University of South-Eastern Norway

www.usn.no

Master's Thesis 2022

Industrial IT and Automation

Multiphase flow metering – estimation of flow velocities of component phases using multimodal sensor suite

Andreas Lund Rasmussen

Faculty of Technology, Natural Sciences, and Maritime Sciences

Campus Porsgrunn

University of South-Eastern Norway

www.usn.no

Course: FMH606 Master's Thesis, 2022

Title: Multiphase flow metering – detection of flow regimes and estimation of flow velocities of component phases using multimodal sensor suite

Number of pages: 194

Keywords: Accelerometer, multiphase, flowrate, neural network

Student: Andreas Lund Rasmussen

Supervisor: Håkon Viumdal & Saba Mylvaganam, USN

External partner: Kjetil Fjalestad, Equinor

Summary:

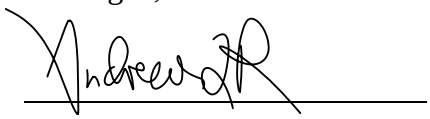
There are many good alternatives to measure single-phase flow with high accuracy, but measuring multiphase flow is challenging. The sand detectors used on Equinor's oil and gas installations use a piezoelectric sensor to measure the acoustic emission caused by sand production. This master thesis investigates if it is possible to make models for multiphase flow using a combination of acoustic emission sensors and other available sensors. If this is possible, then Equinor can use already installed sensors to estimate the multiphase flow on their oil and gas producing wells. This thesis shows that with a shallow neural network it is possible to create good models for gas and total flow rate with acoustic emission sensors and sensors to measure differential pressure over a Venturi. It was more challenging to make good models for the oil and water flow. To improve the results in this thesis it is recommended to try a deep neural network and to have higher focus on frequency analysis.

Preface

This thesis was completed as a part of a four-year part time master's study program at the University of South-Eastern Norway in Industrial IT and Automation. Since I was in a fulltime job during the study the study was done over two semesters.

All the experiments and datalogging were done by Equinor. Dataset 1 was prepared and available in advance of this thesis, while the experiments for dataset 2 were done in the middle of this study. Therefore dataset 2 was not available before the end of the first semester of this thesis. When reading this thesis, it is assumed that the reader has a basic understanding of standard instrumentation in the oil and gas industry and that the reader also has a basic understanding of neural network programming. I would like to thank Kjetil Fjalestad for providing the data and necessary information to complete this thesis. I would also like to thank Håkon Viumdal and Saba Mylvaganam for their supervision.

Veavågen, 05th October 2022



Andreas Lund Rasmussen

Contents

Table of figures

Nomenclature

A_i: *Initial layer input delay state*

B: *Bandwidth [kHz]*

CFD: *Computational Fluid Dynamics*

DI: *Density Indicator*

DAQ: *Data Acquisition System*

FFT: *Fast Fourier Transform*

f_s: *Sampling frequency [kHz]*

GVF: *Gas Volume Fraction*

g: *Gravitational acceleration* $\left[{}^{m}/_{s^2}\right]$

G: *Volumetric gas flow rate* $\left[{}^{m^3}/_{h}\right]$

GO: *Volumetric gas and oil flow rate* $\left[{}^{m^3}/_{h}\right]$

GOW: *Volumetric gas, oil and water flow rate* $\left[{}^{m^3}/_{h}\right]$

GW: *Volumetric gas and water flow rate* $\left[{}^{m^3}/_{h}\right]$

HIC: *Hand operated, Indicating Controller*.

H: Elevation [m]:

K: *Constant for Venturi or Orifice expression*

LER: *Local Equipment Room*

L: *Hamming window length*

MSE: *Mean Square Error*

$\dot{m}$: *Mass flow rate* $[kg/m^3]$

$NARX$: *Nonlinear Autoregressive network with Exogenous inputs*

O: *Volumetric oil flow rate* $\left[{}^{m^3}/_{h}\right]$

OW: *Volumetric oil and water flow rate* $\left[{}^{m^3}/_{h}\right]$

PSD: *Power Spectral Density*

PI: *Pressure Indicator*

$P\&ID$: *Piping & Instrument Diagram*

PDI: *Pressure Differential Indicator*

ΔP: *Differential Pressure* $[bar]$

P_i: *Inlet pressure* $[bar]$

p: Static pressure $[Pa]$

P_o: *Outlet pressure* $[bar]$

Q: $Volumetric\ flow\ rate\ \left[{}^{m^3}/_{h}\right]$

Q_w: $Volumetric\ water\ flow\ rate\ \left[{}^{m^3}/_{h}\right]$

Q_o: $Volumetric\ oil\ flow\ rate\ \left[{}^{m^3}/_{h}\right]$

Q_g: $Volumetric\ gas\ flow\ rate\ \left[{}^{m^3}/_{h}\right]$

Q_{tot}: $Volumetric\ total\ flow\ rate\ \left[{}^{m^3}/_{h}\right]$

R: $Regression\ between\ actual\ and\ predicted\ value$

RMS: $Root\ Mean\ Square$

$RMSE$: $Root\ Mean\ Square\ Error$

SAS: $Safety\ and\ Automation\ System$

SFT: $Short\ Time\ Fourier\ Transform$

TI: $Temperature\ Indicator$

T_i: $Inlet\ Temperature\ [°C]$

T_o: $Outlet\ Temperature\ [°C]$

ΔT: $Delta\ Temperature\ [°C]$

v: $Flow\ velocity\ [{}^{m}/_{s}]$

W: $Volumetric\ water\ flow\ rate\ \left[{}^{m^3}/_{h}\right]$

WLR: $Water\ Liquid\ Ratio$

XI: $Acoustic\ emission\ Indicator$

XI_{adj}: $Adjusted\ acoustic\ emission\ sensor\ reading\ [V_{RMS}]$

XI_{noise}: $Acoustic\ emission\ offset\ measurement\ error\ [V_{RMS}]$

XI_{read}: $Acoustic\ emission\ reading\ [V_{RMS}]$

x: $Input\ value$

$x(t)$: $Input\ time\ series$

x': $Normalized\ input\ value$

x_{min}: $Minimum\ value\ in\ the\ input\ time\ series$

x_{max}: $Maximum\ value\ in\ the\ input\ time\ series$

X_s: $shifted\ inputs$

X_i: $Initial\ input\ layer\ delay\ states$

$y(t)$: $Output\ time\ series$

σ: $Standard\ deviation$

ρ: $Density\ \left[{}^{kg}/_{m^3}\right]$

μ: $Mean\ value$

1 Introduction

A lot of research has been done in the oil and gas industry to find cheaper and better ways of measuring multiphase flow. Equinor has a test rig in Herøya for performing multiphase flow experiments. This thesis uses two datasets that contain samples from experiments that are performed at this test rig. The focus is especially on acoustic emission sensors, but other sensors are also used.

1.1 Background

It is desirable to know how much oil, water, and gas each well produces in an oil and gas installation. Traditionally this has been done by routing a production well to a test separator where the gravitational force is exploited to separate water, oil, and gas before they are measured at their respective outlets. An alternative to a test separator is to install a multiphase flow meter on each production well. There are many good alternatives to measure single-phase flow with high accuracy, but measuring multiphase flow is more challenging. A typical multiphase flow meter often uses a combination of different sensors that together can be used to measure or estimate the multiphase flow with a high accuracy.

1.2 Objective

If a production well has a higher production than the reservoir can tolerate then there is a risk of formation collapse, resulting in a high level of sand production. Sand production in an oil and gas installation can cause erosion of the piping and over time the piping can rupture resulting in a gas leakage. In an oil and gas installation it is common to install sand detectors to monitor the sand production. This way the sand production can be kept under control. The sand detectors used on Equinor's oil and gas installations use a piezoelectric sensor to measure the acoustic emission caused by sand production. A multiphase flow does also generate acoustic emission. Since sand detectors are both cheap and non-intrusive Equinor is interested in using their already installed sand detectors to measure acoustic emission from the multiphase flow. Together with other available sensors it may be possible to estimate the multiphase flow.

1.3 Method used for multiphase flow estimation

This master thesis will use available datasets from Equinor to research if already installed sand detectors can be used as an accurate, cheap, reliable, and non-intrusive sensor to estimate a multiphase flow. During the experiment accelerometers are used to measure acoustic emission instead of sand detectors since they use the same sensor. Frequency analysis is done on the accelerometer data to look for dominating frequency components. The Root-Means-Squared, RMS, value of the acoustic emission sensors are used together with other selected sensors to train models for gas, oil, water, and total flow. The models have been trained using shallow neural networks. Before training the neural networks, the data was normalized using different normalization techniques.

2 Multiphase flow

It is normal that piping in process industries contains multiphase flow. The oil and gas production are typically a multiphase flow of oil, water, and gas. In some situations, there can also be some sand production.

2.1 Flow regimes

A flow regime is a description of the flow structure in a multiphase flow. Plug, stratified, wavy, slug and annular flow are examples of different flow regimes. Figure 1 show the flow regimes in one study done at the University of South-Eastern Norway (Rafael Johansen, 2018). The stratified flow regime contains low amount of gas bubbles, and it is assumed that the flow regime contains low levels of acoustic emission. The annular flow regime on the other hand contains a higher amount of gas bubbles. For this flow regime it is expected that higher levels of acoustic emission are detected.

Notice that the slug flow in Figure 1 is the most chaotic flow regime. This flow regime is expected to contain high levels of acoustic emission from the multiphase flow. Slug is an unstable slow regime that can occur under different situations like start-up or shutdown of a production well in the oil and gas industry. The most severe slugs are often induced in the vertical part of a production well, typically from the seabed to topside. Liquid accumulates at the lower part of the well due to lack of the capacity to lift the liquid to the topside. The gas is blocked by the hydrostatic pressure from the liquid. As a result, the pressure will build up. When the pressure is higher than the hydrostatic pressure the liquid will blow out followed by a gas surge to the topside. The slugs can be controlled by several methods. One way is to simply reduce the choking of the topside choke valve. This will increase pressure upstream of the choke but result in lower production (Simon Pedersen, 2017).

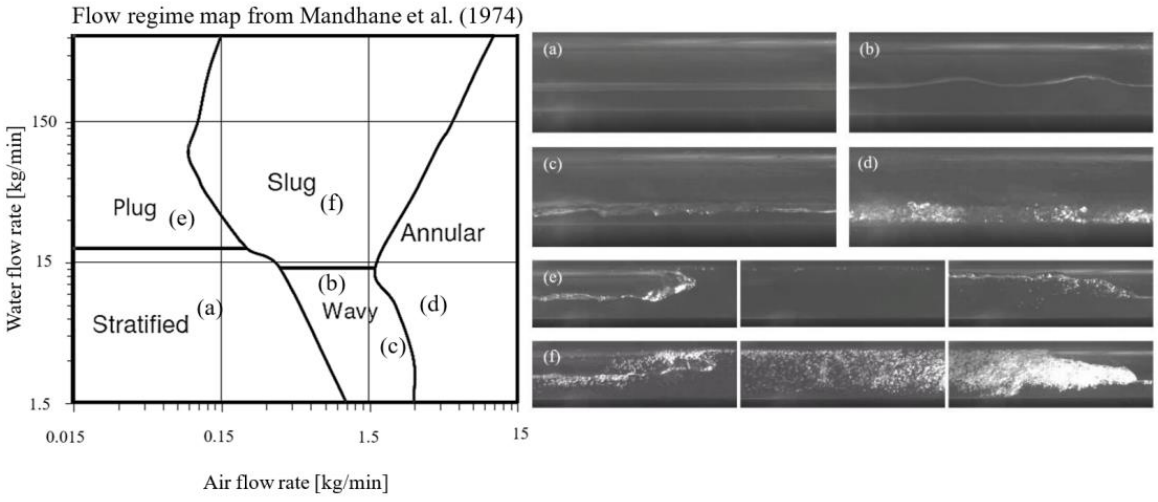


Figure 1: Different multiphase flow regimes are stratified (a), wavy (b/c), annular (d), plug (e) and slug (f). The slug flow is the most chaotic flow regime (Rafael Johansen, 2018)

Figure 2 show a table of the different experiments in dataset 1. The experiments done with gas and oil flow are shown in red, the experiments with gas and water flow are shown in green and the experiments with gas, oil and water flow are shown in blue. When comparing Figure 2 to Figure 1, it is possible to see that most of the experiments have a higher gas flow rate than 3 kg/min and are therefore either annular or slug flow.

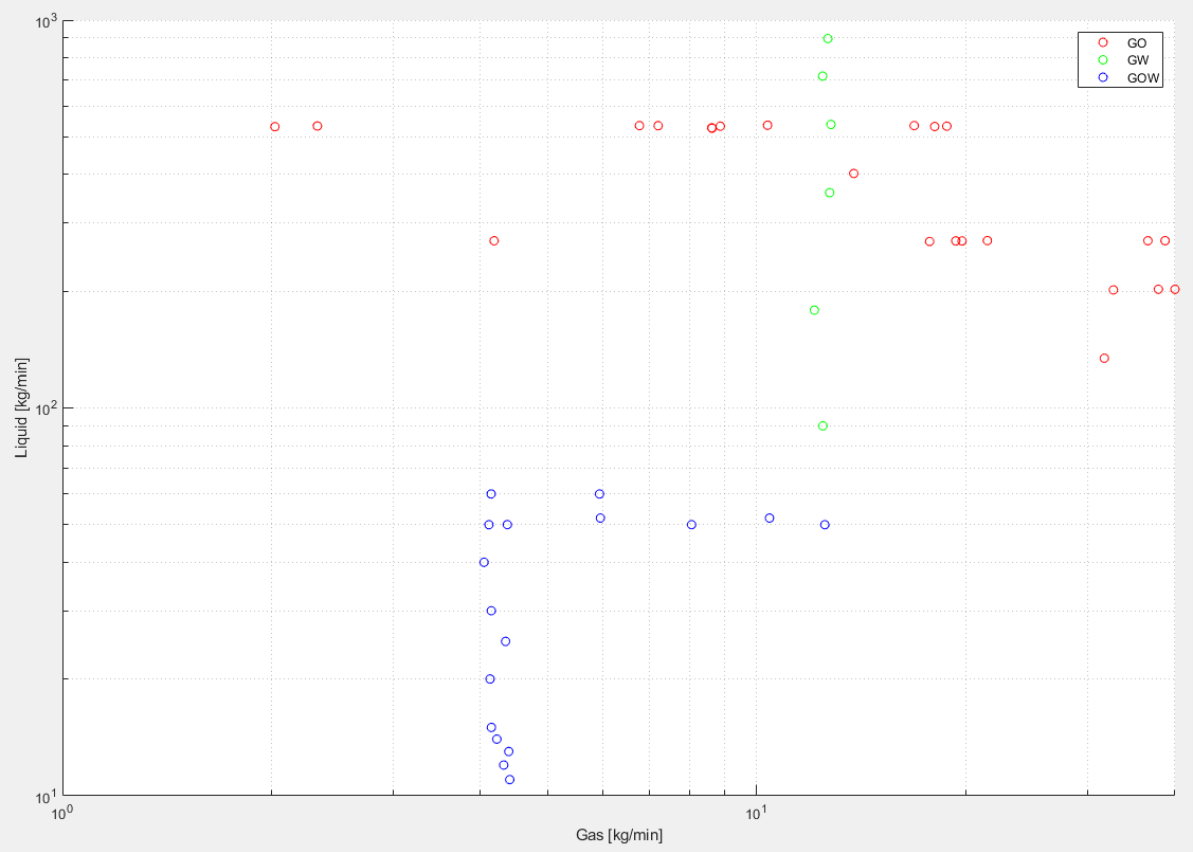


Figure 2: Different liquid and gas mass flow rates in dataset 1 are shown here. Gas and oil flow are shown in red color, gas and water flow are shown in green color, while gas, oil and water are shown in blue. Most of the flow regimes are assumed to be of annular or slug flow regime.

2.2 Test separator

A production well in an offshore oil and gas installation is sometimes rerouted to a test separator to test its capacity. A simplified Piping and Instrumentation Diagram, P&ID, of a test separator is shown in Figure 3. It is difficult to measure multiphase flow, so therefore a test separator can be used to separate the oil, gas, and water so that they can be measured separately. The separator exploits the gravitational force to separate oil, water, and gas inside the separator. The oil will be on the upper layer since it has a lower density than water. When the level in the separator becomes high enough the oil will overflow the weir and exit the oil outlet where the oil flow can be measured by a flow transmitter. The water will be drained from the bottom of the separator, while the gas outlet will be on the top of the separator. The gas and water flow will be measured with its own dedicated flow transmitter. Following this principle, the production capacity of oil, water and gas can be tested with high accuracy.

The test separator is so accurate that it can be used to calibrate multiphase flow meters. The disadvantage is that only one well is tested at each time and it can be long periods of time between each capacity test. Notice that the water and oil level need to be controlled to avoid water from overflowing into the oil chamber and to avoid the oil starting to enter the water outlet due to a low level in the separator. The instrumentation used for controlling the levels inside the separator will not be described in this thesis.

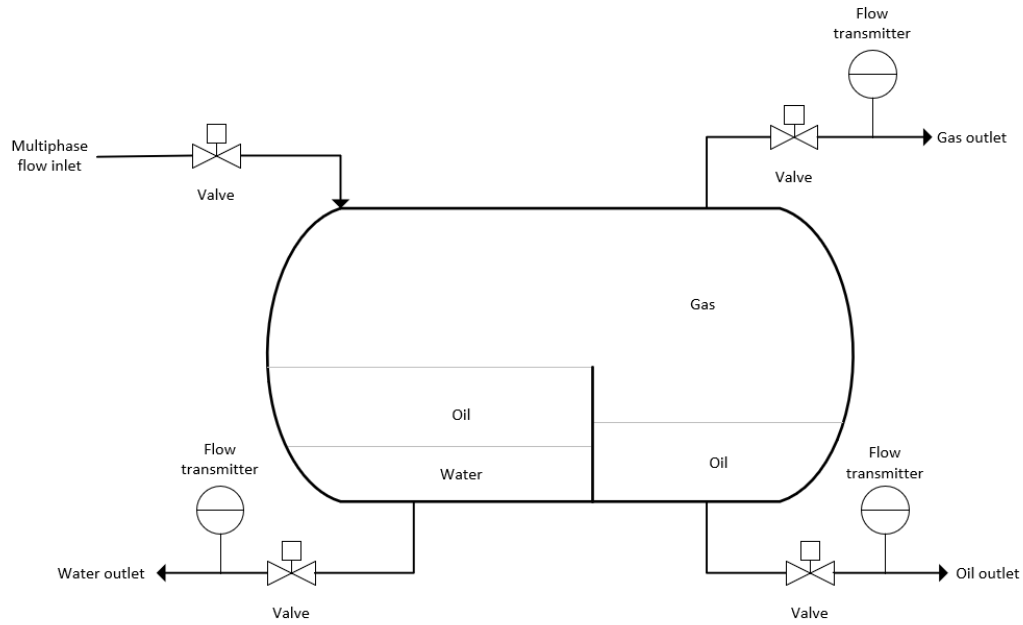


Figure 3: Simplified P&ID of a test separator that separates gas, oil and water before they can be measured at their respective outlets.

2.3 Multiphase flow metering

When using multiphase flow meters, the gas, oil, and water can be measured continuously. Anyway, it is hard to get a highly accurate measurement of all the phases. Therefore, a combination of different measurements is more common in a multiphase meter.

Roxar MPFM 2600 MVG is a multiphase meter from Emerson that use multiple measurement principles to calculate the multiphase flow. The flow meter is shown in Figure 4. Electrical impedance and gamma density measurements are used to determine the phase fractions, while the differential pressure over a Venturi is used to measure the multiphase flow velocity. It can measure Gas Void Faction, GVF, and Water Liquid Ratio, WLR, between 0 – 100%, but the uncertainty is only under 5 % with GVF between 25 – 95%. A 95 % confidence interval is given for the uncertainty and the repeatability is ¼ of the uncertainty. The operating range of the flow is between 1 – 40 m/s (Emerson, 2018). More details can be found on the vendors' website.

Figure 4: Roxar MPFM 2600 MVG from Emerson (Emerson, 2018).

The VIS (Vega Isokinetic Sampling) Multiphase Flowmeter from ABB is another flow meter that uses multiple differential transmitters. The flowmeter is shown in Figure 5. An orifice together with a differential pressure transmitter are used to determine the total multiphase flow. A small portion of the multiphase flow is sampled into an axial separator. Inside the separator the gas is separated from the liquid. The gas is measured by using a differential transmitter and a Venturi, while the liquid is measured from the filling time of a known volume. The liquid discharge valve re-injects the sample from the known volume and back to the main flow. Differential transmitters are used to determine the oil and water level in the known volume. This multiphase meter has an operating range of 80 – 100 % GVF, which makes it perfect for wet gas measurement. The accuracy is as $\pm$3 % of the reading for liquid and gas flow rate, and $\pm$5 % for water flow rate. (ABB, 2022). More details can be found on the vendors' website.

Figure 5: VIS Multiphase Flowmeter from ABB (ABB, 2022)

3 Equinor's multiphase test rig

The experiments are performed in a test rig located in Equinor's research center in Herøya. A small P&ID of this test rig is shown in Figure 6. A multiphase flow consisting of oil, water and gas enters a measuring segment where different sensors are installed to measure different variables.

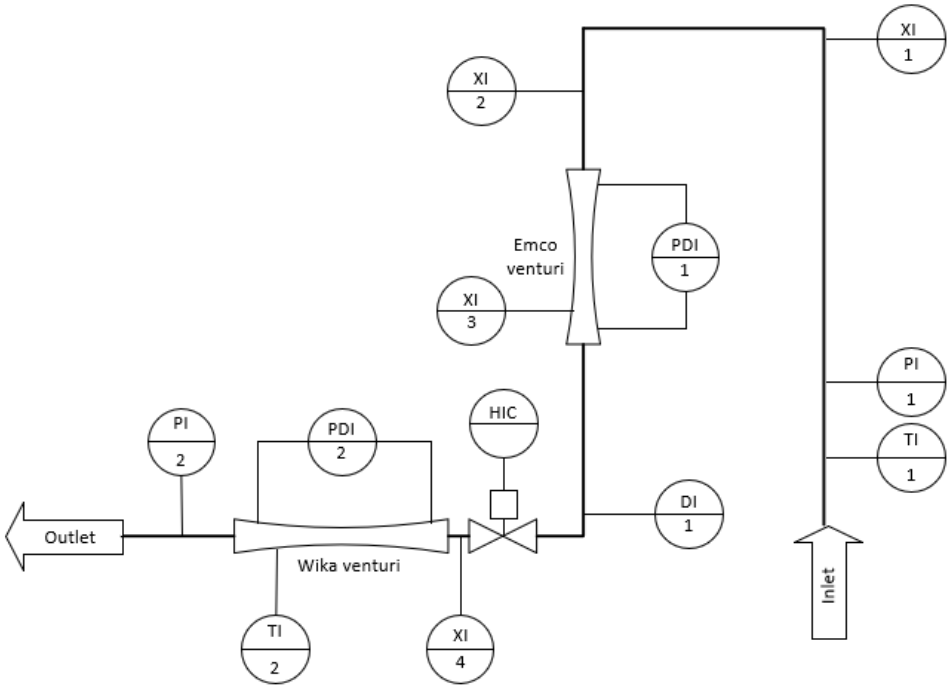


Figure 6: P&ID of Equinor's test rig in Herøya including choke valve (HIC), accelerometers (XI), pressure measurements (PI), differential pressure (PDI), temperature measurements (TI) and densitometer (DI)

The flow enters the test segment on the right side in Figure 6 and exit on the left side. The accelerometers are installed in the first bend (XI1), the second bend (XI2), on the first Venturi (XI3) and after the choke valve (XI4). A differential pressure transmitter (PDI1) is used to indicate the differential pressure over the Emco Venturi. Another differential pressure transmitter (PDI2) is used to measure the differential pressure over the Wika Venturi. In addition, there are pressure transmitters (PI1) on the inlet and outlet of the pipe (PI2) and a densitometer (DI) located upstream the choke. Inlet temperature (TI1) and outlet temperature (TI2) are also measured.

4 Sand detection using an acoustic emission sensor

It is common to use sand detectors to measure the sand production in oil and gas producing wells. By detecting potential sand production, the operator can monitor the situation and take necessary actions to avoid erosion. Erosion is caused by sand particles colliding with the pipe wall resulting in thinning the pipe wall. Over longer periods of time the pipe can rupture. Sand can also clog the cage in a choke valve or erode the cage to such a large degree that it is hard to control the flow with the choke. ClampOn are a well-known supplier of sand detectors used in Equinor's offshore oil and gas installations. Equinor is looking into the possibility to use already installed ClampOn sand detectors to detect acoustic emission that are generated from multiphase flow.

4.1 Working principle of sand detection

When sand particles flow through a 90-degree pipe bend, they will collide with the outer curvature of the pipe wall due to the centrifugal force. The lighter components like gas will be closer to the inner pipe wall. The sand detector uses a piezoelectric element to detect the ultrasonic energy from sand particles colliding with the pipe wall and convert it into electrical energy. The sand rate can be calculated if the flow velocity is known for the same period (ClampOn, 2019). There is a study that shows the relationship between acoustic energy and solid particles in gas flow at different velocities (Yao Yang, 2019). The black graph in Figure 7 show that the acoustic energy from pure gas flow without solid particles is approximately zero. The amount of acoustic energy increased after adding solids to the fluid as shown in the red graph, but the acoustic energy varies with different velocities. The amount of energy decreased with increasing velocity until it reached its lowest point at around 7m/s. After that point the acoustic energy increased with increasing velocity. Adding more solid particles will increase the acoustic energy as shown in the blue graph.

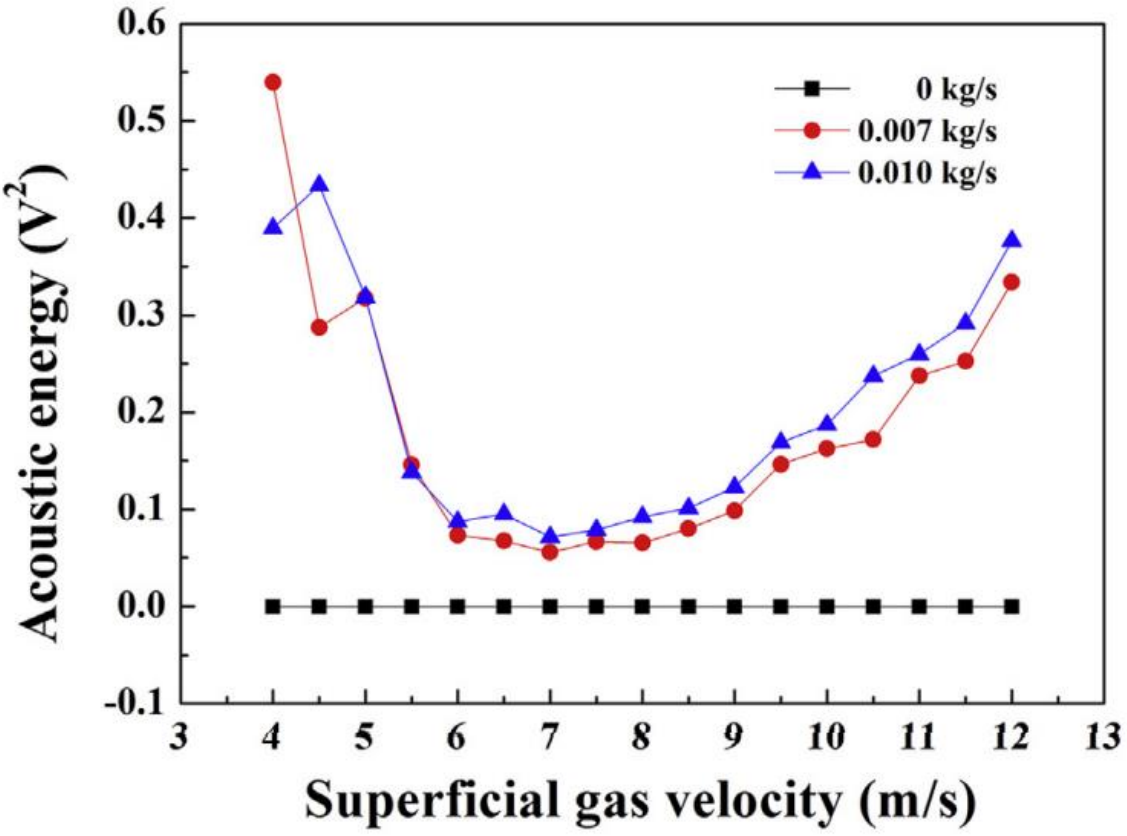


Figure 7: As shown in the red and blue graph the acoustic energy is highest when the superficial gas velocity is either high or low for flow containing solids. The black graph shows the acoustic energy when it is almost zero for pure gas flow (Yao Yang, 2019)

4.2 Installation requirements for sand detectors

When installing the sensor, it is important to have direct metal to metal contact with the pipe and therefore it may be necessary to remove insulation on the pipe or paint on the pipe. Before installing the sensor, a thin layer of silicone compound shall be applied where the sensor has contact with the pipe. The sand detector is installed as close as possible to the impact zone such that the best sand-to-noise ratio is achieved. The most preferable is if the pipe bend changes from vertical upward to horizontal. For gas wells the velocity is higher after a choke, so this location is preferable according to ClampOn. This statement supports the study described in Ch. 4.1. Background noise may also influence the measurement (ClampOn, 2019). Figure 8 show that the ideal location is between zero and three times the pipe diameter downstream the entry of the pipe bend.

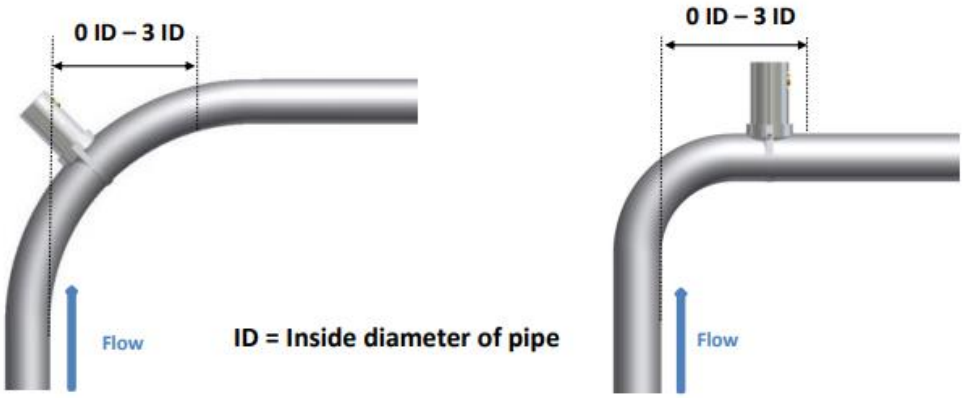


Figure 8: Ideal location of sand detector is a certain length after the inlet of the bend. This length is 0-3 times the inner diameter of the pipe. (ClampOn, 2019).

4.3 Sand detector from ClampOn

The raw signal of the sand detector from the company ClampOn communicates on several different protocols, but RS-485 combined with Modbus RTU protocol is the most relevant for this thesis. Analog 4-20 mA communication is only recommended for trending purposes (ClampOn, 2019).

Figure 9: ClampOn sand detector (ClampOn, 2019)

The datasheet for ClampOn is shown in Table 1. Standard baud rate is 9600 kbps but using ClampOn software the standard baud rate is 19 kbps (Datasheet, 2019).

Table 1: ClampOn DSP Particle Monitor datasheet (Datasheet, 2019)

Model name:	DSP Particle Monitor
Model number:	TSE.DS2I-SA00.A10
Repeatability:	Better than 1%
Minimum flow velocity	0.5 m/s
Minimum sand rate:	0.01 g/s
Signal types:	RS-485 and 4-20 mA
RS-485 (half duplex)	Modbus RTU or proprietary DSP
RS-485 baud rate:	2.4 kbps to 115.2 kbps (19.2 kbps standard)

The sand rate is calculated in grams, either by a computer running ClampOn software or by the control system of the oil and gas installation. The calculation is done according to Eq.4.1 (ClampOn, 2019).

$$Sand\ rate\ [g] = \frac{RAW - Zero}{Step[g^{-1}]} \qquad (Eq.4.1)$$

- *RAW: The raw value of the measurement signal.*
- *Zero: Background noise affecting the raw value.*
- *Step: The raw value is equal to one gram of sand.*

Notice that the raw signal is unitless. The background noise can come from the electrical signal itself or from the acoustic noise in the piping caused by the oil and gas production. The noise will most likely also vary with the flow velocity. The step size is a calculation factor that describes how much the raw signal varies pr. gram of sand. (ClampOn, 2019).

4.4 Wiring diagram for ClampOn sand detector

ClampOn can be integrated into the Safety and Automation System, SAS. The signal from the sand detector contains electrical energy. Since the signal can contain enough energy to potentially ignite a gas leakage in an oil and gas installation a GM D1061S barrier can be used to make the RS-485 signal intrinsically safe. After the signal is converted, the raw values are stored in a virtual machine where SAS can get access to the data. An overview of the system is shown below in Figure 10.

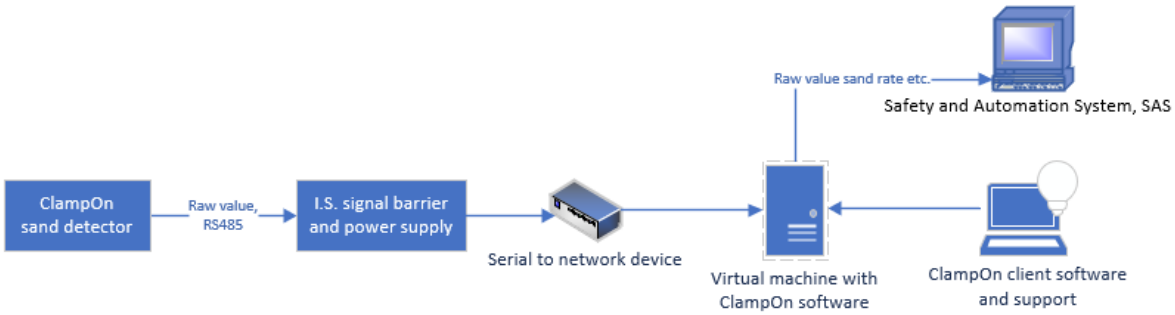


Figure 10: Integration of ClampOn into the Safety Automation System, SAS (ClampOn website, 2022) (ClampOn, 2019).

5 Acoustic emission sensors installed in Equinor's test rig

A key task in this thesis is to measure the acoustic emission from the multiphase flow to estimate different flow velocities and parameters. For this purpose, an accelerometer is used in combination with a signal amplifier, an isolated barrier, and a Data Acquisition, DAQ, unit. The working principle of the accelerometer is the same as for the sand detector and are described in Ch. 4.1. The main difference is if the sensor is used to detect bubbles/droplets or sand particles. Equinor is looking into the possibility of using their already installed sand detectors to measure acoustic emission in multiphase flow.

5.1 Brüel & Kjær accelerometer

An accelerometer type 5704 from Brüel & Kjær is used to measure the acoustic emission in the flow. It is important that the sensor is mounted on a flat clean surface to get good contact with the surface. There is also a risk of contact resonance, but that can be reduced by applying a thin layer of silicone grease between the contact area and the sensor. A picture of the sensor is shown below in Figure 11.

Figure 11: Brüel & Kjær accelerometer type 5704 (Kjær, Acceleration sensor charge type 5704 manual, 2017).

The sensor is working according to the piezoelectric shear principle. Electric charges are generated due to the piezo effect, and they are proportional to the acceleration. In this thesis the focus is to detect acoustic emission from the bubbles inside the pipe (Kjær, Acceleration sensor charge type 5704 manual, 2017).

5.2 Brüel & Kjær amplifier

A Brüel & Kjær charge amplifier type 2667 is used. This amplifier has a differential input, making it less sensible to electromagnetic measurement noise. It also amplifies the measurement signal, making it possible to travel greater lengths (Kjær, Charge amplifier type 2667 manual, 2016). The amplifier is shown in Figure 12 and a part of the datasheet in Table 2.

Figure 12: Brüel & Kjær charge amplifier type 2667 (Kjær, Charge amplifier type 2667 manual, 2016).

Table 2: Charge amplifier type 2667 datasheet (Kjær, Charge amplifier type 2667 manual, 2016)

Model name:	Brüel & Kjær type charge amplifier
Model number:	Type 2667
Sensitivity:	1 mV/pC, ±2% at 80 Hz
Frequency range	1 Hz to 100 kHz ±0.5 dB
High-Pass Filter	2. order, -3 dB at 0.48 Hz

5.3 Pepperl+Fuchs isolated barrier

The signal from the accelerometer contains electrical energy. Since the signal can contain enough energy to potentially ignite a gas leakage in an oil and gas installation, an isolated barrier is used to make the signal intrinsically safe. A Pepperl+Fuchs type KFD2-VR4 are used for this purpose and are shown in Figure 13. The isolated barrier is Ex certified according to international standards and is commonly used in oil and gas installations.

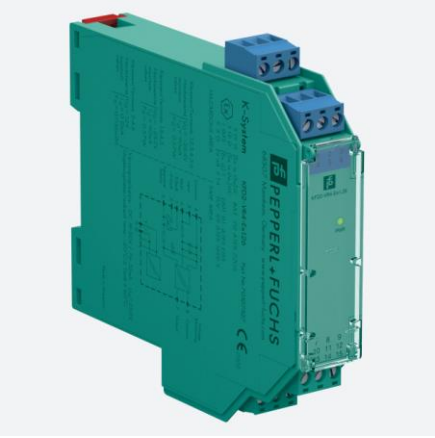


Figure 13: Pepperl+Fuchs KFD2-VR4 Ex barrier (Pepperl+Fuchs, 2022)

5.4 National Instruments Data Acquisition unit

Two NI 9232 modules from National Instruments are used for data acquisition and are shown in Figure 14. Three accelerometers are connected to the first module, while the fourth and last accelerometer are connected to the second module. The unit has high accuracy and a sampling rate of 102.4 kS/s like shown in Table 3.

Figure 14: NI 9232 with three channels is used for data acquisition of the accelerometers

Table 3: NI 9232 DAQ unit datasheet

Model name:	National Instruments
Model number:	NI 9232
Channels:	3
Sample rate:	102.4 kS/s
Gain error:	± 0.1% of gain at 23°C, ±5 °C
Offset error	± 0.023%
High-Pass Filter	2. order, -3 dB at 0.48 Hz

5.5 Wiring diagram for Brüel & Kjær accelerometers

All the components mentioned in the above subchapters are connected in the wiring diagram below in Figure 15. The accelerometer and charge amplifier are installed in a hazardous area in an oil and gas installation, while the isolated barrier and the DAQ unit are in a safe area like a Local Equipment Room, LER.

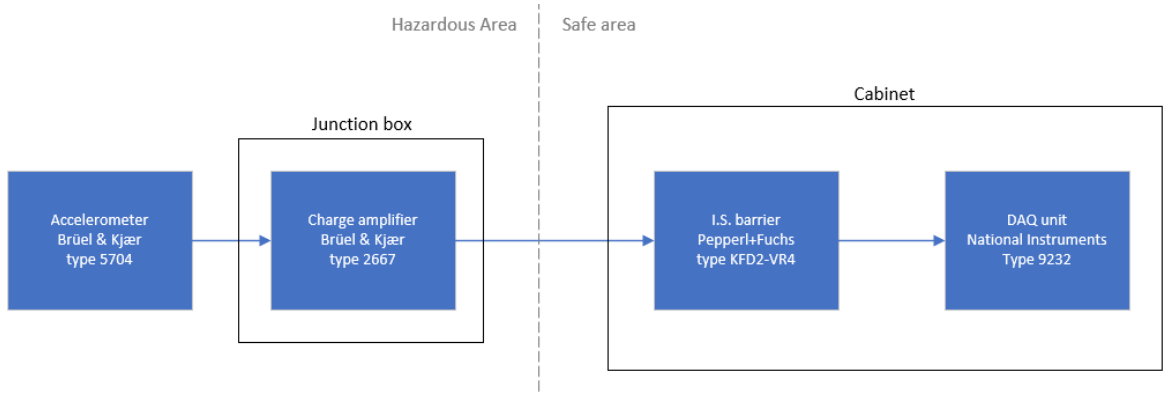


Figure 15: Wiring diagram of accelerometer including isolated barrier and DAQ unit.

6 Analyzing data from experiments

Two different experiments have been done on Equinor's test rig in Herøya. The experiments are called dataset 1 and dataset 2. This chapter focuses on both datasets since dataset 1 was only available at the start of this thesis and dataset 2 was desired to use for further analysis and model development.

6.1 Process conditions for the measured acoustic emission

The level of acoustic emission in the multiphase flow is dependent on process conditions. This sub chapter describes how process conditions are affecting the acoustic emission measurements. Results from research using Computational Fluid Dynamics, CFD, simulations have been used to make assumptions on the acoustic emission in the experiments.

6.1.1 Distribution of phases in and around 90-degree bend

Figure 16 show the distribution between gas and liquid in a two-phase flow around a 90-degree bend. When the flow is annular at the entry of the elbow the gas bubbles tend to move to the inner curvature. (Shouxu Qiao, 2021). This should be taken into consideration when installing the accelerometer. A good location could be in the inner curvature of the bend somewhere. Notice that sand detectors are often located on the outer curvature of the bend or after the bend to detect sand particles like described in Figure 8. Also, sand particles contain higher mass than gas bubbles and this may affect the results.

Two of the accelerometers on Equinor's test rig at Herøya are installed near a 90-degree bend to detect acoustic emission from bubbles. XI1 is located upstream a 90-degree bend that go from a vertical to a horizontal direction. XI2 are located downstream a 90-degree bend that go from a horizontal to a vertical direction. Both are located on the outer curvature of the bend. Look at Ch.3 for the sensor location on the P&ID, but keep in mind that the location on the P&ID is not accurate.

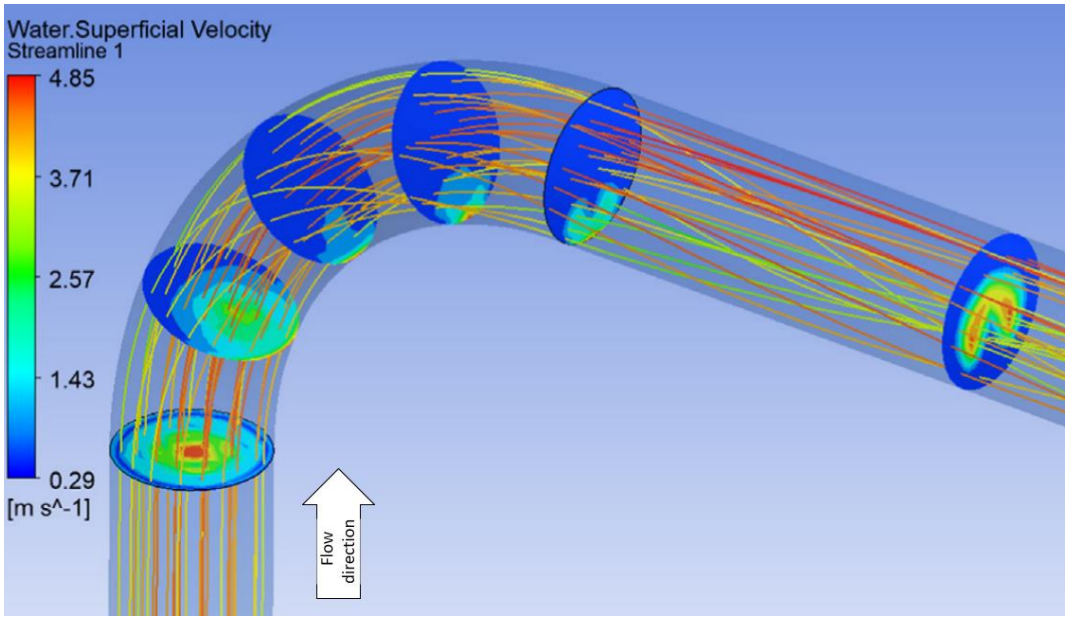


Figure 16: Bubble distribution in two-phase flow around a 90-degree elbow (Shouxu Qiao, 2021)

6.1.2 Venturi effect on gas volume fraction

Bernoulli's equation shown in Eq.6.1 states that the velocity in a pipe will increase if there is a reduction in either static pressure or potential energy.

$$\frac{V^2}{2} + g \cdot h + \frac{p}{\rho} = Constant \qquad (Eq.6.1)$$

where:

- *v [m/s]: Flow velocity*
- *g [m/s^2]: Gravitational acceleration*
- *h [m]: Elevation*
- *p [Pa]: Static pressure*
- *ρ[kg/m^3]: Density*

A Venturi meter contains a restriction of the inner diameter of a pipe segment. The restriction in the pipe diameter results in a reduced static pressure and increased velocity according to Bernoulli's equation. Figure 17 show the result of a CFD simulation done at Instituto Tecnológico de Celaya. It can be observed that the reduction in diameter results in an increased velocity (Efraín Quiroz-Pérez, 2014).

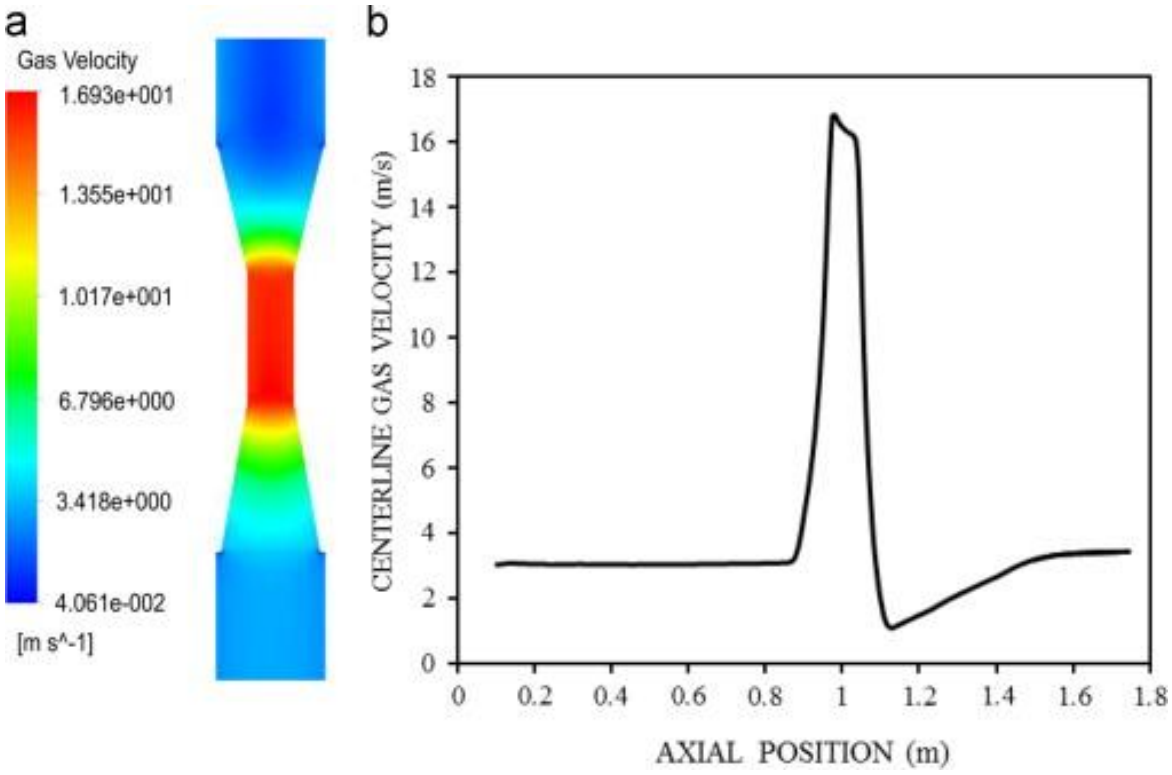


Figure 17: Steady-state gas flow through a Venturi device: (a) color contour and (b) transversal area averaged graph (Efraín Quiroz-Pérez, 2014).

Accelerometer XI3 on Equinor's test rig is installed on the Emco Venturi to detect acoustic emission from bubbles. Look at Ch.3 for the sensor location on the P&ID. CFD simulations from a study at Instituto Tecnológico de Celaya to show the effect from a Venturi device on a gas production well. The results in Figure 18 show that the Venturi effect increases the gas volume fraction in a gas-liquid flow.

$$GVF = \frac{Q_g}{Q_{tot}} \qquad (Eq.6.2)$$

In the study it was observed that the flow is of a mist type and that the gas bubbles tended to flow in the center of the well (Efraín Quiroz-Pérez, 2014).

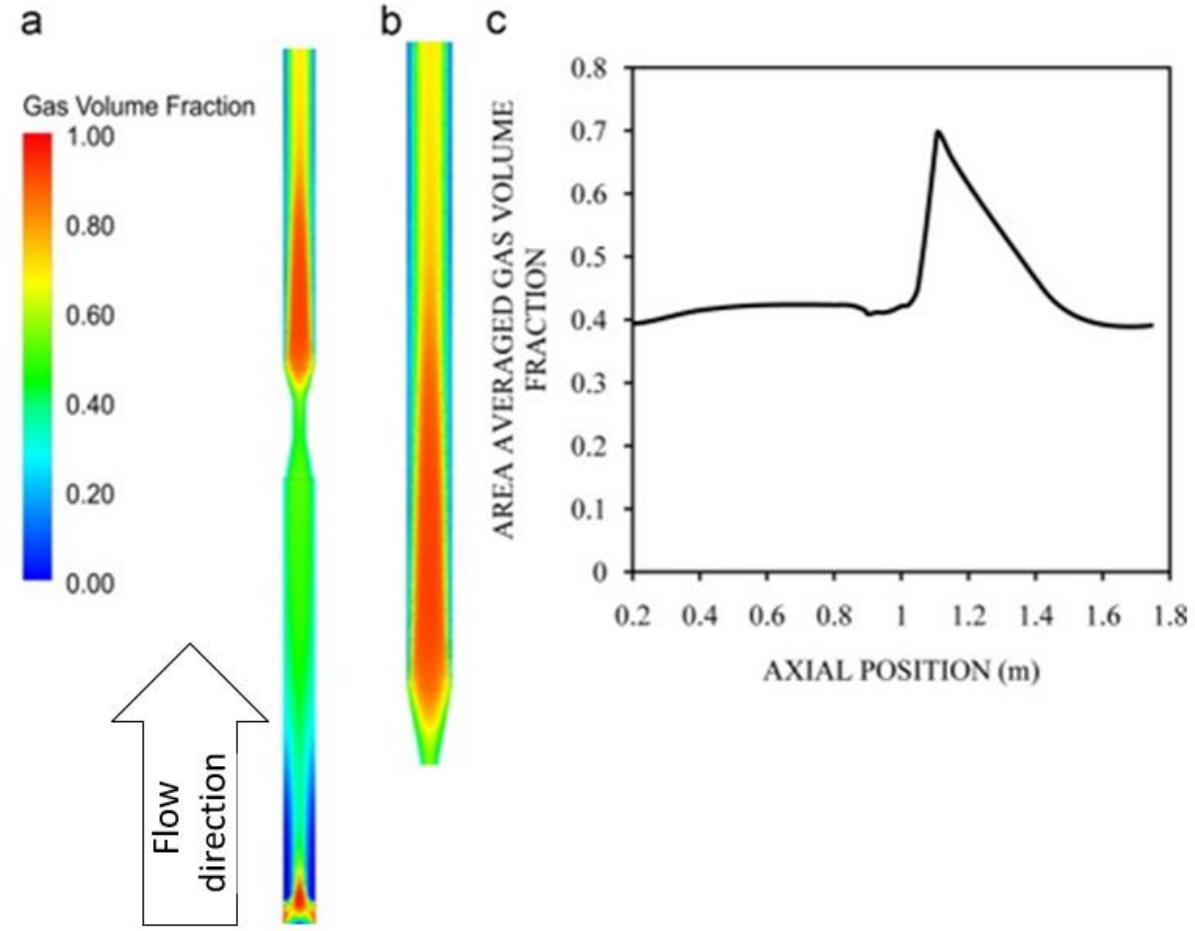


Figure 18: Estimation of gas volume fraction in a Venturi device where: (a) full plane, (b) upper section and (c) transversal area averaged graph (Efraín Quiroz-Pérez, 2014).

6.1.3 Choke valve effect on gas volume fraction

When the choke is neither fully open nor closed it will restrict the flow causing a reduction in static pressure like described in Ch. 6.1.2 for the Venturi. Therefore, it would be reasonable to assume that the choke valve will give the same effect on the gas void fraction as described in Ch. 6.1.2. Anyway, the degree of the effect on the gas void fraction will depend on the choke valve position. Since the valve can give a higher flow restriction than the Venturi it is assumed that if there is gas present, then the choke position will influence the acoustic emission to a higher degree than the Venturi. Accelerometer XI4 on the test rig are installed

downstream the choke and may detect high degree of acoustic emission. Look at Ch.3 for the sensor location on the P&ID.

6.2 Soft sensing strategies

The datasets contain many different variables, but it is also possible to make new variables as a soft sensor. There is no data available for the total differential pressure, total flow, and the delta temperature. These variables may potentially contain some important information that is relevant for the analysis Ch. 6. These variables are calculated as soft sensors and are shown below in Eq.6.3, Eq 6.4 and Eq. 6.5.

$$\Delta P = P_i - P_o \quad \textit{(Eq.6.3)}$$

$$\Delta T = T_i - T_o \quad \textit{(Eq.6.4)}$$

$$Q_{tot} = Q_g + Q_o + Q_w \quad \textit{(Eq.6.5)}$$

The code for calculating the soft sensors in the main code for dataset 2 is shown below. It can also be found in appendix B.

```
%Soft sensors in dataset 2
dP_Train_2 = training_2(:,6) - training_2(:,20);
dP_Test_2 = testing_2(:,6) - testing_2(:,20);
dT_Train_2 = training_2(:,19) - training_2(:,5);
dT_Test_2 = testing_2(:,19) - testing_2(:,5);
Qtot_Train_2 = training_2(:,25) + training_2(:,26) + training_2(:,27);
Qtot_Test_2 = testing_2(:,25) + testing_2(:,26) + testing_2(:,27);
```

6.3 Analysis of the Root Mean Square (RMS) value of the acoustic emission measurements

The acoustic emission sensors in this thesis give a voltage signal that can be used for further analysis. In this thesis it was desirable to use the voltage RMS, V_{RMS}, value of some samples. One V_{RMS} value is calculated from approximately 2000 samples of each experiment in dataset 1. The values are used in the analysis in this chapter only, but the comments in this chapter are assumed to be applicable for both dataset 1 and dataset 2. The calculation is done using the inbuilt RMS function in MATLAB and can be found in appendix H, I, J, K, L, M and N. The function uses the formula shown in Eq. 6.6. The hypothesis is that V_{RMS} value will correlate with the acoustic emission from the multiphase flow.

$$V_{RMS} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}|V_n|^2} \qquad \textit{(Eq.6.6)}$$

6.3.1 V_{RMS} value in single-phase flow

The plot of single-phase flow experiments for pure gas, water, or oil flow are shown in Figure 19. The code used to calculate the values and plot them can be found in appendix G. A pure gas flow rate between 100 – 200 m^3/h seems to give an effect on the differential pressure over both Venturi's but a very small impact on the total differential pressure. When studying the calculated V_{RMS} value from the acoustic emission sensors it seems like there is a high degree of acoustic emission from the gas flow in this flow range, but that the acoustic emission seems to be much lower for lower gas flow rates. For pure oil flow rate between 5 – 60 m^3/h the acoustic emission seems to be higher than for pure gas. In these experiments oil flow was controlled with the choke position, while for the pure gas flow experiments the choke was fully open. The acoustic emission sensor located downstream of the choke seems to be most affected by the change in total differential pressure, but it can also be observed that it correlates with the other acoustic emission sensors. A pure water flow rate between 0 - 60 m^3/h seems to give very little acoustic emission.

There seems to be a correlation between the differential pressure on the Venturis and the total flow rate for both pure gas, oil, and water flow. It is already well known that there is a relationship between the differential pressure over a Venturi and the total flow. Eq. 6.7 follow the ISO-5167 standard and are based on Bernoulli's equation. This equation can be rewritten to volumetric flow as shown in Eq. 6.8. Please note that this equation is used for single phase flow and will not give accurate reading of multi-phase flow. The problem with using this equation for multiphase flow is that the different phases will contain different flow velocity and that it is difficult to estimate the density in a multiphase flow.

$$\dot{m} = K \cdot \sqrt{\Delta P} \cdot \rho \qquad \textit{(Eq.6.7)}$$

$$Q = K \cdot \sqrt{\frac{\Delta P}{\rho}} \qquad \textit{(Eq.6.8)}$$

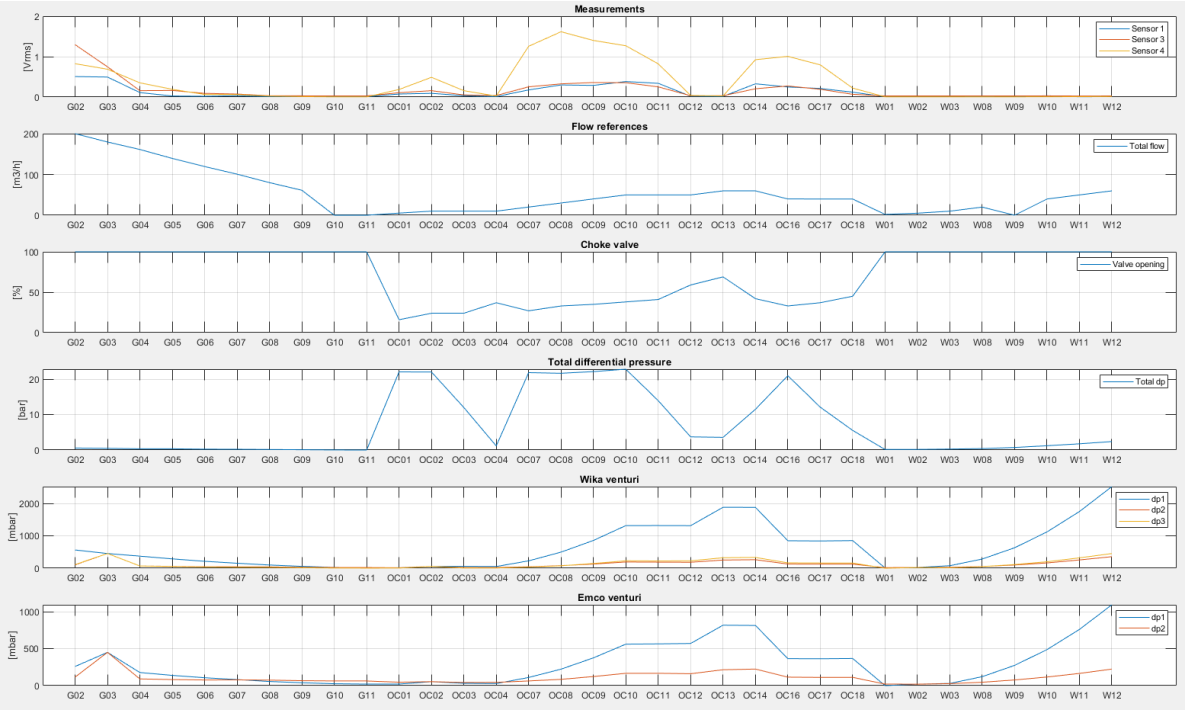


Figure 19: The letters and numbers in the x-axis contain information about the experiments in dataset 1. The first letters explain if the experiments are done on pure gas (G), water (W), or oil (OC) flow. The numbers explain the experiment number.

In Figure 20, it can be observed that a big change in the choke position results in a peak in both total differential pressure and the acoustic emission sensors for a single-phase flow containing both oil and water. This is especially true for the acoustic emission sensor located after the choke valve. It is worth noticing that compared to a change in the choke position, a change in either oil or water flow gives a small effect on the acoustic emission readings.

