2).Cells(j, 2 + 4 * aa).Clear 'Clear pressure
   Next j
   bb = Worksheets(3).Cells(Rows.Count, 1 + 4 * aa).End(xlUp).row 'Last active cell in the column
   For j = 4 To bb
      Worksheets(3).Cells(j, 1 + 4 * aa).Clear 'Clear date
      Worksheets(3).Cells(j, 2 + 4 * aa).Clear 'Clear rate
      Worksheets(3).Cells(j, 3 + 4 * aa).Clear 'clear secondary name test
   Next j
   Worksheets(2).Columns(1 + 4 * aa).Resize(, 2).Locked = True
   Worksheets(3).Columns(1 + 4 * aa).Resize(, 2).Locked = True
   aa = aa + 1
Next i

Worksheets(1).Cells(3, 4) = 1
Worksheets(2).Columns(1).Resize(, 2).Locked = False
Worksheets(2).Cells(2, 1) = "Well 1"
Worksheets(2).Cells(2, 1).Locked = True
Worksheets(2).Cells(3, 1) = "Date"
Worksheets(2).Cells(3, 1).Locked = True
```

Worksheets(2).Cells(3, 2) = "Pressure (psia)"
Worksheets(2).Cells(3, 2).WrapText = True
Worksheets(2).Cells(3, 2).Locked = True
Worksheets(3).Columns(1).Resize(, 2).Locked = False
Worksheets(3).Cells(2, 1) = "Well 1"
Worksheets(3).Cells(2, 1).Locked = True
Worksheets(3).Cells(3, 1) = "Time@end"
Worksheets(3).Cells(3, 1).Locked = True
Worksheets(3).Cells(3, 2) = "Liquid rate (STB/D)"
Worksheets(3).Cells(3, 2).WrapText = True
Worksheets(3).Cells(3, 2).Locked = True
Worksheets(1).Cells(8, 2) = "Well 1"
Worksheets(1).Cells(9, 2) = "Pi(psia)"
Worksheets(1).Cells(9, 3).Interior.ColorIndex = 6
Worksheets(1).Cells(9, 3).Locked = False
Worksheets(1).Cells(10, 2) = "Bo(RB/STB)"
Worksheets(1).Cells(10, 3).Interior.ColorIndex = 6
Worksheets(1).Cells(10, 3).Locked = False
Worksheets(1).Cells(11, 2) = "Viscosity(cp)"
Worksheets(1).Cells(11, 3).Interior.ColorIndex = 6
Worksheets(1).Cells(11, 3).Locked = False
Worksheets(1).Cells(12, 2) = "rw(ft)"
Worksheets(1).Cells(12, 3).Interior.ColorIndex = 6
Worksheets(1).Cells(12, 3).Locked = False
Worksheets(1).Cells(13, 2) = "h(ft)"
Worksheets(1).Cells(13, 3).Interior.ColorIndex = 6
Worksheets(1).Cells(13, 3).Locked = False
Worksheets(1).Cells(14, 2) = "Porosity"
Worksheets(1).Cells(14, 3).Interior.ColorIndex = 6
Worksheets(1).Cells(14, 3).Locked = False
Worksheets(1).Cells(15, 2) = "Ct(psia-1)"
Worksheets(1).Cells(15, 3).Interior.ColorIndex = 6
Worksheets(1).Cells(15, 3).Locked = False

Worksheets(1).Protect Password:="crystal", AllowFormattingCells:=True,
AllowFormattingColumns:=True, AllowFormattingRows:=True
Worksheets(2).Protect Password:="crystal", AllowFormattingCells:=True,
AllowFormattingColumns:=True, AllowFormattingRows:=True
Worksheets(3).Protect Password:="crystal", AllowFormattingCells:=True,
AllowFormattingColumns:=True, AllowFormattingRows:=True
Application.ScreenUpdating = True 'Faster calculations

End Sub