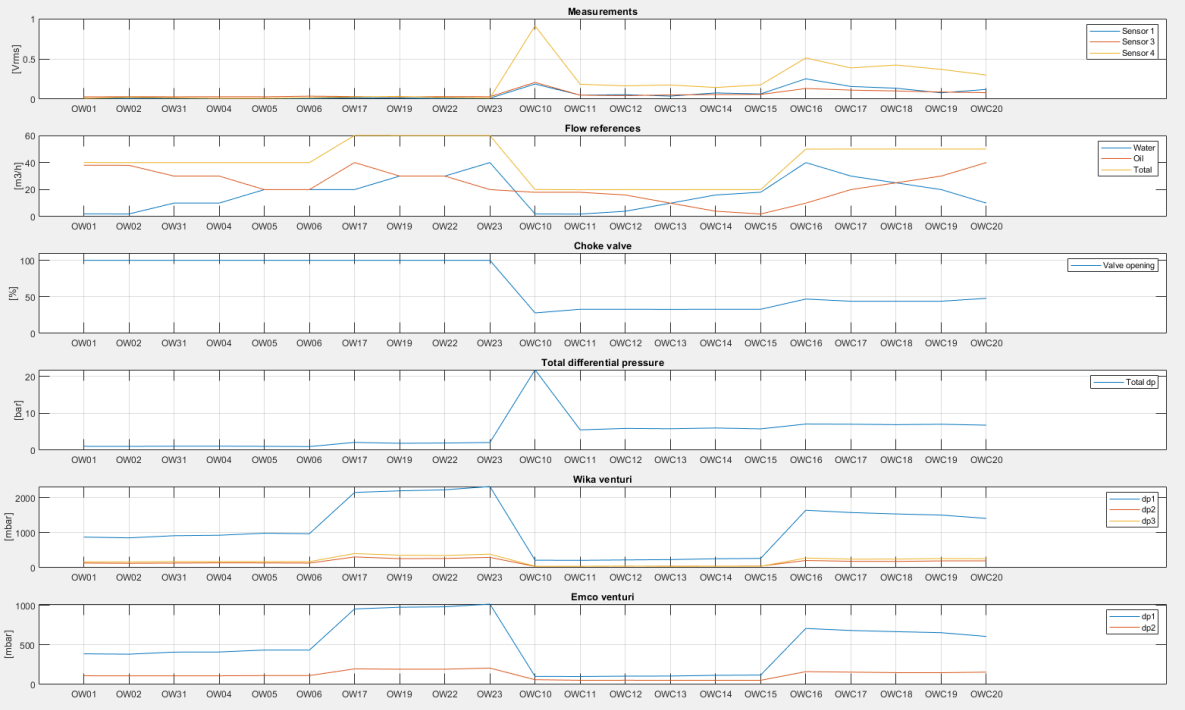


Figure 20: The letters and numbers in the x-axis contain information about the experiments in dataset 1. The first two letters explain that the experiments are done on water and oil (OW) flow. The numbers explain the experiment number.

6.3.2 V_{RMS} value in two phase flow

The V_{RMS} value for two-phase flow experiments is analyzed in this subchapter. The codes used to calculate the values and plot them can be found in appendix C, D, E and F. Figure 21 contain experiments containing both oil and gas flow, while the experiments in Figure 22 contain gas and water flow. They all show the same relationship between total differential pressure and the acoustic emission sensors as mentioned in Ch.6.3.1. The V_{RMS} value seems to correlate very well with the total differential pressure and the total differential pressure is highly dependent on the choke valve. The total flow does also correlate very well with the differential pressure over both Venturi.

Figure 21: The letters and numbers in the x-axis contain information about the experiments in dataset 1. The first two letters explain that the experiments are done on gas and oil (GO) flow. The numbers explain the experiment number.

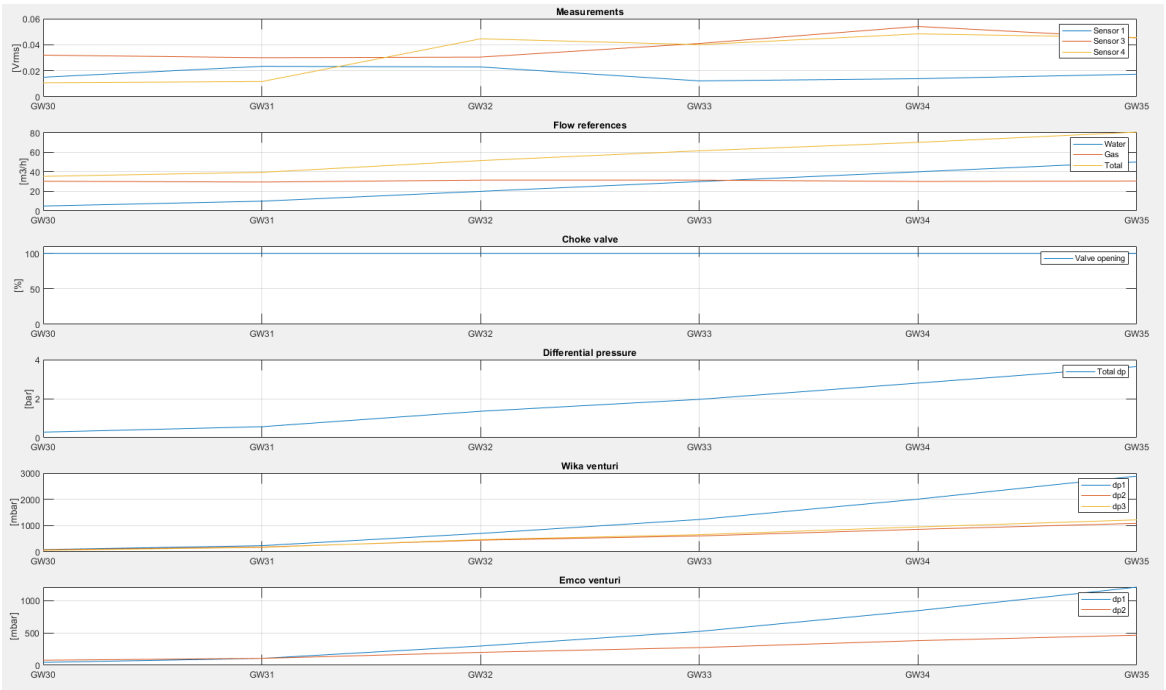


Figure 22: The letters and numbers in the x-axis contain information about the experiments in dataset 1. The first two letters explain that the experiments are done on gas and water (GW) flow. The numbers explain the experiment number.

Figure 23 contain experiments containing oil, water, and gas flow. The figure shows the same relationship between total differential pressure and acoustic emission. It also shows the same relationship between the differential pressure over both Venturi and total flow.

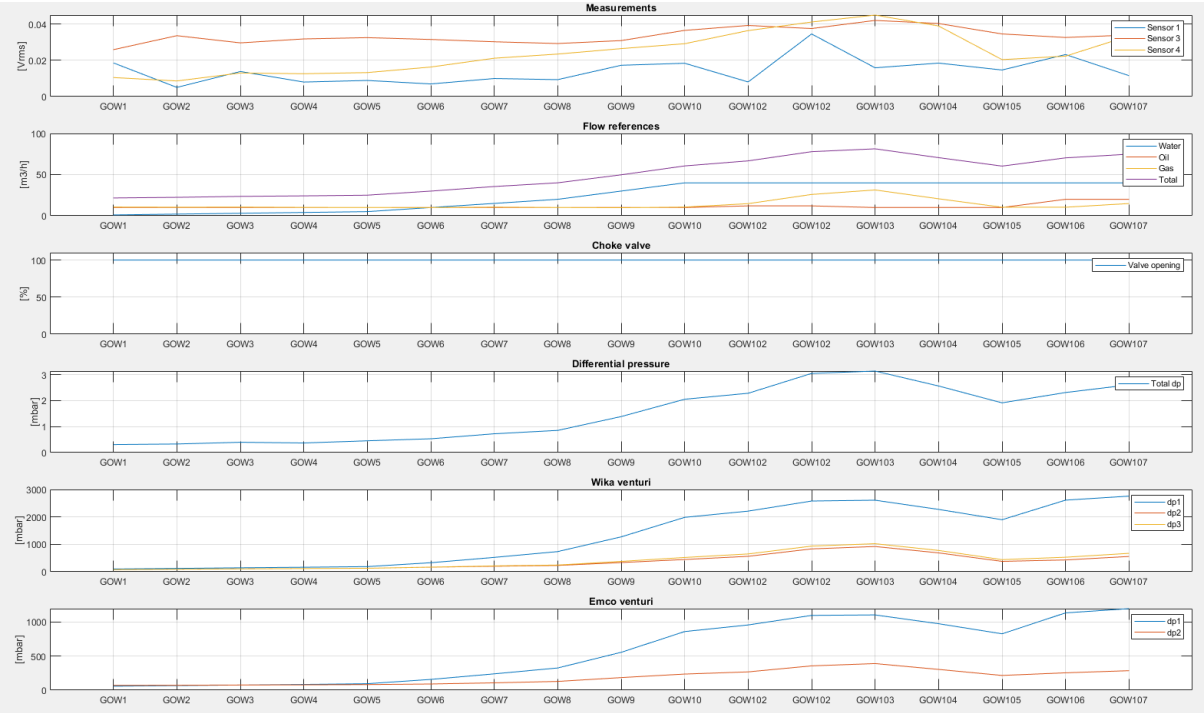


Figure 23: The letters and numbers in the x-axis contain information about the experiments in dataset 1. The first letter explains that the experiments are done on gas, oil, and water (GOW) flow. The numbers explain the experiment number.

6.4 Frequency analysis of the acoustic emission sensors

Frequency can in general be explained as the number of times an event occurs. The purpose of the frequency analysis is to analyze the different frequency components in the dataset and look for patterns. In this chapter approximately 2000 samples from each experiment in dataset 1 were used. The comments regarding dataset 1 in this chapter are also assumed to be applicable for dataset 2. The code used to calculate values and plot them can be found in appendix G.

6.4.1 Sampling rate

The DAQ unit for the accelerometers has a sampling rate of 102.4 kHz like shown in Table 3, but the datasets contain only 51 200 samples pr. second and therefore have a sampling frequency, f_s, of 51.2kHz. To satisfy the Nyquist criterion in Eq. 6.9 the sampling rate, f_s, should be higher than two times the bandwidth, B, to avoid aliasing. This means that the highest frequency component should be lower than 25.6 kHz.

$$f_s > 2B \qquad (Eq.6.9)$$

6.4.2 Power Spectral Density estimate of acoustic emission sensors

The Power Spectral Density, PSD, estimate can generally be described as the frequency distribution of the signal variance (Power Spectral Density, 2022). The pwelch() function in MATLAB is used for PSD analysis. The function normalizes the data with respect to

frequency as shown in the y-axis labels in Figure 24. Normalization of the data makes it easier to compare the decibel value of each frequency component since they are normalized on the same scale.

The upper plot in Figure 24 shows the PSD estimate of one of the acoustic emission sensors. The PSD estimate is done on two phase flow experiments that include both gas and oil flow. The lower plot shows the same PSD estimate, but this time by using a Hamming window. This Hamming window is multiplied with the signal to reduce noise in the dataset and make it easier to see the dominating frequency components. A Hamming window size of L = 500 seem to be a good choice and is therefore used in further analysis in Ch.6.4.4. Increasing the window will reduce the effect. Reducing the window will on the other hand result in more information is lost. This can be observed in the lower plot in Figure 24.

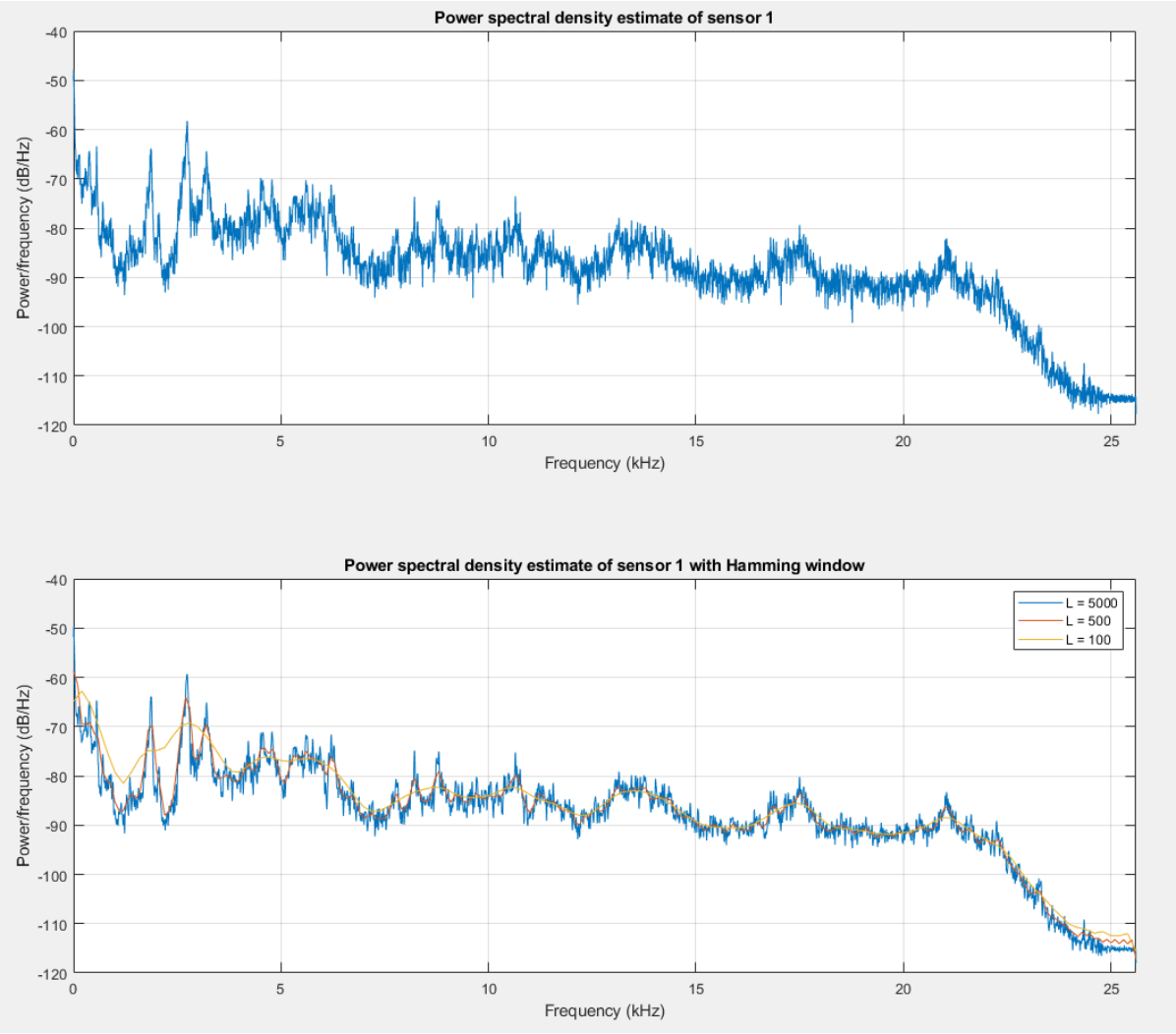


Figure 24: Effect of the size of Hamming window on Power Spectral Density estimates of oil and gas flow. Reducing the window will increase the effect of the windowing function, but more information will be lost. A selection of L = 500 seems to be a good choice for further analysis.

6.4.3 Spectrogram using Short-Time Fourier Transform on acoustic emission sensors

The spectrogram is another method used to analyze the frequency components in the datasets. A spectrogram is a colormap with a visual representation of the frequency components in the dataset like shown in Figure 25. The spectrogram uses Short-Time Fourier Transform, STFT. STFT do a Fourier transforms on a windowed signal. This means that it does several Fourier transforms, but each on a different part of the signal. The Fast Fourier Transform, FFT, on the other hand does a Fourier transform on the entire signal. A STFT is beneficial since the information in each frequency component varies over time. This will avoid that information is lost during the Fourier transform (Kehtarnavaz, 2008).

Figure 25 show the Spectrogram of one of the acoustic emission sensors. The Spectrogram is done on two phase flow experiments that include both gas and oil flow. The lower plot shows the Spectrogram, but this time by using a Hamming window of L = 500, which is the same as for the PSD estimate in Ch.6.4.2. A Hamming window of L = 500 will be used for further analysis in Ch.6.4.4.

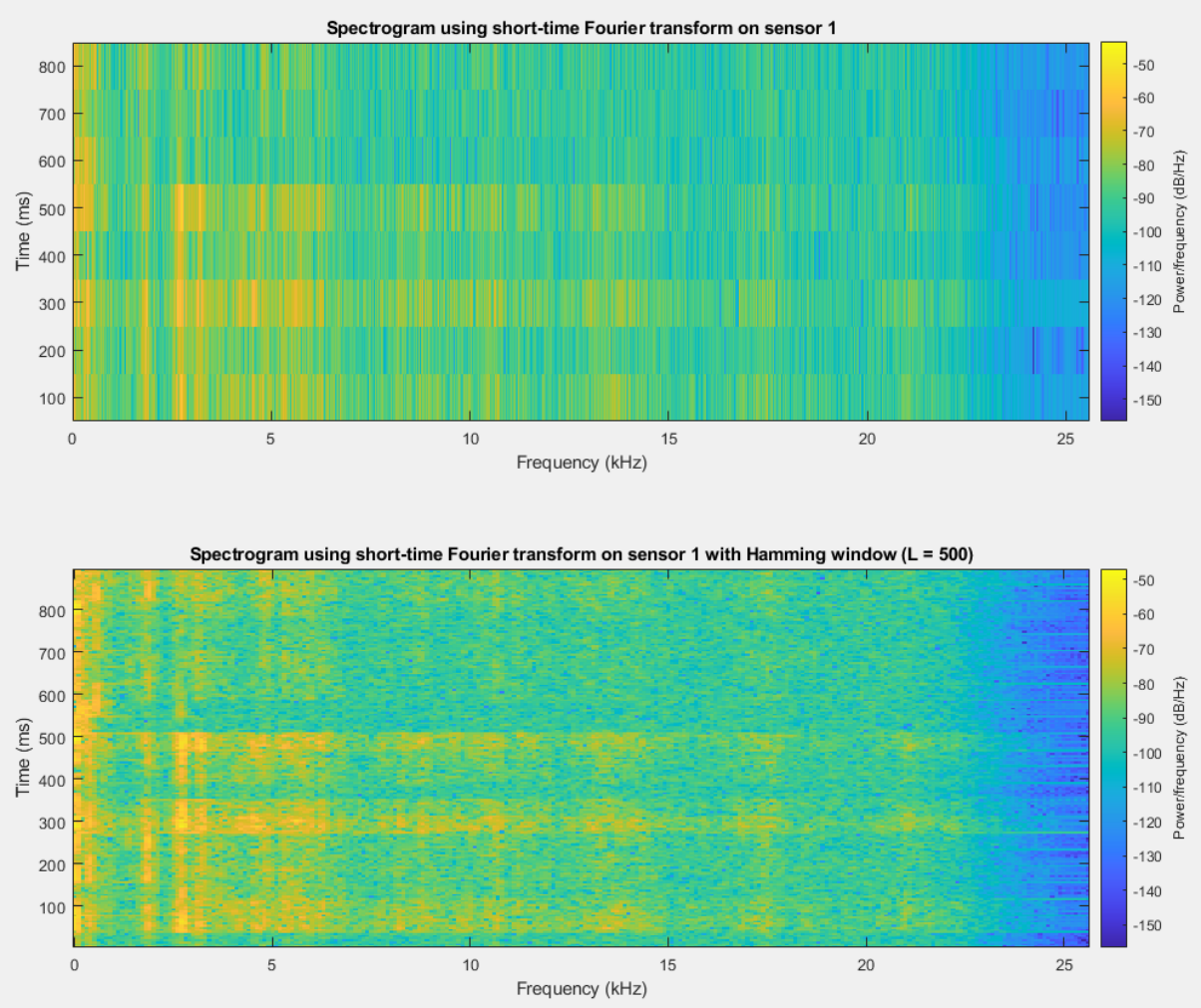


Figure 25: Effect of the size of Hamming window on Spectrogram for an oil and gas flow. The plots show the spectrogram with and without a hamming window of L = 500. A Hamming window seems to give a slightly better visualization of the dominating frequency components.

6.4.4 Results of the frequency analysis

This chapter includes the results of the frequency analysis of the multiphase flow experiments in dataset 1. The results are assumed to also be applicable for dataset 2.

6.4.4.1 Frequency analysis of the single-phase flow

Figure 26, Figure 27 and Figure 28 show the frequency analysis results of oil and water flow. The time series plots show that all the acoustic emission sensors measure higher acoustic emission after approximately 390ms. This is especially true for acoustic emission sensor 4, which is located after the choke valve. The valve was fully open and changed position at approximately 390ms. The valve was 28 % open at approximately 430ms. The period 390 – 430ms is shown as black rectangular boxes in Figure 26, Figure 27 and Figure 28, while highlighted frequencies at the same period is shown in red dotted circles in the PSD and spectrogram plots. The change in choke position is shown in Figure 20.

The PSD estimate shows that the dominating frequency components for sensor 1 and sensor 4 are between 0 – 11 kHz, but sensor 4 also have some peaks around 15 kHz and 22 kHz. Sensor 3 has dominating frequencies between 0 – 6 kHz, around 10 kHz, around 20 kHz. The comments for the PSD estimate are also valid for the spectrogram. The spectrogram also shows that the mentioned frequency components are stronger after 390ms and that they are especially strong during the period of 390 – 430ms, which is when the choke valve changed position from 100 % to 28 % open.

Figure 26: Frequency analysis results for sensor 1 in oil and water flow. The choke position seems to have the highest influence on the acoustic emission and the most important frequencies are between 0 – 11 kHz.

Figure 27: Frequency analysis results for sensor 3 in oil and water flow. A small opening in the choke valve seems to generate higher frequency components than sensor 1.

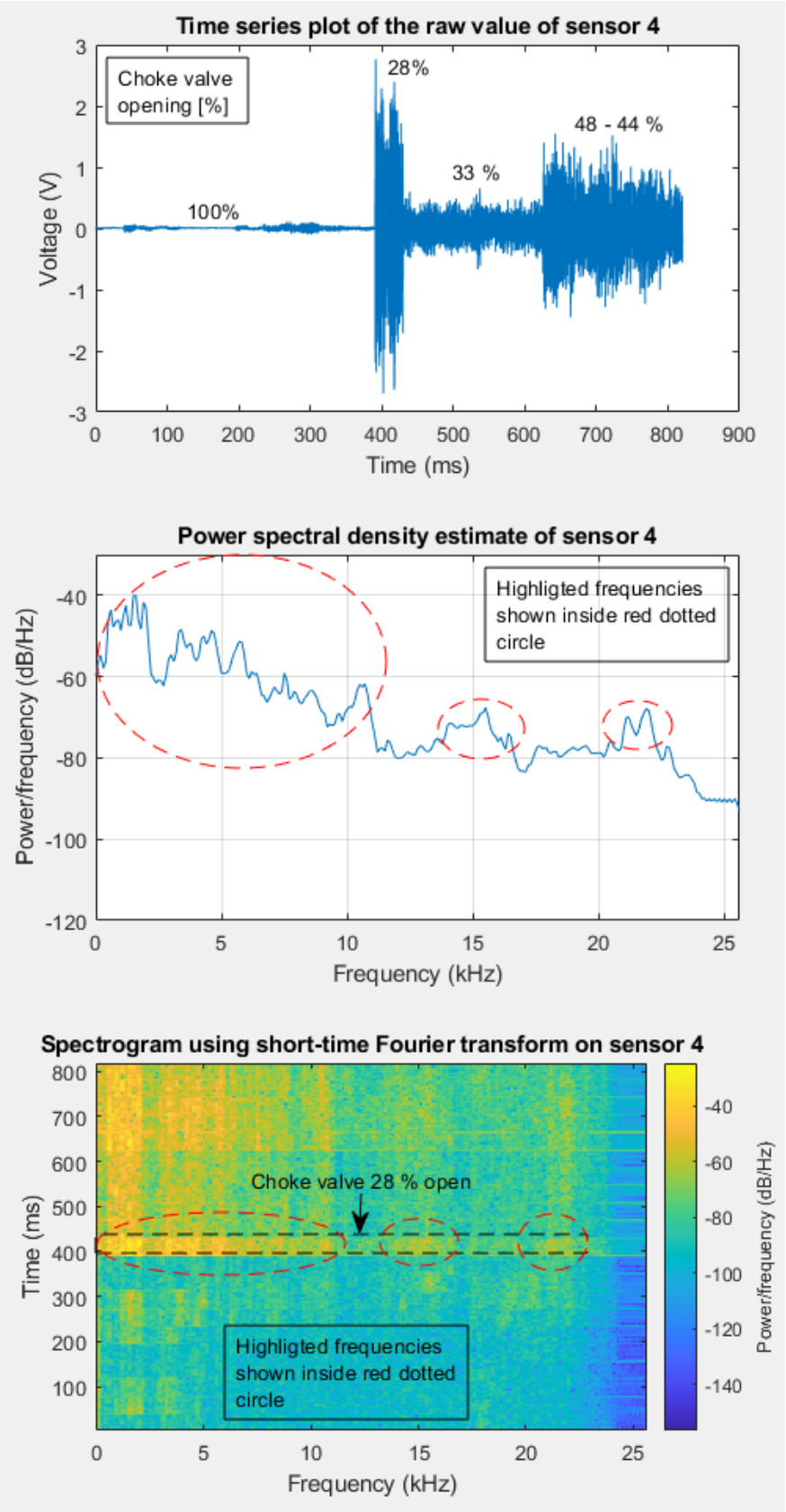


Figure 28: Frequency analysis results for sensor 4 in oil and water flow. This sensor seems to detect the highest levels of acoustic emission and detect many of the same high frequency components as sensor 3.

6.4.4.2 Frequency analysis of two-phase flow

Figure 29, Figure 30 and Figure 31 show the frequency analysis results of gas and oil flow. The time series plots show higher value of acoustic emission around the area 78ms, 312ms, and 508ms. This is the same time as the gas and total flow rate have a peak as shown in Figure 21. In addition, sensors 3 and 4 seem to detect higher acoustic emission after 508ms, which is when the choke position is changed from 100 % to values between 38 % and 50 %. Sensor 1 seems to be unaffected by the acoustic emission from the change of choke valve position.

The spectrogram for sensor 1 measure peaks in several frequency components for sensor 1 around 78ms, 312ms and 508ms. The highest peak seems to be around 3 kHz. Acoustic emission sensor 3 and sensor 4 detect higher levels of acoustic emission mainly around 1.6 kHz when the choke position is changed. This could be explained by the fact that a restriction in the pipe results in an increase in gas volume fraction in a gas-liquid flow, which again should cause more acoustic emission. See Ch.6.1.2 and Ch.6.1.3 for a more detailed explanation.

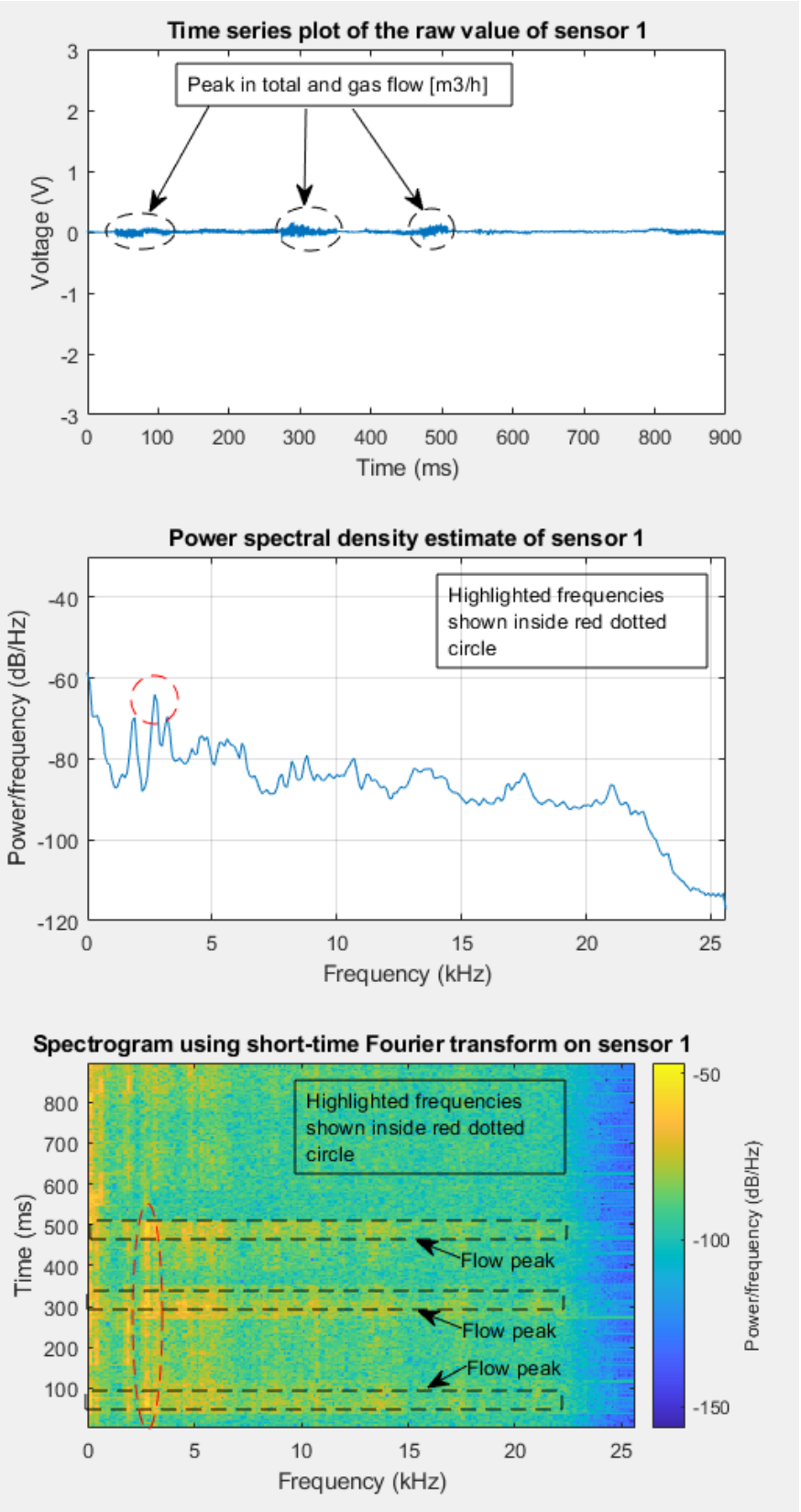


Figure 29: Frequency analysis results for sensor 1 in gas and oil flow. Frequencies around 3 kHz seem to correlate with the gas and total flow.

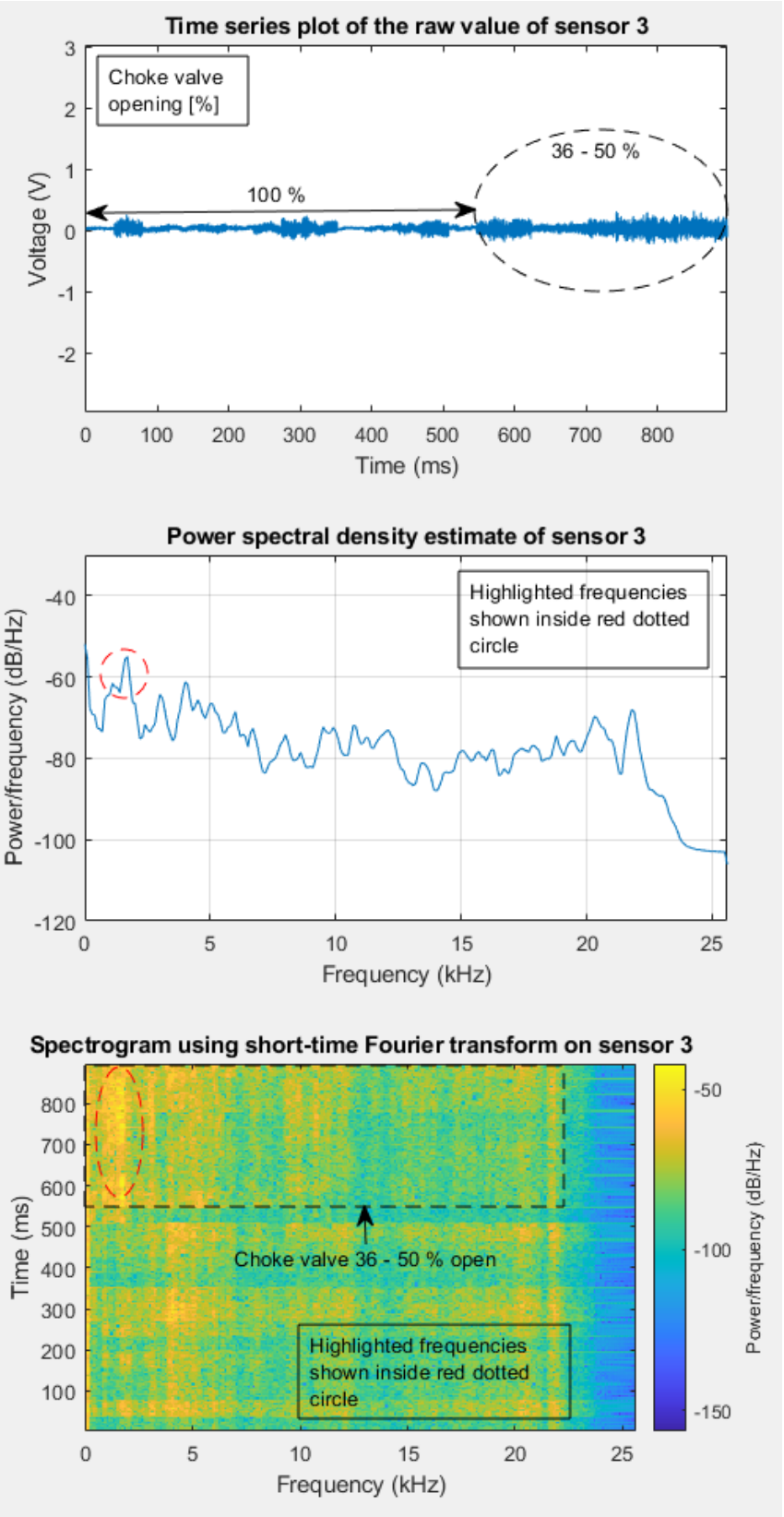


Figure 30: Frequency analysis results for sensor 3 in gas and oil flow. Frequencies around 1.6 kHz seem to correlate with the choke valve position.

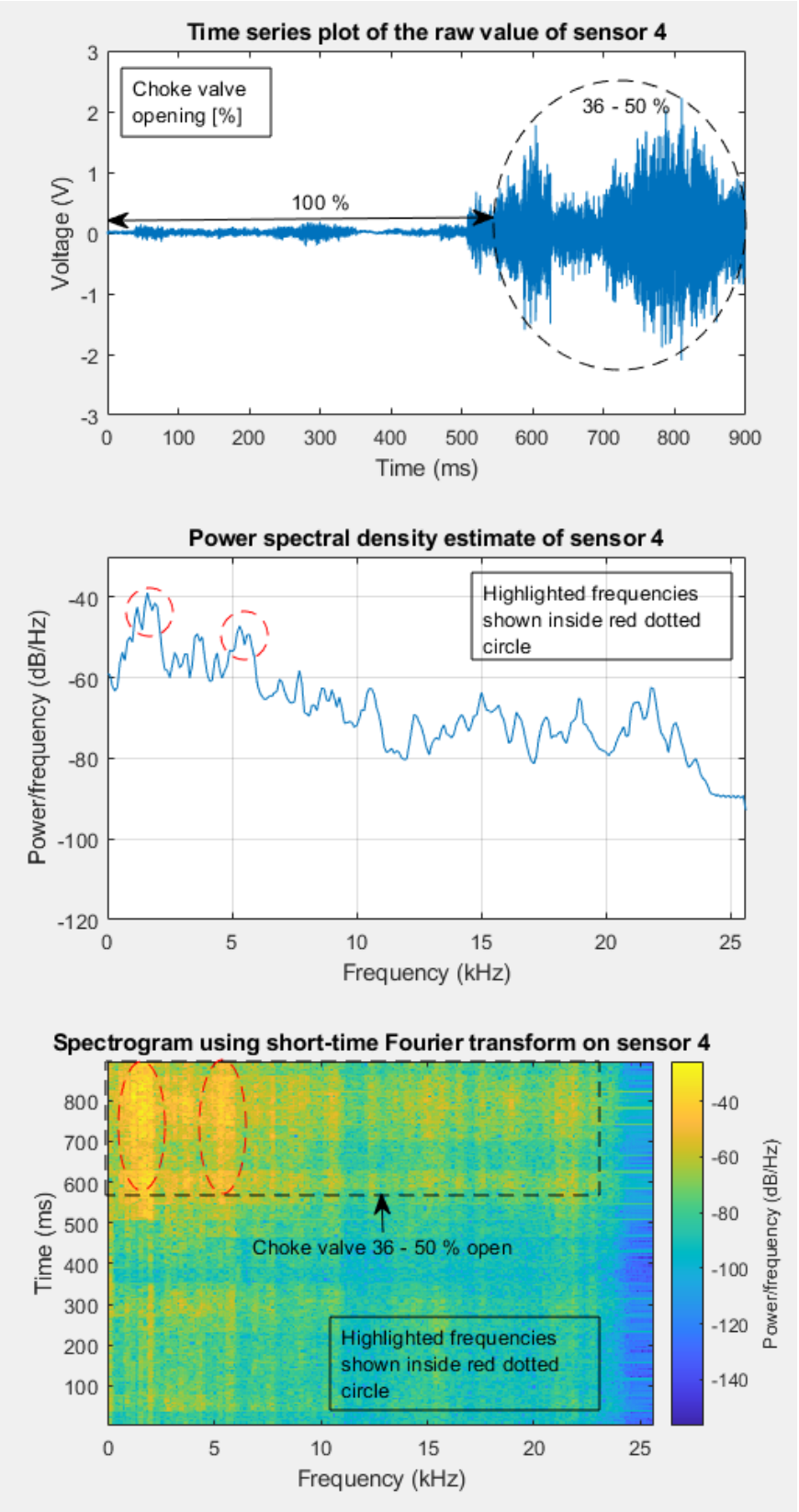


Figure 31: Frequency analysis results for sensor 4 in gas and oil flow. Especially frequencies around 1.6 kHz but also 6kHz seem to correlate with the choke valve position.

Figure 32, Figure 33 and Figure 34 show the frequency analysis results of gas and water. As shown in Figure 22 the water level was increased slowly, from 5 to 50 m^3/h, while the gas flow was stable around 30 m^3/h. The time series plot show that there seem to be little change in reading for sensor 1, while sensor 3 seem to have some small increasing reading over time. Sensor 4 seems to detect very little acoustic emission the first 78ms. After 78ms the water flow has reached 20 m^3/h and at that time it seems like the acoustic emission reading has increased in a high degree.

When studying the spectrogram and PSD estimate it seems like sensor 1 has little information. When studying the spectrogram for sensor 3, it seems like frequency components around 1.7 kHz may increase over time. The other frequency components seem to be unaffected by the change in water flow. There may be a relationship between frequency components around 1.7 kHz and total and water flow. It is harder to see any relationship for sensor 4, but after 78ms there seems to be some acoustic emission on several frequency components.

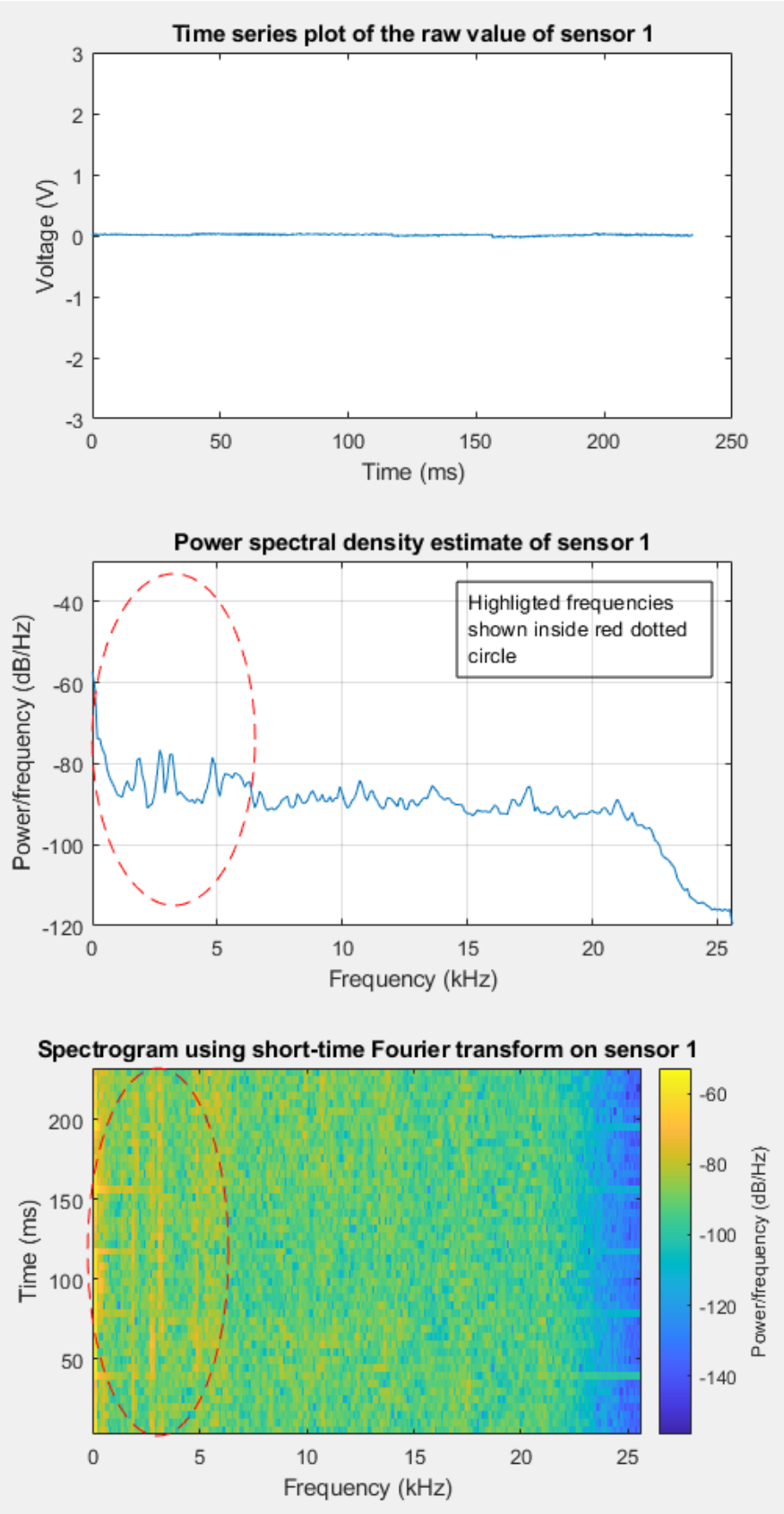


Figure 32: Frequency analysis of sensor 1 in gas and water flow. There seems to be almost zero acoustic emission, where the dominating range may be 0 – 6 kHz.

Figure 33: Frequency analysis of sensor 3 in gas and water flow. There may be some relationship between frequency components around 1.7 kHz and the water flow.

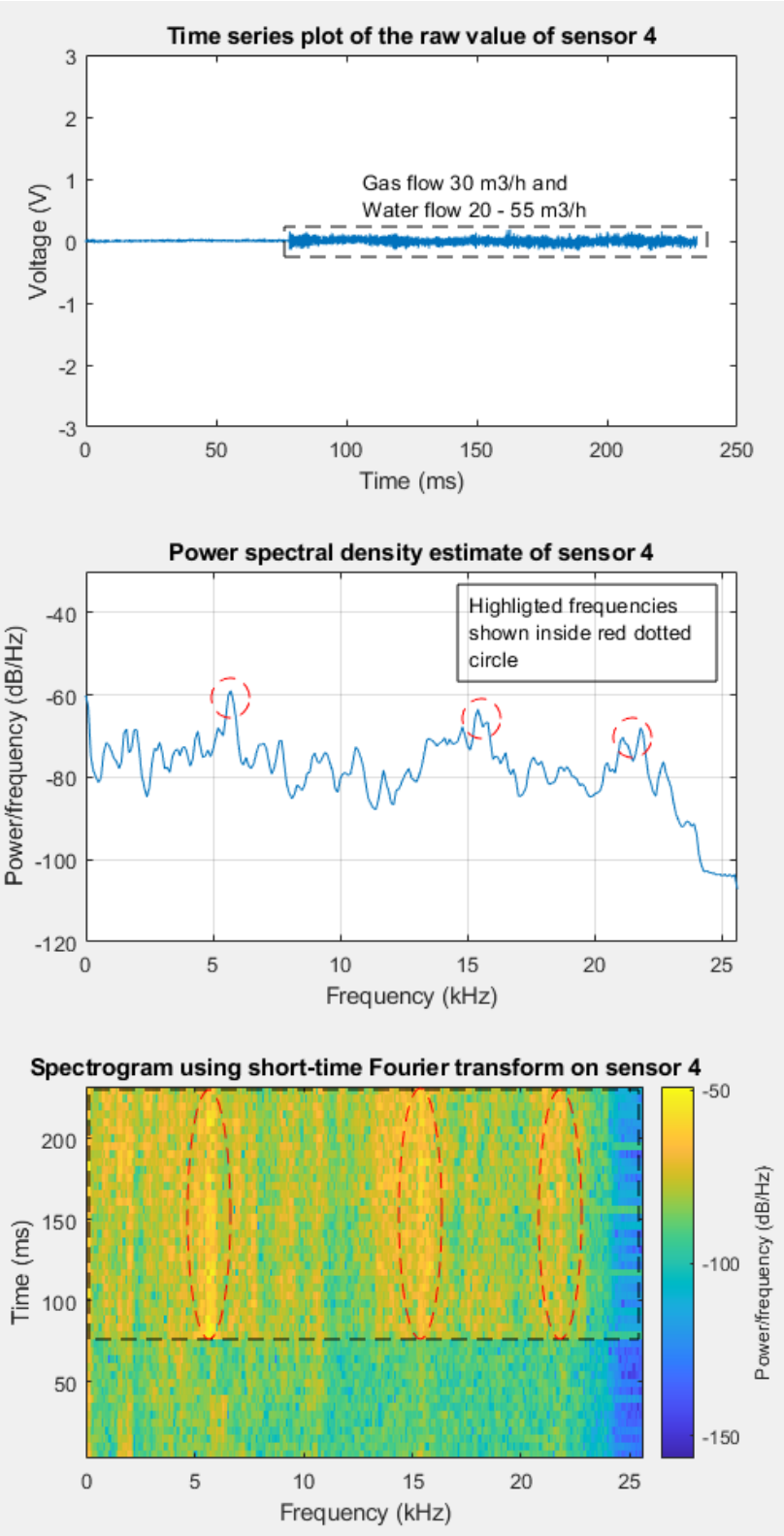


Figure 34: Frequency analysis of sensor 4 in gas and water flow. There seem to be some dominating frequency components after 78ms, but they seem to be constant after 78ms.

Figure 35 show the frequency analysis results of gas, oil, and water. As shown in Figure 23 the gas and oil flow are kept steady at 10 m^3/h for the first 390ms while the water flow increases from 1 – 40 m^3/h. The amplitude on sensor 3 and 4 seem to increase during that time, while sensor 1 is stable.

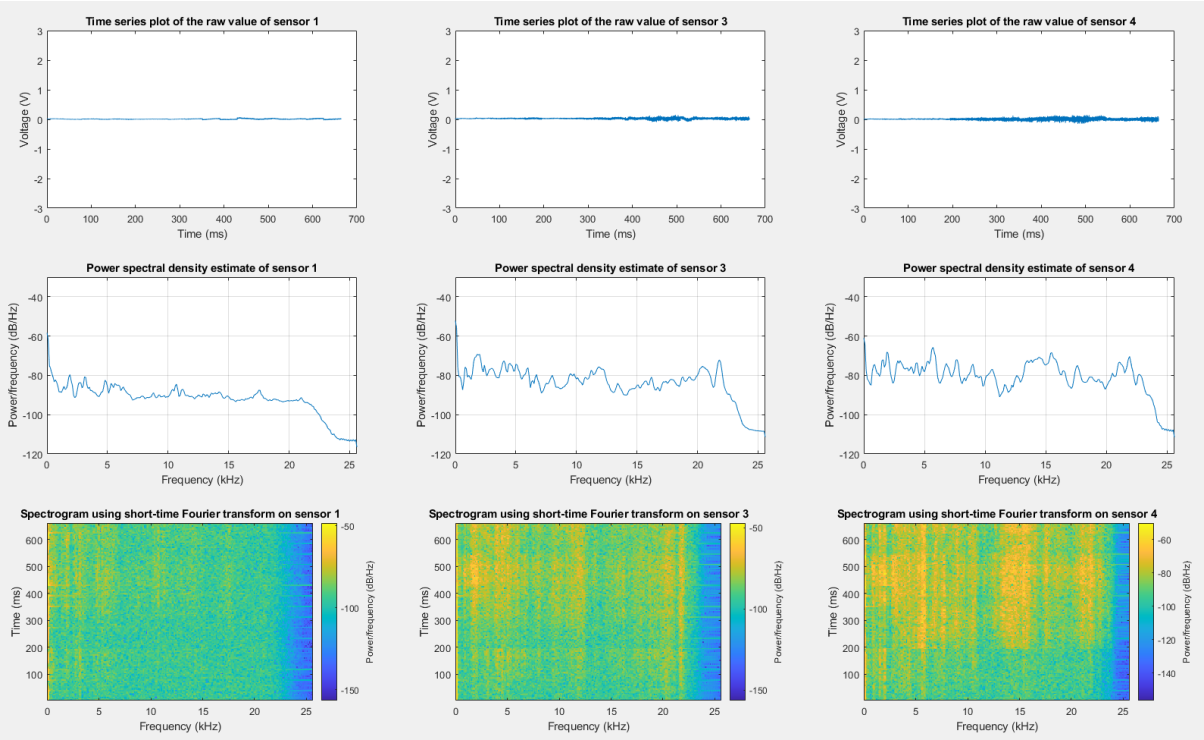


Figure 35: PSD estimation and Spectral analysis of two-phase flow containing gas, oil, and water. Acoustic emission sensor 3 and 4 detect a slight increase in acoustic emission when the water flow is increased.

6.5 Correlation analysis using selected process variables

In this chapter the variables in dataset 2 are analyzed. The code is in appendix B. The purpose is to check for correlation between variables in dataset 2. This analysis is done to reduce the number of variables that are not necessary or relevant for the models.

6.5.1 Correlation between differential pressure and flow measurements

In Figure 36 it is possible to see that there is a correlation between the differential pressure and the different flow rates. The gas flow rate has the highest correlation with differential pressure 2 over a Venturi 2 with 0.82 in correlation. The oil flow and the total flow have a good correlation with differential pressure 1 over the same Venturi. Notice that the total flow and the oil flow rate have almost the same correlation with the same Venturi and the total pressure drop over the test rig. It may be possible to not use a Venturi at all and only use total pressure drop to make a cheaper installation cost.

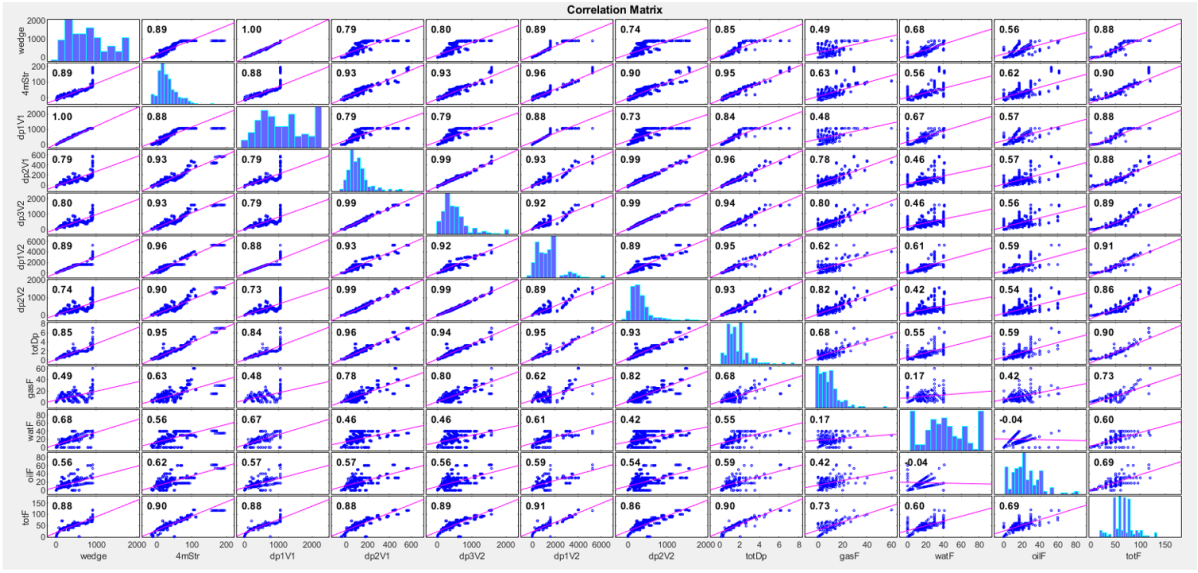


Figure 36: There seems to be some correlation between differential pressure and flow measurements in dataset 2.

6.5.2 Correlation between acoustic emission and flow

The correlation plot in Figure 37 show that there is a low correlation between accelerometer 2 and the gas flow. There seems to be a higher correlation between accelerometer 2 and oil and total flow, but accelerometer 1,3,4 may be enough.

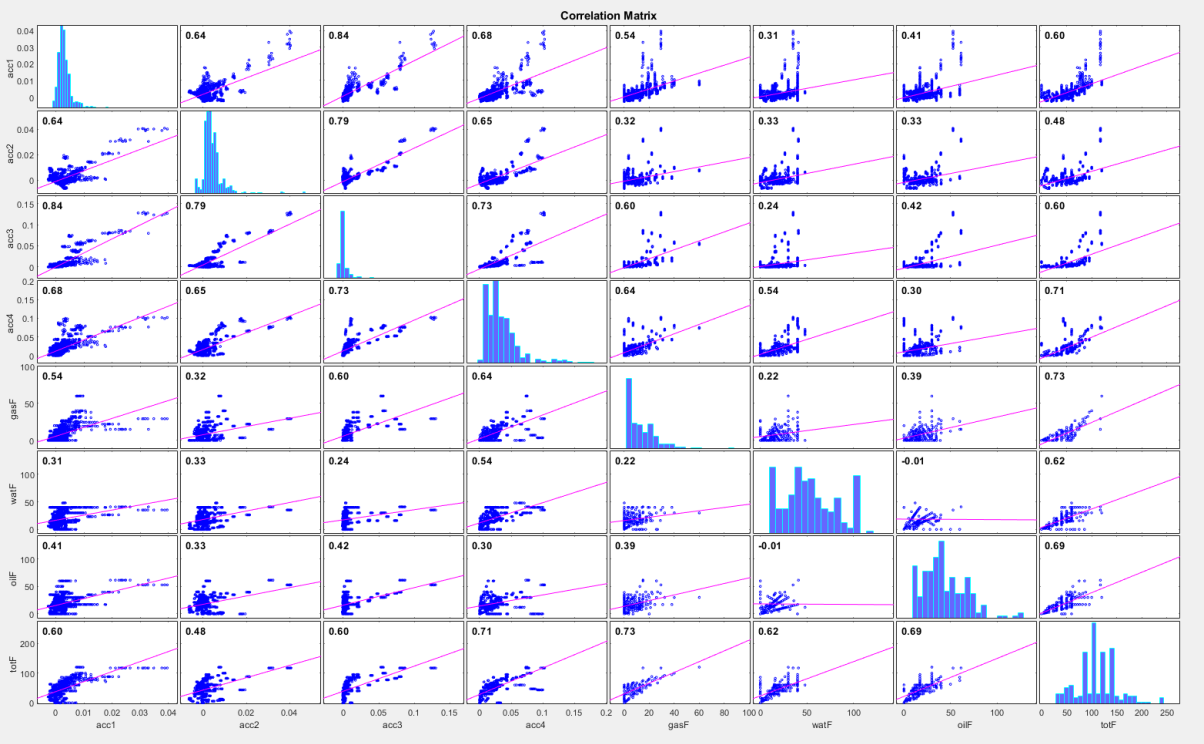


Figure 37: Correlation plot between accelerometers and flow measurements in dataset 2. Acoustic emission sensor 1, 3 and 4 seem to have highest correlation with the different flow measurements.

6.5.3 Correlation between temperature and flow measurements

The correlation plot in Figure 38 shot that the temperature has negligible correlation with the gas flow at the current conditions, but it has some correlation with the water, oil flow and the total flow. Increasing flow rate gives increased temperature of the medium.

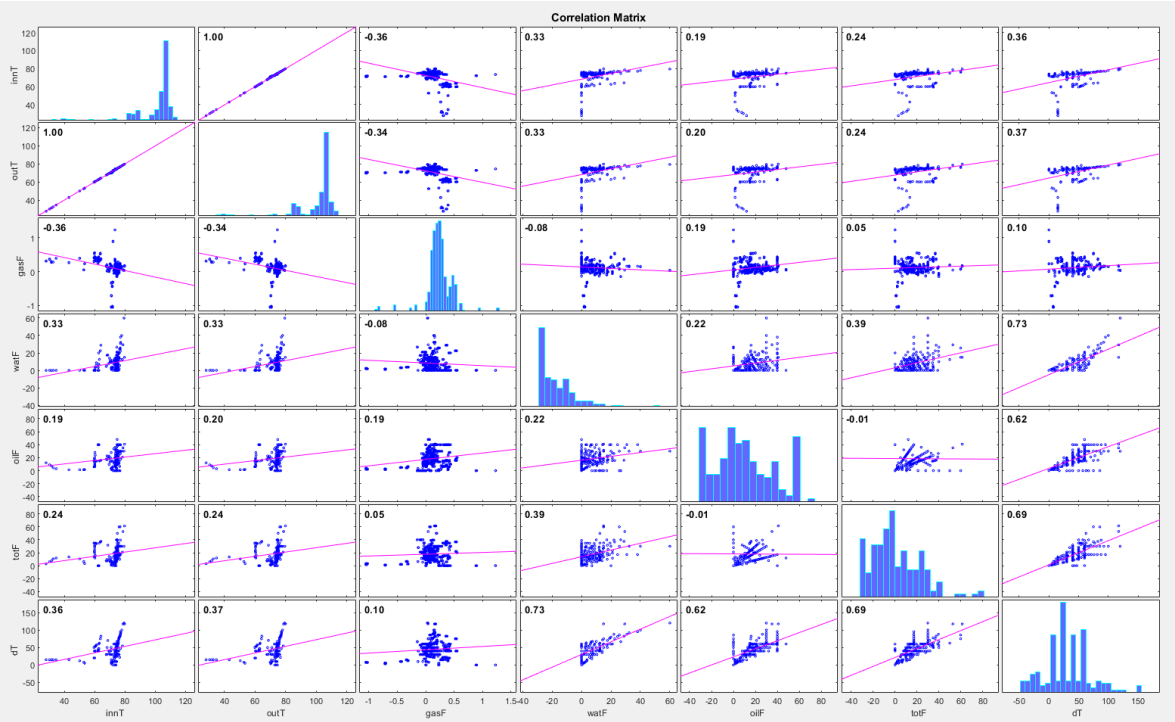


Figure 38: Correlation between temperature and flow measurements in dataset 2 show that in increase in flow may increase the temperature.

6.5.4 Correlation between density and flow measurements

The correlation plot in Figure 39 show that the density has a correlation with both water and gas flow. The measurements from densitometer den3Khas a correlation of 0.31 with the water flow, -0.61 with the gas flow and -0.18 with oil and total flow.

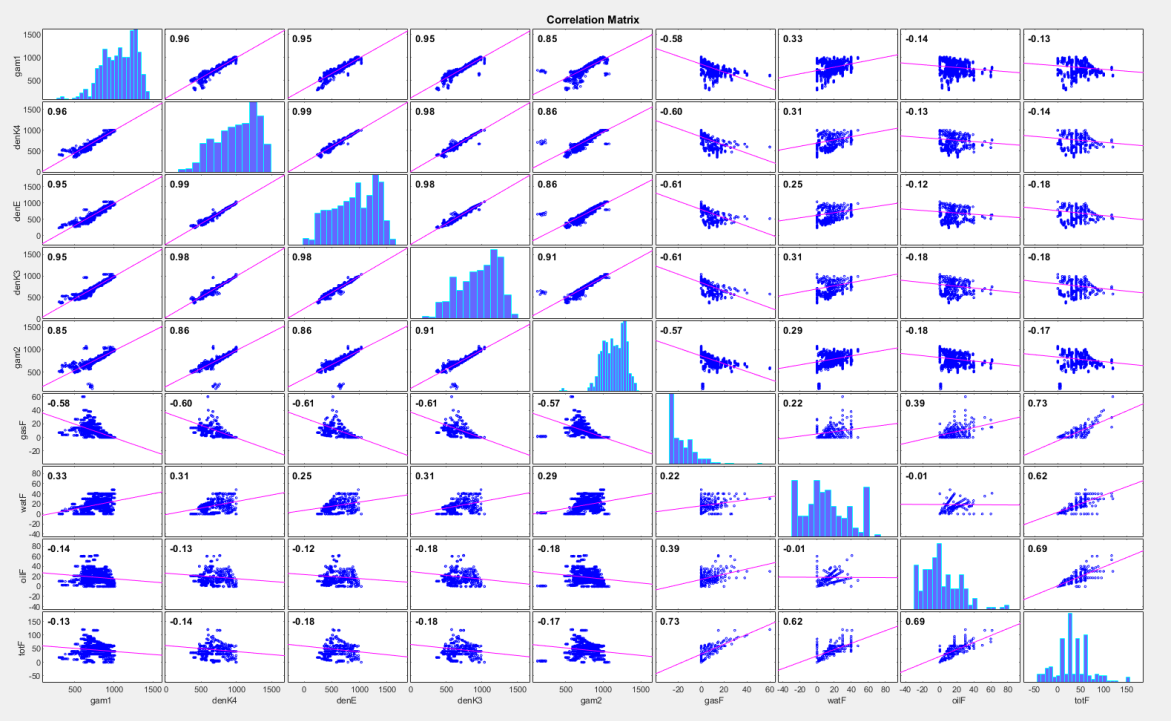


Figure 39: Correlation between density and flow measurements in dataset 2 show that there is a positive correlation with water flow, but a negative correlation with oil, gas and total flow.

6.5.5 Selected process variables after correlation analysis

The following variables below have been selected as input data for training the artificial neural network models in Ch.8.

- density Krohn3
- dp 4m straight
- dp1 Venturi2
- dp2 Venturi2
- Accelerometer 1
- Accelerometer 2
- Accelerometer 3
- Accelerometer 4
- Delta temperature
- Delta pressure

7 Preparing the dataset for the artificial neural networks

Dataset 2 will be used for training and testing artificial neural networks. This chapter will explain how the training and testing matrices are created. It will also explain the normalization of the inputs to the artificial neural networks.

7.1 Preparing the training and testing matrices

All measurements in dataset 2 have a sampling frequency of 1 Hz, except for the accelerometers. The accelerometers have a sampling frequency of 51.2 kHz. When training the artificial neural network all the variables in the dataset need to have the same number of samples and the same time stamp. The pseudocode below calculates the V_{RMS} value of 51 200 samples in the dataset. This is done such that there is only one V_{RMS} value per second for each of the four acoustic emission sensors. The pseudocode runs until it reaches the end, which is determined by the sampling time in seconds. The pseudocode only uses experiment GOW00, but the actual codes in appendix C, D, E and F includes more experiments.

```
for j = 1:4
    for i = 1:1:time
       GOW00_acc_1hz_rms(i,j) = rms(GOW00.testPoint.datAcc.ch(j).data(1+(i-1)*dt:i*dt));
    end
end
```

Figure 40 shows the measured voltage and calculated V_{RMS} for accelerometer 1 in experiment G00. The V_{RMS} is calculated from the first 51 200 samples as shown in the figure.

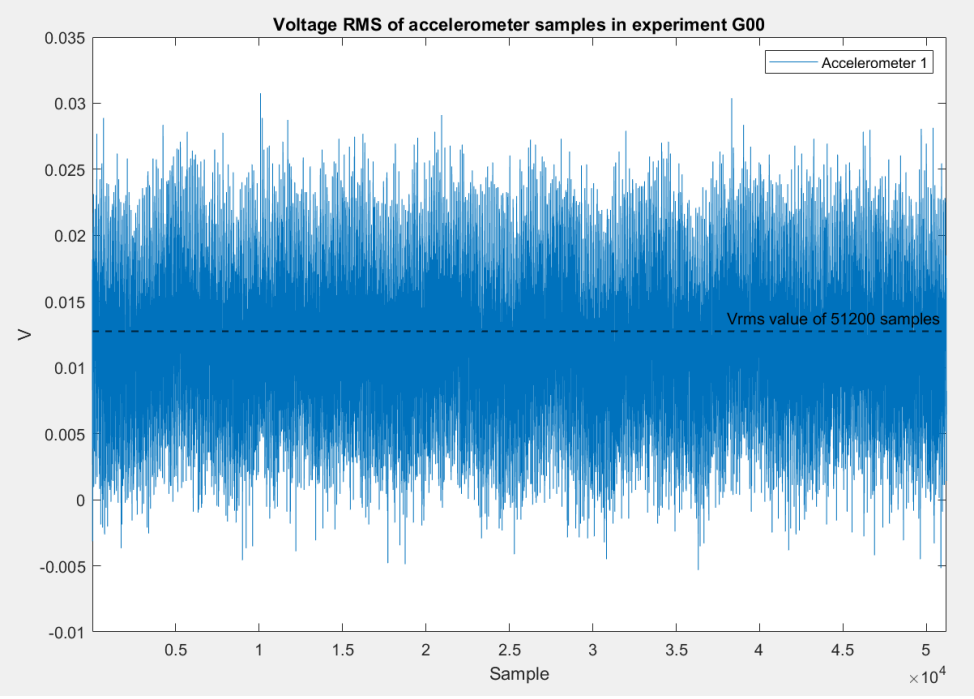


Figure 40: There is one calculated RMS value for each second in the dataset. Each RMS value is calculated from 51 200 samples. Accelerometer 1 in experiment G00 is used in this figure.

The computer used in this thesis struggled with the high amount of data in the MATLAB workspace. It was necessary to have several different codes for making training and testing matrices. The different codes are shown in appendix C, D, E and F. A pseudocode that selects which experiments and samples to use in the training and testing matrices are shown below. The pseudocode uses experiment GOW00 and GOW01 for the training matrix, while GOW143 is used in the testing matrix. The samples are chosen by the start and stop parameter. In the actual code much more experiments are used.

```
training = [GOW00_data(start:stop,:); GOW01_data(start:stop,:);……];
testing = [GOW143_data(start:stop,:);……];
```

The consequence of having different codes is that there are multiple training and testing matrices. In the pseudocode below the different matrices are merged into one matrix for training data and one matrix for testing data. The actual code contains more training and testing matrices than the pseudocode and are shown in Appendix B.

```
training_2 = [twophase_2.training_GO; twophase_2.training_GW;… ];
testing_2 = [twophase_2.testing_GO; twophase_2.testing_GW;… ];
```

The first rows of the training matrix are shown in Figure 41. The first 20 columns contain measurements from different sensors, while the next columns contain the V_{RMS} measurements from the accelerometers and the flow measurements.

Gamma 1 [kg/m3]	Wedge dp [mbar]	Mass flow Krohn4 [kg/m3]	Density Krohn4 [kg/m3]	TI [C]	PI [bar]	Mass flow Endres [kg/h]	Density Endres [kg/m3]	Mass flow Krohn3 [kg/h]	Density Krohn3 [kg/m3]	dp 4m straight [mbar]	dp 1 Venturi 1 [mbar]	dp 2 Venturi 1 [mbar]	Gamma 2 [kg/m3]	Valve [%]	dp 3 Venturi 2 [mbar]	dp 1 Venturi 2 [mbar]	dp 2 Venturi 2 [mbar]	To [C]	Po [bar]	XI1 [Vrms]	XI2 [Vrms]	XI3 [Vrms]	XI4 [Vrms]	Qg [m3/h]	Qw [m3/h]	Qo [m3/h]
500	121	9	383	73	66	7	279	16	433	2,49	173	43	495	100	193	397	158	73	66	0,01	0,01	0,02	0,01	14	0	21
516	122	9	388	73	66	7	289	17	456	1,84	173	44	498	100	192	382	155	73	66	0,01	0,01	0,02	0,01	14	0	21
488	131	9	380	73	66	8	299	17	480	2,36	172	44	502	100	192	393	153	73	66	0,01	0,01	0,02	0,01	14	0	21
498	130	8	373	73	66	8	309	18	504	2,94	171	45	483	100	194	394	157	73	66	0,01	0,01	0,02	0,01	14	0	21
537	120	9	397	73	66	8	320	17	479	2,40	173	43	523	100	188	383	152	73	66	0,01	0,01	0,02	0,01	14	0	21
531	117	10	422	73	66	8	304	16	453	3,16	174	40	565	100	199	403	167	73	66	0,01	0,01	0,02	0,01	14	0	21
498	125	12	448	73	66	7	288	17	450	2,63	176	38	559	100	199	423	162	73	66	0,01	0,01	0,02	0,01	14	0	21
506	122	11	426	73	66	6	272	17	449	2,76	177	39	540	100	198	443	156	73	66	0,01	0,01	0,02	0,01	14	0	21
514	118	10	404	73	66	6	272	17	448	2,91	179	45	526	100	200	444	158	73	66	0,01	0,01	0,02	0,01	14	0	21
495	123	9	381	73	66	6	272	16	444	2,15	178	44	498	100	194	414	159	73	66	0,01	0,01	0,02	0,01	14	0	21
506	121	9	350	73	67	9	299	17	468	1,32	172	40	512	100	187	390	148	73	66	0,01	0,01	0,02	0,01	14	0	21
484	120	9	356	73	67	9	303	17	473	1,65	172	40	529	100	196	386	160	73	66	0,01	0,01	0,02	0,01	14	0	21
520	120	8	361	73	67	9	308	17	473	1,18	172	39	520	100	191	385	151	73	66	0,01	0,01	0,03	0,01	14	0	21
531	121	9	376	73	67	9	312	17	473	2,01	171	39	529	100	185	388	148	73	66	0,01	0,01	0,02	0,01	14	0	21
526	129	10	391	73	67	8	300	17	463	1,78	171	39	525	100	201	400	167	73	66	0,01	0,01	0,02	0,01	14	0	21
530	132	10	398	73	67	7	286	17	453	1,52	171	39	546	100	189	397	154	73	66	0,01	0,01	0,02	0,01	14	0	21
555	133	10	405	73	67	7	286	16	443	1,91	171	40	536	100	193	401	155	73	66	0,01	0,01	0,03	0,01	14	0	21
544	113	10	412	73	67	7	286	17	446	2,00	171	40	525	100	197	405	163	73	66	0,01	0,01	0,02	0,01	14	0	21
564	113	11	429	73	67	8	287	17	449	1,86	170	40	524	100	190	403	156	73	66	0,01	0,01	0,03	0,01	14	0	21
536	120	11	445	73	67	7	285	17	452	0,97	170	41	503	100	197	378	167	73	66	0,01	0,01	0,02	0,01	14	0	21
678	179	19	618	74	67	14	499	21	627	3,84	224	23	676	100	174	529	132	74	66	0,01	0,01	0,02	0,01	7	0	28
641	178	18	607	74	67	14	494	21	622	3,80	224	23	674	100	170	538	126	74	66	0,01	0,01	0,02	0,01	7	0	28
654	177	18	595	74	67	13	488	22	633	3,43	224	23	680	100	173	520	126	74	66	0,01	0,01	0,02	0,01	7	0	28

Figure 41: The first rows of the training matrix in dataset 2 are shown in this figure. The first 20 columns contain the measurements from different sensors, followed by the accelerometer and flow readings.

The accelerometers are expected to read zero when there is no flow in the test rig, but there can be some noise interfering with the measurements. The pseudocode below calculates the V_{RMS} value of each of the four sensors when there is no flow through the test rig. This value is used to adjust the zero value of the acoustic emission sensors. The actual code contains more measurements than shown in the pseudocode below. The actual code can be found in appendix B.

```
XI1_noise_2 = rms([singlephase_2.background_noise(:,21);……]);
XI2_noise_2 = rms([singlephase_2.background_noise(:,22);……]);
XI3_noise_2 = rms([singlephase_2.background_noise(:,23);……]);
XI4_noise_2 = rms([singlephase_2.background_noise(:,24);……]);
```

The following formula in Eq.7.1 is used to adjust the acoustic emission sensors in the training and testing matrices. This is done by subtracting the V_{RMS} value containing only noise from the actual readings.

$$XI_{adj} = XI_{read} - XI_{noise} \quad \textit{(Eq.7.1)}$$

7.2 Normalizing inputs for an artificial neural network

The different variables in the adjusted training and testing matrices have different ranges and may therefore have different levels of influence on the training of the artificial neural networks. While the differential pressure over the Venturi has a range of 1584.99 mbar, accelerometer 4 has a much smaller range of 0.1 V_{RMS} like shown in Table 5. With the current situation the differential pressure over the Venturi will have a big influence on the training, while the accelerometers will have a low influence. This problem can be fixed by normalization. Four different normalization techniques are shown below in Table 4.

Table 4: Different normalization techniques (Google, 2022)

Normalization type	Formula	When to use
Linear scaling	$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$	The samples are uniformly distributed within a range
Clipping	$if\ x > x_{max}, then\ x' = x_{max}$ $if\ x < x_{min}, then\ x' = x_{min}$	The samples contain many outliers
Log Scaling	$x' = log(x)$	The samples follow the power law
Z-score	$x' = \frac{(x - \mu)}{\sigma}$	The samples do not contain extreme outliers

The samples do not seem to follow the power law, so the use of logarithmic scaling is not considered in this thesis. When studying Figure 42 it looks like the samples are distributed around the mean value, with some outliers. Linear scaling and Z-score will therefore be normalization techniques used in this thesis.

7.2.1 Linear scaling normalization

The requirement for using linear scaling is to have uniformly distributed samples. Even though this situation is not a perfect example of this, linear scaling will be used in addition to the Z-score.

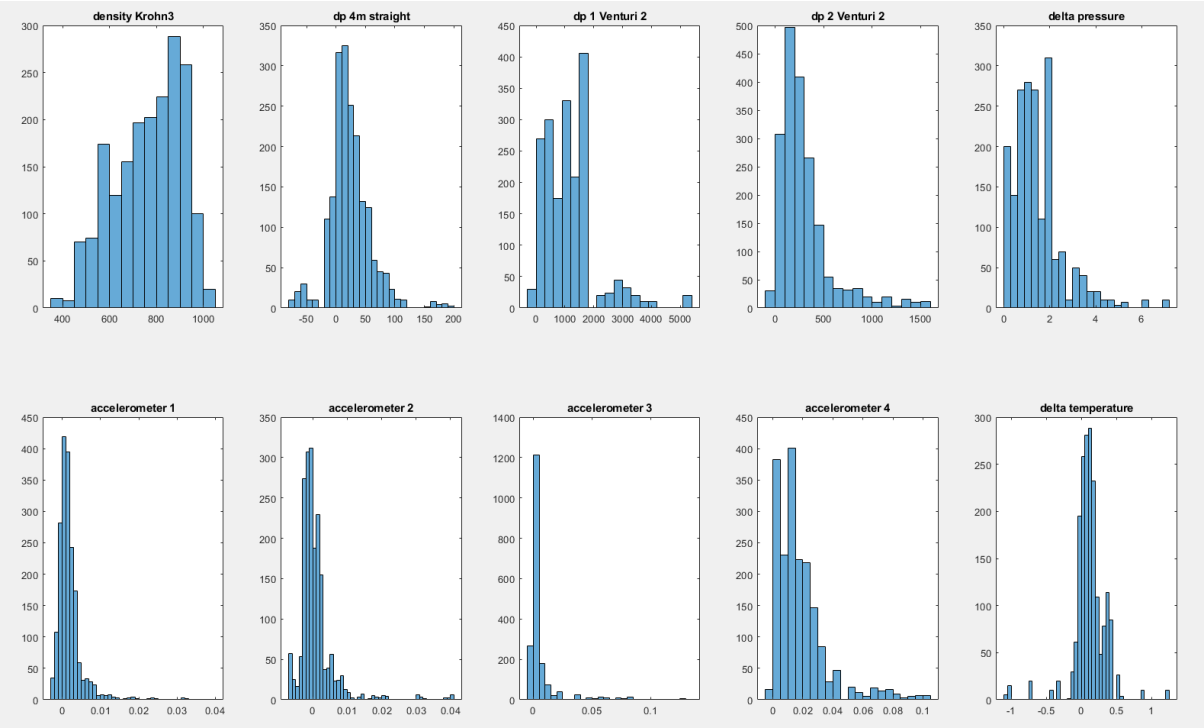


Figure 42: Histogram of samples in dataset 2 show that the samples are not perfectly distributed.

Table 5 show the maximum and minimum values of some variables in dataset 2.

Table 5: Maximum and minimum values

	density Krohn3 [kg/m^3]	dp 4m straight [mbar]	dp1 venturi2 [mbar]	dp2 venturi2 [mbar]	Acc. 1 [V]	Acc. 2 [V]	Acc. 3 [V]	Acc.4 [V]	ΔP [mbar]	ΔT [°C]
x_{maks}	1038,73	194,90	5259,22	1580,31	0,04	0,04	0,13	0,10	7,04	1,23
x_{min}	362,89	-72,32	-6,62	-4,68	0,00	-0,01	0,00	0,00	0,10	-1,05

The range of a variable is defined as the difference between the highest and lowest value as shown in Eq.7.2.

$$x_{range} = x_{max} - x_{min} \qquad (Eq.7.2)$$

The linear scaling normalization is done with the code below. The code is in appendix B.

```
dataMax_2 = max([dataTrainAdj_2; dataTestAdj_2]);
dataMin_2 = min([dataTrainAdj_2; dataTestAdj_2]);
trainLinear_2 = (dataTrainAdj_2 - dataMin_2)./(dataMax_2 - dataMin_2);
testLinear_2 = (dataTestAdj_2 - dataMin_2)./(dataMax_2 - dataMin_2);
```

In the upper plot in Figure 43 the original values are shown, while the lower plot shows the normalized values with the standard range of 0 – 1. Notice that the normalized values from the accelerometers, density and delta temperature seem to have a much higher influence on the neural network with the normalized range of 0 – 1.

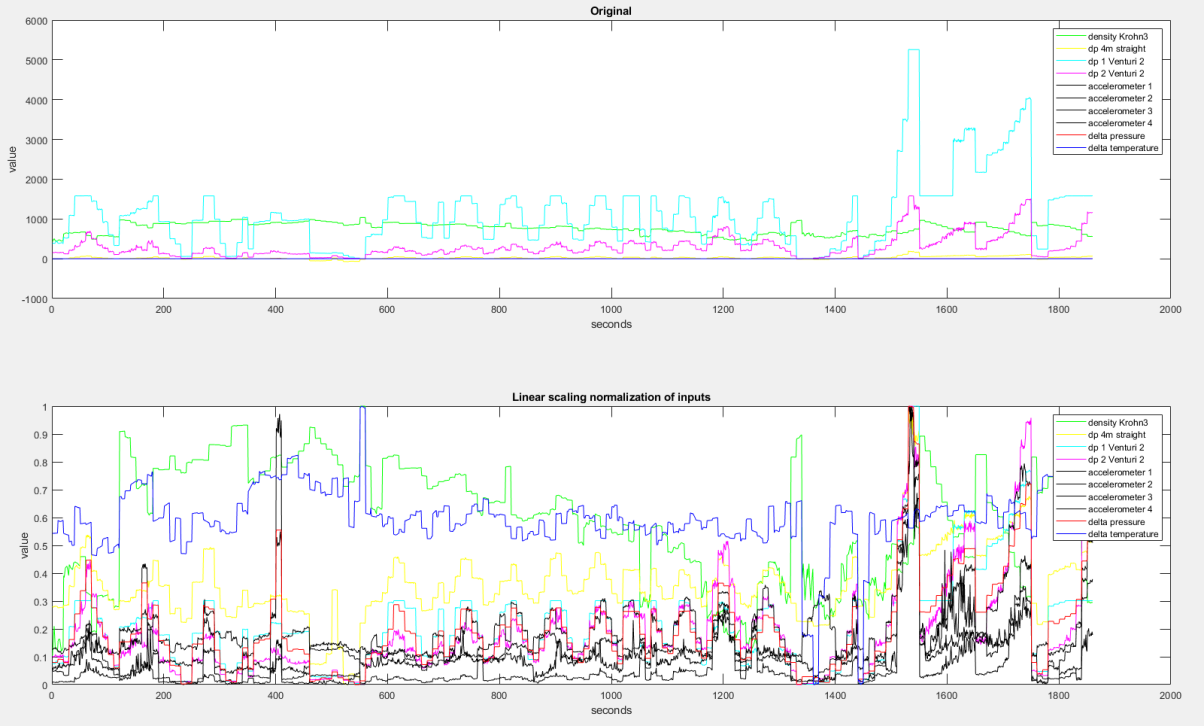


Figure 43: Normalization of training data using linear scaling. Notice that the delta temperature, density, and accelerometers seem to have a higher influence on the artificial neural network when normalized.

Figure 44 contain the same data as Figure 43, but since the plots are separated it is easier to see that there is a correlation between differential pressure measurements and acoustic emission.

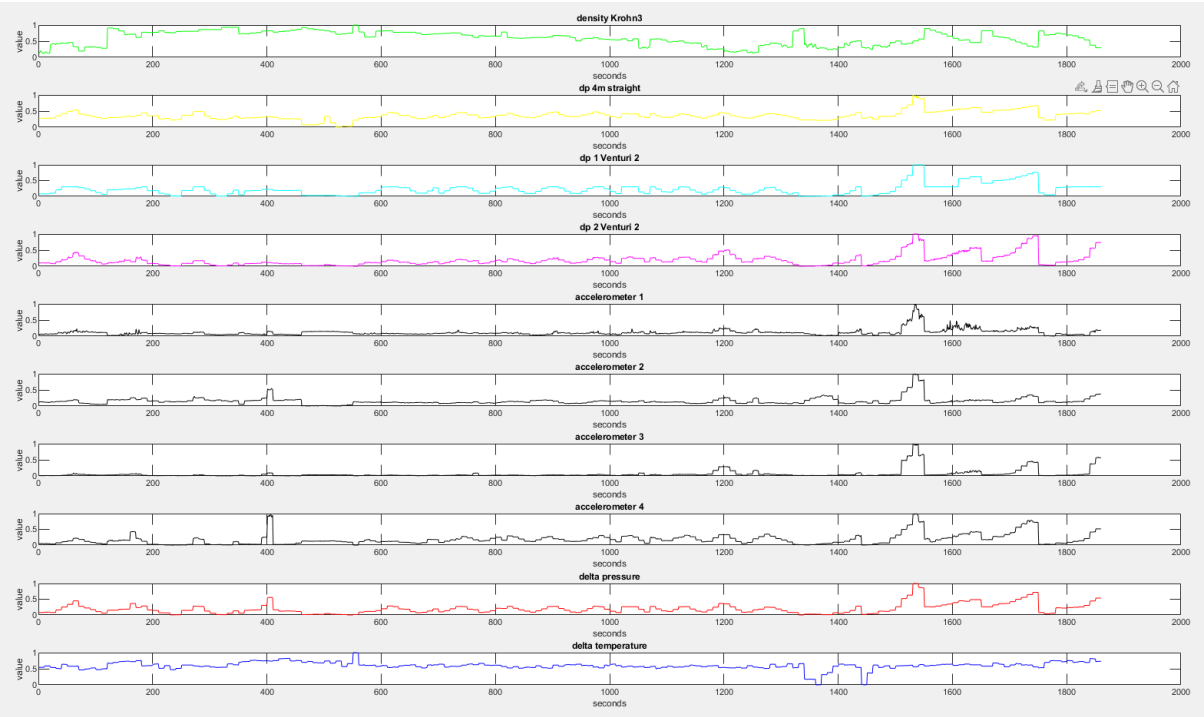


Figure 44: Separated plots of the normalized input data using linear scaling. There seem to be a correlation between acoustic emission and some differential pressure measurements.

7.2.2 Z-score normalization

Since our datasets are not perfectly uniformly distributed then the Z-score method for normalization has also been done. This method does not require uniform distribution of the samples, but extreme outliers can be a problem. The Z-score normalization has been done using the inbuilt MATLAB function zscore(). The code for Z-score normalization of the training and test data in dataset 2 is shown below. The code is in appendix B.

```
trainZscore_2 = zscore(dataTrainAdj_2);
testZscore_2 = zscore(dataTestAdj_2);
```

The results from Z-score normalization are shown in Figure 45 and Figure 46. The plot in Figure 46 is almost identical to Figure 44 except for the y-axis value. In Figure 45 it seems like normalization made the amplitude of the signals more alike, making all the inputs capable of influencing the model to the same degree.

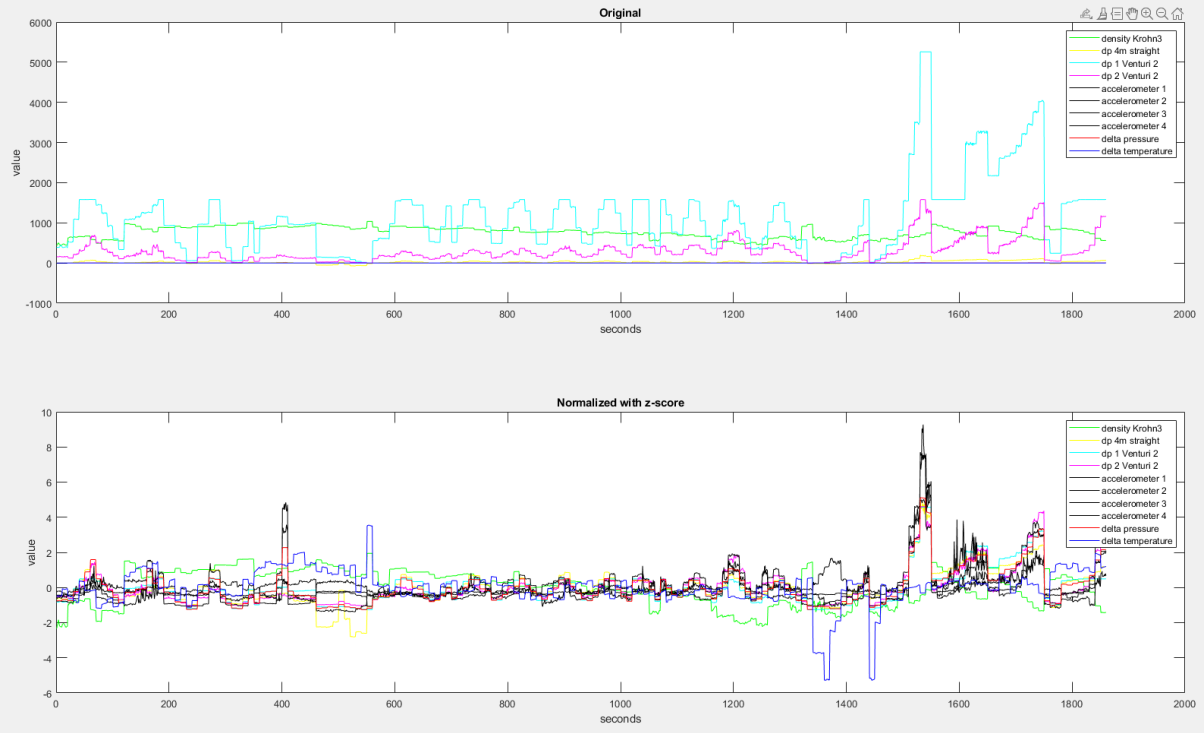


Figure 45: Normalized input data with Z-score method seem to make each measurement more capable of influencing the artificial neural network.

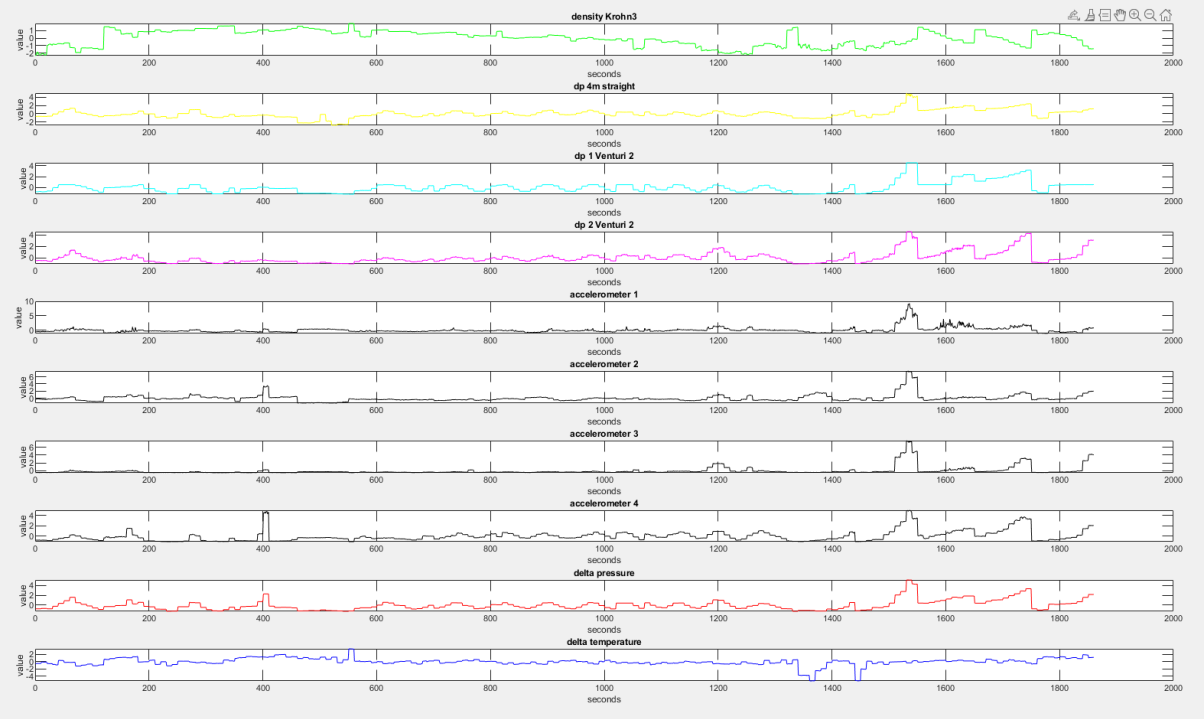


Figure 46: Plot of each of the normalized inputs using Z-score show that there seem to be a correlation between acoustic emission and differential pressure measurements.

8 Shallow neural network time-series modeling

The idea behind artificial neural networks is to make a network of artificial neurons that together imitate the neurons in human brains. Figure 47 show a comparison of an artificial neuron and a biological neuron. An artificial neuron calculates the weighted sum of its inputs, adds the bias, and calculates the output using the activation function.

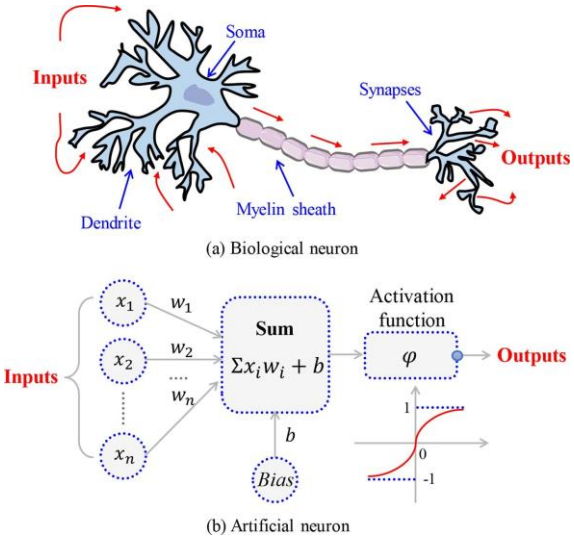


Figure 47: Comparison of an artificial and biological neuron (Xianlin Wang, 2021).

The neural network toolbox in MATLAB R2022b has been used to train all the neural network models. The toolbox is only able to train shallow networks with 1 hidden layer and one output layer. To make more complicated models the deep learning toolbox in MATLAB can be used. A shallow artificial neural network with one input layer, one hidden later and one output layer are shown in Figure 48. The network in Figure 48 have 3 inputs in the input layer, 5 artificial neurons in the hidden layer and 2 outputs in the output layer.

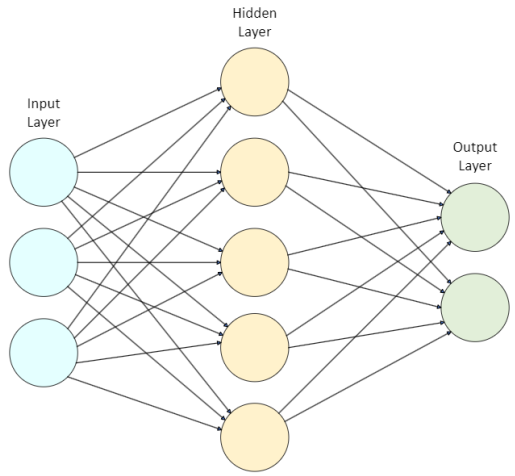


Figure 48: Shallow neural network with 3 inputs in the input layer, 5 neurons in the hidden layer and 2 outputs in the output layer

The Levenberg-Marquardt training algorithm gave the best training results and was therefore used for all the models. A data split of 80% training, 10% for validation and 10% testing was used for all models. The code for making all the figures in this chapter is in appendix B.

8.1 Nonlinear autoregressive with exogenous inputs neural network for multiphase flow estimation

The nonlinear autoregressive neural network with exogenous inputs (NARX) is a dynamic recurrent network. The next value of the time series $y(t)$ is calculated based on previous output values and past values of the inputs $x(t)$. The equation for a NARX model is shown in Eq. 8.1 (MathWorks, 2022).

$$y(t) = f\big(y(t-1), y(t-2), \dots, y(t-d), x(t-1), x(t-2), \dots, x(t-d)\big) \qquad (Eq.8.1)$$

Figure 49 shows a neural network in a parallel architecture where the estimated output is used as feedback. The network in Figure 49 have 8 inputs in the input layer, 5 neurons in the hidden layer and one output in the output later. The activation function in the hidden layer is transoid function, while the output layer is linear function. The time delay is set to 1.

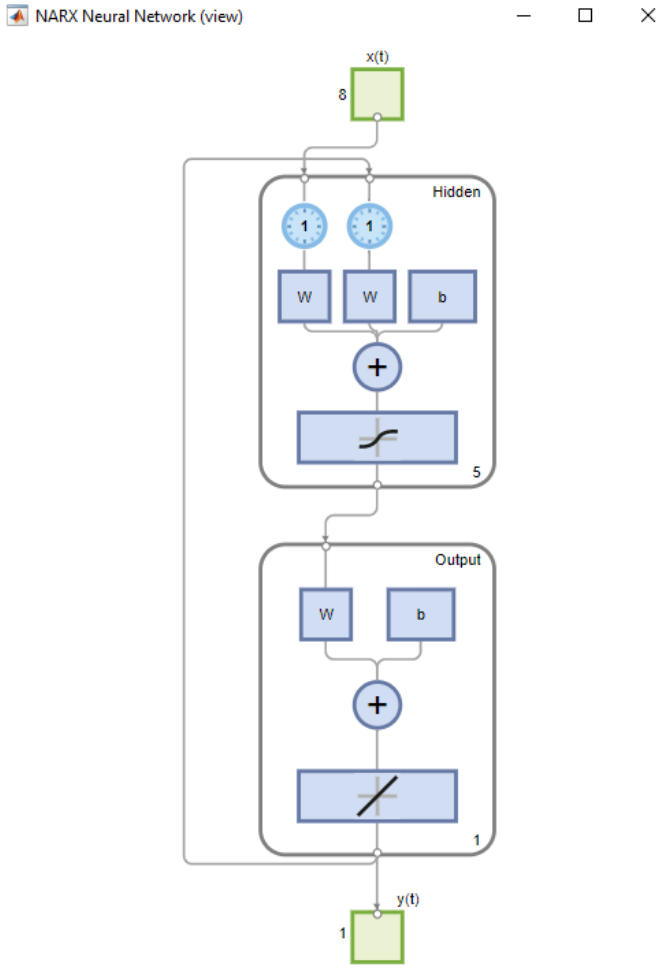


Figure 49: Closed loop NARX shallow neural network with one hidden layer and one output layer. A configuration of 1 time delay and 5 neurons in the hidden layer and 1 output in the output layer.

The NARX model uses the output as feedback. Figure 50 show the testing results of a neural network with a time delay of 1. With a time, delay of 1 it is possible to see that the last output value is based on the previous value and that in this case this has resulted in an error every time there is a change in the setpoint. This may indicate that the previous output value has no relationship with the next.

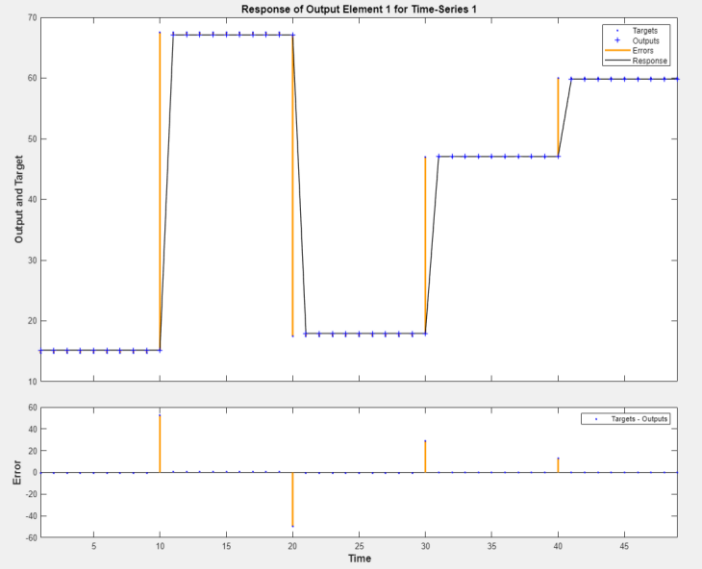


Figure 50: NARX network with a time delay of 1. The next output is based on the previous value.

With a time, delay of 5 the next output is based on the previous five output values. In this case this has resulted in an error every time there is a big change in the output and the effect lasts for the next 5-time stamps like shown in Figure 51. If there is a relationship between the last output and the next this function would have been useful.

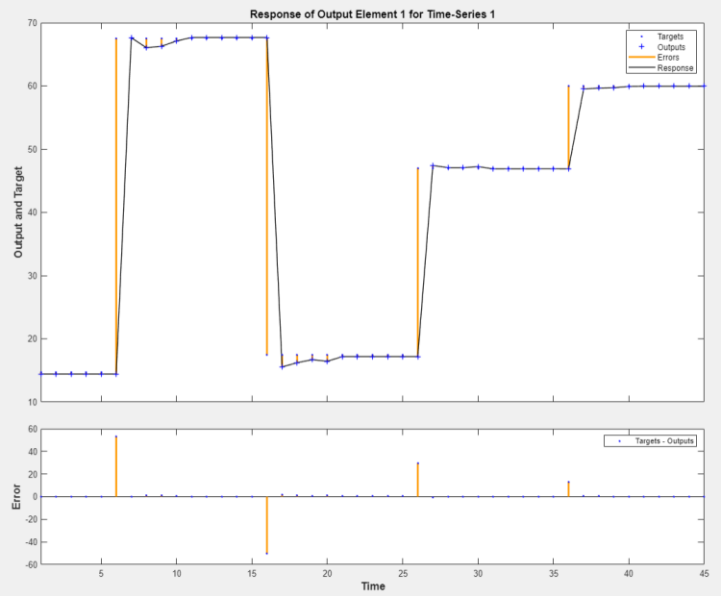


Figure 51: NARX network with a time delay of 5. The next output is based on the previous 5 output values.

Training the NARX networks is done with a series-parallel architecture, also called an open loop configuration. The actual output is available during training and is therefore used as feedback during training. This way the training is done with the highest degree of accuracy since the actual output values are used as feedback. The open loop configuration is shown in Figure 52.

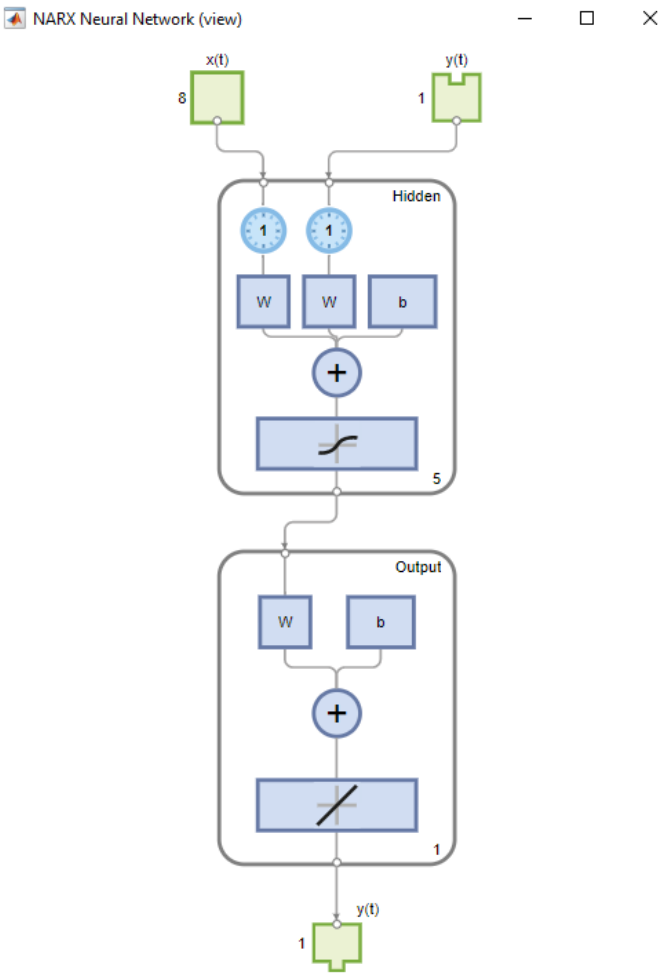


Figure 52: The NARX networks are trained with an open loop configuration using the actual output as feedback. This network has one input layer with 8 inputs, one hidden layer with 5 neurons and one output layer with one output. The time delay is set to 1.

After training the network it will be tested on another dataset. The estimated output will then be used as feedback as shown in Figure 49. Keep in mind that there can be a big difference between the training results and testing results after the loop is closed like shown in Figure 49. The closeloop() function in MATLAB is used to close the loop after the trained model is exported to the workspace. Before the network can be simulated the data needs to be prepared. The inputs and output time series are shifted as many time steps as needed to get enough initial input delay states, X_i, and initial layer delay states, A_i, as needed. The shifted input time series are called X_s. The function is in appendix B.

```
function y = SimulateNARXNetwork(network, inputData, outputData)
    [Xs,Xi, Ai] = preparets(network, inputData, {}, outputData);
    y = sim(network, Xs, Xi, Ai);
end
```

8.1.1 NARX gas flow rate model

It is desired to estimate the gas flow rate. This chapter shows the results of the gas flow models using a NARX neural network.

8.1.1.1 NARX gas flow rate model using 8 inputs and a linear scaling normalization

Training of the NARX network finished after 14 epochs. Training stopped when validation criterion was met after 6 checks. At that time the gradient = 4.1478 and Mu = 0.01. The regression line and mean square error can be shown in Table 6 below. The training results are shown in Figure 53, while the results from testing the model on the testing matrix are shown in Figure 54. Table 6 show the training results.

Table 6: Training results of the NARX gas flow model using 8 inputs and linear scaling normalization

	Observations	**MSE**	**R**
Training	1399	2.8490	0.9816
Validation	175	1.3612	0.9926
Test	175	0.5130	0.9976

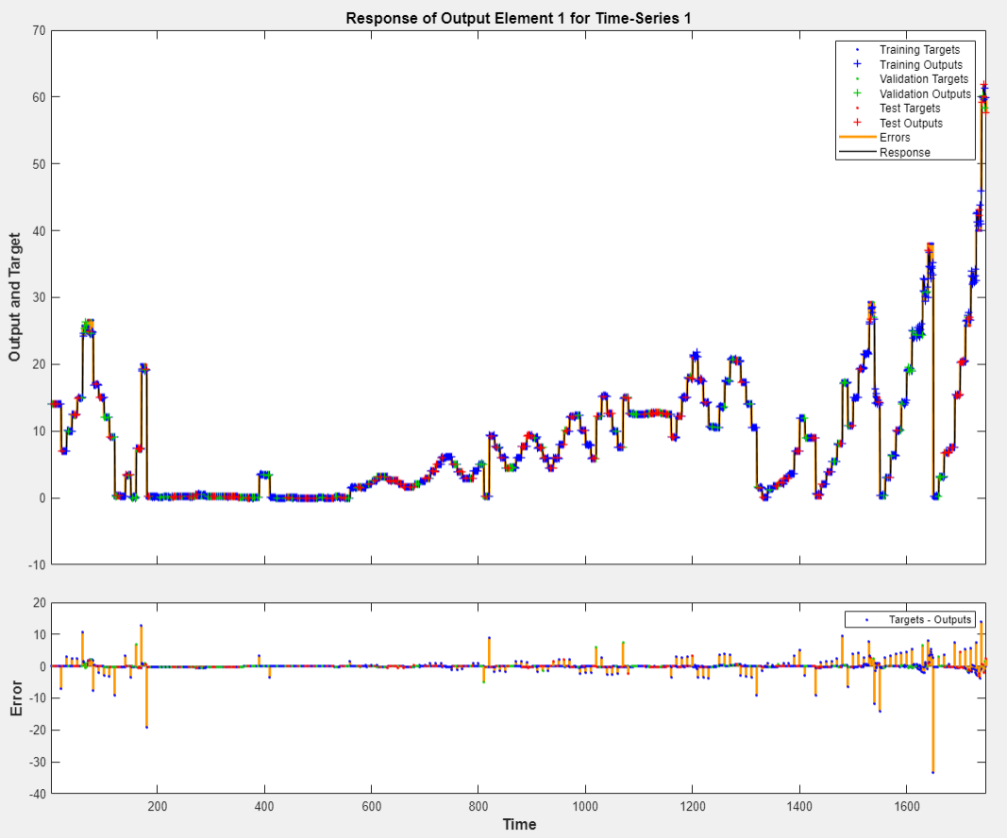


Figure 53: Time series response of the NARX gas flow model using 8 inputs an and linear scaling normalization

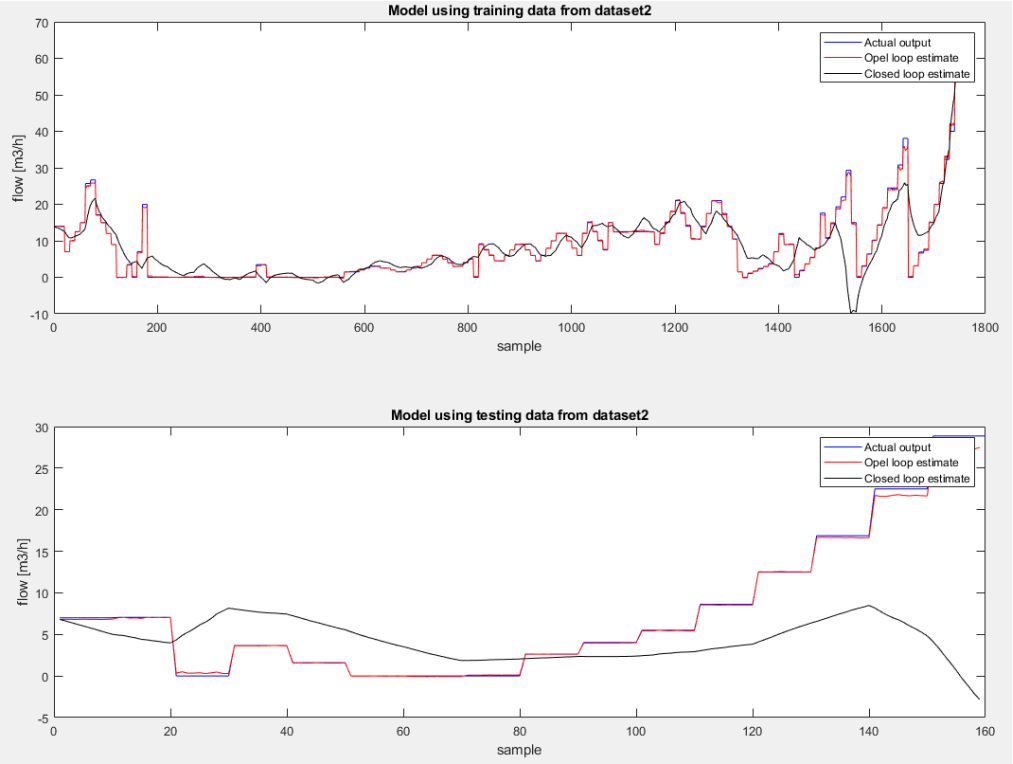


Figure 54: Testing the NARX gas flow model with 8 inputs and linear scaling normalization in both open and closed loop.

8.1.1.2 NARX gas flow rate model using 8 inputs and Z-score normalization

Training of the NARX network finished after 45 epochs. Training stopped when validation criterion was met after 6 checks. At that time the gradient = 22.5961 and Mu = 0.001. The regression line and mean square error can be shown in Table 7. The training results are shown in Figure 55, while the results from testing the model on the testing matrix are shown in Figure 56.

Table 7: Training results of the NARX gas flow model using 8 inputs Z-score normalization

	Observations	**MSE**	**R**
Training	1399	2.7954	0.9830
Validation	175	2.4007	0.9831
Test	175	0.9748	0.9947

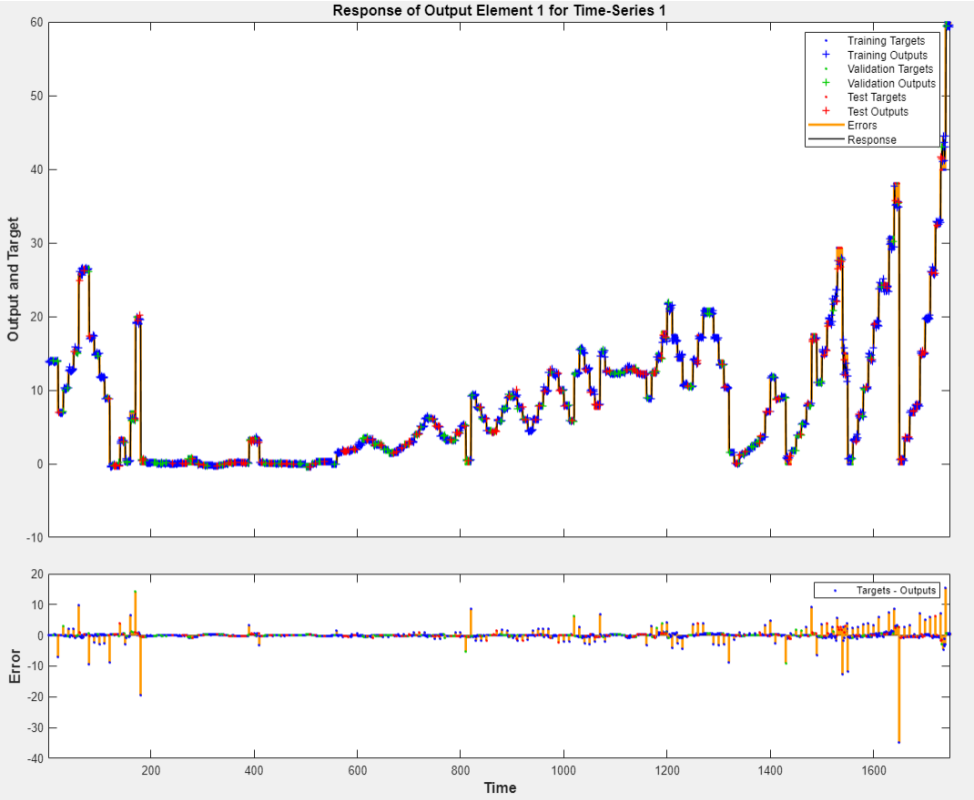


Figure 55: Time series response of the NARX gas flow model using 8 inputs an and Z-score

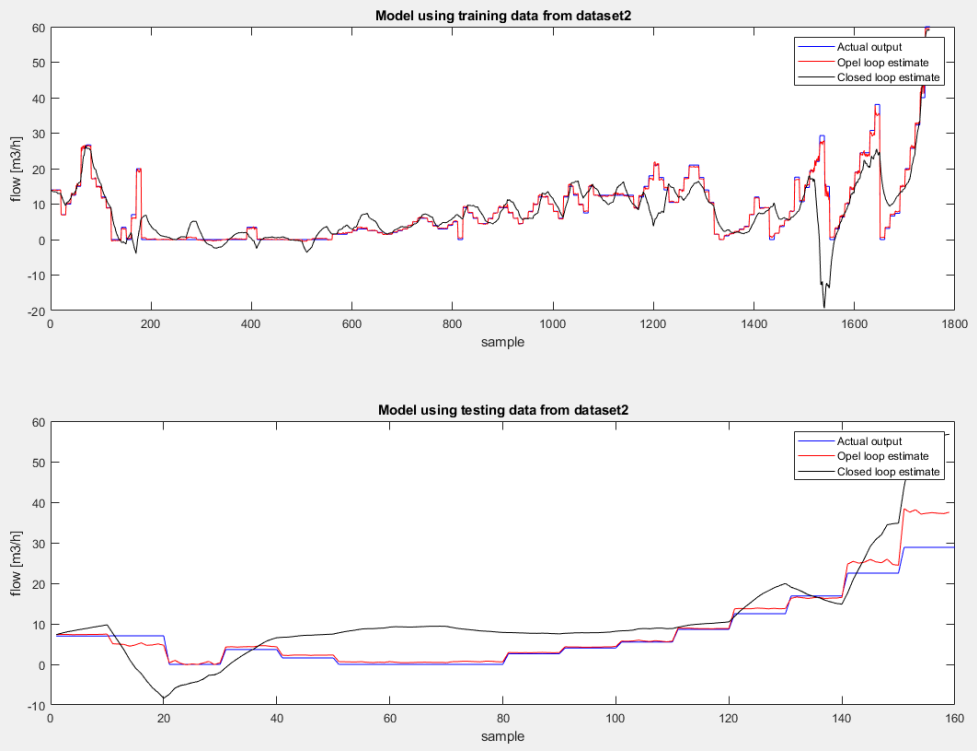


Figure 56: Testing the NARX gas flow model with 8 inputs and Z-score normalization in both open and closed loop.

8.1.2 NARX water flow rate model

It is desired to estimate the oil flow rate. This chapter shows the results of the water flow models using a NARX neural network.

8.1.2.1 NARX water flow model using 7 inputs and linear scaling normalization

Training of the NARX network finished after 19 epochs. Training stopped when validation criterion was met after 6 checks. At that time the gradient = 12.9312 and Mu = 0.001. The regression line and mean square error can be shown in Table 8. The training results are shown in Figure 57, while the results from testing the model on the testing matrix are shown in Figure 58.

Table 8: Training results of the NARX water flow model using 7 inputs and linear scaling

	Observations	**MSE**	**R**
Training	1399	5.5361	0.9806
Validation	175	2.0226	0.9927
Test	175	3.5579	0.9893

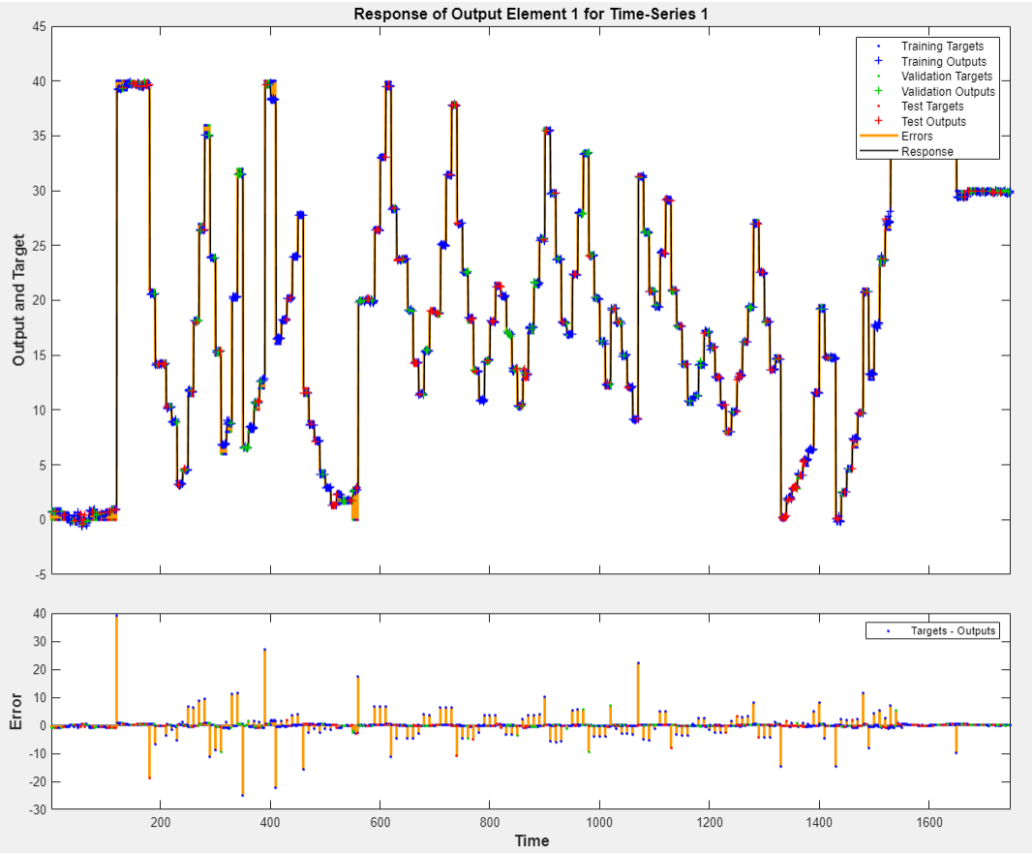


Figure 57: Time series response of the NARX water flow model using 7 inputs and linear scaling normalization

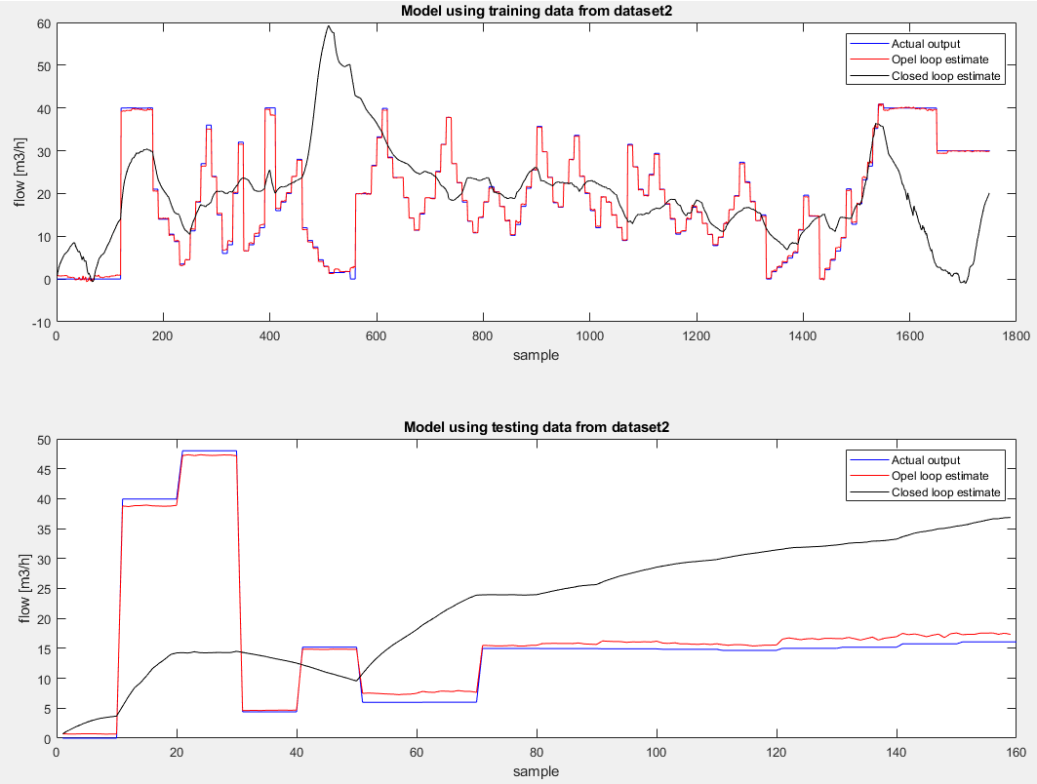


Figure 58: Testing the NARX oil flow model with 7 inputs and linear scaling normalization in both open and closed loop

8.1.2.2 NARX water flow rate model using 7 inputs and Z-score normalization

Training of the NARX network finished after 15 epochs. Training stopped when validation criterion was met after 6 checks. At that time the gradient = 2.3284 and Mu = 0.001. The regression line and mean square error can be shown in Table 9. The training results are shown in Figure 59, while the results from testing the model on the testing matrix are shown in Figure 60.

Table 9: Training results of the NARX water flow model using 7 inputs and Z-score

	Observations	**MSE**	**R**
Training	1399	5.7672	0.9801
Validation	175	2.1298	0.9930
Test	175	2.0271	0.9922

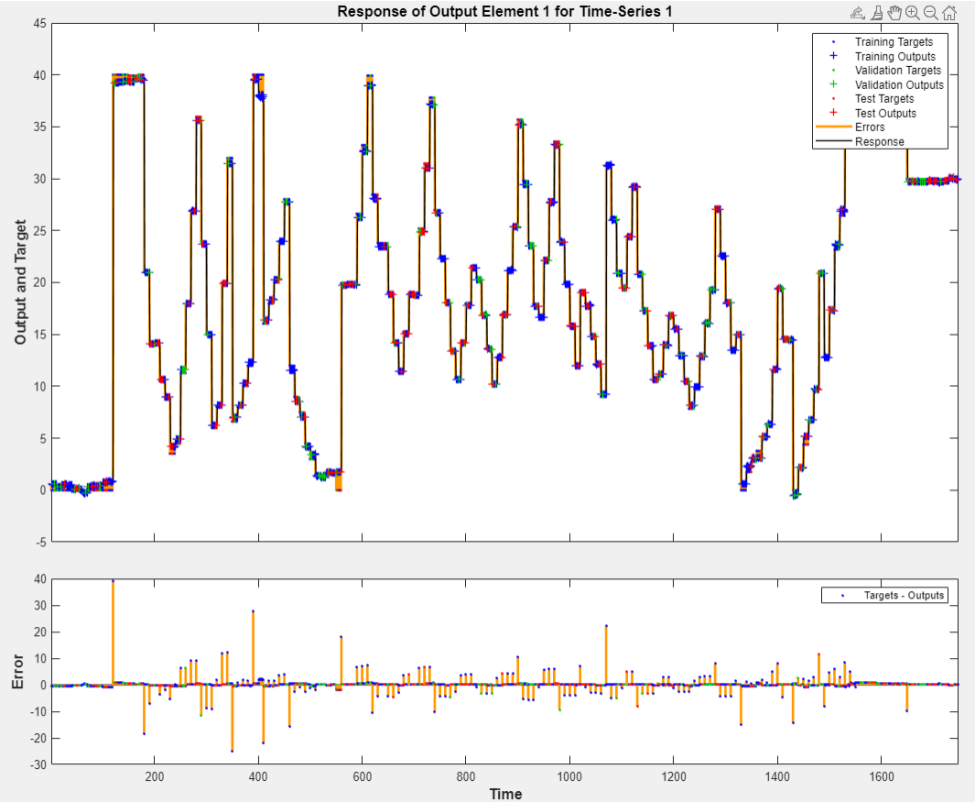


Figure 59: Time series response of the NARX water flow model using 7 inputs and Z-score

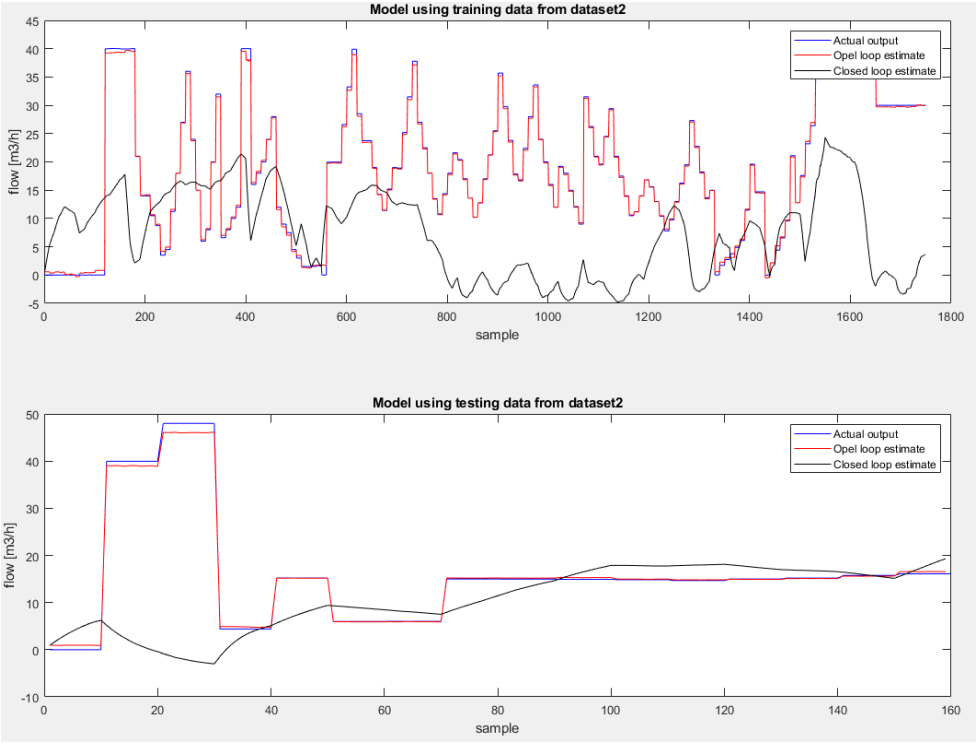


Figure 60: Testing the NARX water flow model with 7 inputs and Z-score normalization in both open and closed loop

8.1.3 NARX oil flow rate model

It is desired to estimate the oil flow. This chapter shows the results of the oil flow models using a NARX neural network.

8.1.3.1 NARX oil flow NARX model using 7 inputs and linear scaling normalization

Training of the NARX network finished after 16 epochs. Training stopped when validation criterion was met after 6 checks. At that time the gradient = 11.6055 and Mu = 0.001. The regression line and mean square error can be shown in Table 10. The training results are shown in Figure 61, while the results from testing the model on the testing matrix are shown in Figure 62.

Table 10: Training results of the NARX oil flow model using 7 inputs and linear scaling normalization

	Observations	**MSE**	**R**
Training	1399	7.0304	0.9740
Validation	175	2.6748	0.9904
Test	175	4.0626	0.9889

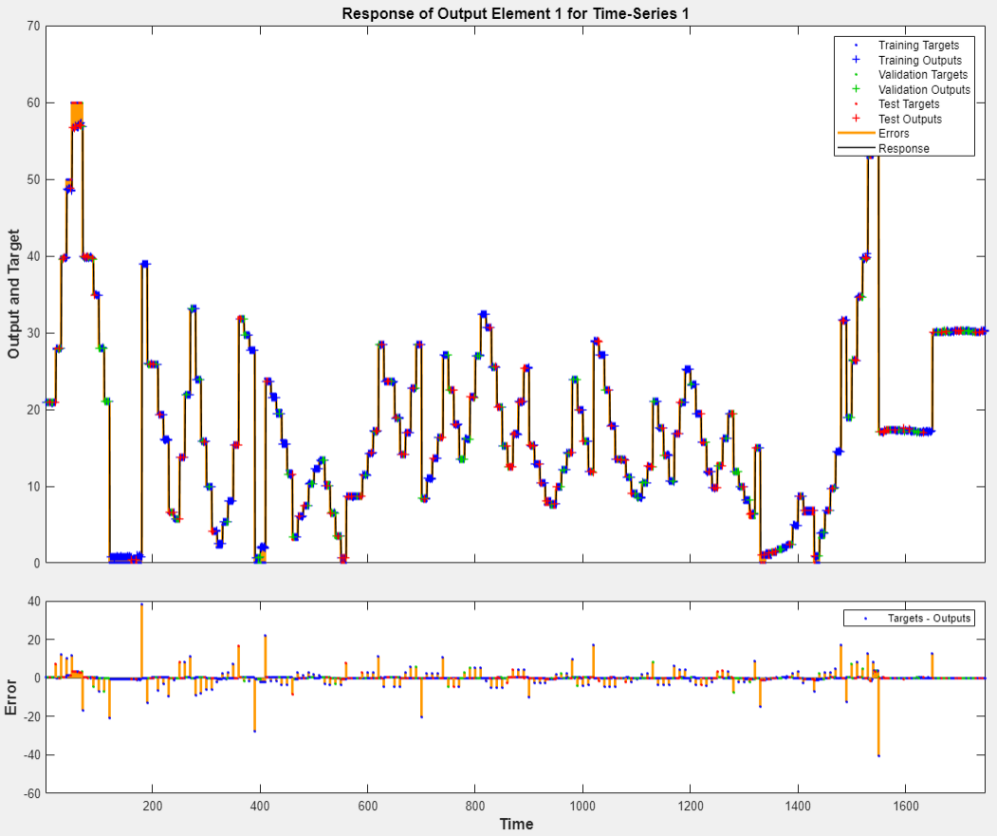


Figure 61: Time series response of the NARX oil flow model using 7 inputs and linear scaling normalization

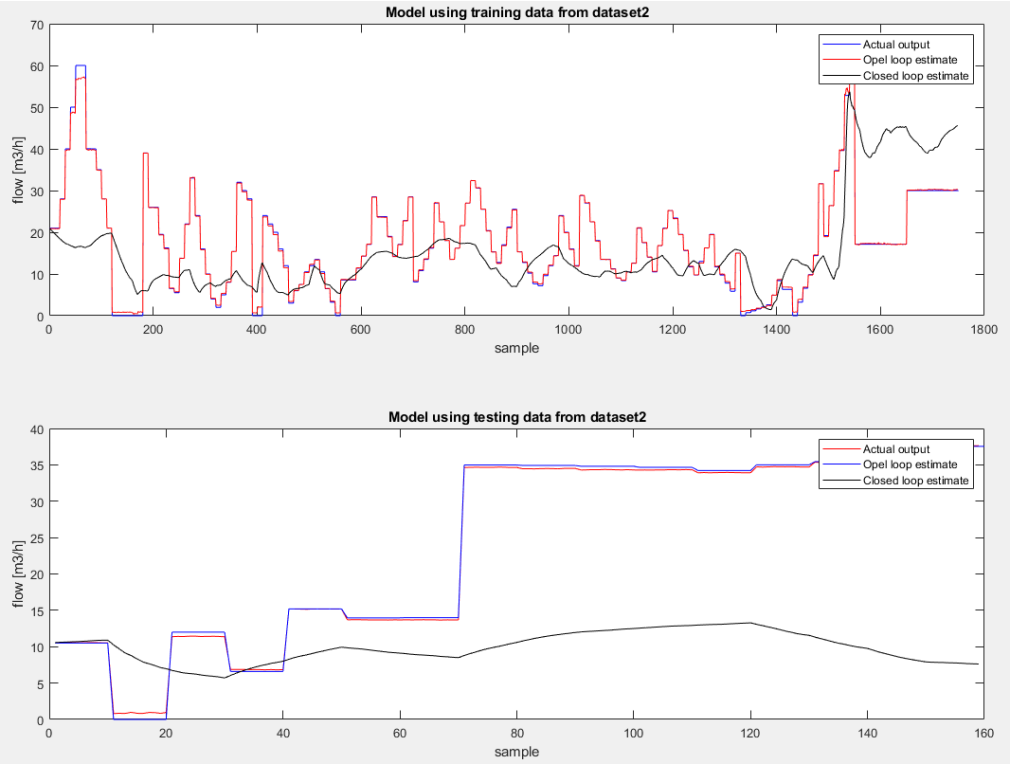


Figure 62: Testing the NARX oil flow model with 7 inputs and linear scaling normalization in both open and closed loop

8.1.3.2 NARX oil flow model using 7 inputs and Z-score normalization

Training of the NARX network finished after 23 epochs. Training stopped when validation criterion was met after 6 checks. When training finished the gradient = 1.5305 and Mu = 0.01. The regression line and mean square error can be shown in Table 11. The training results are shown in Figure 63, while the results from testing the model on the testing matrix are shown in Figure 64.

Table 11: Training results of the NARX oil flow model using 7 inputs and Z-score

	Observations	**MSE**	**R**
Training	1399	6.8774	0.9756
Validation	175	2.5251	0.9922
Test	175	4.4739	0.9823

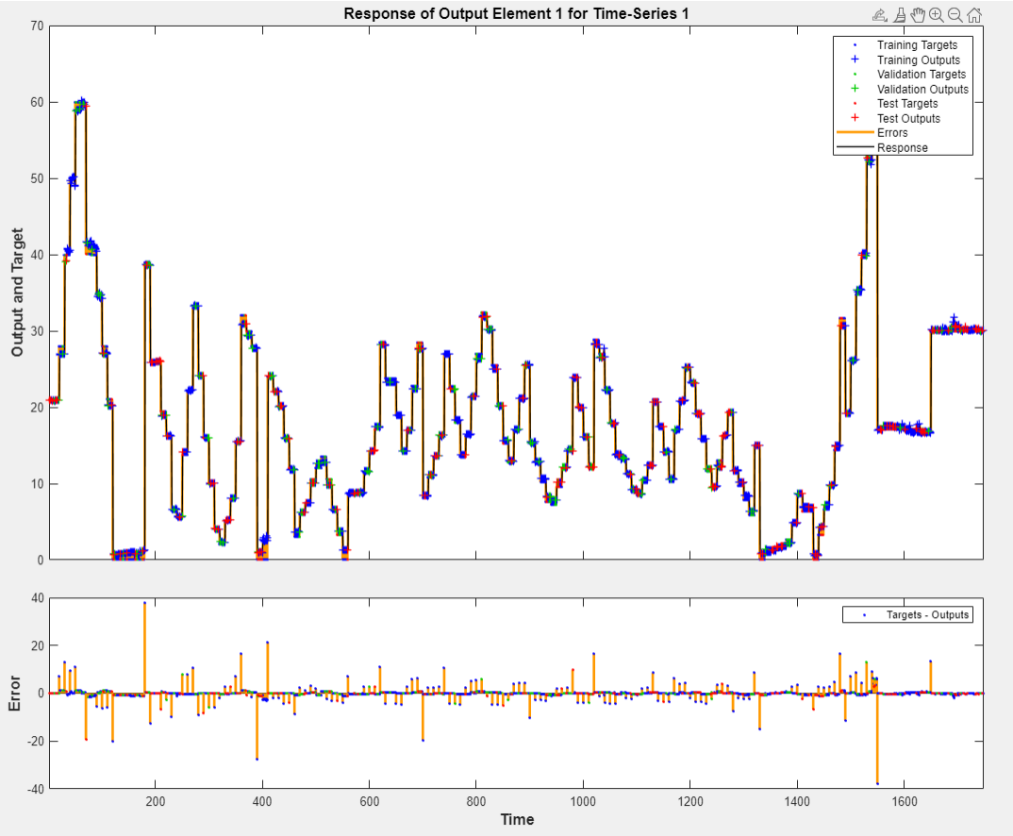


Figure 63: Time series response of the NARX oil flow model using 7 inputs and Z-score

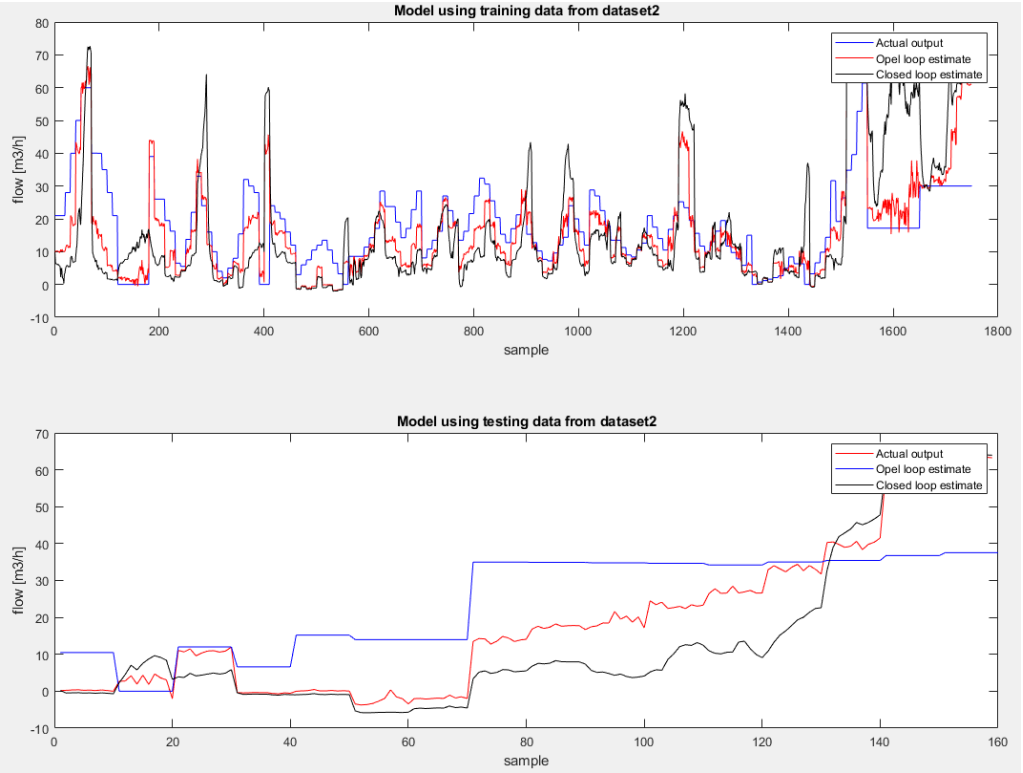


Figure 64: Testing the NARX oil flow model with 7 inputs and Z-score normalization in both open and closed loop

8.1.4 NARX total flow rate model

It is desired to estimate the total flow. This chapter shows the results of the total flow models using a NARX neural network.

8.1.4.1 NARX total flow model using 7 inputs and linear scaling normalization

Training of the NARX network finished after 13 epochs. Training stopped when validation criterion was met after 6 checks. When training finished the gradient = 4.0786 and Mu = 0.1. The regression line and mean square error are shown in Table 12. The training results are shown in Figure 65, while the results from testing the model on the testing matrix are shown in Figure 66.

Table 12: Training results of the NARX total flow model using 7 inputs and linear scaling

	Observations	**MSE**	**R**
Training	1399	16.2172	0.9837
Validation	175	7.9689	0.9926
Test	175	9.5169	0.9908

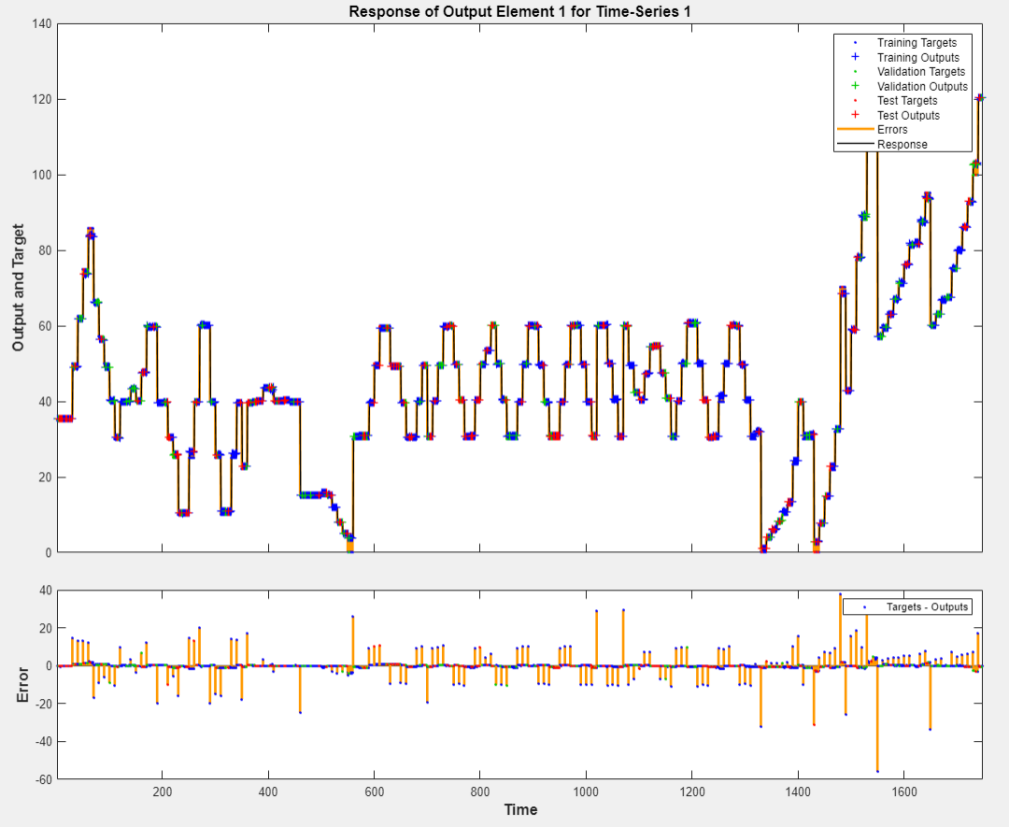


Figure 65: Time series response of the NARX total flow model using 7 inputs and linear scaling

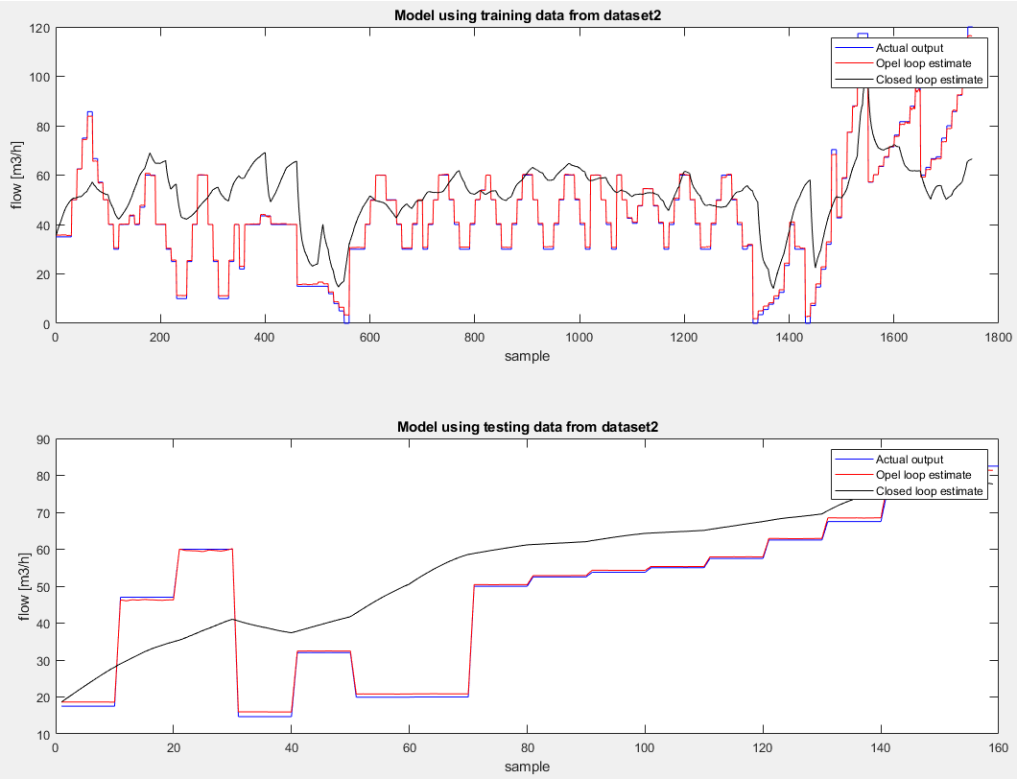


Figure 66: Testing the NARX total flow model with 7 inputs and linear scaling normalization in both open and closed loop

8.1.4.2 NARX total flow model with 7 inputs and a Z-score normalization

Training of the NARX network finished after 15 epochs. Training stopped when validation criterion was met after 6 checks. When the training finished the gradient = 25.4948 and Mu = 0.01. The regression line and mean square error can be shown in Table 13. The training results are shown in Figure 67, while the results from testing the model on the testing matrix are shown in Figure 68.

Table 13: Training results of NARX total flow model using 7 inputs and Z-score

	Observations	**MSE**	**R**
Training	1399	16.4713	0.9838
Validation	175	5.8976	0.9939
Test	175	8.9316	0.9915

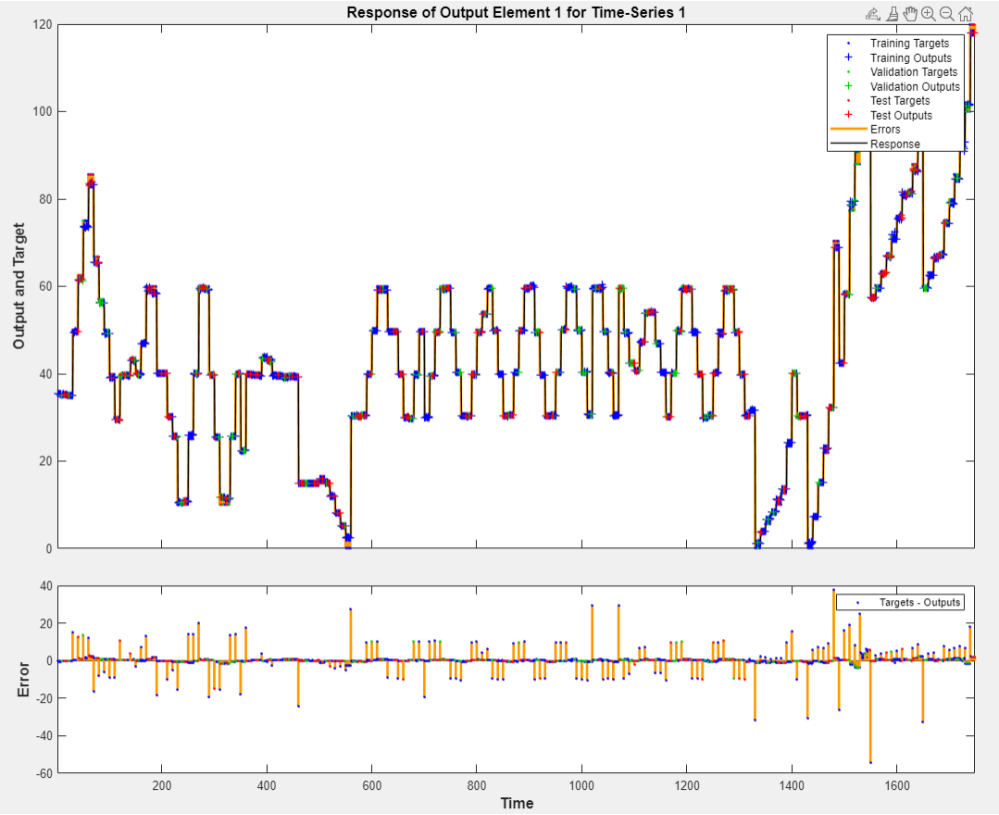


Figure 67: Time series response of the NARX total flow model using 7 inputs and Z-score.

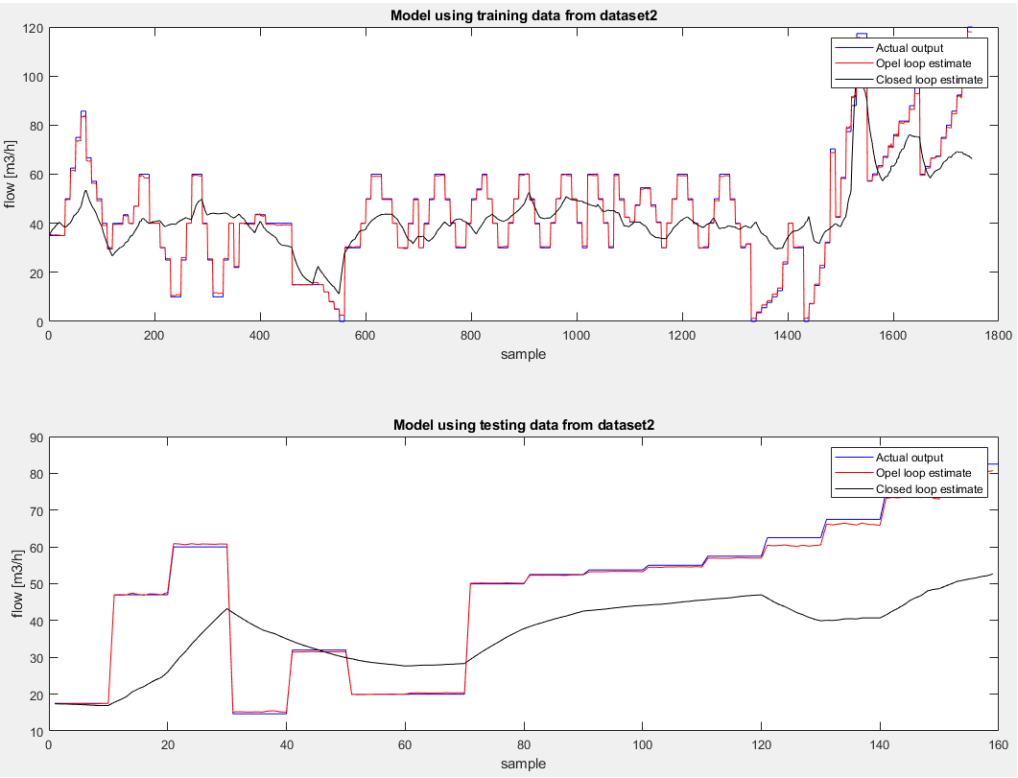


Figure 68: Testing the NARX total flow model with 7 inputs and Z-score normalization in both open and closed loop.

8.2 Nonlinear input-output neural network for multiphase flow estimation

The nonlinear input-output network is like the NARX network. In the NARX network the output $y(t)$ was predicted based on previous input and output values. The nonlinear input-output neural network only uses the previous values of the input $x(t)$ to predict the output $y(t)$ as shown in Eq. 8.2 (MathWorks, Shallow Neural Network Time-Series Prediction and Modeling, 2022).

$$y(t) = f\big(x(t-1), x(t-2), \dots, x(t-d)\big) \qquad \textit{(Eq.8.2)}$$

Figure 69Figure 48 shows a nonlinear input-output neural network. The network in Figure 69Figure 49 have 8 inputs in the input layer, 3 neurons in the hidden layer and one output in the output later. The activation function in the hidden layer is transoid function, while the output layer is linear function. The time delay is set to 1.

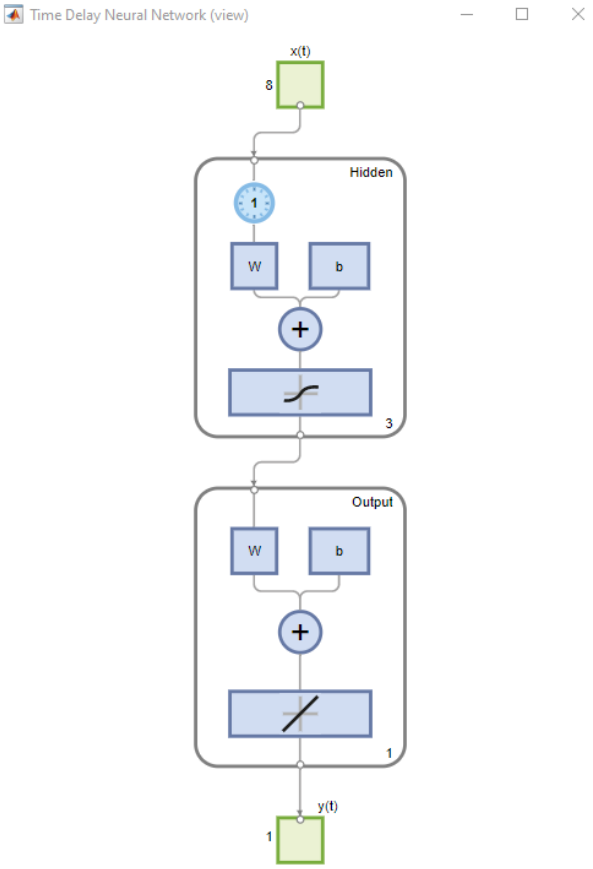


Figure 69: Nonlinear input-output neural network with one hidden layer and one output layer. This network is configured with a time delay of 1, 3 neurons in the hidden layer and 1 output in the output layer.

The NARX neural network models that was trained on data that had been normalized with the Z-score normalization technique in Ch.8.1 gave the best results. After some tests it was verified that this is also true for the nonlinear input-output model. Therefore, all models in this chapter are trained on data that have been normalized with Z-score normalization technique.

Before the network can be simulated the data needs to be prepared. The inputs and output time series are shifted as many time steps as needed to get enough initial input delay states, X_i, and layer delay states, X_i, as needed. The shifted time series are called X_s. The code below shows the functions that are used to prepare the data and simulate the network. The output data is non feedback targets and are therefore inserted into the third input in the preparets() function, while the fourth input in the preparets() function was used for the NARX networks since the output data is in that case the feedback targets. The function is in appendix B.

```
function y = SimulateInputOutputNetwork(network, inputData, outputData)
    [Xs,Xi, Ai] = preparets(network, inputData, outputData);
    y = sim(network, Xs, Xi, Ai);
end
```

8.2.1 Nonlinear input-output models using 7 or 8 variables as input

This chapter shows the results of the flow rate models using the selected variables from the correlation plots in Ch.6.5.

8.2.1.1 Nonlinear input-output gas flow rate model using 8 inputs

Testing with a nonlinear input-output network with 3 neurons in the hidden layer and 1 in time delay. Training of the network finished after 19 epochs. Training stopped when validation criterion was met after 6 checks. When training finished the gradient = 1.4205 and Mu = 0.01. The regression line and mean square error are shown in Table 14. The training and testing results are shown in Figure 70.

Table 14: Training results of nonlinear input-output neural network for gas flow model using 8 inputs and Z-score normalization technique.

	Observations	**MSE**	**R**
Training	1225	7.8461	0.9468
Validation	262	6.1991	0.9748
Test	262	6.1280	0.9604

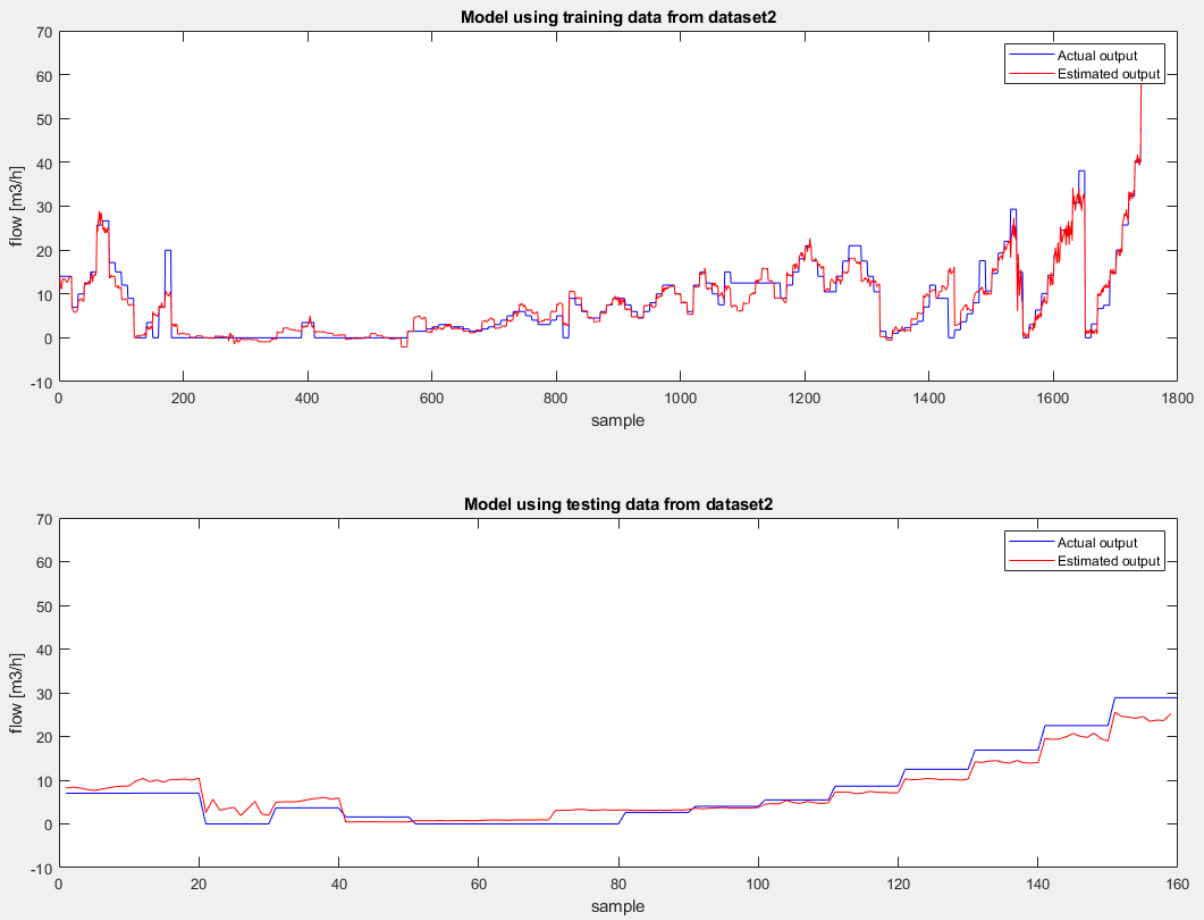


Figure 70: Testing of nonlinear input-output neural network of gas flow model using 8 inputs and Z-score normalization.

8.2.1.2 Nonlinear input-output model of water flow rate using 7 inputs

Testing with a nonlinear input-output network with 2 neurons in the hidden layer and 1 in time delay. Training of the network finished after 65 epochs. Training stopped when validation criterion was met after 6 checks. When training finished the gradient = 0.0477 and Mu = 0.01. The regression line and mean square error are shown in Table 15. The training and testing results are shown in Figure 71.

Table 15: Training results of nonlinear input-output neural network for water flow model using 7 inputs and Z-score normalization technique.

	Observations	**MSE**	**R**
Training	1399	38.5147	0.8577
Validation	175	35.1920	0.8767
Test	175	40.2919	0.8360

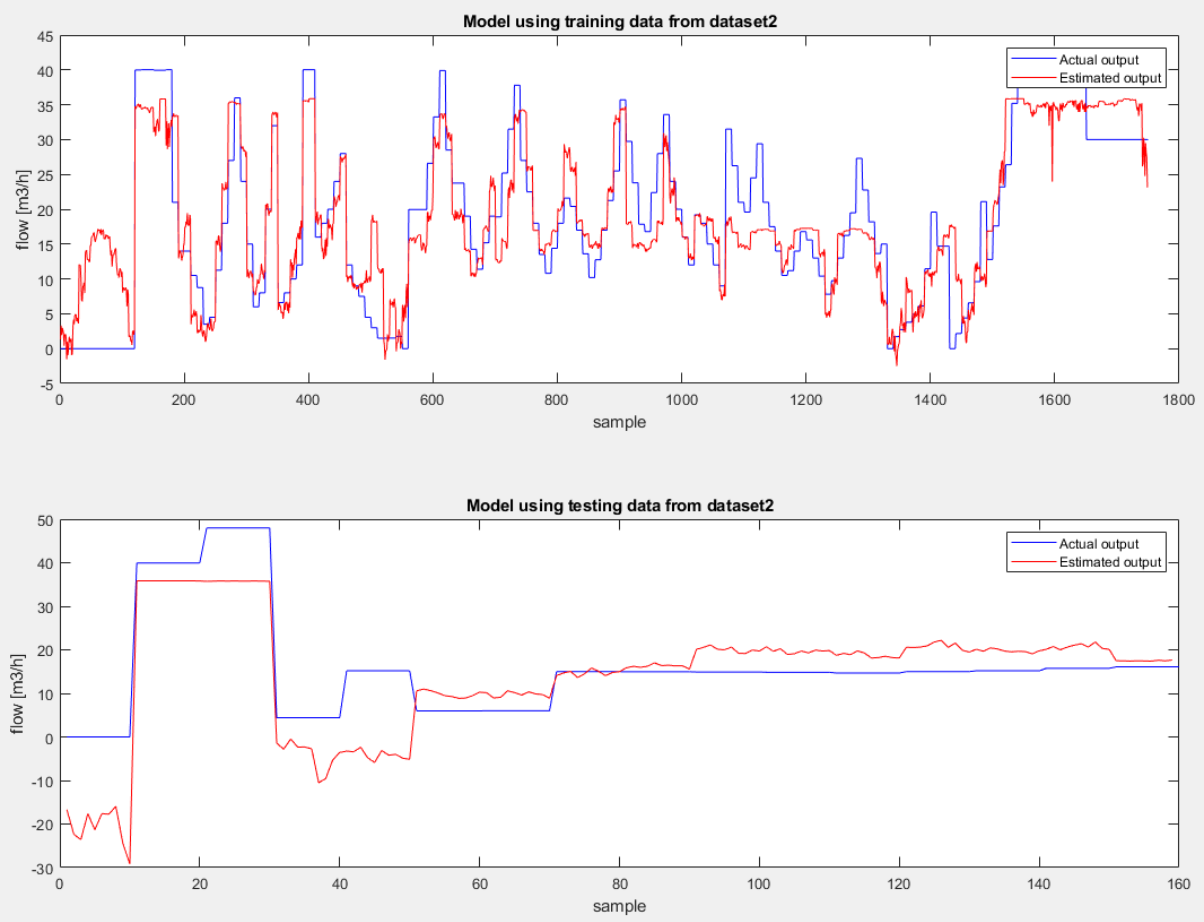


Figure 71: Testing of nonlinear input-output neural network of water flow model using 7 inputs and Z-score normalization.

8.2.1.3 Nonlinear input-output model of oil flow rate using 7 inputs

Testing with a nonlinear input-output network with 2 neurons in the hidden layer and 2 in time delay. Training of the network finished after 29 epochs. Training stopped when validation criterion was met after 6 checks. When training finished the gradient = 0.0123 and Mu = 0.01. The regression line and mean square error are shown in Table 16. The training and testing results are shown in Figure 72.

Table 16: Training results of nonlinear input-output neural network for oil flow model using 7 inputs and Z-score normalization technique

	Observations	**MSE**	**R**
Training	1399	62.0421	0.7609
Validation	175	50.0568	0.7515
Test	175	66.7118	0.6706

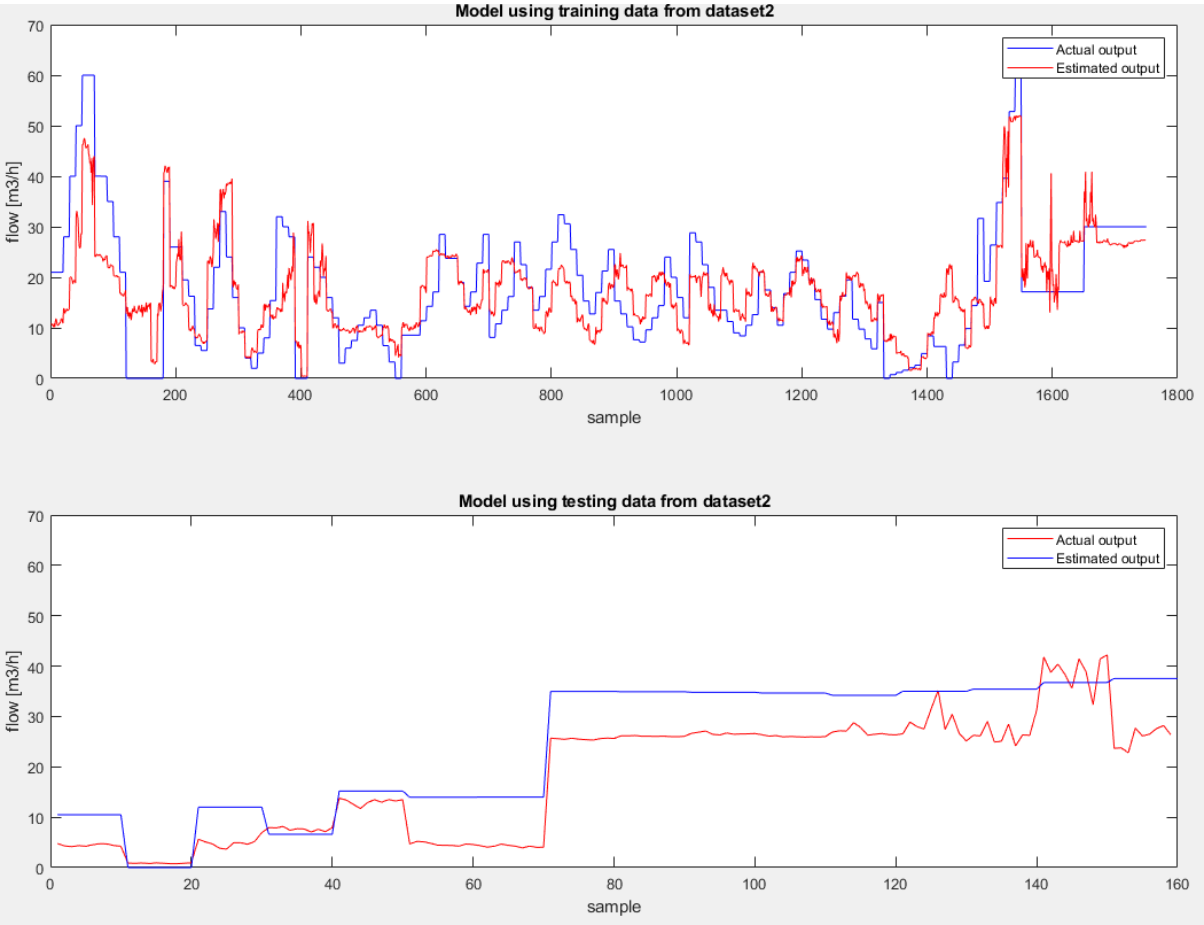


Figure 72: Testing of nonlinear input-output neural network of oil flow model using 7 inputs and Z-score normalization.

8.2.1.4 Nonlinear input-output mode of total flow rate using 7 inputs

Testing with a nonlinear input-output network with 1 neuron in the hidden layer and 1 in time delay. Training of the network finished after 42 epochs. Training stopped when validation criterion was met after 6 checks. When training finished the gradient = 19.9632 and Mu = 0.1. The regression line and mean square error are shown in Table 17. The training and testing results are shown in Figure 73.

Table 17: Training results of nonlinear input-output neural network for total flow model using 7 inputs and Z-score normalization technique

	Observations	**MSE**	**R**
Training	1399	73.9870	0.9247
Validation	175	61.9111	0.9298
Test	175	42.0706	0.9607

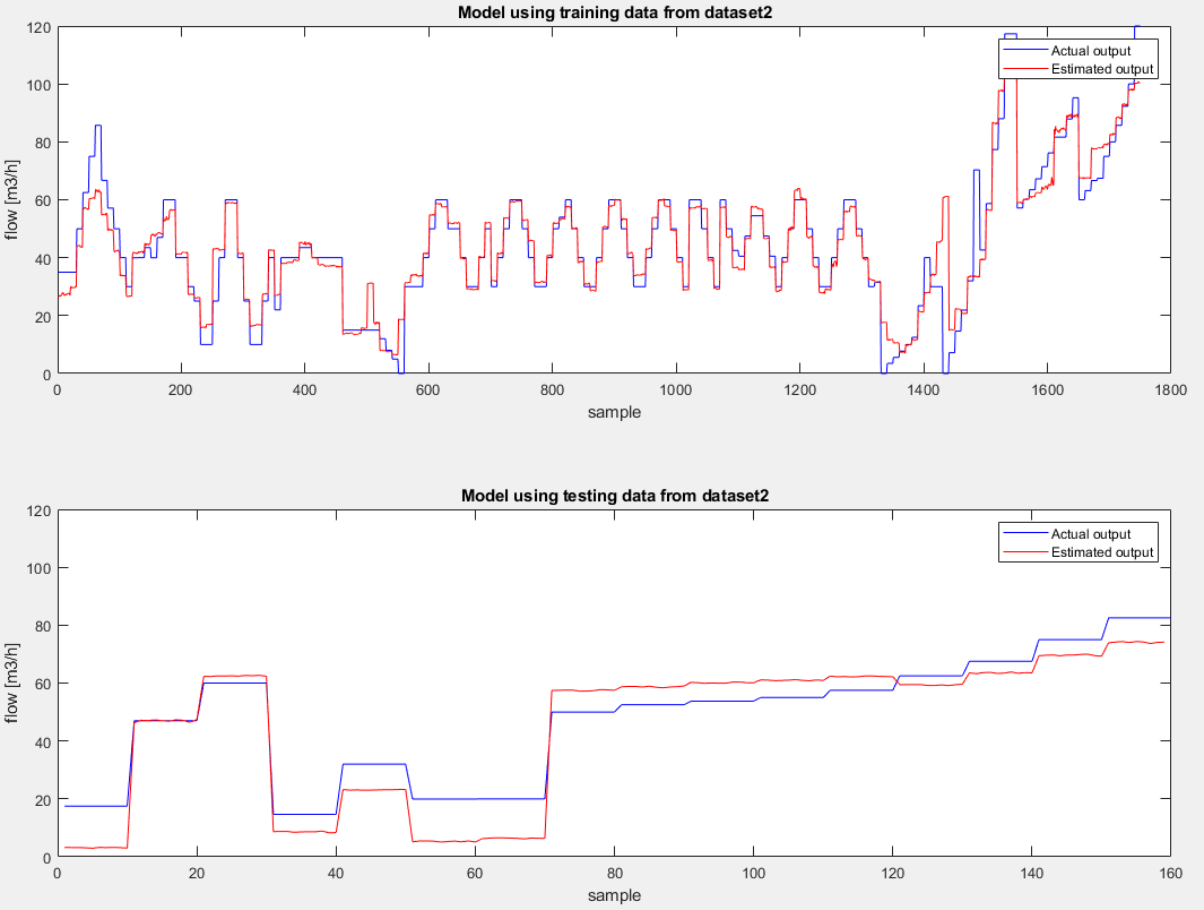


Figure 73: Testing of nonlinear input-output neural network of total flow model using 7 inputs and Z-score normalization.

8.2.2 Nonlinear input-output models using acoustic emission and differential pressure variables as input

It is desired to make a simple cheap flow rate model using only accelerometers and the differential pressure over a Venturi. This chapter shows the results from the neural networks with only accelerometer and Venturi data as inputs.

8.2.2.1 Nonlinear input-output model of gas flow rate using 5 inputs

Testing with a nonlinear input-output network with 2 neurons in the hidden layer and 1 in time delay. Training of the network finished after 25 epochs. Training stopped when validation criterion was met after 6 checks. When training finished the gradient = 7.7355 and Mu = 0.01. The regression line and mean square error are shown in Table 18. The training and testing results are shown in Figure 74 .

Table 18: Training results of nonlinear input-output neural network for gas flow model using 5 inputs and Z-score normalization technique.

	Observations	MSE	R
Training	1399	16.0132	0.8987
Validation	175	13.5451	0.9181
Test	175	12.2629	0.9083

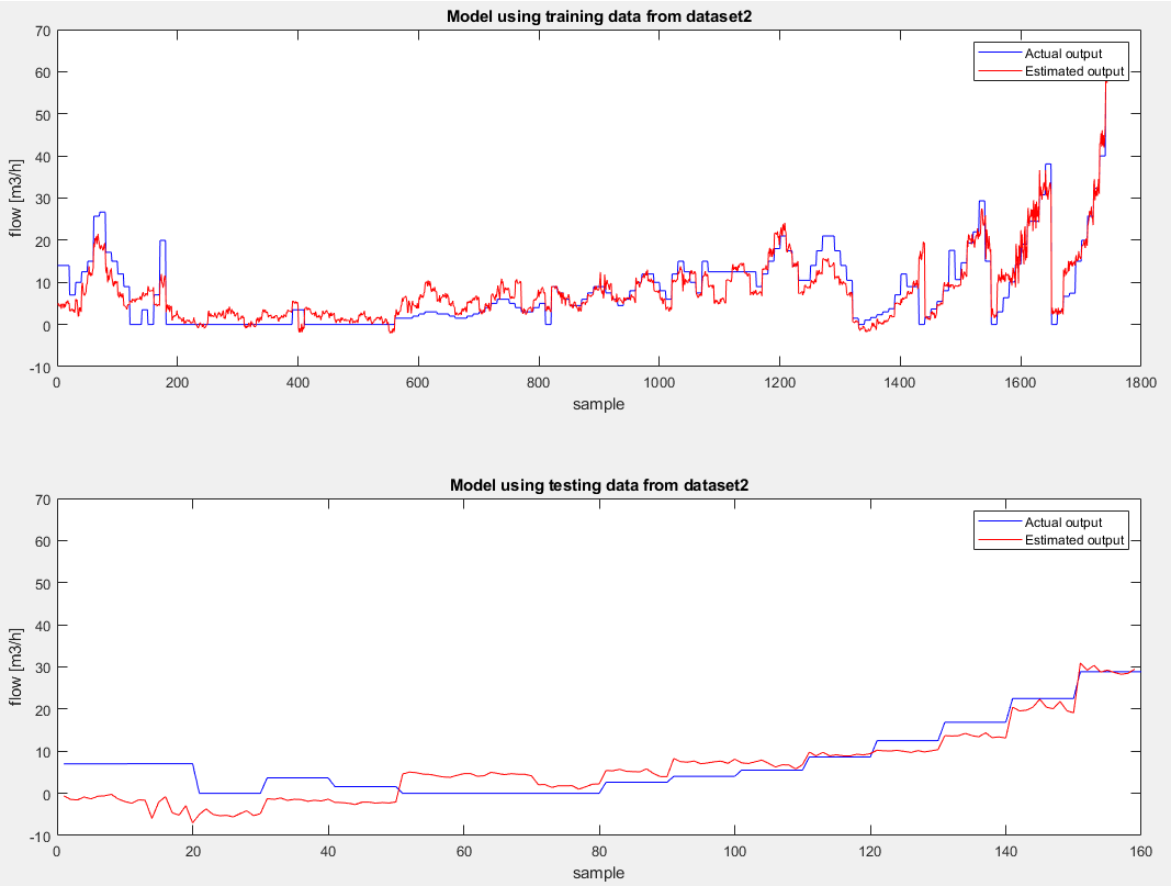


Figure 74: Testing of nonlinear input-output neural network of gas flow model using 5 inputs and Z-score normalization.

8.2.2.2 Nonlinear input-output model of water flow rate using 5 inputs

Testing with a nonlinear input-output network with 2 neurons in the hidden layer and 1 in time delay. Training of the network finished after 24 epochs. Training stopped when validation criterion was met after 6 checks. When training finished the gradient = 0.0267 and Mu = 0.01. The regression line and mean square error are shown in Table 19. The training and testing results are shown in Figure 75.

Table 19: Training results of nonlinear input-output neural network for water flow model using 5 inputs and Z-score normalization technique.

	Observations	**MSE**	**R**
Training	1399	59.2976	0.7673
Validation	175	56.2726	0.7810
Test	175	63.5181	0.7634

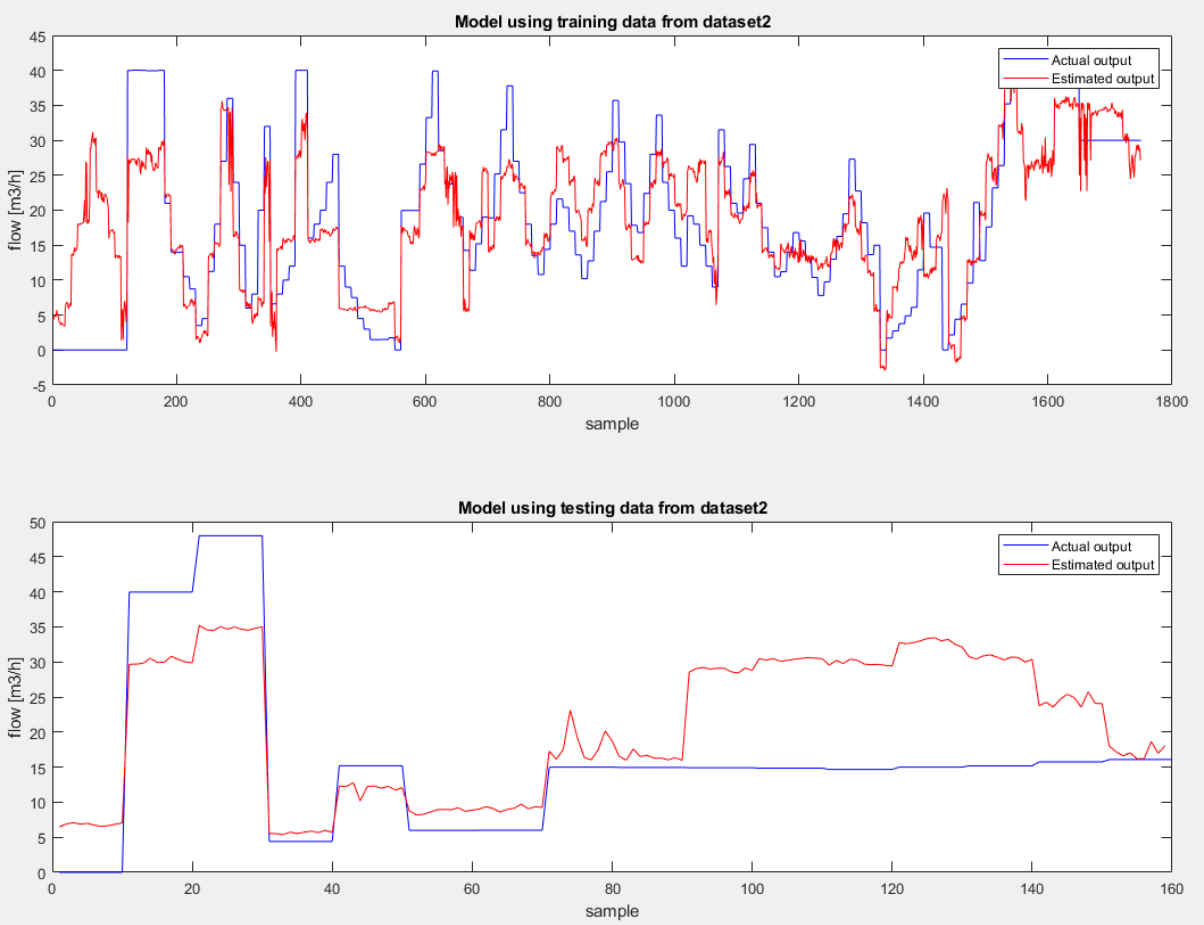


Figure 75: Testing of nonlinear input-output neural network of water flow model using 5 inputs and Z-score normalization.

8.2.2.3 Nonlinear input-output model of oil flow rate using 5 inputs

Testing with a nonlinear input-output network with 2 neurons in the hidden layer and 1 in time delay. Training of the network finished after 14 epochs. Training stopped when validation criterion was met after 6 checks. When training finished the gradient = 0.0253 and Mu = 1.00000000000000e-10. The regression line and mean square error are shown in Table 20. The training and testing results are shown in Figure 76.

Table 20: Training results of nonlinear input-output neural network for oil flow model using 5 inputs and Z-score normalization technique.

	Observations	**MSE**	**R**
Training	1399	67.4215	0.7299
Validation	175	41.8101	0.8121
Test	175	70.9960	0.7072

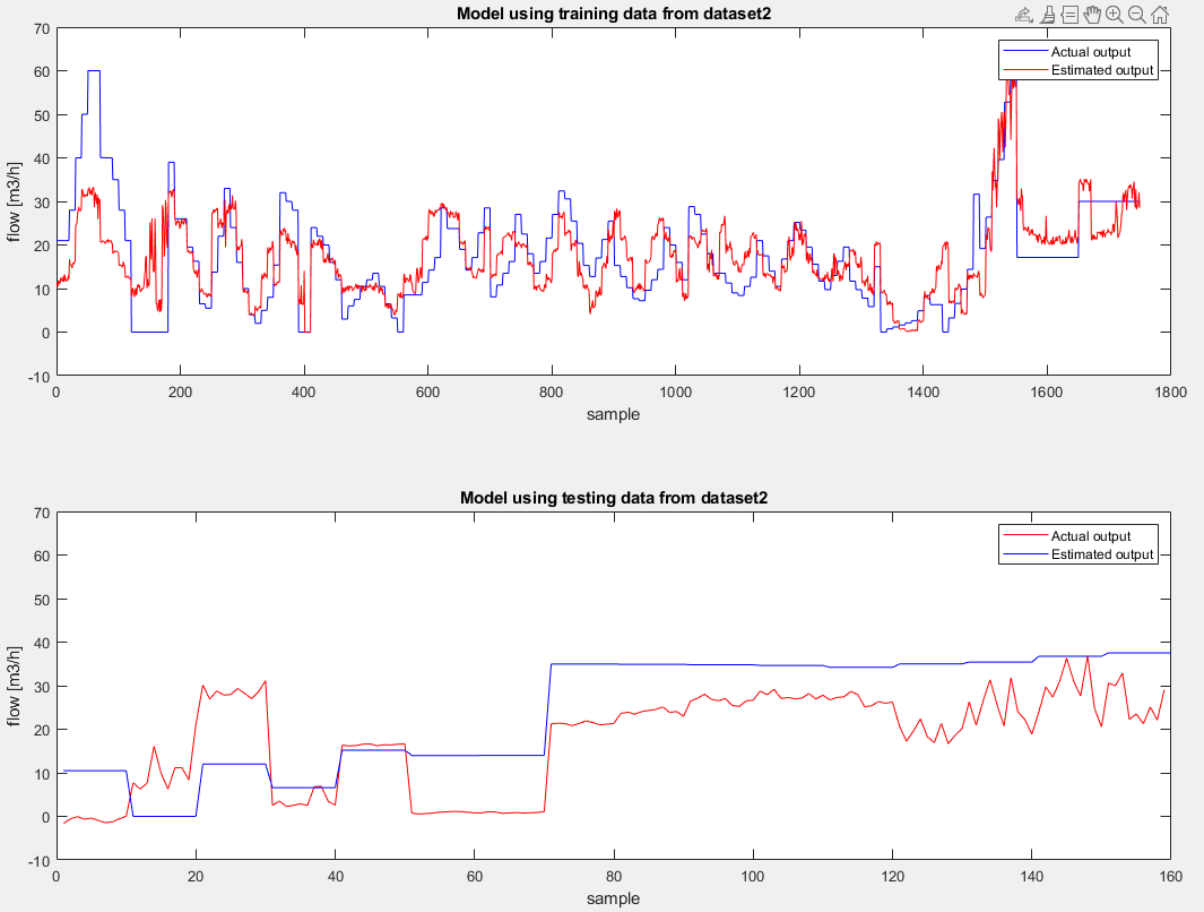


Figure 76: Testing of nonlinear input-output neural network of oil flow model using 5 inputs and Z-score normalization.

8.2.2.4 Nonlinear input-output model of total flow rate using 5 inputs

Testing with a nonlinear input-output network with 2 neurons in the hidden layer and 1 in time delay. Training of the network finished after 104 epochs. Training stopped when validation criterion was met after 6 checks. When training finished the gradient = 0.2210 and Mu = 1.00000000000000e-14. The regression line and mean square error are shown in Table 21. The training and testing results are shown in Figure 77.

Table 21: Training results of nonlinear input-output neural network for total flow model using 5 inputs and Z-score normalization technique.

	Observations	**MSE**	**R**
Training	1399	61.9685	0.9375
Validation	175	92.8446	0.8983
Test	175	61.4880	0.9330

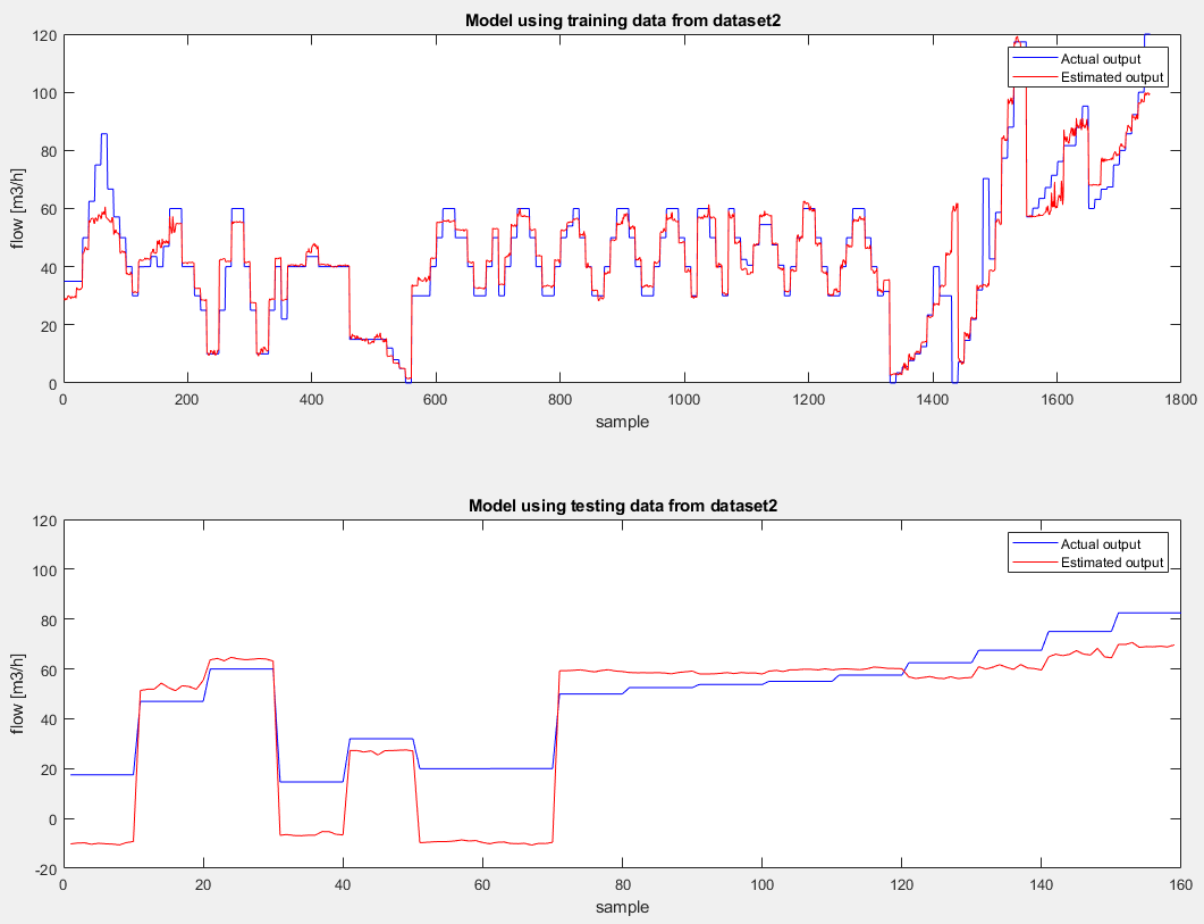


Figure 77: Testing of nonlinear input-output neural network of total flow model using 5 inputs and Z-score normalization.

9 Gas volume fraction estimation

It was tried to make a neural network model for estimating the Gas Volume Fraction, GVF. It gave better results to use the estimated flowrate values from the nonlinear input-output models. The GVF is calculated by dividing the gas flow rate with the total flow rate as shown in Eq.6.2. The equation is the same for the actual and estimated GVF. A pseudocode for the calculation of the actual GVF is shown below. When the total flow is 0 m^3/h the GVF will be undefined and are shown as Nan. If this happens then the GVF value is set to zero. The actual code is in appendix B.

```
GV = Qg./ Qtot;
GV(isnan(GV)) = 0;
```

The GVF can also be estimated by dividing the estimated gas flow rate by the estimated total flow rate. These values were estimated by shallow neural network models in Ch.8. A pseudocode for the calculation of the estimated GVF is shown below. If the GVF is undefined, then it is set to zero. The actual code is in appendix B.

```
GV_simModel = simGasModel{1, 1}' ./ simTotModel{1, 1}';
GV_simModel(isnan(GV_simModel)) = 0;
```

The plots in Figure 78 show the actual GVF in blue color and estimated GVF in red color. The estimated GVF in the upper subplot uses the flow rates that have been estimated by the neural network using 7 inputs for the total flow rate and 8 inputs for the gas flow rate. The lower subplot uses the estimated flow rates from the neural network using only accelerometers and the differential pressure over the Venturi. The upper plot shows a good estimation, while the lower plot also shows good results. The negative values for GVF are not possible and should therefore be set to zero. To improve these results the model for estimation of gas and total flow in Ch.8 need to be improved.

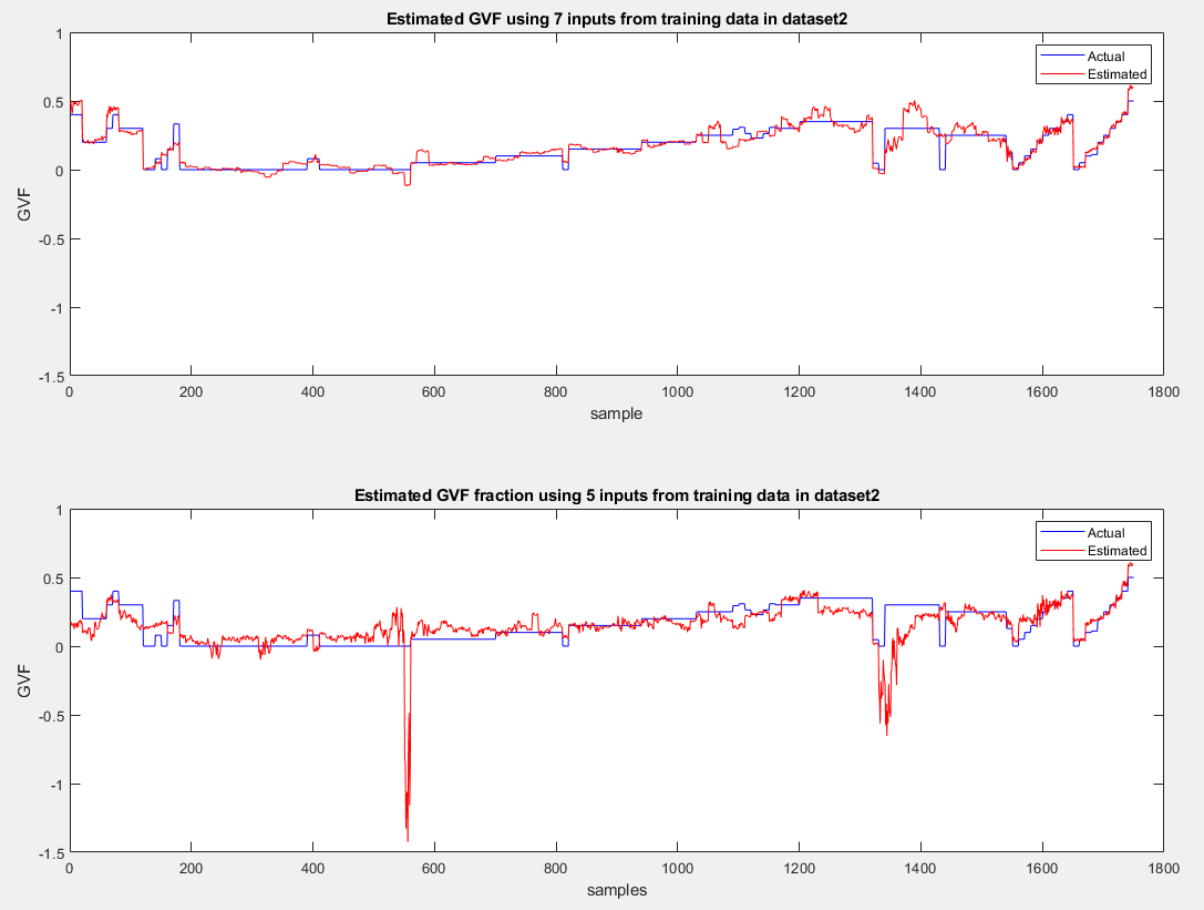


Figure 78: Calculated GVF values based on both actual and estimated flow rates are shown in the plots.

10 Water liquid ratio estimation

It was tried to make a neural network model for estimating the Water Liquid Ratio, WLR. It gave better results to use the estimated flowrate values from the nonlinear input-output models. The WLR is calculated by dividing the water flow rate with the total liquid flow rate as shown in Eq.9.2. The equation is the same for the actual and estimated WLR.

$$GV = \frac{Q_w}{Q_w + Q_o} \qquad \textit{(Eq.9.2)}$$

A pseudocode for the calculation of the actual WLR is shown below. When the liquid flow is 0 m^3/h the WLR will be undefined and are shown as Nan. If the WLR is undefined, then the value is set to zero. The actual code is in appendix B.

```
WLR = Qw./ (Qw + Qo);
WLR(isnan(WLR)) = 0;
```

The WLR can also be estimated by dividing the estimated water flow rate with the estimated liquid flow rate. These flow rate values were estimated by shallow neural network models in Ch.8. A pseudocode for the calculation of the estimated WLR is shown below. If the WLR is undefined, then it is set to zero. The actual code is in appendix B.

```
WLR_simModel= simWatModel{1, 1}' ./ (simWatModel{1, 1}' + simOilModel{1, 1}');
WLR_simModel(isnan(WLR_simModel)) = 0;
```

The plots in Figure 79 show the actual WLR in blue color and estimated WLR in red color. The estimated WLR in the upper subplot uses the flow rates that have been estimated by the neural network using 7 inputs. The lower subplot uses the estimated flow rates from the neural network using only accelerometers and the differential pressure over the Venturi. The plots do not show very good results. This is of course due to the poor estimation of water and oil flow in Ch.8.

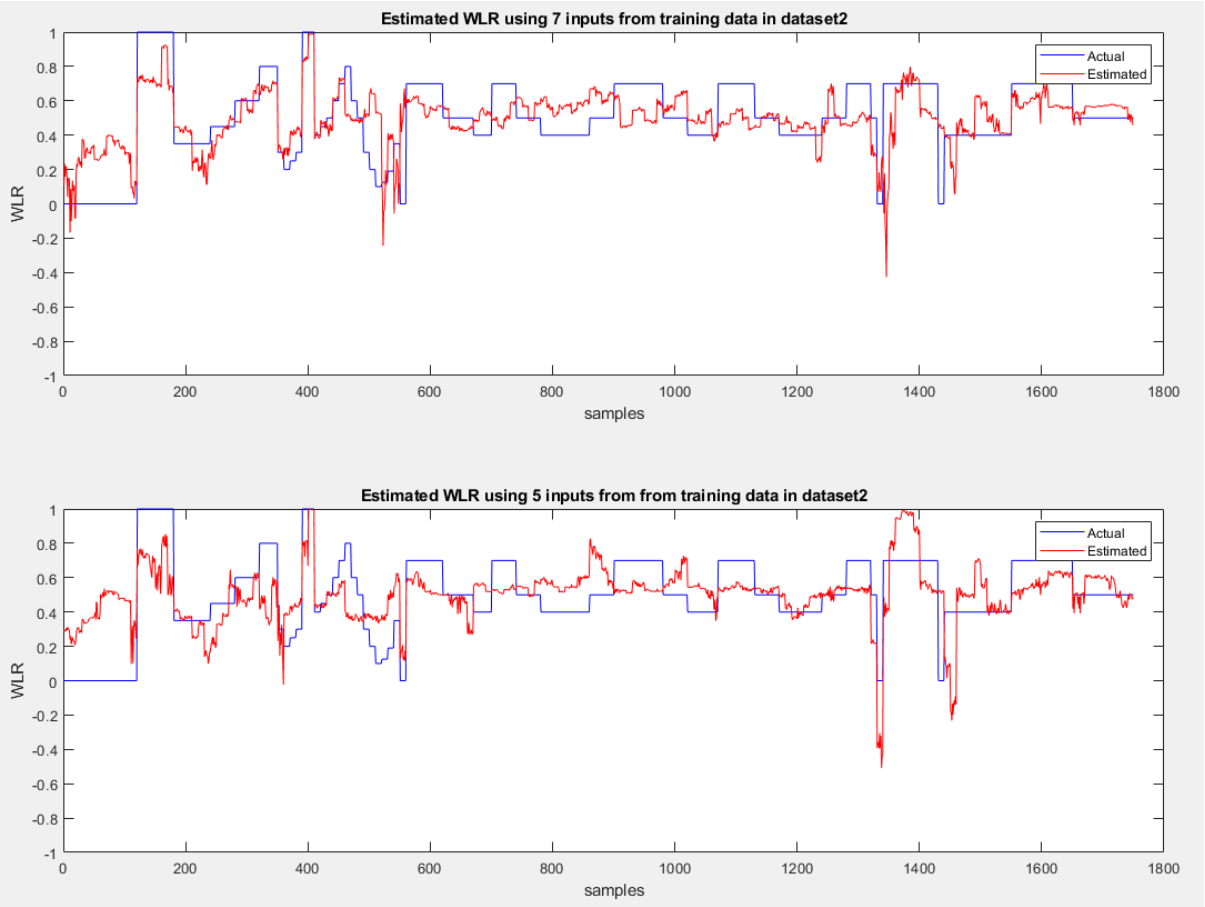


Figure 79: Calculated WLR values based on both actual and estimated flow rates shown in the plots.

11 Results

The most important results will be summarized in this chapter.

11.1 Results of the data analysis

The analysis of the V_{RMS} values showed that there may be a relationship between the total flow rate and the differential pressure over the Wika Venturi, also called Venturi 2. A relationship between the total differential pressure and the measured acoustic emission was also found.

The results from the correlation analysis showed that there is a high degree of correlation between the total flow rate and the differential pressure measurements. The highest correlation was a correlation of 0.91 with the differential pressure 1 over Venturi 2, but the total differential pressure has almost the same result with a correlation of 0.90. A correlation of 0.71 between acoustic emission sensor 4 and the total flow rate was found, but the other acoustic emission sensors also had some correlation with the total flow rate. The gas flow rate seems to correlate with acoustic emission sensor 1, 3, and 4. With a correlation of 0.64 the acoustic emission sensor 4 had the highest correlation with the gas flow. There was also a correlation of 0.82 between the gas flow rate and differential pressure 2 measurement over Venturi 2.

The frequency analysis showed that pure oil flow seemed to give a higher amount of acoustic emission than pure gas flow and that pure water flow seemed to give very little acoustic emission. It seemed like the frequency range between 0 – 11 kHz contained the most important information. Since there was no conclusion, it was decided to keep all frequency components to avoid the risk of filtering away important information.

11.2 Results of the shallow neural network time-series models

The models for gas, oil, water, and total flow were trained with shallow neural networks. Both NARX and Nonlinear Input-Output networks were used for training. The nonlinear input-output model gave the best results when testing on the testing matrix. The results of the nonlinear input-output models using 5 inputs and Z-score normalization can be found in Figure 80. The models in the figure contain 2 neurons in the hidden layer. These 5 inputs are all four acoustic emission sensors and differential pressure 2 over Venturi 2. The GVF and WLR were calculated from the estimated flow velocities and the quality of this calculation is therefore dependent on the accuracy of these estimated values. The negative flow, GVF and WLR values are set equal to zero since negative values are not possible for an oil and gas producing well.

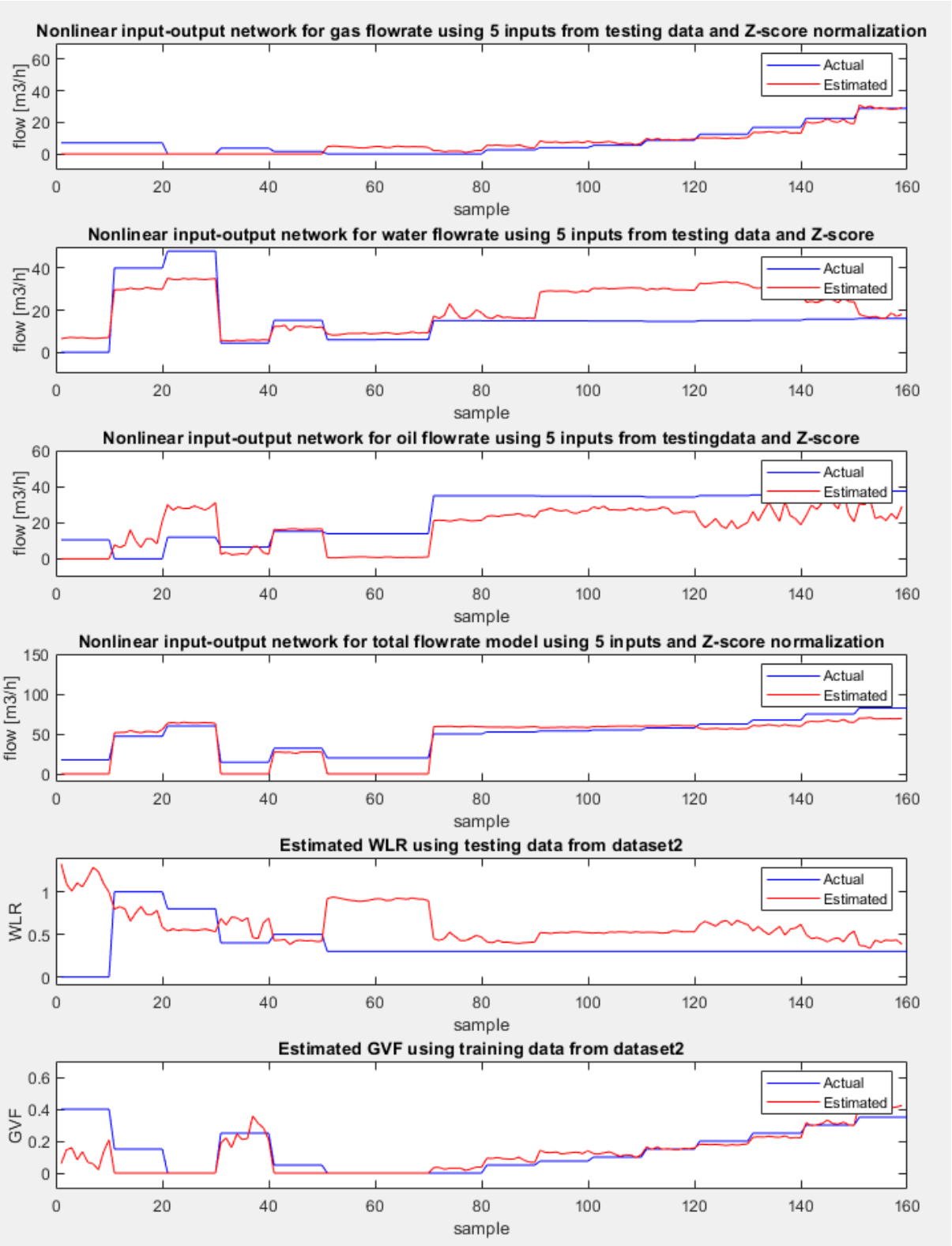


Figure 80: Overview of the performance of the models when using the testing matrix as input. The gas and total flow rate seem to give good results, in addition to GVF. Negative values are set to zero.

12 Discussion

The frequency analysis indicates that pure oil flow gives a higher amount of acoustic emission than pure gas flow and that pure water flow gives almost zero acoustic emission. It is recommended to investigate this finding in future research since there may be a relationship between with the viscosity of the fluid, or some other variable that can be exploited. It seemed like the frequency range between 0 – 11 kHz contained the most important information. Since there was no clear conclusion in the frequency analysis all frequency components were kept avoiding the risk of filtering away important information. This decision may have been a mistake. Further investigation should be done on the frequency analysis to find out if some of the frequency components are noise, but also to research if there are some dominating frequency components in gas, oil, or water flow. If there is a dominating frequency component in, for instance, gas, then the amplitude of that frequency component may correlate with the gas flow rate. In the frequency analysis it was shown that there may be a relationship between frequency components around 1.7 kHz and total and water flow, but this needs to be investigated further.

The models for gas, oil, water, and total flow were trained with shallow neural networks. It is recommended to use a deep neural network in future studies to improve the results. According to MATLAB documentation the NARX model will give more accurate predictions than the nonlinear input-output models since it uses the additional information from previous values of y(t). In this thesis it was observed that the NARX models gave the best training results, but that after closing the loop and simulating the network on the testing matrix the results were not as good as expected. The Nonlinear Input-Output models gave the best results when using the testing matrix. This network does not use the previous values of y(t) as feedback like the NARX network does. This may indicate that there is no relationship between the measured flow rate and the previously measured flow rate. The models using 7-8 inputs gave the best results, but the models with only 5 inputs were used since it gave similar results. The models with 5 inputs are recommended since they are a more cost-effective solution.

The location of the acoustic emission sensors may be vital for the results. It was shown in other studies that the GVF is increased downstream of a flow restriction like a Venturi or choke valve. Note that this is only true in the pipe just after the flow restriction and not for the whole pipe downstream the flow restriction. If this is to be interpreted as something that can be exploited in future studies or if this should be conceived as measurement noise should be investigated. If a sand detector is chosen as the acoustic emission sensor, then the location may not be ideal since it is located to best detect sand particles that collide with the outer curvature of the pipe wall, while gas bubbles seem to flow closer to the inner curvature of a 90-degree bend.

13 Conclusion

Equinor is interested in using their already installed sand detectors to measure acoustic emission from the multiphase flow. This thesis shows that it may be possible to use four acoustic emission sensors in combination with differential pressure measurements over a Venturi to estimate multiphase flow. Shallow nonlinear input-output neural networks were created for gas, oil, water, and total flow. Including more sensors increased the accuracy but since the difference was not significant the more cost-effective solution using only five measurements as input were preferred. In future studies it is recommended to use a deep neural network to improve the results.

There seems to be a direct relationship between the acoustic emission generated by the multiphase flow and the differential pressure, which again correlates with the total flow rate. The RMSE values in Table 22 it shows that the RMSE for oil and water flow rate models are higher than for the gas flow rate model, which makes sense since the results of the oil and water flow rate models were not very accurate and need to be improved. Since the total flow rate is the sum of oil, water, and gas flow rate the RMSE value is higher, but the results are good. The GVF and WLR are calculated from flow rates and are therefore dependent on the accuracy of the flow rate models.

In future studies it is recommended to have a higher focus on frequency analysis to look for dominating frequency components. The most interesting frequency range in this thesis seemed to be 0 – 11 kHz, but since there was no conclusion all frequency components were kept during the whole study. Data processing time has not been considered as important in this thesis. In dataset 1 the flow regime seems to be mainly annular or slug, but this was not investigated in dataset 2.

Table 22: An overview of the RMSE, network algorithm and configuration. The models for oil and water flow have room for improvement, while the gas and total flow models have both good accuracy and low RMSE.

	Network type	Training algorithm	Network configuration	RMSE	Comments
Gas flow model	Nonlinear Input-Output model	Levenberg-Marquardt	1 hidden layer with 2 neurons	4.62	Dataset 2
Oil flow model	Nonlinear Input-Output model	Levenberg-Marquardt	1 hidden layer with 2 neurons	11.40	Dataset 2
Water flow model	Nonlinear Input-Output model	Levenberg-Marquardt	1 hidden layer with 2 neurons	10.87	Dataset 2
Total flow model	Nonlinear Input-Output model	Levenberg-Marquardt	1 hidden layer with 2 neurons	16.65	Dataset 2

14 References

(2022, 10 17). Hentet fra Pepperl+Fuchs: https://www.pepperl-fuchs.com/norway/no/classid_22.htm?view=productdetails&prodid=97249

ABB. (2022, 10 17). *ABB*. Hentet fra https://new.abb.com/products/measurement-products/flow/multiphase-flowmeters/vis-multiphase-flowmeter

ClampOn. (2019). *User manual DSP Particle Monitor TSE.DS21.Sx00.A10.* ClampOn.

ClampOn website. (2022, 10 17). Hentet fra https://www.clampon.com/products/topside/topside-particle-monitor/

Datasheet, C. (2022, 10 17). Hentet fra ClampOn.com: https://www.clampon.com/wp-content/uploads/2019/04/62-320-00138-1.pdf

Efraín Quiroz-Pérez, R. V.-R.-A.-H. (2014). An approach to evaluate Venturi-device effects on gas wells production. *Journal of Petroleum Science and Engineering*.

Emerson. (2018). *Roxar MPFM 2600 MVG Data Sheet.*

Google. (2022, 10 17). *Data Preparation and Feature Engineering for Machine Learning*. Hentet fra https://developers.google.com/machine-learning/data-prep/transform/normalization

Kehtarnavaz, N. (2008). *Digital Signal Processing System Design (Second Edition)*. Hentet fra ScienceDirect: https://www.sciencedirect.com/topics/engineering/short-time-fourier-transform

Kjær, B. &. (2016). *Charge amplifier type 2667 manual.*

Kjær, B. &. (2017). *Acceleration sensor charge type 5704 manual.*

MathWorks. (2022, 10 17). *Design Time Series NARX Feedback Neural Networks*. Hentet fra https://se.mathworks.com/help/deeplearning/ug/design-time-series-narx-feedback-neural-networks.html;jsessionid=3309411cc43bab895eda5ac675e7

MathWorks. (2022, 10 17). *Shallow Neural Network Time-Series Prediction and Modeling*. Hentet fra https://se.mathworks.com/help/deeplearning/gs/neural-network-time-series-prediction-and-modeling.html

Power Spectral Density. (2022, 10 17). Hentet fra Science Direct: https://www.sciencedirect.com/topics/computer-science/power-spectral-density

Rafael Johansen, T. G. (2018). *Long short-term memory neural networks for flow.* ResearchGate.

Shouxu Qiao, W. Z. (2021). *Numerical simulation of single and two-phase flow across 90 degree vertical elbows.* Elsevier.

Simon Pedersen, P. D. (2017). Challenges in slug modeling and control for offshore oil and gas. *Elsevier*.

Xianlin Wang, Y. L. (2021). *Bond strength prediction of concrete-encased steel structures using hybrid machine learning method.* Elsevier.

Yao Yang, L. H. (2019). *Measurement and analysis of flow regimes transition by acoustic and electrostatic signals in vertical pneumatic conveying.* Elsevier.

15 Appendices

Appendix A: Signed task description of this thesis

Appendix B: The main code for dataset 2

Appendix C: Preparing single-phase flow experiment data from dataset 2.

Appendix D: Preparing two-phase flow experiment data from dataset 2.

Appendix E: Preparing the second part of two-phase flow experiment data from dataset 2.

Appendix F: Preparing the third part of two-phase flow experiment data from dataset 2.

Appendix G: The main code data analysis of dataset 1.

Appendix H: Preparing the gas flow experiment data from dataset 1.

Appendix I: Preparing GOW flow experiment data from dataset 1.

Appendix J: Preparing GW flow experiment data from dataset 1.

Appendix K: Preparing GO flow experiment data from dataset 1.

Appendix L: Preparing oil flow experiment data from dataset 1.

Appendix M: Preparing OW flow experiment data from dataset 1.

Appendix N: Preparing water flow experiment data from dataset 1.

Appendix A

Signed task description of this thesis.

FMH606 Master's Thesis

Title: Multiphase flowmetering – estimation of flow velocities of component phases using multimodal sensor suite
USN supervisor: Håkon Viumdal; Saba Mylvaganam

External partner: Kjetil Fjalestad/EQUINOR/, Tonni Franke Johansen/ SINTEF

Task background:
Multiphase flow rig in USN built and modified many times with funding from the industries and Research Council of Norway, has been used in in various CFD studies, testing different flowmeters and predicting ***in real time*** flow regimes in multiphase flows, i.e. water/oil, oil/water, air/oil, air/water for more than three decades. Based on extensive experimental work, USN has developed a set of test matrices with details of different compositions of air, water and oil flows for generating different types of flow regimes for the multiphase flow loop in the process lab of Campus Porsgrunn.

EQUINOR has a multiphase flow rig for similar purposes and has performed various measurements and is planning to perform more measurements.

Along with conventional measurements such as temperature, pressure, flow, and absorption of gamma rays, tomometric measurements using electrical resistance and capacitance tomographic equipment have been also used in these studies. Recently ultrasonic, acoustic emission and acceleration sensors have been used to interrogate multiphase flows. Most of the results are available in systematically organized directories in the cloud.

USN has been collaborating nationally and internationally, in studying multiphase flows using process tomographic techniques. This project aims to build upon these results and has focus on the fusion of data from these different sensor modalities to estimate different parameters of interest in multiphase flow, e.g. composition of the components in the flow (oil, water and gas/air), flowrates of these components, flow regimes, study of bubble and droplets formations and detection of specific events.

Task description:
(1) Brief survey of multiphase flow and multiphase flow metering with focus on the latest developments
(2) Brief survey of the various techniques used in multiphase flow studies based on current technological standards interacting with industrial actors
(3) Develop test matrix for different flow regimes relevant in Equinor Herøya's desired specification.
(4) Analyzing data from experiments done at EQUINOR using the sensor suite available and collecting data from sensors.
(5) Estimating typical flow parameters in addition to different flow velocities in the context of multiphase flow
(6) Analyzing data from the multimodal sensor suite and fusing the data from various sensors, e.g., AE-sensors and/or other available measurements.
(7) Developing new models or extending already existing models in estimating flow related parameters and identification/prediction of flow and flow regimes

(8) Developing algorithms for estimating volume fraction and flow velocities of component phases
(9) Sketch the signal from a typical sand detector to cabinet
(10) Optional: Evaluate the possibility of using already installed sand detectors to estimate multiphase flow
(11) Submitting a Master Thesis following the guidelines of USN with necessary programs and including a well-documented and complete set of all experimental data from the measurements

Student category: (EET, EPE, IIA or PT students)

The task is suitable for online students (not present at the campus): **Reserved for Andreas Lund Rasmussen**

Practical arrangements:
Necessary experimental data will be provide by USN and EQUINOR. This work is closely coupled to an ongoing project SAM (SAM Self Adapting Model-based system for Process Autonomy - SINTEF).

Supervision:
As a general rule, the student is entitled to 15-20 hours of supervision. This includes necessary time for the supervisor to prepare for supervision meetings (reading material to be discussed, etc).

Signatures:

Supervisor (date and signature): 15.02.2022

Student (write clearly in all capitalized letters): ANDREAS LUND RASMUSSEN

Student (date and signature):

16.02.22

Appendix B

The main code for dataset 2.

```
clc;
clear all;

%Load dataset2
G00_2 = load('G00_2');
singlephase_2 = load('singlephase_2');
twophase_2 = load('twophase_2');
twophase2_2 = load('twophase2_2');
twophase3_2 = load('twophase3_2');

%Load models
load('GasFlowModel8InputsLinearNorm');
load('GasFlowModel8InputsZscoreNorm');
load('OilFlowModel7InputsLinearNorm');
load('OilFlowModel7InputsZscoreNorm');
load('WatFlowModel7InputsLinearNorm');
load('WatFlowModel7InputsZscoreNorm');
load('TotFlowModel7InputsLinearNorm');
load('TotFlowModel7InputsZscoreNorm');

load('GasFlowDelayModel5InputsZscoreNorm');
load('GasFlowDelayModel8InputsZscoreNorm');
load('TotFlowDelayModel7InputsZscoreNorm');
load('TotFlowDelayModel5InputsZscoreNorm');
load('WatFlowDelayModel7InputsZscoreNorm');
load('WatFlowDelayModel5InputsZscoreNorm');
load('OilFlowDelayModel7InputsZscoreNorm');
load('OilFlowDelayModel5InputsZscoreNorm');

%Background noise of accelerometers in dataset 2
XI1_noise_2 = rms([singlephase_2.background_noise(:,21);
twophase_2.background_noise(:,21)]);
XI2_noise_2 = rms([singlephase_2.background_noise(:,22);
twophase_2.background_noise(:,22)]);
XI3_noise_2 = rms([singlephase_2.background_noise(:,23);
twophase_2.background_noise(:,23)]);
XI4_noise_2 = rms([singlephase_2.background_noise(:,24);
twophase_2.background_noise(:,24)]);

%Merging into one dataset for training and one for testing
training_2 = [twophase_2.training_GO; twophase_2.training_GW;
twophase_2.training_OW; twophase2_2.dataset_training;
twophase3_2.dataset_training];
testing_2 = [twophase_2.testing_GO; twophase_2.testing_GW; twophase_2.testing_OW;
twophase2_2.dataset_testing; twophase3_2.dataset_testing];

%Soft sensors in dataset 2
dP_Train_2 = training_2(:,6) - training_2(:,20);
dP_Test_2 = testing_2(:,6) - testing_2(:,20);
dT_Train_2 = training_2(:,19) - training_2(:,5);
dT_Test_2 = testing_2(:,19) - testing_2(:,5);
Qtot_Train_2 = training_2(:,25) + training_2(:,26) + training_2(:,27);
```

```
Qtot_Test_2 = testing_2(:,25) + testing_2(:,26) + testing_2(:,27);

%Adjust for background noise on accelerometers and include all relevant inputs
dataTrainAdj_2 = [training_2(:,1:20) (training_2(:,21)-XI1_noise_2)
(training_2(:,22)-XI2_noise_2) (training_2(:,23)-XI3_noise_2) (training_2(:,24)-
XI4_noise_2) training_2(:,25:27) Qtot_Train_2 dP_Train_2 dT_Train_2];
dataTestAdj_2 = [testing_2(:,1:20) (testing_2(:,21)-XI1_noise_2) (testing_2(:,22)-
XI2_noise_2) (testing_2(:,23)-XI3_noise_2) (testing_2(:,24)-XI4_noise_2)
testing_2(:,25:27) Qtot_Test_2 dP_Test_2 dT_Test_2];

%Normalize dataset 2
dataMax_2 = max([dataTrainAdj_2; dataTestAdj_2]);
dataMin_2 = min([dataTrainAdj_2; dataTestAdj_2]);
trainLinear_2 = (dataTrainAdj_2 - dataMin_2)./(dataMax_2 - dataMin_2);
testLinear_2 = (dataTestAdj_2 - dataMin_2)./(dataMax_2 - dataMin_2);
trainZscore_2 = zscore(dataTrainAdj_2);
testZscore_2 = zscore(dataTestAdj_2);

%-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
----

%Gas flow model with inputs from 'Density Krohn3', 'dp 4m straight','dp 2 Venturi
2', accelerometers and delta temperature
Qg_Train8InputsLinear_2 = [trainLinear_2(:,10) trainLinear_2(:,11)
trainLinear_2(:,18) trainLinear_2(:,21) trainLinear_2(:,22) trainLinear_2(:,23)
trainLinear_2(:,24) trainLinear_2(:,30)];
Qg_Test8InputsLinear_2 = [testLinear_2(:,10) testLinear_2(:,11) testLinear_2(:,18)
testLinear_2(:,21) testLinear_2(:,22) testLinear_2(:,23) testLinear_2(:,24)
testLinear_2(:,30)];
Qg_Train8InputsZscore_2 = [trainZscore_2(:,10) trainZscore_2(:,11)
trainZscore_2(:,18) trainZscore_2(:,21) trainZscore_2(:,22) trainZscore_2(:,23)
trainZscore_2(:,24) trainZscore_2(:,30)];
Qg_Test8InputsZscore_2 = [testZscore_2(:,10) testZscore_2(:,11) testZscore_2(:,18)
testZscore_2(:,21) testZscore_2(:,22) testZscore_2(:,23) testZscore_2(:,24)
testZscore_2(:,30)];

%Gas flow model with input from acoustic emission sensors and 'dp 2 Venturi 2'
Qg_Train5InputsZscore_2 = [trainZscore_2(:,18) trainZscore_2(:,21)
trainZscore_2(:,22) trainZscore_2(:,23) trainZscore_2(:,24)];
Qg_Test5InputsZscore_2 = [testZscore_2(:,18) testZscore_2(:,21) testZscore_2(:,22)
testZscore_2(:,23) testZscore_2(:,24)];

%Gas flow model output
Qg_Train_2 = [dataTrainAdj_2(:,25)];
Qg_Test_2 = [dataTestAdj_2(:,25)];

%-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
----

%Water flow estimation model with 'Density Krohn3', 'dp 4m straight', 'dp 1
Venturi 2', accelerometers and delta temperature
Qw_Train7InputsLinear_2 = [trainLinear_2(:,10) trainLinear_2(:,11)
trainLinear_2(:,21) trainLinear_2(:,22) trainLinear_2(:,23) trainLinear_2(:,24)
trainLinear_2(:,30)];
```

```
Qw_Test7InputsLinear_2 = [testLinear_2(:,10) testLinear_2(:,11) testLinear_2(:,21)
testLinear_2(:,22) testLinear_2(:,23) testLinear_2(:,24) testLinear_2(:,30)];
Qw_Train7InputsZscore_2 = [trainZscore_2(:,10) trainZscore_2(:,11)
trainZscore_2(:,21) trainZscore_2(:,22) trainZscore_2(:,23) trainZscore_2(:,24)
trainZscore_2(:,30)];
Qw_Test7InputsZscore_2 = [testZscore_2(:,10) testZscore_2(:,11) testZscore_2(:,21)
testZscore_2(:,22) testZscore_2(:,23) testZscore_2(:,24) testZscore_2(:,30)];

%Water flow model with with input from acoustic emission sensors and 'dp 1 Venturi
2'
Qw_Train5InputsZscore_2 = [trainZscore_2(:,17) trainZscore_2(:,21)
trainZscore_2(:,22) trainZscore_2(:,23) trainZscore_2(:,24)];
Qw_Test5InputsZscore_2 = [testZscore_2(:,17) testZscore_2(:,21) testZscore_2(:,22)
testZscore_2(:,23) testZscore_2(:,24)];

%Water flow model output
Qw_Train_2 = [dataTrainAdj_2(:,26)];
Qw_Test_2 = [dataTestAdj_2(:,26)];

%-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-----

%Oil flow estimation model with 'dp 4m straight', 'dp 1 Venturi 2', accelerometers
and delta temperature
Qo_Train7InputsLinear_2 = [trainLinear_2(:,11) trainLinear_2(:,17)
trainLinear_2(:,21) trainLinear_2(:,22) trainLinear_2(:,23) trainLinear_2(:,24)
trainLinear_2(:,30)];
Qo_Test7InputsLinear_2 = [testLinear_2(:,11) testLinear_2(:,17) testLinear_2(:,21)
testLinear_2(:,22) testLinear_2(:,23) testLinear_2(:,24) testLinear_2(:,30)];
Qo_Train7InputsZscore_2 = [trainZscore_2(:,11) trainZscore_2(:,17)
trainZscore_2(:,21) trainZscore_2(:,22) trainZscore_2(:,23) trainZscore_2(:,24)
trainZscore_2(:,30)];
Qo_Test7InputsZscore_2 = [testZscore_2(:,11) testZscore_2(:,17) testZscore_2(:,21)
testZscore_2(:,22) testZscore_2(:,23) testZscore_2(:,24) testZscore_2(:,30)];

%Oil flow model with with input from acoustic emission sensors and 'dp 1 Venturi
2'
Qo_Train5InputsZscore_2 = [trainZscore_2(:,17) trainZscore_2(:,21)
trainZscore_2(:,22) trainZscore_2(:,23) trainZscore_2(:,24)];
Qo_Test5InputsZscore_2 = [testZscore_2(:,17) testZscore_2(:,21) testZscore_2(:,22)
testZscore_2(:,23) testZscore_2(:,24)];

%Oil flow model output
Qo_Train_2 = [dataTrainAdj_2(:,27)];
Qo_Test_2 = [dataTestAdj_2(:,27)];

%-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-----

%Total flow estimation model with 'dp 4m straight', 'dp 1 Venturi 2',
accelerometers and delta temperature
Qtot_Train7InputsLinear_2 = [trainLinear_2(:,11) trainLinear_2(:,17)
trainLinear_2(:,21) trainLinear_2(:,22) trainLinear_2(:,23) trainLinear_2(:,24)
trainLinear_2(:,30)];
```

```
Qtot_Test7InputsLinear_2 = [testLinear_2(:,11) testLinear_2(:,17)
testLinear_2(:,21) testLinear_2(:,22) testLinear_2(:,23) testLinear_2(:,24)
testLinear_2(:,30)];
Qtot_Train7InputsZscore_2 = [trainZscore_2(:,11) trainZscore_2(:,17)
trainZscore_2(:,21) trainZscore_2(:,22) trainZscore_2(:,23) trainZscore_2(:,24)
trainZscore_2(:,30)];
Qtot_Test7InputsZscore_2 = [testZscore_2(:,11) testZscore_2(:,17)
testZscore_2(:,21) testZscore_2(:,22) testZscore_2(:,23) testZscore_2(:,24)
testZscore_2(:,30)];

%Total flow estimation model with with input from acoustic emission sensors and
'dp 1 Venturi 2'
Qtot_Train5InputsZscore_2 = [trainZscore_2(:,17) trainZscore_2(:,21)
trainZscore_2(:,22) trainZscore_2(:,23) trainZscore_2(:,24)];
Qtot_Test5InputsZscore_2 = [testZscore_2(:,17) testZscore_2(:,21)
testZscore_2(:,22) testZscore_2(:,23) testZscore_2(:,24)];

%Total flow model output
Qtot_Train_2 = dataTrainAdj_2(:,28);
Qtot_Test_2 = dataTestAdj_2(:,28);

%-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-----

%Close loop on NARX neural networks
gasNARXModel8InputsLinearClosed =
closeloop(GasFlowModel8InputsLinearNorm.Network);
gasNARXModel8InputsZscoreClosed =
closeloop(GasFlowModel8InputsZscoreNorm.Network);
oilNARXModel7InputsLinearClosed =
closeloop(OilFlowModel7InputsLinearNorm.Network);
oilNARXModel7InputsZscoreClosed =
closeloop(OilFlowModel7InputsZscoreNorm.Network);
watNARXModel7InputsLinearClosed =
closeloop(WatFlowModel7InputsLinearNorm.Network);
watNARXModel7InputsZscoreClosed =
closeloop(WatFlowModel7InputsZscoreNorm.Network);
totNARXModel7InputsLinearClosed =
closeloop(TotFlowModel7InputsLinearNorm.Network);
totNARXModel7InputsZscoreClosed =
closeloop(TotFlowModel7InputsZscoreNorm.Network);

%Estimate gas flow with a NARX model.
simGasNARXModelTrain8InputsLinearOpen_2 =
seq2con(SimulateNARXNetwork(GasFlowModel8InputsLinearNorm.Network,
con2seq(Qg_Train8InputsLinear_2'), con2seq(Qg_Train_2')));
simGasNARXModelTrain8InputsLinearClosed_2 =
seq2con(SimulateNARXNetwork(gasNARXModel8InputsLinearClosed,
con2seq(Qg_Train8InputsLinear_2'), con2seq(Qg_Train_2')));
simGasNARXModelTest8InputsLinearOpen_2 =
seq2con(SimulateNARXNetwork(GasFlowModel8InputsLinearNorm.Network,
con2seq(Qg_Test8InputsLinear_2'), con2seq(Qg_Test_2')));
simGasNARXModelTest8InputsLinearClosed_2 =
seq2con(SimulateNARXNetwork(gasNARXModel8InputsLinearClosed,
con2seq(Qg_Test8InputsLinear_2'), con2seq(Qg_Test_2')));
```

```
simGasNARXModelTrain8InputsZscoreOpen_2 =
seq2con(SimulateNARXNetwork(GasFlowModel8InputsZscoreNorm.Network,
con2seq(Qg_Train8InputsZscore_2'), con2seq(Qg_Train_2')));
simGasNARXModelTrain8InputsZscoreClosed_2 =
seq2con(SimulateNARXNetwork(gasNARXModel8InputsZscoreClosed,
con2seq(Qg_Train8InputsZscore_2'), con2seq(Qg_Train_2')));
simGasNARXModelTest8InputsZscoreOpen_2 =
seq2con(SimulateNARXNetwork(GasFlowModel8InputsZscoreNorm.Network,
con2seq(Qg_Test8InputsZscore_2'), con2seq(Qg_Test_2')));
simGasNARXModelTest8InputsZscoreClosed_2 =
seq2con(SimulateNARXNetwork(gasNARXModel8InputsZscoreClosed,
con2seq(Qg_Test8InputsZscore_2'), con2seq(Qg_Test_2')));

%Estimate gas flow with a delay model
simGasInputOutputModelTrain8InputsZscore_2 =
seq2con(SimulateInputOutputNetwork(GasFlowDelayModel8InputsZscoreNorm.Network,
con2seq(Qg_Train8InputsZscore_2'), con2seq(Qg_Train_2')));
simGasInputOutputModelTest8InputsZscore_2 =
seq2con(SimulateInputOutputNetwork(GasFlowDelayModel8InputsZscoreNorm.Network,
con2seq(Qg_Test8InputsZscore_2'), con2seq(Qg_Test_2')));
simGasInputOutputModelTrain5InputsZscore_2 =
seq2con(SimulateInputOutputNetwork(GasFlowDelayModel5InputsZscoreNorm.Network,
con2seq(Qg_Train5InputsZscore_2'), con2seq(Qg_Train_2')));
simGasInputOutputModelTest5InputsZscore_2 =
seq2con(SimulateInputOutputNetwork(GasFlowDelayModel5InputsZscoreNorm.Network,
con2seq(Qg_Test5InputsZscore_2'), con2seq(Qg_Test_2')));

%Estimate oil flow with a NARX model
simOilNARXModelTrain7InputsLinearOpen_2 =
seq2con(SimulateNARXNetwork(OilFlowModel7InputsLinearNorm.Network,
con2seq(Qo_Train7InputsLinear_2'), con2seq(Qo_Train_2')));
simOilNARXModelTrain7InputsLinearClosed_2 =
seq2con(SimulateNARXNetwork(oilNARXModel7InputsLinearClosed,
con2seq(Qo_Train7InputsLinear_2'), con2seq(Qo_Train_2')));
simOilNARXModelTest7InputsLinearOpen_2 =
seq2con(SimulateNARXNetwork(OilFlowModel7InputsLinearNorm.Network,
con2seq(Qo_Test7InputsLinear_2'), con2seq(Qo_Test_2')));
simOilNARXModelTest7InputsLinearClosed_2 =
seq2con(SimulateNARXNetwork(oilNARXModel7InputsLinearClosed,
con2seq(Qo_Test7InputsLinear_2'), con2seq(Qo_Test_2')));
simOilNARXModelTrain7InputsZscoreOpen_2 =
seq2con(SimulateNARXNetwork(OilFlowModel7InputsZscoreNorm.Network,
con2seq(Qo_Train7InputsZscore_2'), con2seq(Qo_Train_2')));
simOilNARXModelTrain7InputsZscoreClosed_2 =
seq2con(SimulateNARXNetwork(oilNARXModel7InputsZscoreClosed,
con2seq(Qo_Train7InputsZscore_2'), con2seq(Qo_Train_2')));
simOilNARXModelTest7InputsZscoreOpen_2 =
seq2con(SimulateNARXNetwork(OilFlowModel7InputsZscoreNorm.Network,
con2seq(Qo_Test7InputsZscore_2'), con2seq(Qo_Test_2')));
simOilNARXModelTest7InputsZscoreClosed_2 =
seq2con(SimulateNARXNetwork(oilNARXModel7InputsZscoreClosed,
con2seq(Qo_Test7InputsZscore_2'), con2seq(Qo_Test_2')));

%Estimate oil flow with a delay model
simOilInputOutputModelTrain7InputsZscore_2 =
seq2con(SimulateInputOutputNetwork(OilFlowDelayModel7InputsZscoreNorm.Network,
con2seq(Qo_Train7InputsZscore_2'), con2seq(Qo_Train_2')));
```

```
simOilInputOutputModelTest7InputsZscore_2 =
seq2con(SimulateInputOutputNetwork(OilFlowDelayModel7InputsZscoreNorm.Network,
con2seq(Qo_Test7InputsZscore_2'), con2seq(Qo_Test_2')));
simOilInputOutputModelTrain5InputsZscore_2 =
seq2con(SimulateInputOutputNetwork(OilFlowDelayModel5InputsZscoreNorm.Network,
con2seq(Qo_Train5InputsZscore_2'), con2seq(Qo_Train_2')));
simOilInputOutputModelTest5InputsZscore_2 =
seq2con(SimulateInputOutputNetwork(OilFlowDelayModel5InputsZscoreNorm.Network,
con2seq(Qo_Test5InputsZscore_2'), con2seq(Qo_Test_2')));

%Estimate wat flow with a NARX model
simWatNARXModelTrain7InputsLinearOpen_2 =
seq2con(SimulateNARXNetwork(WatFlowModel7InputsLinearNorm.Network,
con2seq(Qw_Train7InputsLinear_2'), con2seq(Qw_Train_2')));
simWatNARXModelTrain7InputsLinearClosed_2 =
seq2con(SimulateNARXNetwork(watNARXModel7InputsLinearClosed,
con2seq(Qw_Train7InputsLinear_2'), con2seq(Qw_Train_2')));
simWatNARXModelTest7InputsLinearOpen_2 =
seq2con(SimulateNARXNetwork(WatFlowModel7InputsLinearNorm.Network,
con2seq(Qw_Test7InputsLinear_2'), con2seq(Qw_Test_2')));
simWatNARXModelTest7InputsLinearClosed_2 =
seq2con(SimulateNARXNetwork(watNARXModel7InputsLinearClosed,
con2seq(Qw_Test7InputsLinear_2'), con2seq(Qw_Test_2')));
simWatNARXModelTrain7InputsZscoreOpen_2 =
seq2con(SimulateNARXNetwork(WatFlowModel7InputsZscoreNorm.Network,
con2seq(Qw_Train7InputsZscore_2'), con2seq(Qw_Train_2')));
simWatNARXModelTrain7InputsZscoreClosed_2 =
seq2con(SimulateNARXNetwork(watNARXModel7InputsZscoreClosed,
con2seq(Qw_Train7InputsZscore_2'), con2seq(Qw_Train_2')));
simWatNARXModelTest7InputsZscoreOpen_2 =
seq2con(SimulateNARXNetwork(WatFlowModel7InputsZscoreNorm.Network,
con2seq(Qw_Test7InputsZscore_2'), con2seq(Qw_Test_2')));
simWatNARXModelTest7InputsZscoreClosed_2 =
seq2con(SimulateNARXNetwork(watNARXModel7InputsZscoreClosed,
con2seq(Qw_Test7InputsZscore_2'), con2seq(Qw_Test_2')));

%Estimate wat flow with a delay model
simWatInputOutputModelTrain7InputsZscore_2 =
seq2con(SimulateInputOutputNetwork(WatFlowDelayModel7InputsZscoreNorm.Network,
con2seq(Qw_Train7InputsZscore_2'), con2seq(Qw_Train_2')));
simWatInputOutputModelTest7InputsZscore_2 =
seq2con(SimulateInputOutputNetwork(WatFlowDelayModel7InputsZscoreNorm.Network,
con2seq(Qw_Test7InputsZscore_2'), con2seq(Qw_Test_2')));
simWatInputOutputModelTrain5InputsZscore_2 =
seq2con(SimulateInputOutputNetwork(WatFlowDelayModel5InputsZscoreNorm.Network,
con2seq(Qw_Train5InputsZscore_2'), con2seq(Qw_Train_2')));
simWatInputOutputModelTest5InputsZscore_2 =
seq2con(SimulateInputOutputNetwork(WatFlowDelayModel5InputsZscoreNorm.Network,
con2seq(Qw_Test5InputsZscore_2'), con2seq(Qw_Test_2')));

%Estimate total flow with a NARX model
simTotNARXModelTrain7InputsLinearOpen_2 =
seq2con(SimulateNARXNetwork(TotFlowModel7InputsLinearNorm.Network,
con2seq(Qtot_Train7InputsLinear_2'), con2seq(Qtot_Train_2')));
simTotNARXModelTrain7InputsLinearClosed_2 =
seq2con(SimulateNARXNetwork(totNARXModel7InputsLinearClosed,
con2seq(Qtot_Train7InputsLinear_2'), con2seq(Qtot_Train_2')));
```

```
simTotNARXModelTest7InputsLinearOpen_2 =
seq2con(SimulateNARXNetwork(TotFlowModel7InputsLinearNorm.Network,
con2seq(Qtot_Test7InputsLinear_2'), con2seq(Qtot_Test_2')));
simTotNARXModelTest7InputsLinearClosed_2 =
seq2con(SimulateNARXNetwork(totNARXModel7InputsLinearClosed,
con2seq(Qtot_Test7InputsLinear_2'), con2seq(Qtot_Test_2')));
simTotNARXModelTrain7InputsZscoreOpen_2 =
seq2con(SimulateNARXNetwork(TotFlowModel7InputsZscoreNorm.Network,
con2seq(Qtot_Train7InputsZscore_2'), con2seq(Qtot_Train_2')));
simTotNARXModelTrain7InputsZscoreClosed_2 =
seq2con(SimulateNARXNetwork(totNARXModel7InputsZscoreClosed,
con2seq(Qtot_Train7InputsZscore_2'), con2seq(Qtot_Train_2')));
simTotNARXModelTest7InputsZscoreOpen_2 =
seq2con(SimulateNARXNetwork(TotFlowModel7InputsZscoreNorm.Network,
con2seq(Qtot_Test7InputsZscore_2'), con2seq(Qtot_Test_2')));
simTotNARXModelTest7InputsZscoreClosed_2 =
seq2con(SimulateNARXNetwork(totNARXModel7InputsZscoreClosed,
con2seq(Qtot_Test7InputsZscore_2'), con2seq(Qtot_Test_2')));

%Estimate total flow with a delay model
simTotInputOutputModelTrain7InputsZscore_2 =
seq2con(SimulateInputOutputNetwork(TotFlowDelayModel7InputsZscoreNorm.Network,
con2seq(Qtot_Train7InputsZscore_2'), con2seq(Qtot_Train_2')));
simTotInputOutputModelTest7InputsZscore_2 =
seq2con(SimulateInputOutputNetwork(TotFlowDelayModel7InputsZscoreNorm.Network,
con2seq(Qtot_Test7InputsZscore_2'), con2seq(Qtot_Test_2')));
simTotInputOutputModelTrain5InputsZscore_2 =
seq2con(SimulateInputOutputNetwork(TotFlowDelayModel5InputsZscoreNorm.Network,
con2seq(Qtot_Train5InputsZscore_2'), con2seq(Qtot_Train_2')));
simTotInputOutputModelTest5InputsZscore_2 =
seq2con(SimulateInputOutputNetwork(TotFlowDelayModel5InputsZscoreNorm.Network,
con2seq(Qtot_Test5InputsZscore_2'), con2seq(Qtot_Test_2')));

%Calculated water cut from estimated flow
WLR_Train_2 = Qw_Train_2 ./ (Qw_Train_2 + Qo_Train_2);
WLR_Train_2(isnan(WLR_Train_2)) = 0;
WLR_Test_2 = Qw_Test_2 ./ (Qw_Test_2 + Qo_Test_2);
WLR_Test_2(isnan(WLR_Test_2)) = 0;
WLR_simInputOutputModelTrain7InputsZscore_2 =
simWatInputOutputModelTrain7InputsZscore_2{1, 1}' ./
(simWatInputOutputModelTrain7InputsZscore_2{1, 1}' +
simOilInputOutputModelTrain7InputsZscore_2{1, 1}');
WLR_simInputOutputModelTrain7InputsZscore_2(isnan(WLR_simInputOutputModelTrain7Inp
utsZscore_2)) = 0;
WLR_simInputOutputModelTrain5InputsZscore_2 =
simWatInputOutputModelTrain5InputsZscore_2{1, 1}' ./
(simWatInputOutputModelTrain5InputsZscore_2{1, 1}' +
simOilInputOutputModelTrain5InputsZscore_2{1, 1}');
WLR_simInputOutputModelTrain5InputsZscore_2(isnan(WLR_simInputOutputModelTrain5Inp
utsZscore_2)) = 0;
WLR_simInputOutputOutputModelTest7InputsZscore_2 =
simWatInputOutputModelTest7InputsZscore_2{1, 1}' ./
(simWatInputOutputModelTest7InputsZscore_2{1, 1}' +
simOilInputOutputModelTest7InputsZscore_2{1, 1}');
WLR_simInputOutputOutputModelTest7InputsZscore_2(isnan(WLR_simInputOutputOutputMod
elTest7InputsZscore_2)) = 0;
```

```
WLR_simInputOutputModelTest5InputsZscore_2 =
simWatInputOutputModelTest5InputsZscore_2{1, 1}' ./
(simWatInputOutputModelTest5InputsZscore_2{1, 1}' +
simOilInputOutputModelTest5InputsZscore_2{1, 1}');
WLR_simInputOutputModelTest5InputsZscore_2(isnan(WLR_simInputOutputModelTest5Input
sZscore_2)) = 0;

%Calculated gas void fraction from estimated flow
GV_Train_2 = Qg_Train_2 ./ Qtot_Train_2;
GV_Train_2(isnan(GV_Train_2)) = 0;
GV_Test_2 = Qg_Test_2 ./  Qtot_Test_2;
GV_Test_2(isnan(GV_Test_2)) = 0;
GV_simInputOutputModelTrain7InputsZscore_2 =
simGasInputOutputModelTrain8InputsZscore_2{1, 1}' ./
simTotInputOutputModelTrain7InputsZscore_2{1, 1}';
GV_simInputOutputModelTrain7InputsZscore_2(isnan(GV_simInputOutputModelTrain7Input
sZscore_2)) = 0;
GV_simInputOutputModelTrain5InputsZscore_2 =
simGasInputOutputModelTrain5InputsZscore_2{1, 1}' ./
simTotInputOutputModelTrain5InputsZscore_2{1, 1}';
GV_simInputOutputModelTrain5InputsZscore_2(isnan(GV_simInputOutputModelTrain5Input
sZscore_2)) = 0;
GV_simInputOutputModelTest7InputsZscore_2 =
simGasInputOutputModelTest8InputsZscore_2{1, 1}' ./
simTotInputOutputModelTest7InputsZscore_2{1, 1}';
GV_simInputOutputModelTest7InputsZscore_2(isnan(GV_simInputOutputModelTest7InputsZ
score_2)) = 0;
GV_simInputOutputModelTest5InputsZscore_2 =
simGasInputOutputModelTest5InputsZscore_2{1, 1}' ./
simTotInputOutputModelTest5InputsZscore_2{1, 1}';
GV_simInputOutputModelTest5InputsZscore_2(isnan(GV_simInputOutputModelTest5InputsZ
score_2)) = 0;

%Calculate RMS error
RMSE_GasInputOutputModelTest5InputsZscore_2 = sqrt(mean((Qg_Test_2(2:160) -
simGasInputOutputModelTest5InputsZscore_2{1,1}').^2));
RMSE_OilInputOutputModelTest5InputsZscore_2 = sqrt(mean((Qo_Test_2(2:160) -
simOilInputOutputModelTest5InputsZscore_2{1,1}').^2));
RMSE_WatInputOutputModelTest5InputsZscore_2 = sqrt(mean((Qw_Test_2(2:160) -
simWatInputOutputModelTest5InputsZscore_2{1,1}').^2));
RMSE_TotInputOutputModelTest5InputsZscore_2 = sqrt(mean((Qtot_Test_2(2:160) -
simTotInputOutputModelTest5InputsZscore_2{1,1}').^2));
RMSE_GasNARXModelTest8InputsZscoreClosed_2 = sqrt(mean((Qg_Test_2(2:160) -
simGasNARXModelTest8InputsZscoreClosed_2{1,1}').^2));
RMSE_OilNARXModelTest7InputsZscoreClosed_2 = sqrt(mean((Qo_Test_2(2:160) -
simOilNARXModelTest7InputsZscoreClosed_2{1,1}').^2));
RMSE_WatNARXModelTest7InputsZscoreClosed_2 = sqrt(mean((Qw_Test_2(2:160) -
simWatNARXModelTest7InputsZscoreClosed_2{1,1}').^2));
RMSE_TotNARXModelTest7InputsZscoreClosed_2 = sqrt(mean((Qtot_Test_2(2:160) -
simTotNARXModelTest7InputsZscoreClosed_2{1,1}').^2));

%-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-----

%Plots for data preparation
figure('Name','Voltage RMS of accelerometer samples in experiment G00');
```

```
plot(G00_2.testPoint.datAcc.ch(1).data(1:1*51200));
hold on
yline(rms(G00_2.testPoint.datAcc.ch(1).data(1*51200:2*51200)),'--k','Vrms value of
51200 samples','LineWidth',1);
xlim([1,51200]);
title('Voltage RMS of accelerometer samples in experiment G00');
legend('Accelerometer 1');
xlabel('Sample');
ylabel('V');

%Correlation plots
figure('Name','Correlation between differential pressure and flow');
corrplot([dataTrainAdj_2(:,2) dataTrainAdj_2(:,11:13) dataTrainAdj_2(:,16:18)
dataTrainAdj_2(:,29) dataTrainAdj_2(:,25) dataTrainAdj_2(:,26)
dataTrainAdj_2(:,27) dataTrainAdj_2(:,28)], VarNames=["wedge" "4mStr" "dp1V1"
"dp2V1" "dp3V2" "dp1V2" "dp2V2" "totDp" "gasF" "watF" "oilF" "totF"]);
figure('Name','Correlation between accelerometers and flow');
corrplot([dataTrainAdj_2(:,21) dataTrainAdj_2(:,22) dataTrainAdj_2(:,23)
dataTrainAdj_2(:,24) dataTrainAdj_2(:,25) dataTrainAdj_2(:,26)
dataTrainAdj_2(:,27) dataTrainAdj_2(:,28)], VarNames=["acc1" "acc2" "acc3" "acc4"
"gasF" "watF" "oilF" "totF"]);
figure('Name','Correlation between temperature and flow');
corrplot([dataTrainAdj_2(:,5) dataTrainAdj_2(:,19) dataTrainAdj_2(:,30)
dataTrainAdj_2(:,25) dataTrainAdj_2(:,26) dataTrainAdj_2(:,27)
dataTrainAdj_2(:,28)], VarNames=["innT" "outT" "gasF" "watF" "oilF" "totF" "dT"]);
figure('Name','Correlation between density, valve and flow');
corrplot([dataTrainAdj_2(:,1) dataTrainAdj_2(:,4) dataTrainAdj_2(:,8)
dataTrainAdj_2(:,10) dataTrainAdj_2(:,14) dataTrainAdj_2(:,25)
dataTrainAdj_2(:,26) dataTrainAdj_2(:,27) dataTrainAdj_2(:,28)], VarNames=["gam1"
"denK4" "denE" "denK3" "gam2" "gasF" "watF" "oilF" "totF"]);

%Histogram for sample distribution
figure('Name','Sample distribution');
subplot(2,5,1);
histogram(dataTrainAdj_2(:,10));
title('density Krohn3');
subplot(2,5,2);
histogram(dataTrainAdj_2(:,11));
title('dp 4m straight');
subplot(2,5,3);
histogram(dataTrainAdj_2(:,17));
title('dp 1 Venturi 2');
subplot(2,5,4);
histogram(dataTrainAdj_2(:,18));
title('dp 2 Venturi 2');
subplot(2,5,5);
histogram(dataTrainAdj_2(:,29));
title('delta pressure');
subplot(2,5,6);
histogram(dataTrainAdj_2(:,21));
title('accelerometer 1');
subplot(2,5,7);
histogram(dataTrainAdj_2(:,22));
title('accelerometer 2');
subplot(2,5,8);
histogram(dataTrainAdj_2(:,23));
title('accelerometer 3');
```

```
subplot(2,5,9);
histogram(dataTrainAdj_2(:,24));
title('accelerometer 4');
subplot(2,5,10);
histogram(dataTrainAdj_2(:,30));
title('delta temperature');

%Linear scaling normalization of inputs
figure('Name','Linear scaling normalization of inputs');
subplot(2,1,1);
plot(dataTrainAdj_2(:,10),'g');
hold on
plot(dataTrainAdj_2(:,11),'y');
plot(dataTrainAdj_2(:,17),'c');
plot(dataTrainAdj_2(:,18),'m');
plot(dataTrainAdj_2(:,21),'k');
plot(dataTrainAdj_2(:,22),'k');
plot(dataTrainAdj_2(:,23),'k');
plot(dataTrainAdj_2(:,24),'k');
plot(dataTrainAdj_2(:,29),'r');
plot(dataTrainAdj_2(:,30),'b');
title('Original');
legend('density Krohn3','dp 4m straight','dp 1 Venturi 2','dp 2 Venturi
2','accelerometer 1','accelerometer 2','accelerometer 3','accelerometer 4','delta
pressure','delta temperature');
xlabel('seconds');
ylabel('value');
subplot(2,1,2);
plot(trainLinear_2(:,10),'g');
hold on
plot(trainLinear_2(:,11),'y');
plot(trainLinear_2(:,17),'c');
plot(trainLinear_2(:,18),'m');
plot(trainLinear_2(:,21),'k');
plot(trainLinear_2(:,22),'k');
plot(trainLinear_2(:,23),'k');
plot(trainLinear_2(:,24),'k');
plot(trainLinear_2(:,29),'r');
plot(trainLinear_2(:,30),'b');
title('Linear scaling normalization of inputs');
legend('density Krohn3','dp 4m straight','dp 1 Venturi 2','dp 2 Venturi
2','accelerometer 1','accelerometer 2','accelerometer 3','accelerometer 4','delta
pressure','delta temperature')
xlabel('seconds');
ylabel('value');

%Linear scaling normalization of inputs
figure('Name','Linear scaling normalization of inputs');
subplot(10,1,1);
plot(trainLinear_2(:,10),'g');
title('density Krohn3');
xlabel('seconds');
ylabel('value');
subplot(10,1,2);
plot(trainLinear_2(:,11),'y');
title('dp 4m straight');
xlabel('seconds');
```

```
ylabel('value');
subplot(10,1,3);
plot(trainLinear_2(:,17),'c');
title('dp 1 Venturi 2');
xlabel('seconds');
ylabel('value');
subplot(10,1,4);
plot(trainLinear_2(:,18),'m');
title('dp 2 Venturi 2');
xlabel('seconds');
ylabel('value');
subplot(10,1,5);
plot(trainLinear_2(:,21),'k');
title('accelerometer 1');
xlabel('seconds');
ylabel('value');
subplot(10,1,6);
plot(trainLinear_2(:,22),'k');
title('accelerometer 2');
xlabel('seconds');
ylabel('value');
subplot(10,1,7);
plot(trainLinear_2(:,23),'k');
title('accelerometer 3');
xlabel('seconds');
ylabel('value');
subplot(10,1,8);
plot(trainLinear_2(:,24),'k');
title('accelerometer 4');
xlabel('seconds');
ylabel('value');
subplot(10,1,9);
plot(trainLinear_2(:,29),'r');
title('delta pressure');
xlabel('seconds');
ylabel('value');
subplot(10,1,10);
plot(trainLinear_2(:,30),'b');
title('delta temperature');
xlabel('seconds');
ylabel('value');

%Z-score normalization of inputs
figure('Name','Z-score normalization of inputs');
subplot(2,1,1);
plot(dataTrainAdj_2(:,10),'g');
hold on
plot(dataTrainAdj_2(:,11),'y');
plot(dataTrainAdj_2(:,17),'c');
plot(dataTrainAdj_2(:,18),'m');
plot(dataTrainAdj_2(:,21),'k');
plot(dataTrainAdj_2(:,22),'k');
plot(dataTrainAdj_2(:,23),'k');
plot(dataTrainAdj_2(:,24),'k');
plot(dataTrainAdj_2(:,29),'r');
plot(dataTrainAdj_2(:,30),'b');
title('Original');
```

```
legend('density Krohn3','dp 4m straight','dp 1 Venturi 2','dp 2 Venturi
2','accelerometer 1','accelerometer 2','accelerometer 3','accelerometer 4','delta
pressure','delta temperature');
xlabel('seconds');
ylabel('value');
subplot(2,1,2);
plot(trainZscore_2(:,10),'g');
hold on
plot(trainZscore_2(:,11),'y');
plot(trainZscore_2(:,17),'c');
plot(trainZscore_2(:,18),'m');
plot(trainZscore_2(:,21),'k');
plot(trainZscore_2(:,22),'k');
plot(trainZscore_2(:,23),'k');
plot(trainZscore_2(:,24),'k');
plot(trainZscore_2(:,29),'r');
plot(trainZscore_2(:,30),'b');
title('Normalized with z-score');
legend('density Krohn3','dp 4m straight','dp 1 Venturi 2','dp 2 Venturi
2','accelerometer 1','accelerometer 2','accelerometer 3','accelerometer 4','delta
pressure','delta temperature')
xlabel('seconds');
ylabel('value');

%Z-score normalization of inputs
figure('Name','Z-score normalization of inputs');
subplot(10,1,1);
plot(trainZscore_2(:,10),'g');
title('density Krohn3');
xlabel('seconds');
ylabel('value');
subplot(10,1,2);
plot(trainZscore_2(:,11),'y');
title('dp 4m straight');
xlabel('seconds');
ylabel('value');
subplot(10,1,3);
plot(trainZscore_2(:,17),'c');
title('dp 1 Venturi 2');
xlabel('seconds');
ylabel('value');
subplot(10,1,4);
plot(trainZscore_2(:,18),'m');
title('dp 2 Venturi 2');
xlabel('seconds');
ylabel('value');
subplot(10,1,5);
plot(trainZscore_2(:,21),'k');
title('accelerometer 1');
xlabel('seconds');
ylabel('value');
subplot(10,1,6);
plot(trainZscore_2(:,22),'k');
title('accelerometer 2');
xlabel('seconds');
ylabel('value');
subplot(10,1,7);
```

```
plot(trainZscore_2(:,23),'k');
title('accelerometer 3');
xlabel('seconds');
ylabel('value');
subplot(10,1,8);
plot(trainZscore_2(:,24),'k');
title('accelerometer 4');
xlabel('seconds');
ylabel('value');
subplot(10,1,9);
plot(trainZscore_2(:,29),'r');
title('delta pressure');
xlabel('seconds');
ylabel('value');
subplot(10,1,10);
plot(trainZscore_2(:,30),'b');
title('delta temperature');
xlabel('seconds');
ylabel('value');

%Plots for gas flow models
figure('Name','Gas flow NARX model using 8 inputs and linear scaling
normalization');
subplot(2,1,1)
plot(Qg_Train_2,'b');
hold on;
plot(simGasNARXModelTrain8InputsLinearOpen_2{1, 1},'r');
plot(simGasNARXModelTrain8InputsLinearClosed_2{1, 1},'k');
title('Model using training data from dataset2');
legend('Actual output','Opel loop estimate','Closed loop estimate');
xlabel('sample');
ylabel('flow [m3/h]');
subplot(2,1,2);
plot(Qg_Test_2,'b');
hold on;
plot(simGasNARXModelTest8InputsLinearOpen_2{1, 1},'r');
plot(simGasNARXModelTest8InputsLinearClosed_2{1, 1},'k');
title('Model using testing data from dataset2');
legend('Actual output','Opel loop estimate','Closed loop estimate');
xlabel('sample');
ylabel('flow [m3/h]');
figure('Name','Gas flow NARX model using 8 inputs and Z-score normalization');
subplot(2,1,1)
plot(Qg_Train_2,'b');
hold on;
plot(simGasNARXModelTrain8InputsZscoreOpen_2{1, 1},'r');
plot(simGasNARXModelTrain8InputsZscoreClosed_2{1, 1},'k');
title('Model using training data from dataset2');
legend('Actual output','Opel loop estimate','Closed loop estimate');
xlabel('sample');
ylabel('flow [m3/h]');
subplot(2,1,2);
plot(Qg_Test_2,'b');
hold on;
plot(simGasNARXModelTest8InputsZscoreOpen_2{1, 1},'r');
plot(simGasNARXModelTest8InputsZscoreClosed_2{1, 1},'k');
title('Model using testing data from dataset2');
```

```
legend('Actual output','Opel loop estimate','Closed loop estimate');
xlabel('sample');
ylabel('flow [m3/h]');
figure('Name','Gas flow delay model using 8 inputs and Z-score normalization');
subplot(2,1,1)
plot(Qg_Train_2,'b');
hold on;
plot(simGasInputOutputModelTrain8InputsZscore_2{1, 1},'r');
title('Model using training data from dataset2');
legend('Actual output','Estimated output');
xlabel('sample');
ylabel('flow [m3/h]');
subplot(2,1,2);
plot(Qg_Test_2,'b');
hold on;
plot(simGasInputOutputModelTest8InputsZscore_2{1, 1},'r');
title('Model using testing data from dataset2');
legend('Actual output','Estimated output');
xlabel('sample');
ylabel('flow [m3/h]');
ylim([-10,70]);
figure('Name','Gas flow delay model using 5 inputs and Z-score normalization');
subplot(2,1,1)
plot(Qg_Train_2,'b');
hold on;
plot(simGasInputOutputModelTrain5InputsZscore_2{1, 1},'r');
title('Model using training data from dataset2');
legend('Actual output','Estimated output');
xlabel('sample');
ylabel('flow [m3/h]');
subplot(2,1,2);
plot(Qg_Test_2,'b');
hold on;
plot(simGasInputOutputModelTest5InputsZscore_2{1, 1},'r');
title('Model using testing data from dataset2');
legend('Actual output','Estimated output');
xlabel('sample');
ylabel('flow [m3/h]');
ylim([-10,70]);

%Plots for oil flow models
figure('Name','Oil flow NARX model using 7 inputs and linear scaling');
subplot(2,1,1)
plot(Qo_Train_2,'b');
hold on;
plot(simOilNARXModelTrain7InputsLinearOpen_2{1, 1},'r');
plot(simOilNARXModelTrain7InputsLinearClosed_2{1, 1},'k');
title('Model using training data from dataset2');
legend('Actual output','Opel loop estimate','Closed loop estimate');
xlabel('sample');
ylabel('flow [m3/h]');
subplot(2,1,2);
plot(Qo_Test_2,'b');
plot(simOilNARXModelTest7InputsLinearOpen_2{1, 1},'r');
hold on;
plot(Qo_Test_2,'b');
plot(simOilNARXModelTest7InputsLinearClosed_2{1, 1},'k');
```

```
title('Model using testing data from dataset2');
legend('Actual output','Opel loop estimate','Closed loop estimate');
xlabel('sample');
ylabel('flow [m3/h]');

figure('Name','Oil flow NARX model using 7 inputs and Z-score');
subplot(2,1,1)
plot(Qo_Train_2,'b');
hold on;
plot(simOilNARXModelTrain7InputsZscoreOpen_2{1, 1},'r');
plot(simOilNARXModelTrain7InputsZscoreClosed_2{1, 1},'k');
title('Model using training data from dataset2');
legend('Actual output','Opel loop estimate','Closed loop estimate');
xlabel('sample');
ylabel('flow [m3/h]');
subplot(2,1,2);
plot(simOilNARXModelTest7InputsZscoreOpen_2{1, 1},'r');
hold on;
plot(Qo_Test_2,'b');
plot(simOilNARXModelTest7InputsZscoreClosed_2{1, 1},'k');
title('Model using testing data from dataset2');
legend('Actual output','Opel loop estimate','Closed loop estimate');
xlabel('sample');
ylabel('flow [m3/h]');
figure('Name','Oil flow delay model using 7 inputs and Z-score');
subplot(2,1,1)
plot(Qo_Train_2,'b');
hold on;
plot(simOilInputOutputModelTrain7InputsZscore_2{1, 1},'r');
title('Model using training data from dataset2');
legend('Actual output','Estimated output');
xlabel('sample');
ylabel('flow [m3/h]');
subplot(2,1,2);
plot(simOilInputOutputModelTest7InputsZscore_2{1, 1},'r');
hold on;
plot(Qo_Test_2,'b');
title('Model using testing data from dataset2');
legend('Actual output','Estimated output');
xlabel('sample');
ylabel('flow [m3/h]');
ylim([0,70]);
figure('Name','Oil flow delay model using 5 inputs and Z-score');
subplot(2,1,1)
plot(Qo_Train_2,'b');
hold on;
plot(simOilInputOutputModelTrain5InputsZscore_2{1, 1},'r');
title('Model using training data from dataset2');
legend('Actual output','Estimated output');
xlabel('sample');
ylabel('flow [m3/h]');
subplot(2,1,2);
plot(simOilInputOutputModelTest5InputsZscore_2{1, 1},'r');
hold on;
plot(Qo_Test_2,'b');
title('Model using testing data from dataset2');
legend('Actual output','Estimated output');
```

```
xlabel('sample');
ylabel('flow [m3/h]');
ylim([-10,70]);

%Plots for water flow models
figure('Name','Water flow NARX model using 7 inputs and linear scaling');
subplot(2,1,1)
plot(Qw_Train_2,'b');
hold on;
plot(simWatNARXModelTrain7InputsLinearOpen_2{1, 1},'r');
plot(simWatNARXModelTrain7InputsLinearClosed_2{1, 1},'k');
title('Model using training data from dataset2');
legend('Actual output','Opel loop estimate','Closed loop estimate');
xlabel('sample');
ylabel('flow [m3/h]');
subplot(2,1,2);
plot(Qw_Test_2,'b');
hold on;
plot(simWatNARXModelTest7InputsLinearOpen_2{1, 1},'r');
plot(simWatNARXModelTest7InputsLinearClosed_2{1, 1},'k');
title('Model using testing data from dataset2');
legend('Actual output','Opel loop estimate','Closed loop estimate');
xlabel('sample');
ylabel('flow [m3/h]');
figure('Name','Water flow model using 7 inputs and Z-score');
subplot(2,1,1)
plot(Qw_Train_2,'b');
hold on;
plot(simWatNARXModelTrain7InputsZscoreOpen_2{1, 1},'r');
plot(simWatNARXModelTrain7InputsZscoreClosed_2{1, 1},'k');
title('Model using training data from dataset2');
legend('Actual output','Opel loop estimate','Closed loop estimate');
xlabel('sample');
ylabel('flow [m3/h]');
subplot(2,1,2);
plot(Qw_Test_2,'b');
hold on;
plot(simWatNARXModelTest7InputsZscoreOpen_2{1, 1},'r');
plot(simWatNARXModelTest7InputsZscoreClosed_2{1, 1},'k');
title('Model using testing data from dataset2');
legend('Actual output','Opel loop estimate','Closed loop estimate');
xlabel('sample');
ylabel('flow [m3/h]');
figure('Name','Water flow NARX model using 7 inputs and Z-score');
subplot(2,1,1)
plot(Qw_Train_2,'b');
hold on;
plot(simWatNARXModelTrain7InputsZscoreOpen_2{1, 1},'r');
plot(simWatNARXModelTrain7InputsZscoreClosed_2{1, 1},'k');
title('Model using training data from dataset2');
legend('Actual output','Opel loop estimate','Closed loop estimate');
xlabel('sample');
ylabel('flow [m3/h]');
subplot(2,1,2);
plot(Qw_Test_2,'b');
hold on;
plot(simWatNARXModelTest7InputsZscoreOpen_2{1, 1},'r');
```

```
plot(simWatNARXModelTest7InputsZscoreClosed_2{1, 1},'k');
title('Model using testing data from dataset2');
legend('Actual output','Opel loop estimate','Closed loop estimate');
xlabel('sample');
ylabel('flow [m3/h]');
figure('Name','Water flow delay model using 7 inputs and Z-score');
subplot(2,1,1)
plot(Qw_Train_2,'b');
hold on;
plot(simWatInputOutputModelTrain7InputsZscore_2{1, 1},'r');
title('Model using training data from dataset2');
legend('Actual output','Estimated output');
xlabel('sample');
ylabel('flow [m3/h]');
subplot(2,1,2);
plot(Qw_Test_2,'b');
hold on;
plot(simWatInputOutputModelTest7InputsZscore_2{1, 1},'r');
title('Model using testing data from dataset2');
legend('Actual output','Estimated output');
xlabel('sample');
ylabel('flow [m3/h]');
figure('Name','Water flow delay model using 5 inputs and Z-score');
subplot(2,1,1)
plot(Qw_Train_2,'b');
hold on;
plot(simWatInputOutputModelTrain5InputsZscore_2{1, 1},'r');
title('Model using training data from dataset2');
legend('Actual output','Estimated output');
xlabel('sample');
ylabel('flow [m3/h]');
subplot(2,1,2);
plot(Qw_Test_2,'b');
hold on;
plot(simWatInputOutputModelTest5InputsZscore_2{1, 1},'r');
title('Model using testing data from dataset2');
legend('Actual output','Estimated output');
xlabel('sample');
ylabel('flow [m3/h]');

%Plots for total flow
figure('Name','Total flow NARX model using 7 inputs and linear scaling');
subplot(2,1,1)
plot(Qtot_Train_2,'b');
hold on;
plot(simTotNARXModelTrain7InputsLinearOpen_2{1, 1},'r');
plot(simTotNARXModelTrain7InputsLinearClosed_2{1, 1},'k');
title('Model using training data from dataset2');
legend('Actual output','Opel loop estimate','Closed loop estimate');
xlabel('sample');
ylabel('flow [m3/h]');
subplot(2,1,2);
plot(Qtot_Test_2,'b');
hold on;
plot(simTotNARXModelTest7InputsLinearOpen_2{1, 1},'r');
plot(simTotNARXModelTest7InputsLinearClosed_2{1, 1},'k');
title('Model using testing data from dataset2');
```

```
legend('Actual output','Opel loop estimate','Closed loop estimate');
xlabel('sample');
ylabel('flow [m3/h]');
figure('Name','Total flow NARX model using 7 inputs and Z-score');
subplot(2,1,1)
plot(Qtot_Train_2,'b');
hold on;
plot(simTotNARXModelTrain7InputsZscoreOpen_2{1, 1},'r');
plot(simTotNARXModelTrain7InputsZscoreClosed_2{1, 1},'k');
title('Model using training data from dataset2');
legend('Actual output','Opel loop estimate','Closed loop estimate');
xlabel('sample');
ylabel('flow [m3/h]');
subplot(2,1,2);
plot(Qtot_Test_2,'b');
hold on;
plot(simTotNARXModelTest7InputsZscoreOpen_2{1, 1},'r');
plot(simTotNARXModelTest7InputsZscoreClosed_2{1, 1},'k');
title('Model using testing data from dataset2');
legend('Actual output','Opel loop estimate','Closed loop estimate');
xlabel('sample');
ylabel('flow [m3/h]');
figure('Name','Total flow delay model using 7 inputs and Z-score');
subplot(2,1,1)
plot(Qtot_Train_2,'b');
hold on;
plot(simTotInputOutputModelTrain7InputsZscore_2{1, 1},'r');
title('Model using training data from dataset2');
legend('Actual output','Estimated output');
xlabel('sample');
ylabel('flow [m3/h]');
subplot(2,1,2);
plot(Qtot_Test_2,'b');
hold on;
plot(simTotInputOutputModelTest7InputsZscore_2{1, 1},'r');
title('Model using testing data from dataset2');
legend('Actual output','Estimated output');
xlabel('sample');
ylabel('flow [m3/h]');
ylim([0,120]);

figure('Name','Total flow delay model using 5 inputs and Z-score');
subplot(2,1,1)
plot(Qtot_Train_2,'b');
hold on;
plot(simTotInputOutputModelTrain5InputsZscore_2{1, 1},'r');
title('Model using training data from dataset2');
legend('Actual output','Estimated output');
xlabel('sample');
ylabel('flow [m3/h]');
subplot(2,1,2);
plot(Qtot_Test_2,'b');
hold on;
plot(simTotInputOutputModelTest5InputsZscore_2{1, 1},'r');
title('Model using testing data from dataset2');
legend('Actual output','Estimated output');
xlabel('sample');
```

```
ylabel('flow [m3/h]');
ylim([-20,120]);

%Water cut
figure('Name','Water Liquid Ratio');
subplot(2,1,1);
plot(WLR_Train_2,'b');
hold on;
plot(WLR_simInputOutputModelTrain7InputsZscore_2, 'r');
legend('Actual','Estimated');
title('Estimated WLR using 7 inputs from training data in dataset2');
ylim([-1 1]);
xlabel('samples');
ylabel('WLR')
subplot(2,1,2);
plot(WLR_Train_2,'b');
hold on;
plot(WLR_simInputOutputModelTrain5InputsZscore_2, 'r');
legend('Actual','Estimated');
title('Estimated WLR using 5 inputs from from training data in dataset2');
ylim([-1 1]);
xlabel('samples');
ylabel('WLR')

%Gas void fraction
figure('Name','Gas Volume Fraction');
subplot(2,1,1);
plot(GV_Train_2,'b');
hold on;
plot(GV_simInputOutputModelTrain7InputsZscore_2, 'r');
legend('Actual','Estimated');
title('Estimated GVF using 7 inputs from training data in dataset2');
xlabel('sample');
ylim([-1.5 1])
ylabel('GVF')
subplot(2,1,2);
plot(GV_Train_2,'b');
hold on;
plot(GV_simInputOutputModelTrain5InputsZscore_2, 'r');
legend('Actual','Estimated');
title('Estimated GVF fraction using 5 inputs from training data in dataset2');
xlabel('samples');
ylim([-1.5 1]);
ylabel('GVF')

%Results plot
figure('Name',' Overview of the results');
subplot(6,1,1);
plot(Qg_Test_2,'b');
hold on;
plot(max(0, simGasInputOutputModelTest5InputsZscore_2{1, 1}),'r');
title('Nonlinear input-output network for gas flowrate using 5 inputs from testing
data and Z-score normalization');
legend('Actual','Estimated');
xlabel('sample');
ylabel('flow [m3/h]');
ylim([-10,70]);
```

```
subplot(6,1,2);
plot(Qw_Test_2,'b');
hold on;
plot(max(0, simWatInputOutputModelTest5InputsZscore_2{1, 1}),'r');
title('Nonlinear input-output network for water flowrate using 5 inputs from
testing data and Z-score');
legend('Actual','Estimated');
xlabel('sample');
ylabel('flow [m3/h]');
ylim([-10,50]);
subplot(6,1,3);
plot(Qo_Test_2,'b');
hold on;
plot(max(0, simOilInputOutputModelTest5InputsZscore_2{1, 1}),'r');
title('Nonlinear input-output network for oil flowrate using 5 inputs from
testingdata and Z-score');
legend('Actual','Estimated');
xlabel('sample');
ylabel('flow [m3/h]');
ylim([-10,60]);
subplot(6,1,4);
plot(Qtot_Test_2,'b');
hold on;
plot(max(0, simTotInputOutputModelTest5InputsZscore_2{1, 1}),'r');
title('Nonlinear input-output network for total flowrate model using 5 inputs and
Z-score normalization');
legend('Actual','Estimated');
xlabel('sample');
ylabel('flow [m3/h]');
ylim([-10,150]);
subplot(6,1,5)
plot(WLR_Test_2,'b');
hold on;
plot(max(0, WLR_simInputOutputModelTest5InputsZscore_2),'r');
title('Estimated WLR using testing data from dataset2');
legend('Actual','Estimated');
xlabel('sample');
ylabel('WLR');
ylim([-0.1,1.4]);
subplot(6,1,6)
plot(GV_Test_2,'b');
hold on;
plot(max(0, GV_simInputOutputModelTest5InputsZscore_2),'r');
title('Estimated GVF using training data from dataset2');
legend('Actual','Estimated');
xlabel('sample');
ylabel('GVF');
ylim([-0.1,0.7]);



%-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-----

%Function for preparing data before network simulation
```

```
function y = SimulateNARXNetwork(network, inputData, outputData)
    [Xs,Xi, Ai] = preparets(network, inputData, {}, outputData);
    y = sim(network, Xs, Xi, Ai);
end

function y = SimulateInputOutputNetwork(network, inputData, outputData)
    [Xs,Xi, Ai] = preparets(network, inputData, outputData);
    y = sim(network, Xs, Xi, Ai);
end
```

Appendix C

Preparing single-phase flow experiment data from dataset 2.

```
clc;
clear all;

%Load data
O01 = load('O_01.mat');
O01_1secData = load('O_01_1secData.mat');
O02 = load('O_02.mat');
O02_1secData = load('O_02_1secData.mat');
O03 = load('O_03.mat');
O03_1secData = load('O_03_1secData.mat');
O04 = load('O_04.mat');
O04_1secData = load('O_04_1secData.mat');
O05 = load('O_05.mat');
O05_1secData = load('O_05_1secData.mat');
O06 = load('O_06.mat');
O06_1secData = load('O_06_1secData.mat');
O07 = load('O_07.mat');
O07_1secData = load('O_07_1secData.mat');
O08 = load('O_08.mat');
O08_1secData = load('O_08_1secData.mat');
G00 = load('G_00.mat');
G00_1secData = load('G_00_1secData.mat');
G01 = load('G_01.mat');
G01_1secData = load('G_01_1secData.mat');
G02 = load('G_02.mat');
G02_1secData = load('G_02_1secData.mat');
G03 = load('G_03.mat');
G03_1secData = load('G_03_1secData.mat');
G04 = load('G_04.mat');
G04_1secData = load('G_04_1secData.mat');
G05 = load('G_05.mat');
G05_1secData = load('G_05_1secData.mat');
G08 = load('G_08.mat');
G08_1secData = load('G_08_1secData.mat');
W00 = load('W_00.mat');
W00_1secData = load('W_00_1secData.mat');
W00b = load('W_00b.mat');
W00b_1secData = load('W_00b_1secData.mat');
W01 = load('W_01.mat');
W01_1secData = load('W_01_1secData.mat');
W01b = load('W_01b.mat');
W01b_1secData = load('W_01b_1secData.mat');
W01c = load('W_01c.mat');
W01c_1secData = load('W_01c_1secData.mat');
W02 = load('W_02.mat');
W02_1secData = load('W_02_1secData.mat');
W03 = load('W_03.mat');
W03_1secData = load('W_03_1secData.mat');
W04 = load('W_04.mat');
W04_1secData = load('W_04_1secData.mat');
W05= load('W_05.mat');
W05_1secData = load('W_05_1secData.mat');
```

```
W501 = load('W_501.mat');
W501_1secData = load('W_501_1secData.mat');
W501C3 = load('W_501C3.mat');
W501C3_1secData = load('W_501C3_1secData.mat');
W501C6 = load('W_501C6.mat');
W501C6_1secData = load('W_501C6_1secData.mat');
W502 = load('W_502.mat');
W502_1secData = load('W_502_1secData.mat');
W502C3 = load('W_502C3.mat');
W502C3_1secData = load('W_502C3_1secData.mat');
W503 = load('W_503.mat');
W503_1secData = load('W_503_1secData.mat');

%Calculate average RMS with 1Hz samwple rate
N_acc = length(O01.testPoint.datAcc.ch(1).data);
N = length(O01_1secData.dataStruct.data);
dt = 1/O01.testPoint.datAcc.dt;
time = N_acc/dt;

%Select samples in each dataset:
start = 100;
stop = 109;

for j = 1:4
    for i = 1:1:time
        O01_acc_1hz_rms(i,j) = rms(O01.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        O02_acc_1hz_rms(i,j) = rms(O02.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        O03_acc_1hz_rms(i,j) = rms(O03.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        O04_acc_1hz_rms(i,j) = rms(O04.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        O05_acc_1hz_rms(i,j) = rms(O05.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        O06_acc_1hz_rms(i,j) = rms(O06.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        O07_acc_1hz_rms(i,j) = rms(O07.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        O08_acc_1hz_rms(i,j) = rms(O08.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));

        G00_acc_1hz_rms(i,j) = rms(G00.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        G01_acc_1hz_rms(i,j) = rms(G01.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        G02_acc_1hz_rms(i,j) = rms(G02.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        G03_acc_1hz_rms(i,j) = rms(G03.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        G04_acc_1hz_rms(i,j) = rms(G04.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        G05_acc_1hz_rms(i,j) = rms(G05.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        G08_acc_1hz_rms(i,j) = rms(G08.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
```

```
        W00_acc_1hz_rms(i,j) = rms(W00.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        W00b_acc_1hz_rms(i,j) = rms(W00b.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        W01_acc_1hz_rms(i,j) = rms(W01.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        W01b_acc_1hz_rms(i,j) = rms(W01b.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        W01c_acc_1hz_rms(i,j) = rms(W01c.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        W02_acc_1hz_rms(i,j) = rms(W02.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        W03_acc_1hz_rms(i,j) = rms(W03.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        W04_acc_1hz_rms(i,j) = rms(W04.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        W05_acc_1hz_rms(i,j) = rms(W05.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        W501_acc_1hz_rms(i,j) = rms(W501.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        W501C3_acc_1hz_rms(i,j) = rms(W501C3.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        W501C6_acc_1hz_rms(i,j) = rms(W501C6.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        W502_acc_1hz_rms(i,j) = rms(W502.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        W502C3_acc_1hz_rms(i,j) = rms(W502C3.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        W503_acc_1hz_rms(i,j) = rms(W503.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
    end
end

%Data matrices with 1Hz samplingrate
O01_data = [O02_1secData.dataStruct.data O01_acc_1hz_rms
O01.testPoint.gasRef.q*ones([N,1]) O01.testPoint.watRef.q*ones([N,1])
O01.testPoint.oilRef.q*ones([N,1])];
O02_data = [O02_1secData.dataStruct.data O02_acc_1hz_rms
O02.testPoint.gasRef.q*ones([N,1]) O02.testPoint.watRef.q*ones([N,1])
O02.testPoint.oilRef.q*ones([N,1])];
O03_data = [O02_1secData.dataStruct.data O03_acc_1hz_rms
O03.testPoint.gasRef.q*ones([N,1]) O03.testPoint.watRef.q*ones([N,1])
O03.testPoint.oilRef.q*ones([N,1])];
O04_data = [O02_1secData.dataStruct.data O04_acc_1hz_rms
O04.testPoint.gasRef.q*ones([N,1]) O04.testPoint.watRef.q*ones([N,1])
O04.testPoint.oilRef.q*ones([N,1])];
O05_data = [O05_1secData.dataStruct.data O05_acc_1hz_rms
O05.testPoint.gasRef.q*ones([N,1]) O05.testPoint.watRef.q*ones([N,1])
O05.testPoint.oilRef.q*ones([N,1])];
O06_data = [O02_1secData.dataStruct.data O06_acc_1hz_rms
O06.testPoint.gasRef.q*ones([N,1]) O06.testPoint.watRef.q*ones([N,1])
O06.testPoint.oilRef.q*ones([N,1])];
O07_data = [O02_1secData.dataStruct.data O07_acc_1hz_rms
O07.testPoint.gasRef.q*ones([N,1]) O07.testPoint.watRef.q*ones([N,1])
O07.testPoint.oilRef.q*ones([N,1])];
```

```
O08_data = [O08_1secData.dataStruct.data O08_acc_1hz_rms
O08.testPoint.gasRef.q*ones([N,1]) O08.testPoint.watRef.q*ones([N,1])
O08.testPoint.oilRef.q*ones([N,1])];
G00_data = [G00_1secData.dataStruct.data G00_acc_1hz_rms
G00.testPoint.gasRef.q*ones([N,1]) G00.testPoint.watRef.q*ones([N,1])
G00.testPoint.oilRef.q*ones([N,1])];
G01_data = [G01_1secData.dataStruct.data G01_acc_1hz_rms
G01.testPoint.gasRef.q*ones([N,1]) G01.testPoint.watRef.q*ones([N,1])
G01.testPoint.oilRef.q*ones([N,1])];
G02_data = [G02_1secData.dataStruct.data G02_acc_1hz_rms
G02.testPoint.gasRef.q*ones([N,1]) G02.testPoint.watRef.q*ones([N,1])
G02.testPoint.oilRef.q*ones([N,1])];
G03_data = [G03_1secData.dataStruct.data G03_acc_1hz_rms
G03.testPoint.gasRef.q*ones([N,1]) G03.testPoint.watRef.q*ones([N,1])
G03.testPoint.oilRef.q*ones([N,1])];
G04_data = [G04_1secData.dataStruct.data G04_acc_1hz_rms
G04.testPoint.gasRef.q*ones([N,1]) G04.testPoint.watRef.q*ones([N,1])
G04.testPoint.oilRef.q*ones([N,1])];
G05_data = [G05_1secData.dataStruct.data G05_acc_1hz_rms
G05.testPoint.gasRef.q*ones([N,1]) G05.testPoint.watRef.q*ones([N,1])
G05.testPoint.oilRef.q*ones([N,1])];
G08_data = [G08_1secData.dataStruct.data G08_acc_1hz_rms
G08.testPoint.gasRef.q*ones([N,1]) G08.testPoint.watRef.q*ones([N,1])
G08.testPoint.oilRef.q*ones([N,1])];
W00_data = [W00_1secData.dataStruct.data W00_acc_1hz_rms
W00.testPoint.gasRef.q*ones([N,1]) W00.testPoint.watRef.q*ones([N,1])
W00.testPoint.oilRef.q*ones([N,1])];
W00b_data = [W00b_1secData.dataStruct.data W00b_acc_1hz_rms
W00b.testPoint.gasRef.q*ones([N,1]) W00b.testPoint.watRef.q*ones([N,1])
W00b.testPoint.oilRef.q*ones([N,1])];
W01_data = [W01_1secData.dataStruct.data W01_acc_1hz_rms
W01.testPoint.gasRef.q*ones([N,1]) W01.testPoint.watRef.q*ones([N,1])
W01.testPoint.oilRef.q*ones([N,1])];
W01b_data = [W01b_1secData.dataStruct.data W01b_acc_1hz_rms
W01b.testPoint.gasRef.q*ones([N,1]) W01b.testPoint.watRef.q*ones([N,1])
W01b.testPoint.oilRef.q*ones([N,1])];
W01c_data = [W01c_1secData.dataStruct.data W01c_acc_1hz_rms
W01c.testPoint.gasRef.q*ones([N,1]) W01c.testPoint.watRef.q*ones([N,1])
W01c.testPoint.oilRef.q*ones([N,1])];
W02_data = [W02_1secData.dataStruct.data W02_acc_1hz_rms
W02.testPoint.gasRef.q*ones([N,1]) W02.testPoint.watRef.q*ones([N,1])
W02.testPoint.oilRef.q*ones([N,1])];
W03_data = [W03_1secData.dataStruct.data W03_acc_1hz_rms
W03.testPoint.gasRef.q*ones([N,1]) W03.testPoint.watRef.q*ones([N,1])
W03.testPoint.oilRef.q*ones([N,1])];
W04_data = [W04_1secData.dataStruct.data W04_acc_1hz_rms
W04.testPoint.gasRef.q*ones([N,1]) W04.testPoint.watRef.q*ones([N,1])
W04.testPoint.oilRef.q*ones([N,1])];
W05_data = [W05_1secData.dataStruct.data W05_acc_1hz_rms
W05.testPoint.gasRef.q*ones([N,1]) W05.testPoint.watRef.q*ones([N,1])
W05.testPoint.oilRef.q*ones([N,1])];
W501_data = [W501_1secData.dataStruct.data W501_acc_1hz_rms
W501.testPoint.gasRef.q*ones([N,1]) W501.testPoint.watRef.q*ones([N,1])
W501.testPoint.oilRef.q*ones([N,1])];
W501C3_data = [W501C3_1secData.dataStruct.data W501C3_acc_1hz_rms
W501C3.testPoint.gasRef.q*ones([N,1]) W501C3.testPoint.watRef.q*ones([N,1])
W501C3.testPoint.oilRef.q*ones([N,1])];
```

```
W501C6_data = [W501C6_1secData.dataStruct.data W501C6_acc_1hz_rms
W501C6.testPoint.gasRef.q*ones([N,1]) W501C6.testPoint.watRef.q*ones([N,1])
W501C6.testPoint.oilRef.q*ones([N,1])];
W502_data = [W502_1secData.dataStruct.data W502_acc_1hz_rms
W502.testPoint.gasRef.q*ones([N,1]) W502.testPoint.watRef.q*ones([N,1])
W502.testPoint.oilRef.q*ones([N,1])];
W502C3_data = [W502C3_1secData.dataStruct.data W502C3_acc_1hz_rms
W502C3.testPoint.gasRef.q*ones([N,1]) W502C3.testPoint.watRef.q*ones([N,1])
W502C3.testPoint.oilRef.q*ones([N,1])];
W503_data = [W503_1secData.dataStruct.data W03_acc_1hz_rms
W503.testPoint.gasRef.q*ones([N,1]) W503.testPoint.watRef.q*ones([N,1])
W503.testPoint.oilRef.q*ones([N,1])];

%Merging all training data
training_O = [O01_data(start:stop,:); O02_data(start:stop,:);
O03_data(start:stop,:); O04_data(start:stop,:); O05_data(start:stop,:);
O06_data(start:stop,:); O07_data(start:stop,:);O08_data(start:stop,:)];
training_G = [G01_data(start:stop,:); G02_data(start:stop,:);
G03_data(start:stop,:); G04_data(start:stop,:); G05_data(start:stop,:);
G08_data(start:stop,:)];
training_W = [W00b_data(start:stop,:); W01_data(start:stop,:);
W01b_data(start:stop,:); W01c_data(start:stop,:); W02_data(start:stop,:);
W03_data(start:stop,:); W04_data(start:stop,:); W05_data(start:stop,:);
W501_data(start:stop,:); W501C3_data(start:stop,:); W501C6_data(start:stop,:);
W502_data(start:stop,:); W502C3_data(start:stop,:); W503_data(start:stop,:);];

%Merging all testing data
testing_O = [];
testing_G = [];
testing_W = [];

%Background noise
background_noise = [G00_data(start:stop,:); W00_data(start:stop,:)];
```

Appendix D

Preparing two-phase flow experiment data from dataset 2.

```
clc;
clear all;

%Load data
GO01 = load('GO_01.mat');
GO01_1secData = load('GO_01_1secData.mat');
GO01b = load('GO_01b.mat');
GO01b_1secData = load('GO_01b_1secData.mat');
GO02 = load('GO_02.mat');
GO02_1secData = load('GO_02_1secData.mat');
GO03 = load('GO_03.mat');
GO03_1secData = load('GO_03_1secData.mat');
GO04 = load('GO_04.mat');
GO04_1secData = load('GO_04_1secData.mat');
GO05 = load('GO_05.mat');
GO05_1secData = load('GO_05_1secData.mat');
GO06 = load('GO_06.mat');
GO06_1secData = load('GO_06_1secData.mat');
GO07 = load('GO_07.mat');
GO07_1secData = load('GO_07_1secData.mat');
GO08 = load('GO_08.mat');
GO08_1secData = load('GO_08_1secData.mat');
GO09 = load('GO_09.mat');
GO09_1secData = load('GO_09_1secData.mat');
GO10 = load('GO_10.mat');
GO10_1secData = load('GO_10_1secData.mat');
GO11 = load('GO_11.mat');
GO11_1secData = load('GO_11_1secData.mat');
GO12 = load('GO_12.mat');
GO12_1secData = load('GO_12_1secData.mat');
GW501 = load('GW_501.mat');
GW501_1secData = load('GW_501_1secData.mat');
GW502 = load('GW_502.mat');
GW502_1secData = load('GW_502_1secData.mat');
GW503 = load('GW_503.mat');
GW503_1secData = load('GW_503_1secData.mat');
GW503C1 = load('GW_503C1.mat');
GW503C1_1secData = load('GW_503C1_1secData.mat');
GW503C2 = load('GW_503C2.mat');
GW503C2_1secData = load('GW_503C2_1secData.mat');
GW504 = load('GW_504.mat');
GW504_1secData = load('GW_504_1secData.mat');
GW504C1 = load('GW_504C1.mat');
GW504C1_1secData = load('GW_504C1_1secData.mat');
GW504C2 = load('GW_504C2.mat');
GW504C2_1secData = load('GW_504C2_1secData.mat');
GW505 = load('GW_505.mat');
GW505_1secData = load('GW_505_1secData.mat');
OW00 = load('OW_00.mat');
OW00_1secData = load('OW_00_1secData.mat');
OW05 = load('OW_05.mat');
OW05_1secData = load('OW_05_1secData.mat');
```

```
OW05b = load('OW_05b.mat');
OW05b_1secData = load('OW_05b_1secData.mat');
OW06 = load('OW_06.mat');
OW06_1secData = load('OW_06_1secData.mat');
OW06b = load('OW_06b.mat');
OW06b_1secData = load('OW_06b_1secData.mat');
OW07 = load('OW_07.mat');
OW07_1secData = load('OW_07_1secData.mat');
OW08 = load('OW_08.mat');
OW08_1secData = load('OW_08_1secData.mat');
OW09 = load('OW_09.mat');
OW09_1secData = load('OW_09_1secData.mat');
OW10 = load('OW_10.mat');
OW10_1secData = load('OW_11_1secData.mat');
OW11 = load('OW_11.mat');
OW11_1secData = load('OW_11_1secData.mat');
OW12 = load('OW_12.mat');
OW12_1secData = load('OW_12_1secData.mat');
OW13 = load('OW_13.mat');
OW13_1secData = load('OW_13_1secData.mat');
OW14 = load('OW_14.mat');
OW14_1secData = load('OW_14_1secData.mat');
OW15 = load('OW_15.mat');
OW15_1secData = load('OW_15_1secData.mat');
OW16 = load('OW_16.mat');
OW16_1secData = load('OW_16_1secData.mat');
OW17 = load('OW_17.mat');
OW17_1secData = load('OW_17_1secData.mat');
OW18 = load('OW_18.mat');
OW18_1secData = load('OW_18_1secData.mat');
OW19 = load('OW_19.mat');
OW19_1secData = load('OW_19_1secData.mat');
OW20 = load('OW_20.mat');
OW20_1secData = load('OW_20_1secData.mat');
OW31 = load('OW_31.mat');
OW31_1secData = load('OW_31_1secData.mat');
OW501 = load('OW_501.mat');
OW501_1secData = load('OW_501_1secData.mat');
OW502 = load('OW_502.mat');
OW502_1secData = load('OW_502_1secData.mat');
OW503 = load('OW_503.mat');
OW503_1secData = load('OW_503_1secData.mat');
OW504 = load('OW_504.mat');
OW504_1secData = load('OW_504_1secData.mat');
OW505 = load('OW_505.mat');
OW505_1secData = load('OW_505_1secData.mat');
OW506 = load('OW_506.mat');
OW506_1secData = load('OW_506_1secData.mat');
OW507 = load('OW_507.mat');
OW507_1secData = load('OW_507_1secData.mat');
OW508 = load('OW_508.mat');
OW508_1secData = load('OW_508_1secData.mat');
OW509 = load('OW_509.mat');
OW509_1secData = load('OW_509_1secData.mat');
OW510 = load('OW_510.mat');
OW510_1secData = load('OW_510_1secData.mat');
OW511 = load('OW_511.mat');
```

```
OW511_1secData = load('OW_511_1secData.mat');
OW512 = load('OW_512.mat');
OW512_1secData = load('OW_512_1secData.mat');
OW513 = load('OW_513.mat');
OW513_1secData = load('OW_513_1secData.mat');
OW514 = load('OW_514.mat');
OW514_1secData = load('OW_514_1secData.mat');
OW515 = load('OW_515.mat');
OW515_1secData = load('OW_515_1secData.mat');
OW516 = load('OW_516.mat');
OW516_1secData = load('OW_516_1secData.mat');
OW517 = load('OW_517.mat');
OW517_1secData = load('OW_517_1secData.mat');

%Calculate average RMS with 1Hz samwple rate
N_acc = length(GO01.testPoint.datAcc.ch(1).data);
N = length(GO01_1secData.dataStruct.data);
dt = 1/GO01.testPoint.datAcc.dt;
time = N_acc/dt;

%Select samples in each dataset:
start = 100;
stop = 109;

for j = 1:4
    for i = 1:1:time
        GO01_acc_1hz_rms(i,j) = rms(GO01.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GO01b_acc_1hz_rms(i,j) = rms(GO01b.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GO02_acc_1hz_rms(i,j) = rms(GO02.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GO03_acc_1hz_rms(i,j) = rms(GO03.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GO04_acc_1hz_rms(i,j) = rms(GO04.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GO05_acc_1hz_rms(i,j) = rms(GO05.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GO06_acc_1hz_rms(i,j) = rms(GO06.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GO07_acc_1hz_rms(i,j) = rms(GO07.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GO08_acc_1hz_rms(i,j) = rms(GO08.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GO09_acc_1hz_rms(i,j) = rms(GO09.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GO10_acc_1hz_rms(i,j) = rms(GO10.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GO11_acc_1hz_rms(i,j) = rms(GO11.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GO12_acc_1hz_rms(i,j) = rms(GO12.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GW501_acc_1hz_rms(i,j) = rms(GW501.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GW502_acc_1hz_rms(i,j) = rms(GW502.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
```

```
        GW503_acc_1hz_rms(i,j) = rms(GW503.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GW503C1_acc_1hz_rms(i,j) = rms(GW503C1.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GW503C2_acc_1hz_rms(i,j) = rms(GW503C2.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GW504_acc_1hz_rms(i,j) = rms(GW504.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GW504C1_acc_1hz_rms(i,j) = rms(GW504C1.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GW504C2_acc_1hz_rms(i,j) = rms(GW504C2.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GW505_acc_1hz_rms(i,j) = rms(GW505.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW00_acc_1hz_rms(i,j) = rms(OW00.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW05_acc_1hz_rms(i,j) = rms(OW05.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW05b_acc_1hz_rms(i,j) = rms(OW05b.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW06_acc_1hz_rms(i,j) = rms(OW06.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW06b_acc_1hz_rms(i,j) = rms(OW06b.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW07_acc_1hz_rms(i,j) = rms(OW07.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW08_acc_1hz_rms(i,j) = rms(OW08.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW09_acc_1hz_rms(i,j) = rms(OW09.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW10_acc_1hz_rms(i,j) = rms(OW10.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW11_acc_1hz_rms(i,j) = rms(OW11.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW12_acc_1hz_rms(i,j) = rms(OW12.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW13_acc_1hz_rms(i,j) = rms(OW13.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW14_acc_1hz_rms(i,j) = rms(OW14.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW15_acc_1hz_rms(i,j) = rms(OW15.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW16_acc_1hz_rms(i,j) = rms(OW16.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW17_acc_1hz_rms(i,j) = rms(OW17.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW18_acc_1hz_rms(i,j) = rms(OW18.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW19_acc_1hz_rms(i,j) = rms(OW19.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW20_acc_1hz_rms(i,j) = rms(OW20.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW31_acc_1hz_rms(i,j) = rms(OW31.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW501_acc_1hz_rms(i,j) = rms(OW501.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
```

```
        OW502_acc_1hz_rms(i,j) = rms(OW502.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW503_acc_1hz_rms(i,j) = rms(OW503.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW504_acc_1hz_rms(i,j) = rms(OW504.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW505_acc_1hz_rms(i,j) = rms(OW505.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW506_acc_1hz_rms(i,j) = rms(OW506.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW507_acc_1hz_rms(i,j) = rms(OW507.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW508_acc_1hz_rms(i,j) = rms(OW508.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW509_acc_1hz_rms(i,j) = rms(OW509.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW510_acc_1hz_rms(i,j) = rms(OW510.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW511_acc_1hz_rms(i,j) = rms(OW511.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW512_acc_1hz_rms(i,j) = rms(OW512.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW513_acc_1hz_rms(i,j) = rms(OW513.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW514_acc_1hz_rms(i,j) = rms(OW514.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW515_acc_1hz_rms(i,j) = rms(OW515.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW516_acc_1hz_rms(i,j) = rms(OW516.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        OW517_acc_1hz_rms(i,j) = rms(OW517.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
    end
end

%Data matrices with 1Hz samplingrate
GO01_data = [GO01_1secData.dataStruct.data GO01_acc_1hz_rms
GO01.testPoint.gasRef.q*ones([N,1]) GO01.testPoint.watRef.q*ones([N,1])
GO01.testPoint.oilRef.q*ones([N,1])];
GO01b_data = [GO01b_1secData.dataStruct.data GO01b_acc_1hz_rms
GO01b.testPoint.gasRef.q*ones([301,1]) GO01b.testPoint.watRef.q*ones([N,1])
GO01b.testPoint.oilRef.q*ones([N,1])];
GO02_data = [GO02_1secData.dataStruct.data GO02_acc_1hz_rms
GO02.testPoint.gasRef.q*ones([N,1]) GO02.testPoint.watRef.q*ones([N,1])
GO02.testPoint.oilRef.q*ones([N,1])];
GO03_data = [GO03_1secData.dataStruct.data GO03_acc_1hz_rms
GO03.testPoint.gasRef.q*ones([N,1]) GO03.testPoint.watRef.q*ones([N,1])
GO03.testPoint.oilRef.q*ones([N,1])];
GO04_data = [GO04_1secData.dataStruct.data GO04_acc_1hz_rms
GO04.testPoint.gasRef.q*ones([N,1]) GO04.testPoint.watRef.q*ones([N,1])
GO04.testPoint.oilRef.q*ones([N,1])];
GO05_data = [GO05_1secData.dataStruct.data GO05_acc_1hz_rms
GO05.testPoint.gasRef.q*ones([N,1]) GO05.testPoint.watRef.q*ones([N,1])
GO05.testPoint.oilRef.q*ones([N,1])];
GO06_data = [GO06_1secData.dataStruct.data GO06_acc_1hz_rms
GO06.testPoint.gasRef.q*ones([N,1]) GO06.testPoint.watRef.q*ones([N,1])
GO06.testPoint.oilRef.q*ones([N,1])];
```

```
GO07_data = [GO07_1secData.dataStruct.data GO07_acc_1hz_rms
GO07.testPoint.gasRef.q*ones([N,1]) GO07.testPoint.watRef.q*ones([N,1])
GO07.testPoint.oilRef.q*ones([N,1])];
GO08_data = [GO08_1secData.dataStruct.data GO08_acc_1hz_rms
GO08.testPoint.gasRef.q*ones([N,1]) GO08.testPoint.watRef.q*ones([N,1])
GO08.testPoint.oilRef.q*ones([N,1])];
GO09_data = [GO09_1secData.dataStruct.data GO09_acc_1hz_rms
GO09.testPoint.gasRef.q*ones([N,1]) GO09.testPoint.watRef.q*ones([N,1])
GO09.testPoint.oilRef.q*ones([N,1])];
GO10_data = [GO10_1secData.dataStruct.data GO10_acc_1hz_rms
GO10.testPoint.gasRef.q*ones([N,1]) GO10.testPoint.watRef.q*ones([N,1])
GO10.testPoint.oilRef.q*ones([N,1])];
GO11_data = [GO11_1secData.dataStruct.data GO11_acc_1hz_rms
GO11.testPoint.gasRef.q*ones([N,1]) GO11.testPoint.watRef.q*ones([N,1])
GO11.testPoint.oilRef.q*ones([N,1])];
GO12_data = [GO12_1secData.dataStruct.data GO12_acc_1hz_rms
GO12.testPoint.gasRef.q*ones([N,1]) GO12.testPoint.watRef.q*ones([N,1])
GO12.testPoint.oilRef.q*ones([N,1])];
GW501_data = [GW501_1secData.dataStruct.data GW501_acc_1hz_rms
GW501.testPoint.gasRef.q*ones([N,1]) GW501.testPoint.watRef.q*ones([N,1])
GW501.testPoint.oilRef.q*ones([N,1])];
GW502_data = [GW502_1secData.dataStruct.data GW502_acc_1hz_rms
GW502.testPoint.gasRef.q*ones([N,1]) GW502.testPoint.watRef.q*ones([N,1])
GW502.testPoint.oilRef.q*ones([N,1])];
GW503_data = [GW503_1secData.dataStruct.data GW503_acc_1hz_rms
GW503.testPoint.gasRef.q*ones([N,1]) GW503.testPoint.watRef.q*ones([N,1])
GW503.testPoint.oilRef.q*ones([N,1])];
GW503C1_data = [GW503C1_1secData.dataStruct.data GW503C1_acc_1hz_rms
GW503C1.testPoint.gasRef.q*ones([N,1]) GW503C1.testPoint.watRef.q*ones([N,1])
GW503C1.testPoint.oilRef.q*ones([N,1])];
GW503C2_data = [GW503C2_1secData.dataStruct.data GW503C2_acc_1hz_rms
GW503C2.testPoint.gasRef.q*ones([N,1]) GW503C2.testPoint.watRef.q*ones([N,1])
GW503C2.testPoint.oilRef.q*ones([N,1])];
GW504_data = [GW504_1secData.dataStruct.data GW504_acc_1hz_rms
GW501.testPoint.gasRef.q*ones([N,1]) GW504.testPoint.watRef.q*ones([N,1])
GW504.testPoint.oilRef.q*ones([N,1])];
GW504C1_data = [GW504C1_1secData.dataStruct.data GW504C1_acc_1hz_rms
GW504C1.testPoint.gasRef.q*ones([N,1]) GW504C1.testPoint.watRef.q*ones([N,1])
GW504C1.testPoint.oilRef.q*ones([N,1])];
GW504C2_data = [GW504C2_1secData.dataStruct.data GW504C2_acc_1hz_rms
GW504C2.testPoint.gasRef.q*ones([N,1]) GW504C2.testPoint.watRef.q*ones([N,1])
GW504C2.testPoint.oilRef.q*ones([N,1])];
GW505_data = [GW505_1secData.dataStruct.data GW505_acc_1hz_rms
GW505.testPoint.gasRef.q*ones([N,1]) GW505.testPoint.watRef.q*ones([N,1])
GW505.testPoint.oilRef.q*ones([N,1])];
OW00_data = [OW00_1secData.dataStruct.data OW00_acc_1hz_rms
OW00.testPoint.gasRef.q*ones([N,1]) OW00.testPoint.watRef.q*ones([N,1])
OW00.testPoint.oilRef.q*ones([N,1])];
OW05_data = [OW05_1secData.dataStruct.data OW05_acc_1hz_rms
OW05.testPoint.gasRef.q*ones([N,1]) OW05.testPoint.watRef.q*ones([N,1])
OW05.testPoint.oilRef.q*ones([N,1])];
OW05b_data = [OW05b_1secData.dataStruct.data OW05b_acc_1hz_rms
OW05b.testPoint.gasRef.q*ones([N,1]) OW05b.testPoint.watRef.q*ones([N,1])
OW05b.testPoint.oilRef.q*ones([N,1])];
OW06_data = [OW06_1secData.dataStruct.data OW06_acc_1hz_rms
OW06.testPoint.gasRef.q*ones([N,1]) OW06.testPoint.watRef.q*ones([N,1])
OW06.testPoint.oilRef.q*ones([N,1])];
```

```
OW06b_data = [OW06b_1secData.dataStruct.data OW06b_acc_1hz_rms
OW06b.testPoint.gasRef.q*ones([N,1]) OW06b.testPoint.watRef.q*ones([N,1])
OW06b.testPoint.oilRef.q*ones([N,1])];
OW07_data = [OW07_1secData.dataStruct.data OW07_acc_1hz_rms
OW07.testPoint.gasRef.q*ones([N,1]) OW07.testPoint.watRef.q*ones([N,1])
OW07.testPoint.oilRef.q*ones([N,1])];
OW08_data = [OW08_1secData.dataStruct.data OW08_acc_1hz_rms
OW08.testPoint.gasRef.q*ones([N,1]) OW08.testPoint.watRef.q*ones([N,1])
OW08.testPoint.oilRef.q*ones([N,1])];
OW09_data = [OW09_1secData.dataStruct.data OW09_acc_1hz_rms
OW09.testPoint.gasRef.q*ones([N,1]) OW09.testPoint.watRef.q*ones([N,1])
OW09.testPoint.oilRef.q*ones([N,1])];
OW10_data = [OW10_1secData.dataStruct.data OW10_acc_1hz_rms
OW10.testPoint.gasRef.q*ones([N,1]) OW10.testPoint.watRef.q*ones([N,1])
OW10.testPoint.oilRef.q*ones([N,1])];
OW11_data = [OW11_1secData.dataStruct.data OW11_acc_1hz_rms
OW11.testPoint.gasRef.q*ones([N,1]) OW11.testPoint.watRef.q*ones([N,1])
OW11.testPoint.oilRef.q*ones([N,1])];
OW12_data = [OW12_1secData.dataStruct.data OW12_acc_1hz_rms
OW12.testPoint.gasRef.q*ones([N,1]) OW12.testPoint.watRef.q*ones([N,1])
OW12.testPoint.oilRef.q*ones([N,1])];
OW13_data = [OW13_1secData.dataStruct.data OW13_acc_1hz_rms
OW13.testPoint.gasRef.q*ones([N,1]) OW13.testPoint.watRef.q*ones([N,1])
OW13.testPoint.oilRef.q*ones([N,1])];
OW14_data = [OW14_1secData.dataStruct.data OW14_acc_1hz_rms
OW14.testPoint.gasRef.q*ones([N,1]) OW14.testPoint.watRef.q*ones([N,1])
OW14.testPoint.oilRef.q*ones([N,1])];
OW15_data = [OW15_1secData.dataStruct.data OW15_acc_1hz_rms
OW15.testPoint.gasRef.q*ones([N,1]) OW15.testPoint.watRef.q*ones([N,1])
OW15.testPoint.oilRef.q*ones([N,1])];
OW16_data = [OW16_1secData.dataStruct.data OW16_acc_1hz_rms
OW16.testPoint.gasRef.q*ones([N,1]) OW16.testPoint.watRef.q*ones([N,1])
OW16.testPoint.oilRef.q*ones([N,1])];
OW17_data = [OW17_1secData.dataStruct.data OW17_acc_1hz_rms
OW17.testPoint.gasRef.q*ones([N,1]) OW17.testPoint.watRef.q*ones([N,1])
OW17.testPoint.oilRef.q*ones([N,1])];
OW18_data = [OW18_1secData.dataStruct.data OW18_acc_1hz_rms
OW18.testPoint.gasRef.q*ones([N,1]) OW18.testPoint.watRef.q*ones([N,1])
OW18.testPoint.oilRef.q*ones([N,1])];
OW19_data = [OW19_1secData.dataStruct.data OW19_acc_1hz_rms
OW19.testPoint.gasRef.q*ones([N,1]) OW19.testPoint.watRef.q*ones([N,1])
OW19.testPoint.oilRef.q*ones([N,1])];
OW20_data = [OW20_1secData.dataStruct.data OW20_acc_1hz_rms
OW20.testPoint.gasRef.q*ones([N,1]) OW20.testPoint.watRef.q*ones([N,1])
OW20.testPoint.oilRef.q*ones([N,1])];
OW31_data = [OW31_1secData.dataStruct.data OW31_acc_1hz_rms
OW31.testPoint.gasRef.q*ones([N,1]) OW31.testPoint.watRef.q*ones([N,1])
OW31.testPoint.oilRef.q*ones([N,1])];
OW501_data = [OW501_1secData.dataStruct.data OW501_acc_1hz_rms
OW501.testPoint.gasRef.q*ones([N,1]) OW501.testPoint.watRef.q*ones([N,1])
OW501.testPoint.oilRef.q*ones([N,1])];
OW502_data = [OW502_1secData.dataStruct.data OW502_acc_1hz_rms
OW502.testPoint.gasRef.q*ones([N,1]) OW502.testPoint.watRef.q*ones([N,1])
OW502.testPoint.oilRef.q*ones([N,1])];
OW503_data = [OW503_1secData.dataStruct.data OW503_acc_1hz_rms
OW503.testPoint.gasRef.q*ones([N,1]) OW503.testPoint.watRef.q*ones([N,1])
OW503.testPoint.oilRef.q*ones([N,1])];
```

```
OW504_data = [OW504_1secData.dataStruct.data OW504_acc_1hz_rms
OW504.testPoint.gasRef.q*ones([N,1]) OW504.testPoint.watRef.q*ones([N,1])
OW504.testPoint.oilRef.q*ones([N,1])];
OW505_data = [OW505_1secData.dataStruct.data OW505_acc_1hz_rms
OW505.testPoint.gasRef.q*ones([N,1]) OW505.testPoint.watRef.q*ones([N,1])
OW505.testPoint.oilRef.q*ones([N,1])];
OW506_data = [OW506_1secData.dataStruct.data OW506_acc_1hz_rms
OW506.testPoint.gasRef.q*ones([N,1]) OW506.testPoint.watRef.q*ones([N,1])
OW506.testPoint.oilRef.q*ones([N,1])];
OW507_data = [OW507_1secData.dataStruct.data OW507_acc_1hz_rms
OW507.testPoint.gasRef.q*ones([N,1]) OW507.testPoint.watRef.q*ones([N,1])
OW507.testPoint.oilRef.q*ones([N,1])];
OW508_data = [OW508_1secData.dataStruct.data OW508_acc_1hz_rms
OW508.testPoint.gasRef.q*ones([N,1]) OW508.testPoint.watRef.q*ones([N,1])
OW508.testPoint.oilRef.q*ones([N,1])];
OW509_data = [OW509_1secData.dataStruct.data OW509_acc_1hz_rms
OW509.testPoint.gasRef.q*ones([N,1]) OW509.testPoint.watRef.q*ones([N,1])
OW509.testPoint.oilRef.q*ones([N,1])];
OW510_data = [OW510_1secData.dataStruct.data OW510_acc_1hz_rms
OW510.testPoint.gasRef.q*ones([N,1]) OW510.testPoint.watRef.q*ones([N,1])
OW510.testPoint.oilRef.q*ones([N,1])];
OW511_data = [OW511_1secData.dataStruct.data OW511_acc_1hz_rms
OW511.testPoint.gasRef.q*ones([N,1]) OW511.testPoint.watRef.q*ones([N,1])
OW511.testPoint.oilRef.q*ones([N,1])];
OW512_data = [OW512_1secData.dataStruct.data OW512_acc_1hz_rms
OW512.testPoint.gasRef.q*ones([N,1]) OW512.testPoint.watRef.q*ones([N,1])
OW512.testPoint.oilRef.q*ones([N,1])];
OW513_data = [OW513_1secData.dataStruct.data OW513_acc_1hz_rms
OW513.testPoint.gasRef.q*ones([N,1]) OW513.testPoint.watRef.q*ones([N,1])
OW513.testPoint.oilRef.q*ones([N,1])];
OW514_data = [OW514_1secData.dataStruct.data OW514_acc_1hz_rms
OW514.testPoint.gasRef.q*ones([N,1]) OW514.testPoint.watRef.q*ones([N,1])
OW514.testPoint.oilRef.q*ones([N,1])];
OW515_data = [OW515_1secData.dataStruct.data OW515_acc_1hz_rms
OW515.testPoint.gasRef.q*ones([N,1]) OW515.testPoint.watRef.q*ones([N,1])
OW515.testPoint.oilRef.q*ones([N,1])];
OW516_data = [OW516_1secData.dataStruct.data OW516_acc_1hz_rms
OW516.testPoint.gasRef.q*ones([N,1]) OW516.testPoint.watRef.q*ones([N,1])
OW516.testPoint.oilRef.q*ones([N,1])];
OW517_data = [OW517_1secData.dataStruct.data OW517_acc_1hz_rms
OW517.testPoint.gasRef.q*ones([N,1]) OW517.testPoint.watRef.q*ones([N,1])
OW517.testPoint.oilRef.q*ones([N,1])];

%Merging all training data
training_GO = [GO01_data(start:stop,:); GO01b_data(start:stop,:);
GO03_data(start:stop,:); GO04_data(start:stop,:); GO05_data(start:stop,:);
GO06_data(start:stop,:); GO07_data(start:stop,:); GO08_data(start:stop,:);
GO09_data(start:stop,:); GO10_data(start:stop,:); GO11_data(start:stop,:);
GO12_data(start:stop,:)];
training_GW = [GW501_data(start:stop,:); GW502_data(start:stop,:);
GW503_data(start:stop,:); GW504_data(start:stop,:); GW504C1_data(start:stop,:);
GW505_data(start:stop,:)];
training_OW = [OW05_data(start:stop,:); OW05b_data(start:stop,:);
OW06_data(start:stop,:); OW06b_data(start:stop,:); OW07_data(start:stop,:);
OW08_data(start:stop,:); OW09_data(start:stop,:); OW10_data(start:stop,:);
OW11_data(start:stop,:); OW12_data(start:stop,:); OW13_data(start:stop,:);
OW14_data(start:stop,:); OW15_data(start:stop,:); OW16_data(start:stop,:);
```

```
OW17_data(start:stop,:); OW18_data(start:stop,:); OW19_data(start:stop,:);
OW31_data(start:stop,:); OW501_data(start:stop,:); OW502_data(start:stop,:);
OW503_data(start:stop,:); GW503C1_data(start:stop,:); GW503C2_data(start:stop,:);
OW504_data(start:stop,:); OW505_data(start:stop,:); OW506_data(start:stop,:);
OW507_data(start:stop,:); OW508_data(start:stop,:); OW509_data(start:stop,:);
OW510_data(start:stop,:); OW511_data(start:stop,:); OW512_data(start:stop,:);
OW513_data(start:stop,:); OW514_data(start:stop,:); OW515_data(start:stop,:);
OW516_data(start:stop,:); OW517_data(start:stop,:)];

%Merging all testing data
testing_GO = [GO02_data(start:stop,:)];
testing_GW = [GW504C2_data(start:stop,:)];
testing_OW = [OW20_data(start:stop,:)];

%Background noise
background_noise = [OW00_data(start:stop,:)];
```

Appendix E

Preparing the second part of two-phase flow experiment data from dataset 2.

```
clc;
clear all;

% Load data
GOW00 = load('GOW_00.mat');
GOW00_1secData = load('GOW_00_1secData.mat');
GOW01 = load('GOW_01.mat');
GOW01_1secData = load('GOW_01_1secData.mat');
GOW01b = load('GOW_01b.mat');
GOW01b_1secData = load('GOW_01b_1secData.mat');
GOW01c = load('GOW_01c.mat');
GOW01c_1secData = load('GOW_01c_1secData.mat');
GOW02 = load('GOW_02.mat');
GOW02_1secData = load('GOW_02_1secData.mat');
GOW03 = load('GOW_03.mat');
GOW03_1secData = load('GOW_03_1secData.mat');
GOW04 = load('GOW_04.mat');
GOW04_1secData = load('GOW_04_1secData.mat');
GOW05 = load('GOW_05.mat');
GOW05_1secData = load('GOW_05_1secData.mat');
GOW06 = load('GOW_06.mat');
GOW06_1secData = load('GOW_06_1secData.mat');
GOW06b = load('GOW_06b.mat');
GOW06b_1secData = load('GOW_06b_1secData.mat');
GOW07 = load('GOW_07.mat');
GOW07_1secData = load('GOW_07_1secData.mat');
GOW08 = load('GOW_08.mat');
GOW08_1secData = load('GOW_08_1secData.mat');
GOW09 = load('GOW_09.mat');
GOW09_1secData = load('GOW_09_1secData.mat');
GOW10 = load('GOW_10.mat');
GOW10_1secData = load('GOW_10_1secData.mat');
GOW11 = load('GOW_11.mat');
GOW11_1secData = load('GOW_11_1secData.mat');
GOW25 = load('GOW_25.mat');
GOW25_1secData = load('GOW_25_1secData.mat');
GOW26 = load('GOW_26.mat');
GOW26_1secData = load('GOW_26_1secData.mat');
GOW27 = load('GOW_27.mat');
GOW27_1secData = load('GOW_27_1secData.mat');
GOW28 = load('GOW_28.mat');
GOW28_1secData = load('GOW_28_1secData.mat');
GOW29 = load('GOW_29.mat');
GOW29_1secData = load('GOW_29_1secData.mat');
GOW30 = load('GOW_30.mat');
GOW30_1secData = load('GOW_30_1secData.mat');
GOW31 = load('GOW_31.mat');
GOW31_1secData = load('GOW_31_1secData.mat');
GOW32 = load('GOW_32.mat');
GOW32_1secData = load('GOW_32_1secData.mat');
GOW33 = load('GOW_33.mat');
GOW33_1secData = load('GOW_33_1secData.mat');
```

```
GOW34 = load('GOW_34.mat');
GOW34_1secData = load('GOW_34_1secData.mat');
GOW35 = load('GOW_35.mat');
GOW35_1secData = load('GOW_35_1secData.mat');
GOW36 = load('GOW_36.mat');

GOW36_1secData = load('GOW_36_1secData.mat');
GOW37 = load('GOW_37.mat');
GOW37_1secData = load('GOW_37_1secData.mat');
GOW38 = load('GOW_38.mat');
GOW38_1secData = load('GOW_38_1secData.mat');
GOW39 = load('GOW_39.mat');
GOW39_1secData = load('GOW_39_1secData.mat');
GOW40 = load('GOW_40.mat');
GOW40_1secData = load('GOW_40_1secData.mat');
GOW41 = load('GOW_41.mat');
GOW41_1secData = load('GOW_41_1secData.mat');
GOW42 = load('GOW_42.mat');
GOW42_1secData = load('GOW_42_1secData.mat');
GOW43 = load('GOW_43.mat');
GOW43_1secData = load('GOW_43_1secData.mat');
GOW44 = load('GOW_44.mat');
GOW44_1secData = load('GOW_44_1secData.mat');
GOW45 = load('GOW_45.mat');
GOW45_1secData = load('GOW_45_1secData.mat');
GOW46 = load('GOW_46.mat');
GOW46_1secData = load('GOW_46_1secData.mat');
GOW47 = load('GOW_47.mat');
GOW47_1secData = load('GOW_47_1secData.mat');
GOW48 = load('GOW_48.mat');
GOW48_1secData = load('GOW_48_1secData.mat');
GOW49 = load('GOW_49.mat');
GOW49_1secData = load('GOW_49_1secData.mat');
GOW50 = load('GOW_50.mat');
GOW50_1secData = load('GOW_50_1secData.mat');
GOW51 = load('GOW_51.mat');
GOW51_1secData = load('GOW_51_1secData.mat');
GOW52 = load('GOW_52.mat');
GOW52_1secData = load('GOW_52_1secData.mat');
GOW53 = load('GOW_53.mat');
GOW53_1secData = load('GOW_53_1secData.mat');
GOW54 = load('GOW_54.mat');
GOW54_1secData = load('GOW_54_1secData.mat');
GOW55 = load('GOW_55.mat');
GOW55_1secData = load('GOW_55_1secData.mat');
GOW56 = load('GOW_56.mat');
GOW56_1secData = load('GOW_56_1secData.mat');
GOW60 = load('GOW_60.mat');
GOW60_1secData = load('GOW_60_1secData.mat');
GOW61 = load('GOW_61.mat');
GOW61_1secData = load('GOW_61_1secData.mat');
GOW62 = load('GOW_62.mat');
GOW62_1secData = load('GOW_62_1secData.mat');
GOW63 = load('GOW_63.mat');
GOW63_1secData = load('GOW_63_1secData.mat');
GOW64 = load('GOW_64.mat');
GOW64_1secData = load('GOW_64_1secData.mat');
```

```
GOW69 = load('GOW_69.mat');
GOW69_1secData = load('GOW_69_1secData.mat');
GOW70 = load('GOW_70.mat');
GOW70_1secData = load('GOW_70_1secData.mat');
GOW71 = load('GOW_71.mat');
GOW71_1secData = load('GOW_71_1secData.mat');
GOW74 = load('GOW_74.mat');
GOW74_1secData = load('GOW_74_1secData.mat');
GOW75 = load('GOW_75.mat');
GOW75_1secData = load('GOW_75_1secData.mat');
GOW76 = load('GOW_76.mat');
GOW76_1secData = load('GOW_76_1secData.mat');
GOW77 = load('GOW_77.mat');
GOW77_1secData = load('GOW_77_1secData.mat');
GOW78 = load('GOW_78.mat');
GOW78_1secData = load('GOW_78_1secData.mat');
GOW79 = load('GOW_79.mat');
GOW79_1secData = load('GOW_79_1secData.mat');
GOW80 = load('GOW_80.mat');
GOW80_1secData = load('GOW_80_1secData.mat');
GOW82 = load('GOW_82.mat');
GOW82_1secData = load('GOW_82_1secData.mat');
GOW83 = load('GOW_83.mat');
GOW83_1secData = load('GOW_83_1secData.mat');
GOW84 = load('GOW_84.mat');
GOW84_1secData = load('GOW_84_1secData.mat');
GOW85 = load('GOW_85.mat');
GOW85_1secData = load('GOW_85_1secData.mat');
GOW86 = load('GOW_86.mat');
GOW86_1secData = load('GOW_86_1secData.mat');
GOW87 = load('GOW_87.mat');
GOW87_1secData = load('GOW_87_1secData.mat');
GOW88 = load('GOW_88.mat');
GOW88_1secData = load('GOW_88_1secData.mat');
GOW89 = load('GOW_89.mat');
GOW89_1secData = load('GOW_89_1secData.mat');
GOW90 = load('GOW_90.mat');
GOW90_1secData = load('GOW_90_1secData.mat');
GOW91 = load('GOW_91.mat');
GOW91_1secData = load('GOW_91_1secData.mat');
GOW92 = load('GOW_92.mat');
GOW92_1secData = load('GOW_92_1secData.mat');
GOW93 = load('GOW_93.mat');
GOW93_1secData = load('GOW_93_1secData.mat');
GOW94 = load('GOW_94.mat');
GOW94_1secData = load('GOW_94_1secData.mat');
GOW95 = load('GOW_95.mat');
GOW95_1secData = load('GOW_95_1secData.mat');
GOW96 = load('GOW_96.mat');
GOW96_1secData = load('GOW_96_1secData.mat');
GOW97 = load('GOW_97.mat');
GOW97_1secData = load('GOW_97_1secData.mat');
GOW100b = load('GOW_100b.mat');
GOW100b_1secData = load('GOW_100b_1secData.mat');
GOW118 = load('GOW_118.mat');
GOW118_1secData = load('GOW_118_1secData.mat');
GOW119 = load('GOW_119.mat');
```

```
GOW119_1secData = load('GOW_118_1secData.mat');
GOW120 = load('GOW_120.mat');
GOW120_1secData = load('GOW_120_1secData.mat');
GOW121 = load('GOW_121.mat');
GOW121_1secData = load('GOW_121_1secData.mat');
GOW122 = load('GOW_122.mat');
GOW122_1secData = load('GOW_122_1secData.mat');
GOW123 = load('GOW_123.mat');
GOW123_1secData = load('GOW_123_1secData.mat');
GOW124 = load('GOW_124.mat');
GOW124_1secData = load('GOW_124_1secData.mat');
GOW125 = load('GOW_125.mat');
GOW125_1secData = load('GOW_125_1secData.mat');
GOW126 = load('GOW_126.mat');
GOW126_1secData = load('GOW_126_1secData.mat');
GOW127 = load('GOW_127.mat');
GOW127_1secData = load('GOW_127_1secData.mat');
GOW142 = load('GOW_142.mat');
GOW142_1secData = load('GOW_142_1secData.mat');
GOW143 = load('GOW_143.mat');
GOW143_1secData = load('GOW_143_1secData.mat');
GOW144 = load('GOW_144.mat');
GOW144_1secData = load('GOW_144_1secData.mat');
GOW145 = load('GOW_145.mat');
GOW145_1secData = load('GOW_145_1secData.mat');
GOW145b = load('GOW_145b.mat');
GOW145b_1secData = load('GOW_145b_1secData.mat');
GOW146 = load('GOW_146.mat');
GOW146_1secData = load('GOW_146_1secData.mat');
GOW147 = load('GOW_147.mat');
GOW147_1secData = load('GOW_147_1secData.mat');
GOW148 = load('GOW_148.mat');
GOW148_1secData = load('GOW_148_1secData.mat');
GOW149 = load('GOW_149.mat');
GOW149_1secData = load('GOW_149_1secData.mat');
GOW150 = load('GOW_150.mat');
GOW150_1secData = load('GOW_150_1secData.mat');
GOW150b = load('GOW_150b.mat');
GOW150b_1secData = load('GOW_150b_1secData.mat');
GOW151 = load('GOW_151.mat');
GOW151_1secData = load('GOW_151_1secData.mat');
GOW152 = load('GOW_152.mat');
GOW152_1secData = load('GOW_152_1secData.mat');
GOW153 = load('GOW_153.mat');
GOW153_1secData = load('GOW_153_1secData.mat');
GOW154 = load('GOW_154.mat');
GOW154_1secData = load('GOW_154_1secData.mat');
GOW155 = load('GOW_155.mat');
GOW155_1secData = load('GOW_155_1secData.mat');
GOW156 = load('GOW_156.mat');
GOW156_1secData = load('GOW_156_1secData.mat');
GOW157 = load('GOW_157.mat');
GOW157_1secData = load('GOW_157_1secData.mat');
GOW157b = load('GOW_157b.mat');
GOW157b_1secData = load('GOW_157b_1secData.mat');
GOW158 = load('GOW_158.mat');
GOW158_1secData = load('GOW_158_1secData.mat');
```

```
GOW159 = load('GOW_159.mat');
GOW159_1secData = load('GOW_159_1secData.mat');
GOW161 = load('GOW_161.mat');
GOW161_1secData = load('GOW_161_1secData.mat');
GOW162 = load('GOW_162.mat');
GOW162_1secData = load('GOW_162_1secData.mat');
GOW163b = load('GOW_163b.mat');
GOW163b_1secData = load('GOW_163b_1secData.mat');
GOW164 = load('GOW_164.mat');
GOW164_1secData = load('GOW_164_1secData.mat');
GOW165 = load('GOW_165.mat');
GOW165_1secData = load('GOW_165_1secData.mat');
GOW166 = load('GOW_166.mat');
GOW166_1secData = load('GOW_166_1secData.mat');
GOW167 = load('GOW_167.mat');
GOW167_1secData = load('GOW_167_1secData.mat');
GOW168 = load('GOW_168.mat');
GOW168_1secData = load('GOW_168_1secData.mat');
GOW169 = load('GOW_169.mat');
GOW169_1secData = load('GOW_169_1secData.mat');
GOW170 = load('GOW_170.mat');
GOW170_1secData = load('GOW_170_1secData.mat');

%Calculate average RMS with 1Hz samwple rate
N_acc = length(GOW40.testPoint.datAcc.ch(1).data);
N = length(GOW40_1secData.dataStruct.data);
dt = 1/GOW40.testPoint.datAcc.dt;
time = N_acc/dt;

%Select samples in each dataset:
start = 100;
stop = 109;

for j = 1:4
    for i = 1:1:time
        GOW00_acc_1hz_rms(i,j) = rms(GOW00.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW01_acc_1hz_rms(i,j) = rms(GOW01.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW01b_acc_1hz_rms(i,j) = rms(GOW01b.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW01c_acc_1hz_rms(i,j) = rms(GOW01c.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW02_acc_1hz_rms(i,j) = rms(GOW02.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW03_acc_1hz_rms(i,j) = rms(GOW03.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW04_acc_1hz_rms(i,j) = rms(GOW04.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW05_acc_1hz_rms(i,j) = rms(GOW05.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW06_acc_1hz_rms(i,j) = rms(GOW06.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW06b_acc_1hz_rms(i,j) = rms(GOW06b.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW07_acc_1hz_rms(i,j) = rms(GOW07.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
```

```
        GOW08_acc_1hz_rms(i,j) = rms(GOW08.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW09_acc_1hz_rms(i,j) = rms(GOW09.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW10_acc_1hz_rms(i,j) = rms(GOW10.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW11_acc_1hz_rms(i,j) = rms(GOW11.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW25_acc_1hz_rms(i,j) = rms(GOW25.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW26_acc_1hz_rms(i,j) = rms(GOW26.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW27_acc_1hz_rms(i,j) = rms(GOW27.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW28_acc_1hz_rms(i,j) = rms(GOW28.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW29_acc_1hz_rms(i,j) = rms(GOW29.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW30_acc_1hz_rms(i,j) = rms(GOW30.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW31_acc_1hz_rms(i,j) = rms(GOW31.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW32_acc_1hz_rms(i,j) = rms(GOW32.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW33_acc_1hz_rms(i,j) = rms(GOW33.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW34_acc_1hz_rms(i,j) = rms(GOW34.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW35_acc_1hz_rms(i,j) = rms(GOW35.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW36_acc_1hz_rms(i,j) = rms(GOW36.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW37_acc_1hz_rms(i,j) = rms(GOW37.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW38_acc_1hz_rms(i,j) = rms(GOW38.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW39_acc_1hz_rms(i,j) = rms(GOW39.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW40_acc_1hz_rms(i,j) = rms(GOW40.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW41_acc_1hz_rms(i,j) = rms(GOW41.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW42_acc_1hz_rms(i,j) = rms(GOW42.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW43_acc_1hz_rms(i,j) = rms(GOW43.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW44_acc_1hz_rms(i,j) = rms(GOW44.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW45_acc_1hz_rms(i,j) = rms(GOW45.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW46_acc_1hz_rms(i,j) = rms(GOW46.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW47_acc_1hz_rms(i,j) = rms(GOW47.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW48_acc_1hz_rms(i,j) = rms(GOW48.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
```

```
        GOW49_acc_1hz_rms(i,j) = rms(GOW49.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW50_acc_1hz_rms(i,j) = rms(GOW50.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW51_acc_1hz_rms(i,j) = rms(GOW51.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW52_acc_1hz_rms(i,j) = rms(GOW52.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW53_acc_1hz_rms(i,j) = rms(GOW53.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW54_acc_1hz_rms(i,j) = rms(GOW54.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW55_acc_1hz_rms(i,j) = rms(GOW55.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW56_acc_1hz_rms(i,j) = rms(GOW56.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW60_acc_1hz_rms(i,j) = rms(GOW60.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW61_acc_1hz_rms(i,j) = rms(GOW61.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW62_acc_1hz_rms(i,j) = rms(GOW62.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW63_acc_1hz_rms(i,j) = rms(GOW63.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW64_acc_1hz_rms(i,j) = rms(GOW64.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW69_acc_1hz_rms(i,j) = rms(GOW69.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW70_acc_1hz_rms(i,j) = rms(GOW70.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW71_acc_1hz_rms(i,j) = rms(GOW71.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW74_acc_1hz_rms(i,j) = rms(GOW74.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW75_acc_1hz_rms(i,j) = rms(GOW75.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW76_acc_1hz_rms(i,j) = rms(GOW76.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW77_acc_1hz_rms(i,j) = rms(GOW77.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW78_acc_1hz_rms(i,j) = rms(GOW78.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW79_acc_1hz_rms(i,j) = rms(GOW79.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW80_acc_1hz_rms(i,j) = rms(GOW80.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW82_acc_1hz_rms(i,j) = rms(GOW82.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW83_acc_1hz_rms(i,j) = rms(GOW83.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW84_acc_1hz_rms(i,j) = rms(GOW84.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW85_acc_1hz_rms(i,j) = rms(GOW85.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW86_acc_1hz_rms(i,j) = rms(GOW86.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
```

```
        GOW87_acc_1hz_rms(i,j) = rms(GOW87.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW88_acc_1hz_rms(i,j) = rms(GOW88.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW89_acc_1hz_rms(i,j) = rms(GOW89.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW90_acc_1hz_rms(i,j) = rms(GOW90.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW91_acc_1hz_rms(i,j) = rms(GOW91.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW92_acc_1hz_rms(i,j) = rms(GOW92.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW93_acc_1hz_rms(i,j) = rms(GOW93.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW94_acc_1hz_rms(i,j) = rms(GOW94.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW95_acc_1hz_rms(i,j) = rms(GOW95.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW96_acc_1hz_rms(i,j) = rms(GOW96.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW97_acc_1hz_rms(i,j) = rms(GOW97.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW100b_acc_1hz_rms(i,j) = rms(GOW100b.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW118_acc_1hz_rms(i,j) = rms(GOW118.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW119_acc_1hz_rms(i,j) = rms(GOW119.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW120_acc_1hz_rms(i,j) = rms(GOW120.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW121_acc_1hz_rms(i,j) = rms(GOW121.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW122_acc_1hz_rms(i,j) = rms(GOW122.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW123_acc_1hz_rms(i,j) = rms(GOW123.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW124_acc_1hz_rms(i,j) = rms(GOW124.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW125_acc_1hz_rms(i,j) = rms(GOW125.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW126_acc_1hz_rms(i,j) = rms(GOW126.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW127_acc_1hz_rms(i,j) = rms(GOW127.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW142_acc_1hz_rms(i,j) = rms(GOW142.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW143_acc_1hz_rms(i,j) = rms(GOW143.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW144_acc_1hz_rms(i,j) = rms(GOW144.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW145_acc_1hz_rms(i,j) = rms(GOW145.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW145b_acc_1hz_rms(i,j) = rms(GOW145b.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW146_acc_1hz_rms(i,j) = rms(GOW146.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
```

```
        GOW147_acc_1hz_rms(i,j) = rms(GOW147.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW148_acc_1hz_rms(i,j) = rms(GOW148.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW149_acc_1hz_rms(i,j) = rms(GOW149.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW150_acc_1hz_rms(i,j) = rms(GOW150.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW150b_acc_1hz_rms(i,j) = rms(GOW150b.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW151_acc_1hz_rms(i,j) = rms(GOW151.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW152_acc_1hz_rms(i,j) = rms(GOW152.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW153_acc_1hz_rms(i,j) = rms(GOW153.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW154_acc_1hz_rms(i,j) = rms(GOW154.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW155_acc_1hz_rms(i,j) = rms(GOW155.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW156_acc_1hz_rms(i,j) = rms(GOW156.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW157_acc_1hz_rms(i,j) = rms(GOW157.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW157b_acc_1hz_rms(i,j) = rms(GOW157b.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW158_acc_1hz_rms(i,j) = rms(GOW158.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW159_acc_1hz_rms(i,j) = rms(GOW159.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW161_acc_1hz_rms(i,j) = rms(GOW161.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW162_acc_1hz_rms(i,j) = rms(GOW162.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW163b_acc_1hz_rms(i,j) = rms(GOW163b.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW164_acc_1hz_rms(i,j) = rms(GOW164.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW165_acc_1hz_rms(i,j) = rms(GOW165.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW166_acc_1hz_rms(i,j) = rms(GOW166.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW167_acc_1hz_rms(i,j) = rms(GOW167.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW168_acc_1hz_rms(i,j) = rms(GOW168.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW169_acc_1hz_rms(i,j) = rms(GOW169.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW170_acc_1hz_rms(i,j) = rms(GOW170.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
    end
end

%Data matrices with 1Hz samplingrate
GOW00_data = [GOW00_1secData.dataStruct.data GOW00_acc_1hz_rms
GOW00.testPoint.gasRef.q*ones([N,1]) GOW00.testPoint.watRef.q*ones([N,1])
GOW00.testPoint.oilRef.q*ones([N,1])];
```

```
GOW01_data = [GOW01_1secData.dataStruct.data GOW01_acc_1hz_rms
GOW01.testPoint.gasRef.q*ones([N,1]) GOW01.testPoint.watRef.q*ones([N,1])
GOW01.testPoint.oilRef.q*ones([N,1])];
GOW01b_data = [GOW01b_1secData.dataStruct.data GOW01b_acc_1hz_rms
GOW01b.testPoint.gasRef.q*ones([N,1]) GOW01b.testPoint.watRef.q*ones([N,1])
GOW01b.testPoint.oilRef.q*ones([N,1])];
GOW01c_data = [GOW01c_1secData.dataStruct.data GOW01c_acc_1hz_rms
GOW01c.testPoint.gasRef.q*ones([N,1]) GOW01c.testPoint.watRef.q*ones([N,1])
GOW01c.testPoint.oilRef.q*ones([N,1])];
GOW02_data = [GOW02_1secData.dataStruct.data GOW02_acc_1hz_rms
GOW02.testPoint.gasRef.q*ones([N,1]) GOW02.testPoint.watRef.q*ones([N,1])
GOW02.testPoint.oilRef.q*ones([N,1])];
GOW03_data = [GOW03_1secData.dataStruct.data GOW03_acc_1hz_rms
GOW03.testPoint.gasRef.q*ones([N,1]) GOW03.testPoint.watRef.q*ones([N,1])
GOW03.testPoint.oilRef.q*ones([N,1])];
GOW04_data = [GOW04_1secData.dataStruct.data GOW04_acc_1hz_rms
GOW04.testPoint.gasRef.q*ones([N,1]) GOW04.testPoint.watRef.q*ones([N,1])
GOW04.testPoint.oilRef.q*ones([N,1])];
GOW05_data = [GOW05_1secData.dataStruct.data GOW05_acc_1hz_rms
GOW05.testPoint.gasRef.q*ones([N,1]) GOW05.testPoint.watRef.q*ones([N,1])
GOW05.testPoint.oilRef.q*ones([N,1])];
GOW06_data = [GOW06_1secData.dataStruct.data GOW06_acc_1hz_rms
GOW06.testPoint.gasRef.q*ones([N,1]) GOW06.testPoint.watRef.q*ones([N,1])
GOW06.testPoint.oilRef.q*ones([N,1])];
GOW06b_data = [GOW06b_1secData.dataStruct.data GOW06b_acc_1hz_rms
GOW06b.testPoint.gasRef.q*ones([N,1]) GOW06b.testPoint.watRef.q*ones([N,1])
GOW06b.testPoint.oilRef.q*ones([N,1])];
GOW07_data = [GOW07_1secData.dataStruct.data GOW07_acc_1hz_rms
GOW07.testPoint.gasRef.q*ones([N,1]) GOW07.testPoint.watRef.q*ones([N,1])
GOW07.testPoint.oilRef.q*ones([N,1])];
GOW08_data = [GOW08_1secData.dataStruct.data GOW08_acc_1hz_rms
GOW08.testPoint.gasRef.q*ones([N,1]) GOW08.testPoint.watRef.q*ones([N,1])
GOW08.testPoint.oilRef.q*ones([N,1])];
GOW09_data = [GOW09_1secData.dataStruct.data GOW09_acc_1hz_rms
GOW09.testPoint.gasRef.q*ones([N,1]) GOW09.testPoint.watRef.q*ones([N,1])
GOW09.testPoint.oilRef.q*ones([N,1])];
GOW10_data = [GOW10_1secData.dataStruct.data GOW10_acc_1hz_rms
GOW10.testPoint.gasRef.q*ones([N,1]) GOW10.testPoint.watRef.q*ones([N,1])
GOW10.testPoint.oilRef.q*ones([N,1])];
GOW11_data = [GOW11_1secData.dataStruct.data GOW11_acc_1hz_rms
GOW11.testPoint.gasRef.q*ones([N,1]) GOW11.testPoint.watRef.q*ones([N,1])
GOW11.testPoint.oilRef.q*ones([N,1])];
GOW25_data = [GOW25_1secData.dataStruct.data GOW25_acc_1hz_rms
GOW25.testPoint.gasRef.q*ones([N,1]) GOW25.testPoint.watRef.q*ones([N,1])
GOW25.testPoint.oilRef.q*ones([N,1])];
GOW26_data = [GOW26_1secData.dataStruct.data GOW26_acc_1hz_rms
GOW26.testPoint.gasRef.q*ones([N,1]) GOW26.testPoint.watRef.q*ones([N,1])
GOW26.testPoint.oilRef.q*ones([N,1])];
GOW27_data = [GOW27_1secData.dataStruct.data GOW27_acc_1hz_rms
GOW27.testPoint.gasRef.q*ones([N,1]) GOW27.testPoint.watRef.q*ones([N,1])
GOW27.testPoint.oilRef.q*ones([N,1])];
GOW28_data = [GOW28_1secData.dataStruct.data GOW28_acc_1hz_rms
GOW28.testPoint.gasRef.q*ones([N,1]) GOW28.testPoint.watRef.q*ones([N,1])
GOW28.testPoint.oilRef.q*ones([N,1])];
GOW29_data = [GOW29_1secData.dataStruct.data GOW29_acc_1hz_rms
GOW29.testPoint.gasRef.q*ones([N,1]) GOW29.testPoint.watRef.q*ones([N,1])
GOW29.testPoint.oilRef.q*ones([N,1])];
```

```
GOW30_data = [GOW30_1secData.dataStruct.data GOW30_acc_1hz_rms
GOW30.testPoint.gasRef.q*ones([N,1]) GOW30.testPoint.watRef.q*ones([N,1])
GOW30.testPoint.oilRef.q*ones([N,1])];
GOW31_data = [GOW31_1secData.dataStruct.data GOW31_acc_1hz_rms
GOW31.testPoint.gasRef.q*ones([N,1]) GOW31.testPoint.watRef.q*ones([N,1])
GOW31.testPoint.oilRef.q*ones([N,1])];
GOW32_data = [GOW32_1secData.dataStruct.data GOW32_acc_1hz_rms
GOW32.testPoint.gasRef.q*ones([N,1]) GOW32.testPoint.watRef.q*ones([N,1])
GOW32.testPoint.oilRef.q*ones([N,1])];
GOW33_data = [GOW33_1secData.dataStruct.data GOW33_acc_1hz_rms
GOW33.testPoint.gasRef.q*ones([N,1]) GOW33.testPoint.watRef.q*ones([N,1])
GOW33.testPoint.oilRef.q*ones([N,1])];
GOW34_data = [GOW34_1secData.dataStruct.data GOW34_acc_1hz_rms
GOW34.testPoint.gasRef.q*ones([N,1]) GOW34.testPoint.watRef.q*ones([N,1])
GOW34.testPoint.oilRef.q*ones([N,1])];
GOW35_data = [GOW35_1secData.dataStruct.data GOW35_acc_1hz_rms
GOW35.testPoint.gasRef.q*ones([N,1]) GOW35.testPoint.watRef.q*ones([N,1])
GOW35.testPoint.oilRef.q*ones([N,1])];
GOW36_data = [GOW36_1secData.dataStruct.data GOW36_acc_1hz_rms
GOW36.testPoint.gasRef.q*ones([N,1]) GOW36.testPoint.watRef.q*ones([N,1])
GOW36.testPoint.oilRef.q*ones([N,1])];
GOW37_data = [GOW37_1secData.dataStruct.data GOW37_acc_1hz_rms
GOW37.testPoint.gasRef.q*ones([N,1]) GOW37.testPoint.watRef.q*ones([N,1])
GOW37.testPoint.oilRef.q*ones([N,1])];
GOW38_data = [GOW38_1secData.dataStruct.data GOW38_acc_1hz_rms
GOW38.testPoint.gasRef.q*ones([N,1]) GOW38.testPoint.watRef.q*ones([N,1])
GOW38.testPoint.oilRef.q*ones([N,1])];
GOW39_data = [GOW39_1secData.dataStruct.data GOW39_acc_1hz_rms
GOW39.testPoint.gasRef.q*ones([N,1]) GOW39.testPoint.watRef.q*ones([N,1])
GOW39.testPoint.oilRef.q*ones([N,1])];
GOW40_data = [GOW40_1secData.dataStruct.data GOW40_acc_1hz_rms
GOW40.testPoint.gasRef.q*ones([N,1]) GOW40.testPoint.watRef.q*ones([N,1])
GOW40.testPoint.oilRef.q*ones([N,1])];
GOW41_data = [GOW41_1secData.dataStruct.data GOW41_acc_1hz_rms
GOW41.testPoint.gasRef.q*ones([N,1]) GOW41.testPoint.watRef.q*ones([N,1])
GOW41.testPoint.oilRef.q*ones([N,1])];
GOW42_data = [GOW42_1secData.dataStruct.data GOW42_acc_1hz_rms
GOW42.testPoint.gasRef.q*ones([N,1]) GOW42.testPoint.watRef.q*ones([N,1])
GOW42.testPoint.oilRef.q*ones([N,1])];
GOW43_data = [GOW43_1secData.dataStruct.data GOW43_acc_1hz_rms
GOW43.testPoint.gasRef.q*ones([N,1]) GOW43.testPoint.watRef.q*ones([N,1])
GOW43.testPoint.oilRef.q*ones([N,1])];
GOW44_data = [GOW44_1secData.dataStruct.data GOW44_acc_1hz_rms
GOW44.testPoint.gasRef.q*ones([N,1]) GOW44.testPoint.watRef.q*ones([N,1])
GOW44.testPoint.oilRef.q*ones([N,1])];
GOW45_data = [GOW45_1secData.dataStruct.data GOW45_acc_1hz_rms
GOW45.testPoint.gasRef.q*ones([N,1]) GOW45.testPoint.watRef.q*ones([N,1])
GOW45.testPoint.oilRef.q*ones([N,1])];
GOW46_data = [GOW46_1secData.dataStruct.data GOW46_acc_1hz_rms
GOW46.testPoint.gasRef.q*ones([N,1]) GOW46.testPoint.watRef.q*ones([N,1])
GOW46.testPoint.oilRef.q*ones([N,1])];
GOW47_data = [GOW47_1secData.dataStruct.data GOW47_acc_1hz_rms
GOW47.testPoint.gasRef.q*ones([N,1]) GOW47.testPoint.watRef.q*ones([N,1])
GOW47.testPoint.oilRef.q*ones([N,1])];
GOW48_data = [GOW48_1secData.dataStruct.data GOW48_acc_1hz_rms
GOW48.testPoint.gasRef.q*ones([N,1]) GOW48.testPoint.watRef.q*ones([N,1])
GOW48.testPoint.oilRef.q*ones([N,1])];
```

```
GOW49_data = [GOW49_1secData.dataStruct.data GOW49_acc_1hz_rms
GOW49.testPoint.gasRef.q*ones([N,1]) GOW49.testPoint.watRef.q*ones([N,1])
GOW49.testPoint.oilRef.q*ones([N,1])];
GOW50_data = [GOW50_1secData.dataStruct.data GOW50_acc_1hz_rms
GOW50.testPoint.gasRef.q*ones([N,1]) GOW50.testPoint.watRef.q*ones([N,1])
GOW50.testPoint.oilRef.q*ones([N,1])];
GOW51_data = [GOW51_1secData.dataStruct.data GOW51_acc_1hz_rms
GOW51.testPoint.gasRef.q*ones([N,1]) GOW51.testPoint.watRef.q*ones([N,1])
GOW51.testPoint.oilRef.q*ones([N,1])];
GOW52_data = [GOW52_1secData.dataStruct.data GOW52_acc_1hz_rms
GOW52.testPoint.gasRef.q*ones([N,1]) GOW52.testPoint.watRef.q*ones([N,1])
GOW52.testPoint.oilRef.q*ones([N,1])];
GOW53_data = [GOW53_1secData.dataStruct.data GOW53_acc_1hz_rms
GOW53.testPoint.gasRef.q*ones([N,1]) GOW53.testPoint.watRef.q*ones([N,1])
GOW53.testPoint.oilRef.q*ones([N,1])];
GOW54_data = [GOW54_1secData.dataStruct.data GOW54_acc_1hz_rms
GOW54.testPoint.gasRef.q*ones([N,1]) GOW54.testPoint.watRef.q*ones([N,1])
GOW54.testPoint.oilRef.q*ones([N,1])];
GOW55_data = [GOW55_1secData.dataStruct.data GOW55_acc_1hz_rms
GOW55.testPoint.gasRef.q*ones([N,1]) GOW55.testPoint.watRef.q*ones([N,1])
GOW55.testPoint.oilRef.q*ones([N,1])];
GOW56_data = [GOW56_1secData.dataStruct.data GOW56_acc_1hz_rms
GOW56.testPoint.gasRef.q*ones([N,1]) GOW56.testPoint.watRef.q*ones([N,1])
GOW56.testPoint.oilRef.q*ones([N,1])];
GOW60_data = [GOW60_1secData.dataStruct.data GOW60_acc_1hz_rms
GOW60.testPoint.gasRef.q*ones([N,1]) GOW60.testPoint.watRef.q*ones([N,1])
GOW60.testPoint.oilRef.q*ones([N,1])];
GOW61_data = [GOW61_1secData.dataStruct.data GOW61_acc_1hz_rms
GOW61.testPoint.gasRef.q*ones([N,1]) GOW61.testPoint.watRef.q*ones([N,1])
GOW61.testPoint.oilRef.q*ones([N,1])];
GOW62_data = [GOW60_1secData.dataStruct.data GOW62_acc_1hz_rms
GOW62.testPoint.gasRef.q*ones([N,1]) GOW62.testPoint.watRef.q*ones([N,1])
GOW62.testPoint.oilRef.q*ones([N,1])];
GOW63_data = [GOW63_1secData.dataStruct.data GOW63_acc_1hz_rms
GOW63.testPoint.gasRef.q*ones([N,1]) GOW63.testPoint.watRef.q*ones([N,1])
GOW63.testPoint.oilRef.q*ones([N,1])];
GOW64_data = [GOW64_1secData.dataStruct.data GOW64_acc_1hz_rms
GOW64.testPoint.gasRef.q*ones([N,1]) GOW64.testPoint.watRef.q*ones([N,1])
GOW64.testPoint.oilRef.q*ones([N,1])];
GOW69_data = [GOW69_1secData.dataStruct.data GOW69_acc_1hz_rms
GOW69.testPoint.gasRef.q*ones([N,1]) GOW69.testPoint.watRef.q*ones([N,1])
GOW69.testPoint.oilRef.q*ones([N,1])];
GOW70_data = [GOW70_1secData.dataStruct.data GOW70_acc_1hz_rms
GOW70.testPoint.gasRef.q*ones([N,1]) GOW70.testPoint.watRef.q*ones([N,1])
GOW70.testPoint.oilRef.q*ones([N,1])];
GOW71_data = [GOW71_1secData.dataStruct.data GOW71_acc_1hz_rms
GOW70.testPoint.gasRef.q*ones([N,1]) GOW71.testPoint.watRef.q*ones([N,1])
GOW71.testPoint.oilRef.q*ones([N,1])];
GOW74_data = [GOW74_1secData.dataStruct.data GOW74_acc_1hz_rms
GOW70.testPoint.gasRef.q*ones([N,1]) GOW74.testPoint.watRef.q*ones([N,1])
GOW74.testPoint.oilRef.q*ones([N,1])];
GOW75_data = [GOW75_1secData.dataStruct.data GOW75_acc_1hz_rms
GOW70.testPoint.gasRef.q*ones([N,1]) GOW75.testPoint.watRef.q*ones([N,1])
GOW75.testPoint.oilRef.q*ones([N,1])];
GOW76_data = [GOW76_1secData.dataStruct.data GOW76_acc_1hz_rms
GOW70.testPoint.gasRef.q*ones([N,1]) GOW76.testPoint.watRef.q*ones([N,1])
GOW76.testPoint.oilRef.q*ones([N,1])];
```

```
GOW77_data = [GOW77_1secData.dataStruct.data GOW77_acc_1hz_rms
GOW70.testPoint.gasRef.q*ones([N,1]) GOW77.testPoint.watRef.q*ones([N,1])
GOW77.testPoint.oilRef.q*ones([N,1])];
GOW78_data = [GOW78_1secData.dataStruct.data GOW78_acc_1hz_rms
GOW70.testPoint.gasRef.q*ones([N,1]) GOW78.testPoint.watRef.q*ones([N,1])
GOW78.testPoint.oilRef.q*ones([N,1])];
GOW79_data = [GOW79_1secData.dataStruct.data GOW79_acc_1hz_rms
GOW70.testPoint.gasRef.q*ones([N,1]) GOW79.testPoint.watRef.q*ones([N,1])
GOW79.testPoint.oilRef.q*ones([N,1])];
GOW80_data = [GOW80_1secData.dataStruct.data GOW80_acc_1hz_rms
GOW80.testPoint.gasRef.q*ones([N,1]) GOW80.testPoint.watRef.q*ones([N,1])
GOW80.testPoint.oilRef.q*ones([N,1])];
GOW82_data = [GOW82_1secData.dataStruct.data GOW82_acc_1hz_rms
GOW82.testPoint.gasRef.q*ones([N,1]) GOW82.testPoint.watRef.q*ones([N,1])
GOW82.testPoint.oilRef.q*ones([N,1])];
GOW83_data = [GOW83_1secData.dataStruct.data GOW83_acc_1hz_rms
GOW83.testPoint.gasRef.q*ones([N,1]) GOW83.testPoint.watRef.q*ones([N,1])
GOW83.testPoint.oilRef.q*ones([N,1])];
GOW84_data = [GOW84_1secData.dataStruct.data GOW84_acc_1hz_rms
GOW84.testPoint.gasRef.q*ones([N,1]) GOW84.testPoint.watRef.q*ones([N,1])
GOW84.testPoint.oilRef.q*ones([N,1])];
GOW85_data = [GOW85_1secData.dataStruct.data GOW85_acc_1hz_rms
GOW85.testPoint.gasRef.q*ones([N,1]) GOW85.testPoint.watRef.q*ones([N,1])
GOW85.testPoint.oilRef.q*ones([N,1])];
GOW86_data = [GOW86_1secData.dataStruct.data GOW86_acc_1hz_rms
GOW86.testPoint.gasRef.q*ones([N,1]) GOW86.testPoint.watRef.q*ones([N,1])
GOW86.testPoint.oilRef.q*ones([N,1])];
GOW87_data = [GOW87_1secData.dataStruct.data GOW87_acc_1hz_rms
GOW87.testPoint.gasRef.q*ones([N,1]) GOW87.testPoint.watRef.q*ones([N,1])
GOW87.testPoint.oilRef.q*ones([N,1])];
GOW88_data = [GOW88_1secData.dataStruct.data GOW88_acc_1hz_rms
GOW88.testPoint.gasRef.q*ones([N,1]) GOW88.testPoint.watRef.q*ones([N,1])
GOW88.testPoint.oilRef.q*ones([N,1])];
GOW89_data = [GOW89_1secData.dataStruct.data GOW89_acc_1hz_rms
GOW89.testPoint.gasRef.q*ones([N,1]) GOW89.testPoint.watRef.q*ones([N,1])
GOW89.testPoint.oilRef.q*ones([N,1])];
GOW90_data = [GOW90_1secData.dataStruct.data GOW90_acc_1hz_rms
GOW90.testPoint.gasRef.q*ones([N,1]) GOW90.testPoint.watRef.q*ones([N,1])
GOW90.testPoint.oilRef.q*ones([N,1])];
GOW91_data = [GOW91_1secData.dataStruct.data GOW91_acc_1hz_rms
GOW91.testPoint.gasRef.q*ones([N,1]) GOW91.testPoint.watRef.q*ones([N,1])
GOW91.testPoint.oilRef.q*ones([N,1])];
GOW92_data = [GOW92_1secData.dataStruct.data GOW92_acc_1hz_rms
GOW92.testPoint.gasRef.q*ones([N,1]) GOW92.testPoint.watRef.q*ones([N,1])
GOW92.testPoint.oilRef.q*ones([N,1])];
GOW93_data = [GOW93_1secData.dataStruct.data GOW93_acc_1hz_rms
GOW93.testPoint.gasRef.q*ones([N,1]) GOW93.testPoint.watRef.q*ones([N,1])
GOW93.testPoint.oilRef.q*ones([N,1])];
GOW94_data = [GOW94_1secData.dataStruct.data GOW94_acc_1hz_rms
GOW94.testPoint.gasRef.q*ones([N,1]) GOW94.testPoint.watRef.q*ones([N,1])
GOW94.testPoint.oilRef.q*ones([N,1])];
GOW95_data = [GOW95_1secData.dataStruct.data GOW95_acc_1hz_rms
GOW95.testPoint.gasRef.q*ones([N,1]) GOW95.testPoint.watRef.q*ones([N,1])
GOW95.testPoint.oilRef.q*ones([N,1])];
GOW96_data = [GOW96_1secData.dataStruct.data GOW96_acc_1hz_rms
GOW96.testPoint.gasRef.q*ones([N,1]) GOW96.testPoint.watRef.q*ones([N,1])
GOW96.testPoint.oilRef.q*ones([N,1])];
```

```
GOW97_data = [GOW97_1secData.dataStruct.data GOW97_acc_1hz_rms
GOW97.testPoint.gasRef.q*ones([N,1]) GOW97.testPoint.watRef.q*ones([N,1])
GOW97.testPoint.oilRef.q*ones([N,1])];
GOW100b_data = [GOW100b_1secData.dataStruct.data GOW100b_acc_1hz_rms
GOW100b.testPoint.gasRef.q*ones([N,1]) GOW100b.testPoint.watRef.q*ones([N,1])
GOW100b.testPoint.oilRef.q*ones([N,1])];
GOW118_data = [GOW118_1secData.dataStruct.data GOW118_acc_1hz_rms
GOW118.testPoint.gasRef.q*ones([N,1]) GOW118.testPoint.watRef.q*ones([N,1])
GOW118.testPoint.oilRef.q*ones([N,1])];
GOW119_data = [GOW119_1secData.dataStruct.data GOW119_acc_1hz_rms
GOW119.testPoint.gasRef.q*ones([N,1]) GOW119.testPoint.watRef.q*ones([N,1])
GOW119.testPoint.oilRef.q*ones([N,1])];
GOW120_data = [GOW120_1secData.dataStruct.data GOW120_acc_1hz_rms
GOW120.testPoint.gasRef.q*ones([N,1]) GOW120.testPoint.watRef.q*ones([N,1])
GOW120.testPoint.oilRef.q*ones([N,1])];
GOW121_data = [GOW121_1secData.dataStruct.data GOW121_acc_1hz_rms
GOW121.testPoint.gasRef.q*ones([N,1]) GOW121.testPoint.watRef.q*ones([N,1])
GOW121.testPoint.oilRef.q*ones([N,1])];
GOW122_data = [GOW122_1secData.dataStruct.data GOW122_acc_1hz_rms
GOW122.testPoint.gasRef.q*ones([N,1]) GOW122.testPoint.watRef.q*ones([N,1])
GOW122.testPoint.oilRef.q*ones([N,1])];
GOW123_data = [GOW123_1secData.dataStruct.data GOW123_acc_1hz_rms
GOW123.testPoint.gasRef.q*ones([N,1]) GOW123.testPoint.watRef.q*ones([N,1])
GOW123.testPoint.oilRef.q*ones([N,1])];
GOW124_data = [GOW124_1secData.dataStruct.data GOW124_acc_1hz_rms
GOW124.testPoint.gasRef.q*ones([N,1]) GOW124.testPoint.watRef.q*ones([N,1])
GOW124.testPoint.oilRef.q*ones([N,1])];
GOW125_data = [GOW125_1secData.dataStruct.data GOW125_acc_1hz_rms
GOW125.testPoint.gasRef.q*ones([N,1]) GOW125.testPoint.watRef.q*ones([N,1])
GOW125.testPoint.oilRef.q*ones([N,1])];
GOW126_data = [GOW126_1secData.dataStruct.data GOW126_acc_1hz_rms
GOW126.testPoint.gasRef.q*ones([N,1]) GOW126.testPoint.watRef.q*ones([N,1])
GOW126.testPoint.oilRef.q*ones([N,1])];
GOW127_data = [GOW127_1secData.dataStruct.data GOW127_acc_1hz_rms
GOW127.testPoint.gasRef.q*ones([N,1]) GOW127.testPoint.watRef.q*ones([N,1])
GOW127.testPoint.oilRef.q*ones([N,1])];
GOW142_data = [GOW142_1secData.dataStruct.data GOW142_acc_1hz_rms
GOW142.testPoint.gasRef.q*ones([N,1]) GOW142.testPoint.watRef.q*ones([N,1])
GOW142.testPoint.oilRef.q*ones([N,1])];
GOW143_data = [GOW143_1secData.dataStruct.data GOW143_acc_1hz_rms
GOW143.testPoint.gasRef.q*ones([N,1]) GOW143.testPoint.watRef.q*ones([N,1])
GOW143.testPoint.oilRef.q*ones([N,1])];
GOW144_data = [GOW144_1secData.dataStruct.data GOW144_acc_1hz_rms
GOW144.testPoint.gasRef.q*ones([N,1]) GOW144.testPoint.watRef.q*ones([N,1])
GOW144.testPoint.oilRef.q*ones([N,1])];
GOW145_data = [GOW145_1secData.dataStruct.data GOW145_acc_1hz_rms
GOW145.testPoint.gasRef.q*ones([N,1]) GOW145.testPoint.watRef.q*ones([N,1])
GOW145.testPoint.oilRef.q*ones([N,1])];
GOW145b_data = [GOW145b_1secData.dataStruct.data GOW145b_acc_1hz_rms
GOW145b.testPoint.gasRef.q*ones([N,1]) GOW145b.testPoint.watRef.q*ones([N,1])
GOW145b.testPoint.oilRef.q*ones([N,1])];
GOW146_data = [GOW146_1secData.dataStruct.data GOW146_acc_1hz_rms
GOW146.testPoint.gasRef.q*ones([N,1]) GOW146.testPoint.watRef.q*ones([N,1])
GOW146.testPoint.oilRef.q*ones([N,1])];
GOW147_data = [GOW147_1secData.dataStruct.data GOW147_acc_1hz_rms
GOW147.testPoint.gasRef.q*ones([N,1]) GOW147.testPoint.watRef.q*ones([N,1])
GOW147.testPoint.oilRef.q*ones([N,1])];
```

```
GOW148_data = [GOW148_1secData.dataStruct.data GOW148_acc_1hz_rms
GOW148.testPoint.gasRef.q*ones([N,1]) GOW148.testPoint.watRef.q*ones([N,1])
GOW148.testPoint.oilRef.q*ones([N,1])];
GOW149_data = [GOW149_1secData.dataStruct.data GOW149_acc_1hz_rms
GOW149.testPoint.gasRef.q*ones([N,1]) GOW149.testPoint.watRef.q*ones([N,1])
GOW149.testPoint.oilRef.q*ones([N,1])];
GOW150_data = [GOW150_1secData.dataStruct.data GOW150_acc_1hz_rms
GOW150.testPoint.gasRef.q*ones([N,1]) GOW150.testPoint.watRef.q*ones([N,1])
GOW150.testPoint.oilRef.q*ones([N,1])];
GOW150b_data = [GOW150b_1secData.dataStruct.data GOW150b_acc_1hz_rms
GOW150b.testPoint.gasRef.q*ones([N,1]) GOW150b.testPoint.watRef.q*ones([N,1])
GOW150b.testPoint.oilRef.q*ones([N,1])];
GOW151_data = [GOW151_1secData.dataStruct.data GOW151_acc_1hz_rms
GOW151.testPoint.gasRef.q*ones([N,1]) GOW151.testPoint.watRef.q*ones([N,1])
GOW151.testPoint.oilRef.q*ones([N,1])];
GOW152_data = [GOW152_1secData.dataStruct.data GOW152_acc_1hz_rms
GOW152.testPoint.gasRef.q*ones([N,1]) GOW152.testPoint.watRef.q*ones([N,1])
GOW152.testPoint.oilRef.q*ones([N,1])];
GOW153_data = [GOW153_1secData.dataStruct.data GOW153_acc_1hz_rms
GOW153.testPoint.gasRef.q*ones([N,1]) GOW153.testPoint.watRef.q*ones([N,1])
GOW153.testPoint.oilRef.q*ones([N,1])];
GOW154_data = [GOW154_1secData.dataStruct.data GOW154_acc_1hz_rms
GOW154.testPoint.gasRef.q*ones([N,1]) GOW154.testPoint.watRef.q*ones([N,1])
GOW154.testPoint.oilRef.q*ones([N,1])];
GOW155_data = [GOW155_1secData.dataStruct.data GOW155_acc_1hz_rms
GOW155.testPoint.gasRef.q*ones([N,1]) GOW155.testPoint.watRef.q*ones([N,1])
GOW155.testPoint.oilRef.q*ones([N,1])];
GOW156_data = [GOW156_1secData.dataStruct.data GOW156_acc_1hz_rms
GOW156.testPoint.gasRef.q*ones([N,1]) GOW156.testPoint.watRef.q*ones([N,1])
GOW156.testPoint.oilRef.q*ones([N,1])];
GOW157_data = [GOW157_1secData.dataStruct.data GOW157_acc_1hz_rms
GOW157.testPoint.gasRef.q*ones([N,1]) GOW157.testPoint.watRef.q*ones([N,1])
GOW157.testPoint.oilRef.q*ones([N,1])];
GOW157b_data = [GOW157b_1secData.dataStruct.data GOW157b_acc_1hz_rms
GOW157b.testPoint.gasRef.q*ones([N,1]) GOW157b.testPoint.watRef.q*ones([N,1])
GOW157b.testPoint.oilRef.q*ones([N,1])];
GOW158_data = [GOW158_1secData.dataStruct.data GOW158_acc_1hz_rms
GOW158.testPoint.gasRef.q*ones([N,1]) GOW158.testPoint.watRef.q*ones([N,1])
GOW158.testPoint.oilRef.q*ones([N,1])];
GOW159_data = [GOW159_1secData.dataStruct.data GOW159_acc_1hz_rms
GOW159.testPoint.gasRef.q*ones([N,1]) GOW159.testPoint.watRef.q*ones([N,1])
GOW159.testPoint.oilRef.q*ones([N,1])];
GOW161_data = [GOW161_1secData.dataStruct.data GOW161_acc_1hz_rms
GOW161.testPoint.gasRef.q*ones([N,1]) GOW161.testPoint.watRef.q*ones([N,1])
GOW161.testPoint.oilRef.q*ones([N,1])];
GOW162_data = [GOW162_1secData.dataStruct.data GOW162_acc_1hz_rms
GOW162.testPoint.gasRef.q*ones([N,1]) GOW162.testPoint.watRef.q*ones([N,1])
GOW162.testPoint.oilRef.q*ones([N,1])];
GOW163b_data = [GOW163b_1secData.dataStruct.data GOW163b_acc_1hz_rms
GOW163b.testPoint.gasRef.q*ones([N,1]) GOW163b.testPoint.watRef.q*ones([N,1])
GOW163b.testPoint.oilRef.q*ones([N,1])];
GOW164_data = [GOW164_1secData.dataStruct.data GOW164_acc_1hz_rms
GOW164.testPoint.gasRef.q*ones([N,1]) GOW164.testPoint.watRef.q*ones([N,1])
GOW164.testPoint.oilRef.q*ones([N,1])];
GOW165_data = [GOW165_1secData.dataStruct.data GOW165_acc_1hz_rms
GOW165.testPoint.gasRef.q*ones([N,1]) GOW165.testPoint.watRef.q*ones([N,1])
GOW165.testPoint.oilRef.q*ones([N,1])];
```

```
GOW166_data = [GOW166_1secData.dataStruct.data GOW166_acc_1hz_rms
GOW166.testPoint.gasRef.q*ones([N,1]) GOW166.testPoint.watRef.q*ones([N,1])
GOW166.testPoint.oilRef.q*ones([N,1])];
GOW167_data = [GOW167_1secData.dataStruct.data GOW167_acc_1hz_rms
GOW167.testPoint.gasRef.q*ones([N,1]) GOW167.testPoint.watRef.q*ones([N,1])
GOW167.testPoint.oilRef.q*ones([N,1])];
GOW168_data = [GOW168_1secData.dataStruct.data GOW168_acc_1hz_rms
GOW168.testPoint.gasRef.q*ones([N,1]) GOW168.testPoint.watRef.q*ones([N,1])
GOW168.testPoint.oilRef.q*ones([N,1])];
GOW169_data = [GOW169_1secData.dataStruct.data GOW169_acc_1hz_rms
GOW169.testPoint.gasRef.q*ones([N,1]) GOW169.testPoint.watRef.q*ones([N,1])
GOW169.testPoint.oilRef.q*ones([N,1])];
GOW170_data = [GOW170_1secData.dataStruct.data GOW170_acc_1hz_rms
GOW170.testPoint.gasRef.q*ones([N,1]) GOW170.testPoint.watRef.q*ones([N,1])
GOW170.testPoint.oilRef.q*ones([N,1])];

%Merging all data
training = [GOW00_data(start:stop,:); GOW01_data(start:stop,:);
GOW01b_data(start:stop,:); GOW01c_data(start:stop,:); GOW02_data(start:stop,:);
GOW03_data(start:stop,:); GOW04_data(start:stop,:); GOW05_data(start:stop,:);
GOW06_data(start:stop,:); GOW06b_data(start:stop,:);  GOW07_data(start:stop,:);
GOW08_data(start:stop,:); GOW09_data(start:stop,:); GOW10_data(start:stop,:);
GOW11_data(start:stop,:); GOW25_data(start:stop,:); GOW26_data(start:stop,:);
GOW27_data(start:stop,:); GOW28_data(start:stop,:); GOW29_data(start:stop,:);
GOW30_data(start:stop,:); GOW31_data(start:stop,:); GOW32_data(start:stop,:);
GOW33_data(start:stop,:); GOW34_data(start:stop,:); GOW35_data(start:stop,:);
GOW36_data(start:stop,:); GOW37_data(start:stop,:); GOW38_data(start:stop,:);
GOW39_data(start:stop,:); GOW40_data(start:stop,:); GOW41_data(start:stop,:);
GOW42_data(start:stop,:); GOW43_data(start:stop,:); GOW44_data(start:stop,:);
GOW45_data(start:stop,:); GOW46_data(start:stop,:); GOW47_data(start:stop,:);
GOW48_data(start:stop,:); GOW49_data(start:stop,:); GOW50_data(start:stop,:);
GOW51_data(start:stop,:); GOW52_data(start:stop,:); GOW53_data(start:stop,:);
GOW54_data(start:stop,:); GOW55_data(start:stop,:); GOW56_data(start:stop,:);
GOW60_data(start:stop,:); GOW61_data(start:stop,:); GOW62_data(start:stop,:);
GOW63_data(start:stop,:); GOW64_data(start:stop,:); GOW69_data(start:stop,:);
GOW70_data(start:stop,:); GOW71_data(start:stop,:); GOW74_data(start:stop,:);
GOW75_data(start:stop,:); GOW76_data(start:stop,:); GOW77_data(start:stop,:);
GOW78_data(start:stop,:); GOW79_data(start:stop,:); GOW80_data(start:stop,:);
GOW82_data(start:stop,:); GOW83_data(start:stop,:); GOW84_data(start:stop,:);
GOW85_data(start:stop,:); GOW86_data(start:stop,:); GOW87_data(start:stop,:);
GOW88_data(start:stop,:); GOW89_data(start:stop,:); GOW90_data(start:stop,:);
GOW91_data(start:stop,:); GOW92_data(start:stop,:); GOW93_data(start:stop,:);
GOW94_data(start:stop,:); GOW95_data(start:stop,:); GOW96_data(start:stop,:);
GOW97_data(start:stop,:); GOW100b_data(start:stop,:); GOW118_data(start:stop,:);
GOW119_data(start:stop,:); GOW120_data(start:stop,:); GOW121_data(start:stop,:);
GOW122_data(start:stop,:); GOW123_data(start:stop,:); GOW124_data(start:stop,:);
GOW125_data(start:stop,:); GOW126_data(start:stop,:); GOW127_data(start:stop,:);
GOW142_data(start:stop,:); GOW143_data(start:stop,:); GOW144_data(start:stop,:);
GOW145_data(start:stop,:); GOW145b_data(start:stop,:); GOW146_data(start:stop,:);
GOW147_data(start:stop,:); GOW148_data(start:stop,:); GOW149_data(start:stop,:);
GOW150_data(start:stop,:); GOW150b_data(start:stop,:); GOW151_data(start:stop,:);
GOW152_data(start:stop,:); GOW153_data(start:stop,:); GOW154_data(start:stop,:);
GOW155_data(start:stop,:); GOW156_data(start:stop,:); GOW157_data(start:stop,:);
GOW157b_data(start:stop,:); GOW158_data(start:stop,:); GOW159_data(start:stop,:);
GOW161_data(start:stop,:); GOW162_data(start:stop,:); GOW163b_data(start:stop,:);
GOW164_data(start:stop,:); GOW165_data(start:stop,:); GOW166_data(start:stop,:);
```

```
GOW167_data(start:stop,:); GOW168_data(start:stop,:); GOW169_data(start:stop,:);
GOW170_data(start:stop,:)];
testing = [GOW143_data(start:stop,:)];
```

Appendix F

Preparing the third part of two-phase flow experiment data from dataset 2.

```
clc;
clear all;

% Load data
GOW171 = load('GOW_171.mat');
GOW171_1secData = load('GOW_171_1secData.mat');
GOW501 = load('GOW_501.mat');
GOW501_1secData = load('GOW_501_1secData.mat');
GOW502 = load('GOW_502.mat');
GOW502_1secData = load('GOW_502_1secData.mat');
GOW503 = load('GOW_503.mat');
GOW503_1secData = load('GOW_503_1secData.mat');
GOW504 = load('GOW_504.mat');
GOW504_1secData = load('GOW_504_1secData.mat');
GOW505 = load('GOW_505.mat');
GOW505_1secData = load('GOW_505_1secData.mat');
GOW506 = load('GOW_506.mat');
GOW506_1secData = load('GOW_506_1secData.mat');
GOW507 = load('GOW_507.mat');
GOW507_1secData = load('GOW_507_1secData.mat');
GOW508 = load('GOW_508.mat');
GOW508_1secData = load('GOW_508_1secData.mat');
GOW509 = load('GOW_509.mat');
GOW509_1secData = load('GOW_509_1secData.mat');
GOW510 = load('GOW_510.mat');
GOW510_1secData = load('GOW_510_1secData.mat');
GOW511 = load('GOW_511.mat');
GOW511_1secData = load('GOW_511_1secData.mat');

%Calculate average RMS with 1Hz samwple rate
N_acc = length(GOW171.testPoint.datAcc.ch(1).data);
N = length(GOW171_1secData.dataStruct.data);
dt = 1/GOW171.testPoint.datAcc.dt;
time = N_acc/dt;

%Select samples in each dataset:
start = 100;
stop = 109;

for j = 1:4
    for i = 1:1:time
        GOW171_acc_1hz_rms(i,j) = rms(GOW171.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW501_acc_1hz_rms(i,j) = rms(GOW501.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW502_acc_1hz_rms(i,j) = rms(GOW502.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW503_acc_1hz_rms(i,j) = rms(GOW503.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW504_acc_1hz_rms(i,j) = rms(GOW504.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
```

```
        GOW505_acc_1hz_rms(i,j) = rms(GOW505.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW506_acc_1hz_rms(i,j) = rms(GOW506.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW507_acc_1hz_rms(i,j) = rms(GOW507.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW508_acc_1hz_rms(i,j) = rms(GOW508.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW509_acc_1hz_rms(i,j) = rms(GOW509.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW510_acc_1hz_rms(i,j) = rms(GOW510.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
        GOW511_acc_1hz_rms(i,j) = rms(GOW511.testPoint.datAcc.ch(j).data(1+(i-
1)*dt:i*dt));
    end
end

%Data matrices with 1Hz samplingrate
GOW171_data = [GOW171_1secData.dataStruct.data GOW171_acc_1hz_rms
GOW171.testPoint.gasRef.q*ones([N,1]) GOW171.testPoint.watRef.q*ones([N,1])
GOW171.testPoint.oilRef.q*ones([N,1])];
GOW501_data = [GOW501_1secData.dataStruct.data GOW501_acc_1hz_rms
GOW501.testPoint.gasRef.q*ones([N,1]) GOW501.testPoint.watRef.q*ones([N,1])
GOW501.testPoint.oilRef.q*ones([N,1])];
GOW502_data = [GOW502_1secData.dataStruct.data GOW502_acc_1hz_rms
GOW502.testPoint.gasRef.q*ones([N,1]) GOW502.testPoint.watRef.q*ones([N,1])
GOW502.testPoint.oilRef.q*ones([N,1])];
GOW503_data = [GOW503_1secData.dataStruct.data GOW503_acc_1hz_rms
GOW503.testPoint.gasRef.q*ones([N,1]) GOW503.testPoint.watRef.q*ones([N,1])
GOW503.testPoint.oilRef.q*ones([N,1])];
GOW504_data = [GOW504_1secData.dataStruct.data GOW504_acc_1hz_rms
GOW504.testPoint.gasRef.q*ones([N,1]) GOW504.testPoint.watRef.q*ones([N,1])
GOW504.testPoint.oilRef.q*ones([N,1])];
GOW505_data = [GOW505_1secData.dataStruct.data GOW505_acc_1hz_rms
GOW505.testPoint.gasRef.q*ones([N,1]) GOW505.testPoint.watRef.q*ones([N,1])
GOW505.testPoint.oilRef.q*ones([N,1])];
GOW506_data = [GOW506_1secData.dataStruct.data GOW506_acc_1hz_rms
GOW506.testPoint.gasRef.q*ones([N,1]) GOW506.testPoint.watRef.q*ones([N,1])
GOW506.testPoint.oilRef.q*ones([N,1])];
GOW507_data = [GOW507_1secData.dataStruct.data GOW507_acc_1hz_rms
GOW507.testPoint.gasRef.q*ones([N,1]) GOW507.testPoint.watRef.q*ones([N,1])
GOW507.testPoint.oilRef.q*ones([N,1])];
GOW508_data = [GOW508_1secData.dataStruct.data GOW508_acc_1hz_rms
GOW508.testPoint.gasRef.q*ones([N,1]) GOW508.testPoint.watRef.q*ones([N,1])
GOW508.testPoint.oilRef.q*ones([N,1])];
GOW509_data = [GOW509_1secData.dataStruct.data GOW509_acc_1hz_rms
GOW509.testPoint.gasRef.q*ones([N,1]) GOW509.testPoint.watRef.q*ones([N,1])
GOW509.testPoint.oilRef.q*ones([N,1])];
GOW510_data = [GOW510_1secData.dataStruct.data GOW510_acc_1hz_rms
GOW510.testPoint.gasRef.q*ones([N,1]) GOW510.testPoint.watRef.q*ones([N,1])
GOW510.testPoint.oilRef.q*ones([N,1])];
GOW511_data = [GOW511_1secData.dataStruct.data GOW511_acc_1hz_rms
GOW511.testPoint.gasRef.q*ones([N,1]) GOW511.testPoint.watRef.q*ones([N,1])
GOW511.testPoint.oilRef.q*ones([N,1])];


% %Merging into one training dataset and normalizing the data (excluded GOW43)
```

```
dataset_testing = [GOW171_data(start:stop,:); GOW501_data(start:stop,:);
GOW502_data(start:stop,:); GOW503_data(start:stop,:); GOW504_data(start:stop,:);
GOW505_data(start:stop,:); GOW506_data(start:stop,:); GOW507_data(start:stop,:);
GOW508_data(start:stop,:); GOW509_data(start:stop,:); GOW510_data(start:stop,:);
GOW511_data(start:stop,:)];
dataset_training = [];
```

Appendix G

The main code for data analysis of dataset 1.

```
clc;
clear all;

% Load data
load('G_rms_data.mat');
load('O_rms_data.mat');
load('W_rms_data.mat');
load('OW_rms_data.mat');
load('GO_rms_data.mat');
load('GW_rms_data.mat');
load('GOW_rms_data.mat');

%Sampling frequency
fs = 51.2e3;
L = 500;

%X-axis for time vs. amplitude plots
GO_Xvector = linspace(0, length(GO_ae1_data)/fs*1000, length(GO_ae1_data));
GW_Xvector = linspace(0, length(GW_ae1_data)/fs*1000, length(GW_ae1_data));
OW_Xvector = linspace(0, length(OW_ae1_data)/fs*1000, length(OW_ae1_data));
GOW_Xvector = linspace(0, length(GOW_ae1_data)/fs*1000, length(GOW_ae1_data));

%Total volumetric liquid flow
GO_liq_Qv = GO_Oref_Qv;
OW_liq_Qv = OW_Oref_Qv + OW_Wref_Qv;
GW_liq_Qv = GW_Wref_Qv;
GOW_liq_Qv = GOW_Wref_Qv + GOW_Oref_Qv;

%Total volumetric flow
GO_tot_Qv = GO_liq_Qv + GO_Gref_Qv;
OW_tot_Qv = OW_liq_Qv + OW_Gref_Qv;
GW_tot_Qv = GW_liq_Qv + GW_Gref_Qv;
GOW_tot_Qv = GOW_liq_Qv + GOW_Gref_Qv;

%Total differential pressure over test segment
W_tot_dp = W_Pi - W_Po;
O_tot_dp = O_Pi - O_Po;
G_tot_dp = G_Pi - G_Po;
GO_tot_dp = GO_Pi - GO_Po;
OW_tot_dp = OW_Pi - OW_Po;
GW_tot_dp = GW_Pi - GW_Po;
GOW_tot_dp = GOW_Pi - GOW_Po;

%Gas mass flow rate [kg/min]
GO_Gref_Qm = GO_Gref_Qv.*GO_Gref_rho/60;
GW_Gref_Qm = GW_Gref_Qv.*GW_Gref_rho/60;
GOW_Gref_Qm = GOW_Gref_Qv.*GOW_Gref_rho/60;

%Oil mass flow rate [kg/min]
GO_Oref_Qm = GO_Oref_Qv.*GO_Oref_rho/60;
OW_Oref_Qm = OW_Oref_Qv.*OW_Oref_rho/60;
GOW_Oref_Qm = GOW_Oref_Qv.*GOW_Oref_rho/60;
```

```
%Water mass flow rate [kg/min]
GW_Wref_Qm = GW_Wref_Qv.*GW_Wref_rho/60;
OW_Wref_Qm = OW_Wref_Qv.*OW_Wref_rho/60;
GOW_Wref_Qm = GOW_Wref_Qv.*GOW_Wref_rho/60;

%Total liquid mas flow
GO_liq_Qm = GO_Oref_Qm;
GW_liq_Qm = GW_Wref_Qm;
OW_liq_Qm = OW_Oref_Qm + OW_Wref_Qm;
GOW_liq_Qm = GOW_Wref_Qv + GOW_Oref_Qv;

%Singlephase RMS plot
figure('Name','Singlephase RMS plot')
subplot(6,1,1)
plot([G_ae1_rms O_ae1_rms W_ae1_rms]);
hold on
plot([G_ae3_rms O_ae3_rms W_ae3_rms]);
plot([G_ae4_rms O_ae4_rms W_ae4_rms]);
hold off
title('Measurements');
legend('Sensor 1','Sensor 3','Sensor 4');
ylabel('[Vrms]');
grid on
set(gca,'xtick',[1:33],'xticklabel',{'G02';'G03';'G04';'G05';'G06';'G07';'G08';'G0
9';'G10';'G11';'OC01';'OC02';'OC03';'OC04';'OC07';'OC08';'OC09';'OC10';'OC11';'OC1
2';'OC13';'OC14';'OC16';'OC17';'OC18';'W01';'W02';'W03';'W08';'W09';'W10';'W11';'W
12'});
subplot(6,1,2)
plot([G_Gref_Qv O_Oref_Qv W_Wref_Qv]);
title('Flow references');
legend('Total flow');
ylabel('[m3/h]');
grid on
set(gca,'xtick',[1:33],'xticklabel',{'G02';'G03';'G04';'G05';'G06';'G07';'G08';'G0
9';'G10';'G11';'OC01';'OC02';'OC03';'OC04';'OC07';'OC08';'OC09';'OC10';'OC11';'OC1
2';'OC13';'OC14';'OC16';'OC17';'OC18';'W01';'W02';'W03';'W08';'W09';'W10';'W11';'W
12'});
subplot(6,1,3)
plot([G_HIC O_HIC W_HIC]);
title('Choke valve');
legend('Valve opening');
ylabel('[%]');
grid on
set(gca,'xtick',[1:33],'xticklabel',{'G02';'G03';'G04';'G05';'G06';'G07';'G08';'G0
9';'G10';'G11';'OC01';'OC02';'OC03';'OC04';'OC07';'OC08';'OC09';'OC10';'OC11';'OC1
2';'OC13';'OC14';'OC16';'OC17';'OC18';'W01';'W02';'W03';'W08';'W09';'W10';'W11';'W
12'});
subplot(6,1,4)
plot([G_tot_dp O_tot_dp W_tot_dp]);
title('Total differential pressure');
legend('Total dp');
ylabel('[bar]');
grid on
set(gca,'xtick',[1:33],'xticklabel',{'G02';'G03';'G04';'G05';'G06';'G07';'G08';'G0
9';'G10';'G11';'OC01';'OC02';'OC03';'OC04';'OC07';'OC08';'OC09';'OC10';'OC11';'OC1
```

```
2';'OC13';'OC14';'OC16';'OC17';'OC18';'W01';'W02';'W03';'W08';'W09';'W10';'W11';'W
12'});
subplot(6,1,5)
plot([G_wikaDp1 O_wikaDp1 W_wikaDp1]);
hold on
plot([G_wikaDp2 O_wikaDp2 W_wikaDp2]);
plot([G_wikaDp3 O_wikaDp3 W_wikaDp3]);
title('Wika venturi');
legend('dp1','dp2','dp3');
ylabel('[mbar]');
grid on
set(gca,'xtick',[1:33],'xticklabel',{'G02';'G03';'G04';'G05';'G06';'G07';'G08';'G0
9';'G10';'G11';'OC01';'OC02';'OC03';'OC04';'OC07';'OC08';'OC09';'OC10';'OC11';'OC1
2';'OC13';'OC14';'OC16';'OC17';'OC18';'W01';'W02';'W03';'W08';'W09';'W10';'W11';'W
12'});
subplot(6,1,6)
plot([G_EmcoDp1 O_EmcoDp1 W_EmcoDp1]);
hold on
plot([G_EmcoDp2 O_EmcoDp2 W_EmcoDp2]);
title('Emco venturi');
legend('dp1','dp2');
ylabel('[mbar]');
grid on
set(gca,'xtick',[1:33],'xticklabel',{'G02';'G03';'G04';'G05';'G06';'G07';'G08';'G0
9';'G10';'G11';'OC01';'OC02';'OC03';'OC04';'OC07';'OC08';'OC09';'OC10';'OC11';'OC1
2';'OC13';'OC14';'OC16';'OC17';'OC18';'W01';'W02';'W03';'W08';'W09';'W10';'W11';'W
12'});

%Water and oil RMS plot
figure('Name','OW RMS plot')
subplot(6,1,1)
plot(OW_ae1_rms);
hold on
plot(OW_ae3_rms);
plot(OW_ae4_rms);
hold off
grid on
title('Measurements');
legend('Sensor 1','Sensor 3','Sensor 4');
ylabel('[Vrms]');
grid on
set(gca,'xtick',[1:21],'xticklabel',{'OW01';'OW02';'OW31';'OW04';'OW05';'OW06';'OW
17';'OW19';'OW22';'OW23';'OWC10';'OWC11';'OWC12';'OWC13';'OWC14';'OWC15';'OWC16';'
OWC17';'OWC18';'OWC19';'OWC20';});
subplot(6,1,2)
plot(OW_Wref_Qv);
hold on
plot(OW_Oref_Qv);
plot(OW_tot_Qv);
title('Flow references');
legend('Water','Oil','Total');
ylabel('[m3/h]');
set(gca,'xtick',[1:21],'xticklabel',{'OW01';'OW02';'OW31';'OW04';'OW05';'OW06';'OW
17';'OW19';'OW22';'OW23';'OWC10';'OWC11';'OWC12';'OWC13';'OWC14';'OWC15';'OWC16';'
OWC17';'OWC18';'OWC19';'OWC20';});
subplot(6,1,3)
plot(OW_HIC);
```

```
title('Choke valve');
legend('Valve opening');
ylim([0,110]);
ylabel('[%]');
grid on
set(gca,'xtick',[1:21],'xticklabel',{'OW01';'OW02';'OW31';'OW04';'OW05';'OW06';'OW
17';'OW19';'OW22';'OW23';'OWC10';'OWC11';'OWC12';'OWC13';'OWC14';'OWC15';'OWC16';'
OWC17';'OWC18';'OWC19';'OWC20';});
subplot(6,1,4)
plot(OW_tot_dp);
title('Total differential pressure');
legend('Total dp');
ylabel('[bar]');
grid on
set(gca,'xtick',[1:21],'xticklabel',{'OW01';'OW02';'OW31';'OW04';'OW05';'OW06';'OW
17';'OW19';'OW22';'OW23';'OWC10';'OWC11';'OWC12';'OWC13';'OWC14';'OWC15';'OWC16';'
OWC17';'OWC18';'OWC19';'OWC20';});
subplot(6,1,5)
plot(OW_wikaDp1);
hold on
plot(OW_wikaDp2);
plot(OW_wikaDp3);
title('Wika venturi');
legend('dp1','dp2','dp3');
ylabel('[mbar]');
grid on
set(gca,'xtick',[1:21],'xticklabel',{'OW01';'OW02';'OW31';'OW04';'OW05';'OW06';'OW
17';'OW19';'OW22';'OW23';'OWC10';'OWC11';'OWC12';'OWC13';'OWC14';'OWC15';'OWC16';'
OWC17';'OWC18';'OWC19';'OWC20';});
subplot(6,1,6)
plot(OW_EmcoDp1);
hold on
plot(OW_EmcoDp2);
title('Emco venturi');
legend('dp1','dp2');
ylabel('[mbar]');
grid on
set(gca,'xtick',[1:21],'xticklabel',{'OW01';'OW02';'OW31';'OW04';'OW05';'OW06';'OW
17';'OW19';'OW22';'OW23';'OWC10';'OWC11';'OWC12';'OWC13';'OWC14';'OWC15';'OWC16';'
OWC17';'OWC18';'OWC19';'OWC20';});

%Gas and oil RMS plot
figure('Name','GO RMS plot')
subplot(6,1,1)
plot(GO_ae1_rms);
hold on
plot(GO_ae3_rms);
plot(GO_ae4_rms);
hold off
title('Measurements');
legend('Sensor 1','Sensor 3','Sensor 4');
ylabel('[Vrms]');
grid on
set(gca,'xtick',[1:23],'xticklabel',{'GO01';'GO01C';'GO02C';'GO03C';'GO04C';'GO05'
;'GO05C';'GO06';'GO09';'GO17';'GO18';'GO19';'GO22';'GOC11';'GOC12';'GOC13';'GOC14'
;'GOC15';'GOC16';'GOC17';'GOC18';'GOC19';'GOC20';});
subplot(6,1,2)
```

```
plot(GO_Wref_Qv);
hold on
plot(GO_Oref_Qv);
plot(GO_Gref_Qv);
plot(GO_tot_Qv);
title('Flow references');
legend('Water','Oil','Gas','Total');
ylabel('[m3/h]');
grid on
set(gca,'xtick',[1:23],'xticklabel',{'GO01';'GO01C';'GO02C';'GO03C';'GO04C';'GO05'
;'GO05C';'GO06';'GO09';'GO17';'GO18';'GO19';'GO22';'GOC11';'GOC12';'GOC13';'GOC14'
;'GOC15';'GOC16';'GOC17';'GOC18';'GOC19';'GOC20';});
subplot(6,1,3)
plot(GO_HIC);
title('Choke valve');
legend('Valve opening');
ylabel('[%]');
grid on
set(gca,'xtick',[1:23],'xticklabel',{'GO01';'GO01C';'GO02C';'GO03C';'GO04C';'GO05'
;'GO05C';'GO06';'GO09';'GO17';'GO18';'GO19';'GO22';'GOC11';'GOC12';'GOC13';'GOC14'
;'GOC15';'GOC16';'GOC17';'GOC18';'GOC19';'GOC20';});
subplot(6,1,4)
plot(GO_tot_dp);
title('Total differential pressure');
legend('Total dp');
ylabel('[bar]');
grid on
set(gca,'xtick',[1:23],'xticklabel',{'GO01';'GO01C';'GO02C';'GO03C';'GO04C';'GO05'
;'GO05C';'GO06';'GO09';'GO17';'GO18';'GO19';'GO22';'GOC11';'GOC12';'GOC13';'GOC14'
;'GOC15';'GOC16';'GOC17';'GOC18';'GOC19';'GOC20';});
subplot(6,1,5)
plot(GO_wikaDp1);
hold on
plot(GO_wikaDp2);
plot(GO_wikaDp3);
title('Wika venturi');
legend('dp1','dp2','dp3');
ylabel('[mbar]');
grid on
set(gca,'xtick',[1:23],'xticklabel',{'GO01';'GO01C';'GO02C';'GO03C';'GO04C';'GO05'
;'GO05C';'GO06';'GO09';'GO17';'GO18';'GO19';'GO22';'GOC11';'GOC12';'GOC13';'GOC14'
;'GOC15';'GOC16';'GOC17';'GOC18';'GOC19';'GOC20';});
subplot(6,1,6)
plot(GO_EmcoDp1);
hold on
plot(GO_EmcoDp2);
title('Emco venturi');
legend('dp1','dp2');
ylabel('[mbar]');
grid on
set(gca,'xtick',[1:23],'xticklabel',{'GO01';'GO01C';'GO02C';'GO03C';'GO04C';'GO05'
;'GO05C';'GO06';'GO09';'GO17';'GO18';'GO19';'GO22';'GOC11';'GOC12';'GOC13';'GOC14'
;'GOC15';'GOC16';'GOC17';'GOC18';'GOC19';'GOC20';});

%Gas and water RMS plot
figure('Name','GW RMS plot')
subplot(6,1,1)
```

```
plot(GW_ae1_rms);
hold on
plot(GW_ae3_rms);
plot(GW_ae4_rms);
hold off
title('Measurements');
legend('Sensor 1','Sensor 3','Sensor 4');
ylabel('[Vrms]');
grid on
set(gca,'xtick',[1:6],'xticklabel',{'GW30';'GW31';'GW32';'GW33';'GW34';'GW35';});
subplot(6,1,2)
plot(GW_Wref_Qv);
hold on
plot(GW_Gref_Qv);
plot(GW_tot_Qv);
title('Flow references');
legend('Water','Gas','Total');
ylabel('[m3/h]');
grid on
set(gca,'xtick',[1:6],'xticklabel',{'GW30';'GW31';'GW32';'GW33';'GW34';'GW35';});
subplot(6,1,3)
plot(GW_HIC);
title('Choke valve');
legend('Valve opening');
ylim([0,110]);
ylabel('[%]');
grid on
set(gca,'xtick',[1:6],'xticklabel',{'GW30';'GW31';'GW32';'GW33';'GW34';'GW35';});
subplot(6,1,4)
plot(GW_tot_dp);
title('Differential pressure');
legend('Total dp');
ylabel('[bar]');
grid on
set(gca,'xtick',[1:6],'xticklabel',{'GW30';'GW31';'GW32';'GW33';'GW34';'GW35';});
subplot(6,1,5)
plot(GW_wikaDp1);
hold on
plot(GW_wikaDp2);
plot(GW_wikaDp3);
title('Wika venturi');
legend('dp1','dp2','dp3');
ylabel('[mbar]');
grid on
set(gca,'xtick',[1:6],'xticklabel',{'GW30';'GW31';'GW32';'GW33';'GW34';'GW35';});
subplot(6,1,6)
plot(GW_EmcoDp1);
hold on
plot(GW_EmcoDp2);
title('Emco venturi');
legend('dp1','dp2');
ylabel('[mbar]');
grid on
set(gca,'xtick',[1:6],'xticklabel',{'GW30';'GW31';'GW32';'GW33';'GW34';'GW35';});

%Gas oil and water RMS plot
figure('Name','GOW RMS plot')
```

```
subplot(6,1,1)
plot(GOW_ae1_rms);
hold on
plot(GOW_ae3_rms);
plot(GOW_ae4_rms);
hold off
title('Measurements');
legend('Sensor 1','Sensor 3','Sensor 4');
ylabel('[Vrms]');
grid on
set(gca,'xtick',[1:17],'xticklabel',{'GOW1';'GOW2';'GOW3';'GOW4';'GOW5';'GOW6';'GO
W7';'GOW8';'GOW9';'GOW10';'GOW102';'GOW102';'GOW103';'GOW104';'GOW105';'GOW106';'G
OW107';});
subplot(6,1,2)
plot(GOW_Wref_Qv);
hold on
plot(GOW_Oref_Qv);
plot(GOW_Gref_Qv);
plot(GOW_tot_Qv);
title('Flow references');
legend('Water','Oil','Gas','Total');
ylabel('[m3/h]');
grid on
set(gca,'xtick',[1:17],'xticklabel',{'GOW1';'GOW2';'GOW3';'GOW4';'GOW5';'GOW6';'GO
W7';'GOW8';'GOW9';'GOW10';'GOW102';'GOW102';'GOW103';'GOW104';'GOW105';'GOW106';'G
OW107';});
subplot(6,1,3)
plot(GOW_HIC);
title('Choke valve');
legend('Valve opening');
ylim([0,110]);
ylabel('[%]');
grid on
set(gca,'xtick',[1:17],'xticklabel',{'GOW1';'GOW2';'GOW3';'GOW4';'GOW5';'GOW6';'GO
W7';'GOW8';'GOW9';'GOW10';'GOW102';'GOW102';'GOW103';'GOW104';'GOW105';'GOW106';'G
OW107';});
subplot(6,1,4)
plot(GOW_tot_dp);
title('Differential pressure');
legend('Total dp');
ylabel('[bar]');
grid on
set(gca,'xtick',[1:17],'xticklabel',{'GOW1';'GOW2';'GOW3';'GOW4';'GOW5';'GOW6';'GO
W7';'GOW8';'GOW9';'GOW10';'GOW102';'GOW102';'GOW103';'GOW104';'GOW105';'GOW106';'G
OW107';});
subplot(6,1,5)
plot(GOW_wikaDp1);
hold on
plot(GOW_wikaDp2);
plot(GOW_wikaDp3);
title('Wika venturi');
legend('dp1','dp2','dp3');
ylabel('[mbar]');
grid on
set(gca,'xtick',[1:17],'xticklabel',{'GOW1';'GOW2';'GOW3';'GOW4';'GOW5';'GOW6';'GO
W7';'GOW8';'GOW9';'GOW10';'GOW102';'GOW102';'GOW103';'GOW104';'GOW105';'GOW106';'G
OW107';});
```

```
subplot(6,1,6)
plot(GOW_EmcoDp1);
hold on
plot(GOW_EmcoDp2);
title('Emco venturi');
legend('dp1','dp2');
ylabel('[mbar]');
grid on
set(gca,'xtick',[1:17],'xticklabel',{'GOW1';'GOW2';'GOW3';'GOW4';'GOW5';'GOW6';'GO
W7';'GOW8';'GOW9';'GOW10';'GOW102';'GOW102';'GOW103';'GOW104';'GOW105';'GOW106';'G
OW107';});

%FREQUENCY ALANSIS-------------------------------------------------------------

%Frequency vs. time plots
figure('Name','Hamming window selection for Pwelch method')
subplot(2,1,1);
pwelch(GO_ae1_data,[],[],[],fs);
title('Power spectral density estimate of sensor 1');
subplot(2,1,2);
pwelch(GO_ae1_data,hamming(5000),[],[],fs);
hold on;
pwelch(GO_ae1_data,hamming(500),[],[],fs);
pwelch(GO_ae1_data,hamming(100),[],[],fs);
title('Power spectral density estimate of sensor 1 with Hamming window');
legend('L = 5000', 'L = 500', 'L = 100');

figure('Name','Hamming window selection for Spectogram')
subplot(2,1,1);
spectrogram(GO_ae1_data,[],[],[],fs);
title('Spectrogram using short-time Fourier transform on sensor 1');
subplot(2,1,2);
spectrogram(GO_ae1_data,hamming(500),[],[],fs);
title('Spectrogram using short-time Fourier transform on sensor 1 with Hamming
window (L = 500)');

figure('Name','GO frequency analysis')
subplot(3,3,1)
plot(GO_Xvector, GO_ae1_data);
title('Time series plot of the raw value of sensor 1');
xlabel('Time (ms)');
ylabel('Voltage (V)');
ylim([-3 3]);
subplot(3,3,2)
plot(GO_Xvector, GO_ae3_data);
title('Time series plot of the raw value of sensor 3');
xlabel('Time (ms)');
ylabel('Voltage (V)');
ylim([-3 3]);
subplot(3,3,3)
plot(GO_Xvector, GO_ae4_data);
title('Time series plot of the raw value of sensor 4');
xlabel('Time (ms)');
ylabel('Voltage (V)');
ylim([-3 3]);
subplot(3,3,4)
pwelch(GO_ae1_data,hamming(L),[],[],fs);
```

```
title('Power spectral density estimate of sensor 1');
ylim([-120 -30]);
subplot(3,3,5)
pwelch(GO_ae3_data,hamming(L),[],[],fs);
title('Power spectral density estimate of sensor 3');
ylim([-120 -30]);
subplot(3,3,6)
pwelch(GO_ae4_data,hamming(L),[],[],fs);
title('Power spectral density estimate of sensor 4');
ylim([-120 -30]);
subplot(3,3,7)
spectrogram(GO_ae1_data,hamming(L),[],[],fs);
title('Spectrogram using short-time Fourier transform on sensor 1');
subplot(3,3,8)
spectrogram(GO_ae3_data,hamming(L),[],[],fs);
title('Spectrogram using short-time Fourier transform on sensor 3');
subplot(3,3,9)
spectrogram(GO_ae4_data,hamming(L),[],[],fs);
title('Spectrogram using short-time Fourier transform on sensor 4');

figure('Name','GW frequency analysis')
subplot(3,3,1)
plot(GW_Xvector, GW_ae1_data);
title('Time series plot of the raw value of sensor 1');
xlabel('Time (ms)');
ylabel('Voltage (V)');
ylim([-3 3]);
subplot(3,3,2)
plot(GW_Xvector, GW_ae3_data);
title('Time series plot of the raw value of sensor 3');
xlabel('Time (ms)');
ylabel('Voltage (V)');
ylim([-3 3]);
subplot(3,3,3)
plot(GW_Xvector, GW_ae4_data);
title('Time series plot of the raw value of sensor 4');
xlabel('Time (ms)');
ylabel('Voltage (V)');
ylim([-3 3]);
subplot(3,3,4)
pwelch(GW_ae1_data,hamming(L),[],[],fs);
title('Power spectral density estimate of sensor 1');
ylim([-120 -30]);
subplot(3,3,5)
pwelch(GW_ae3_data,hamming(L),[],[],fs);
title('Power spectral density estimate of sensor 3');
ylim([-120 -30]);
subplot(3,3,6)
pwelch(GW_ae4_data,hamming(L),[],[],fs);
title('Power spectral density estimate of sensor 4');
ylim([-120 -30]);
subplot(3,3,7)
spectrogram(GW_ae1_data,hamming(L),[],[],fs);
title('Spectrogram using short-time Fourier transform on sensor 1');
subplot(3,3,8)
spectrogram(GW_ae3_data,hamming(L),[],[],fs);
title('Spectrogram using short-time Fourier transform on sensor 3');
```

```
subplot(3,3,9)
spectrogram(GW_ae4_data,hamming(L),[],[],fs);
title('Spectrogram using short-time Fourier transform on sensor 4');

figure('Name','OW frequency analysis')
subplot(3,3,1)
plot(OW_Xvector, OW_ae1_data);
title('Time series plot of the raw value of sensor 1');
xlabel('Time (ms)');
ylabel('Voltage (V)');
ylim([-3 3]);
subplot(3,3,2)
plot(OW_Xvector, OW_ae3_data);
title('Time series plot of the raw value of sensor 3');
xlabel('Time (ms)');
ylabel('Voltage (V)');
ylim([-3 3]);
subplot(3,3,3)
plot(OW_Xvector, OW_ae4_data);
title('Time series plot of the raw value of sensor 4');
xlabel('Time (ms)');
ylabel('Voltage (V)');
ylim([-3 3]);
subplot(3,3,4)
pwelch(OW_ae1_data,hamming(L),[],[],fs);
title('Power spectral density estimate of sensor 1');
ylim([-120 -30]);
subplot(3,3,5)
pwelch(OW_ae3_data,hamming(L),[],[],fs);
ylim([-120 -30]);
title('Power spectral density estimate of sensor 3');
subplot(3,3,6)
pwelch(OW_ae4_data,hamming(L),[],[],fs);
title('Power spectral density estimate of sensor 4');
ylim([-120 -30]);
subplot(3,3,7)
spectrogram(OW_ae1_data,hamming(L),[],[],fs);
title('Spectrogram using short-time Fourier transform on sensor 1');
subplot(3,3,8)
spectrogram(OW_ae3_data,hamming(L),[],[],fs);
title('Spectrogram using short-time Fourier transform on sensor 3');
subplot(3,3,9)
spectrogram(OW_ae4_data,hamming(L),[],[],fs);
title('Spectrogram using short-time Fourier transform on sensor 4');

figure('Name','GOW frequency plot')
subplot(3,3,1)
plot(GOW_Xvector, GOW_ae1_data);
title('Time series plot of the raw value of sensor 1');
xlabel('Time (ms)');
ylabel('Voltage (V)');
ylim([-3 3]);
subplot(3,3,2)
plot(GOW_Xvector, GOW_ae3_data);
title('Time series plot of the raw value of sensor 3');
xlabel('Time (ms)');
ylabel('Voltage (V)');
```

```
ylim([-3 3]);
subplot(3,3,3)
plot(GOW_Xvector, GOW_ae4_data);
title('Time series plot of the raw value of sensor 4');
xlabel('Time (ms)');
ylabel('Voltage (V)');
ylim([-3 3]);
subplot(3,3,4)
pwelch(GOW_ae1_data,hamming(L),[],[],fs);
title('Power spectral density estimate of sensor 1');
ylim([-120 -30]);
subplot(3,3,5)
pwelch(GOW_ae3_data,hamming(L),[],[],fs);
title('Power spectral density estimate of sensor 3');
ylim([-120 -30]);
subplot(3,3,6)
pwelch(GOW_ae4_data,hamming(L),[],[],fs);
title('Power spectral density estimate of sensor 4');
ylim([-120 -30]);
subplot(3,3,7)
spectrogram(GOW_ae1_data,hamming(L),[],[],fs);
title('Spectrogram using short-time Fourier transform on sensor 1');
subplot(3,3,8)
spectrogram(GOW_ae3_data,hamming(L),[],[],fs);
title('Spectrogram using short-time Fourier transform on sensor 3');
subplot(3,3,9)
spectrogram(GOW_ae4_data,hamming(L),[],[],fs);
title('Spectrogram using short-time Fourier transform on sensor 4');

figure('Name','Plot')
scatter(GO_Gref_Qm, GO_liq_Qm, 'red');
hold on;
scatter(GW_Gref_Qm, GW_liq_Qm, 'green');
scatter(GOW_Gref_Qm, GOW_liq_Qm, 'blue');
hold off;
legend('GO','GW','GOW');
ylabel('Liquid [kg/min]');
xlabel('Gas [kg/min]');
set(gca,'xscale','log');
set(gca,'yscale','log');
yline(13.5);
xline(3);
grid minor
```

Appendix H

Preparing the gas flow experiment data from dataset 1.

```
clc;
clear all;

% %Load dataset
G02 = load('G02.mat');
G03 = load('G03.mat');
G04 = load('G04.mat');
G05 = load('G05.mat');
G06 = load('G06.mat');
G07 = load('G07.mat');
G08 = load('G08.mat');
G09 = load('G09.mat');
G10 = load('G10.mat');
G11 = load('G11.mat');

%Choose samples
start = 7705601;
stop = 7707600;

%Samples
G02_ae1_data = G02.testPoint.data(1).vData.ch(1).data(start:stop);
G03_ae1_data = G03.testPoint.data(1).vData.ch(1).data(start:stop);
G04_ae1_data = G04.testPoint.data(1).vData.ch(1).data(start:stop);
G05_ae1_data = G05.testPoint.data(1).vData.ch(1).data(start:stop);
G06_ae1_data = G06.testPoint.data(1).vData.ch(1).data(start:stop);
G07_ae1_data = G07.testPoint.data(1).vData.ch(1).data(start:stop);
G08_ae1_data = G08.testPoint.data(1).vData.ch(1).data(start:stop);
G09_ae1_data = G09.testPoint.data(1).vData.ch(1).data(start:stop);
G10_ae1_data = G10.testPoint.data(1).vData.ch(1).data(start:stop);
G11_ae1_data = G11.testPoint.data(1).vData.ch(1).data(start:stop);
G02_ae3_data = G02.testPoint.data(1).vData.ch(3).data(start:stop);
G03_ae3_data = G03.testPoint.data(1).vData.ch(3).data(start:stop);
G04_ae3_data = G04.testPoint.data(1).vData.ch(3).data(start:stop);
G05_ae3_data = G05.testPoint.data(1).vData.ch(3).data(start:stop);
G06_ae3_data = G06.testPoint.data(1).vData.ch(3).data(start:stop);
G07_ae3_data = G07.testPoint.data(1).vData.ch(3).data(start:stop);
G08_ae3_data = G08.testPoint.data(1).vData.ch(3).data(start:stop);
G09_ae3_data = G09.testPoint.data(1).vData.ch(3).data(start:stop);
G10_ae3_data = G10.testPoint.data(1).vData.ch(3).data(start:stop);
G11_ae3_data = G11.testPoint.data(1).vData.ch(3).data(start:stop);
G02_ae4_data = G02.testPoint.data(1).vData.ch(4).data(start:stop);
G03_ae4_data = G03.testPoint.data(1).vData.ch(4).data(start:stop);
G04_ae4_data = G04.testPoint.data(1).vData.ch(4).data(start:stop);
G05_ae4_data = G05.testPoint.data(1).vData.ch(4).data(start:stop);
G06_ae4_data = G06.testPoint.data(1).vData.ch(4).data(start:stop);
G07_ae4_data = G07.testPoint.data(1).vData.ch(4).data(start:stop);
G08_ae4_data = G08.testPoint.data(1).vData.ch(4).data(start:stop);
G09_ae4_data = G09.testPoint.data(1).vData.ch(4).data(start:stop);
G10_ae4_data = G10.testPoint.data(1).vData.ch(4).data(start:stop);
G11_ae4_data = G11.testPoint.data(1).vData.ch(4).data(start:stop);

% %Calculate RMS
```

```
G02_ae1_rms = rms(G02_ae1_data);
G03_ae1_rms = rms(G03_ae1_data);
G04_ae1_rms = rms(G04_ae1_data);
G05_ae1_rms = rms(G05_ae1_data);
G06_ae1_rms = rms(G06_ae1_data);
G07_ae1_rms = rms(G07_ae1_data);
G08_ae1_rms = rms(G08_ae1_data);
G09_ae1_rms = rms(G09_ae1_data);
G10_ae1_rms = rms(G10_ae1_data);
G11_ae1_rms = rms(G11_ae1_data);
G02_ae3_rms = rms(G02_ae3_data);
G03_ae3_rms = rms(G03_ae3_data);
G04_ae3_rms = rms(G04_ae3_data);
G05_ae3_rms = rms(G05_ae3_data);
G06_ae3_rms = rms(G06_ae3_data);
G07_ae3_rms = rms(G07_ae3_data);
G08_ae3_rms = rms(G08_ae3_data);
G09_ae3_rms = rms(G09_ae3_data);
G10_ae3_rms = rms(G10_ae3_data);
G11_ae3_rms = rms(G11_ae3_data);
G02_ae4_rms = rms(G02_ae4_data);
G03_ae4_rms = rms(G03_ae4_data);
G04_ae4_rms = rms(G04_ae4_data);
G05_ae4_rms = rms(G05_ae4_data);
G06_ae4_rms = rms(G06_ae4_data);
G07_ae4_rms = rms(G07_ae4_data);
G08_ae4_rms = rms(G08_ae4_data);
G09_ae4_rms = rms(G09_ae4_data);
G10_ae4_rms = rms(G10_ae4_data);
G11_ae4_rms = rms(G11_ae4_data);

%Make lists of acoustic measurements
G_ae1_rms = [G02_ae1_rms G03_ae1_rms G04_ae1_rms G05_ae1_rms G06_ae1_rms
G07_ae1_rms G08_ae1_rms G09_ae1_rms G10_ae1_rms G11_ae1_rms];
G_ae3_rms = [G02_ae3_rms G03_ae3_rms G04_ae3_rms G05_ae3_rms G06_ae3_rms
G07_ae3_rms G08_ae3_rms G09_ae3_rms G10_ae3_rms G11_ae3_rms];
G_ae4_rms = [G02_ae4_rms G03_ae4_rms G04_ae4_rms G05_ae4_rms G06_ae4_rms
G07_ae4_rms G08_ae4_rms G09_ae4_rms G10_ae4_rms G11_ae4_rms];

%Make lists of flow measurements
G_Gref_Qv = [G02.testPoint.gasRef.q G03.testPoint.gasRef.q G04.testPoint.gasRef.q
G05.testPoint.gasRef.q G06.testPoint.gasRef.q G07.testPoint.gasRef.q
G08.testPoint.gasRef.q G09.testPoint.gasRef.q G10.testPoint.gasRef.q
G11.testPoint.gasRef.q];

%Make lists of valve positions and pressure measurements
G_HIC = [G02.testPoint.HIC.pos G03.testPoint.HIC.pos G04.testPoint.HIC.pos
G05.testPoint.HIC.pos G06.testPoint.HIC.pos G07.testPoint.HIC.pos
G08.testPoint.HIC.pos G09.testPoint.HIC.pos G10.testPoint.HIC.pos
G11.testPoint.HIC.pos];
G_Pi = [G02.testPoint.press.in G03.testPoint.press.in G04.testPoint.press.in
G05.testPoint.press.in G06.testPoint.press.in G07.testPoint.press.in
G08.testPoint.press.in G09.testPoint.press.in G10.testPoint.press.in
G11.testPoint.press.in];
G_Po = [G02.testPoint.press.out G03.testPoint.press.out G04.testPoint.press.out
G05.testPoint.press.out G06.testPoint.press.out G07.testPoint.press.out
```

```
G08.testPoint.press.out G09.testPoint.press.out G10.testPoint.press.out
G11.testPoint.press.out];
G_wikaDp1 = [G02.testPoint.Wika.dp1 G03.testPoint.Wika.dp1 G04.testPoint.Wika.dp1
G05.testPoint.Wika.dp1 G06.testPoint.Wika.dp1 G07.testPoint.Wika.dp1
G08.testPoint.Wika.dp1 G09.testPoint.Wika.dp1 G10.testPoint.Wika.dp1
G11.testPoint.Wika.dp1];
G_wikaDp2 = [G02.testPoint.Wika.dp2 G03.testPoint.Wika.dp1 G04.testPoint.Wika.dp2
G05.testPoint.Wika.dp2 G06.testPoint.Wika.dp2 G07.testPoint.Wika.dp2
G08.testPoint.Wika.dp2 G09.testPoint.Wika.dp2 G10.testPoint.Wika.dp2
G11.testPoint.Wika.dp2];
G_wikaDp3 = [G02.testPoint.Wika.dp3 G03.testPoint.Wika.dp1 G04.testPoint.Wika.dp3
G05.testPoint.Wika.dp3 G06.testPoint.Wika.dp3 G07.testPoint.Wika.dp3
G08.testPoint.Wika.dp3 G09.testPoint.Wika.dp3 G10.testPoint.Wika.dp3
G11.testPoint.Wika.dp3];
G_EmcoDp1 = [G02.testPoint.Emco.dp1 G03.testPoint.Wika.dp1 G04.testPoint.Emco.dp1
G05.testPoint.Emco.dp1 G06.testPoint.Emco.dp1 G07.testPoint.Emco.dp1
G08.testPoint.Emco.dp1 G09.testPoint.Emco.dp1 G10.testPoint.Emco.dp1
G11.testPoint.Emco.dp1];
G_EmcoDp2 = [G02.testPoint.Emco.dp2 G03.testPoint.Wika.dp1 G04.testPoint.Emco.dp2
G05.testPoint.Emco.dp2 G06.testPoint.Emco.dp2 G07.testPoint.Emco.dp2
G08.testPoint.Emco.dp2 G09.testPoint.Emco.dp2 G10.testPoint.Emco.dp2
G11.testPoint.Emco.dp2];
```

Appendix I

Preparing GOW flow experiment data from dataset 1.

```
clc;
clear all;

% Load data
GOW01 = load('GOW01.mat');
GOW02 = load('GOW02.mat');
GOW03 = load('GOW03.mat');
GOW04 = load('GOW04.mat');
GOW05 = load('GOW05.mat');
GOW06 = load('GOW06.mat');
GOW07 = load('GOW07.mat');
GOW08 = load('GOW08.mat');
GOW09 = load('GOW09.mat');
GOW10 = load('GOW10.mat');
GOW101 = load('GOW101.mat');
GOW102 = load('GOW102.mat');
GOW103 = load('GOW103.mat');
GOW104 = load('GOW104.mat');
GOW105 = load('GOW105.mat');
GOW106 = load('GOW106.mat');
GOW107 = load('GOW107.mat');

%Choose samples
start = 7705601;
stop = 7707600;

%Samples
GOW01_ae1_data = GOW01.testPoint.data(1).vData.ch(1).data(start:stop);
GOW02_ae1_data = GOW02.testPoint.data(1).vData.ch(1).data(start:stop);
GOW03_ae1_data = GOW03.testPoint.data(1).vData.ch(1).data(start:stop);
GOW04_ae1_data = GOW04.testPoint.data(1).vData.ch(1).data(start:stop);
GOW05_ae1_data = GOW05.testPoint.data(1).vData.ch(1).data(start:stop);
GOW06_ae1_data = GOW06.testPoint.data(1).vData.ch(1).data(start:stop);
GOW07_ae1_data = GOW07.testPoint.data(1).vData.ch(1).data(start:stop);
GOW08_ae1_data = GOW08.testPoint.data(1).vData.ch(1).data(start:stop);
GOW09_ae1_data = GOW09.testPoint.data(1).vData.ch(1).data(start:stop);
GOW10_ae1_data = GOW10.testPoint.data(1).vData.ch(1).data(start:stop);
GOW101_ae1_data = GOW101.testPoint.data(1).vData.ch(1).data(start:stop);
GOW102_ae1_data = GOW102.testPoint.data(1).vData.ch(1).data(start:stop);
GOW103_ae1_data = GOW103.testPoint.data(1).vData.ch(1).data(start:stop);
GOW104_ae1_data = GOW104.testPoint.data(1).vData.ch(1).data(start:stop);
GOW105_ae1_data = GOW105.testPoint.data(1).vData.ch(1).data(start:stop);
GOW106_ae1_data = GOW106.testPoint.data(1).vData.ch(1).data(start:stop);
GOW107_ae1_data = GOW107.testPoint.data(1).vData.ch(1).data(start:stop);
GOW01_ae3_data = GOW01.testPoint.data(1).vData.ch(3).data(start:stop);
GOW02_ae3_data = GOW02.testPoint.data(1).vData.ch(3).data(start:stop);
GOW03_ae3_data = GOW03.testPoint.data(1).vData.ch(3).data(start:stop);
GOW04_ae3_data = GOW04.testPoint.data(1).vData.ch(3).data(start:stop);
GOW05_ae3_data = GOW05.testPoint.data(1).vData.ch(3).data(start:stop);
GOW06_ae3_data = GOW06.testPoint.data(1).vData.ch(3).data(start:stop);
GOW07_ae3_data = GOW07.testPoint.data(1).vData.ch(3).data(start:stop);
GOW08_ae3_data = GOW08.testPoint.data(1).vData.ch(3).data(start:stop);
```

```
GOW09_ae3_data = GOW09.testPoint.data(1).vData.ch(3).data(start:stop);
GOW10_ae3_data = GOW10.testPoint.data(1).vData.ch(3).data(start:stop);
GOW101_ae3_data = GOW101.testPoint.data(1).vData.ch(3).data(start:stop);
GOW102_ae3_data = GOW102.testPoint.data(1).vData.ch(3).data(start:stop);
GOW103_ae3_data = GOW103.testPoint.data(1).vData.ch(3).data(start:stop);
GOW104_ae3_data = GOW104.testPoint.data(1).vData.ch(3).data(start:stop);
GOW105_ae3_data = GOW105.testPoint.data(1).vData.ch(3).data(start:stop);
GOW106_ae3_data = GOW106.testPoint.data(1).vData.ch(3).data(start:stop);
GOW107_ae3_data = GOW107.testPoint.data(1).vData.ch(3).data(start:stop);
GOW01_ae4_data = GOW01.testPoint.data(1).vData.ch(4).data(start:stop);
GOW02_ae4_data = GOW02.testPoint.data(1).vData.ch(4).data(start:stop);
GOW03_ae4_data = GOW03.testPoint.data(1).vData.ch(4).data(start:stop);
GOW04_ae4_data = GOW04.testPoint.data(1).vData.ch(4).data(start:stop);
GOW05_ae4_data = GOW05.testPoint.data(1).vData.ch(4).data(start:stop);
GOW06_ae4_data = GOW06.testPoint.data(1).vData.ch(4).data(start:stop);
GOW07_ae4_data = GOW07.testPoint.data(1).vData.ch(4).data(start:stop);
GOW08_ae4_data = GOW08.testPoint.data(1).vData.ch(4).data(start:stop);
GOW09_ae4_data = GOW09.testPoint.data(1).vData.ch(4).data(start:stop);
GOW10_ae4_data = GOW10.testPoint.data(1).vData.ch(4).data(start:stop);
GOW101_ae4_data = GOW101.testPoint.data(1).vData.ch(4).data(start:stop);
GOW102_ae4_data = GOW102.testPoint.data(1).vData.ch(4).data(start:stop);
GOW103_ae4_data = GOW103.testPoint.data(1).vData.ch(4).data(start:stop);
GOW104_ae4_data = GOW104.testPoint.data(1).vData.ch(4).data(start:stop);
GOW105_ae4_data = GOW105.testPoint.data(1).vData.ch(4).data(start:stop);
GOW106_ae4_data = GOW106.testPoint.data(1).vData.ch(4).data(start:stop);
GOW107_ae4_data = GOW107.testPoint.data(1).vData.ch(4).data(start:stop);

%Calculate RMS
GOW01_ae1_rms = rms(GOW01_ae1_data);
GOW02_ae1_rms = rms(GOW02_ae1_data);
GOW03_ae1_rms = rms(GOW03_ae1_data);
GOW04_ae1_rms = rms(GOW04_ae1_data);
GOW05_ae1_rms = rms(GOW05_ae1_data);
GOW06_ae1_rms = rms(GOW06_ae1_data);
GOW07_ae1_rms = rms(GOW07_ae1_data);
GOW08_ae1_rms = rms(GOW08_ae1_data);
GOW09_ae1_rms = rms(GOW09_ae1_data);
GOW10_ae1_rms = rms(GOW10_ae1_data);
GOW101_ae1_rms = rms(GOW101_ae1_data);
GOW102_ae1_rms = rms(GOW102_ae1_data);
GOW103_ae1_rms = rms(GOW103_ae1_data);
GOW104_ae1_rms = rms(GOW104_ae1_data);
GOW105_ae1_rms = rms(GOW105_ae1_data);
GOW106_ae1_rms = rms(GOW106_ae1_data);
GOW107_ae1_rms = rms(GOW107_ae1_data);
GOW01_ae3_rms = rms(GOW01_ae3_data);
GOW02_ae3_rms = rms(GOW02_ae3_data);
GOW03_ae3_rms = rms(GOW03_ae3_data);
GOW04_ae3_rms = rms(GOW04_ae3_data);
GOW05_ae3_rms = rms(GOW05_ae3_data);
GOW06_ae3_rms = rms(GOW06_ae3_data);
GOW07_ae3_rms = rms(GOW07_ae3_data);
GOW08_ae3_rms = rms(GOW08_ae3_data);
GOW09_ae3_rms = rms(GOW09_ae3_data);
GOW10_ae3_rms = rms(GOW10_ae3_data);
GOW101_ae3_rms = rms(GOW101_ae3_data);
GOW102_ae3_rms = rms(GOW102_ae3_data);
```

```
GOW103_ae3_rms = rms(GOW103_ae3_data);
GOW104_ae3_rms = rms(GOW104_ae3_data);
GOW105_ae3_rms = rms(GOW105_ae3_data);
GOW106_ae3_rms = rms(GOW106_ae3_data);
GOW107_ae3_rms = rms(GOW107_ae3_data);
GOW01_ae4_rms = rms(GOW01_ae4_data);
GOW02_ae4_rms = rms(GOW02_ae4_data);
GOW03_ae4_rms = rms(GOW03_ae4_data);
GOW04_ae4_rms = rms(GOW04_ae4_data);
GOW05_ae4_rms = rms(GOW05_ae4_data);
GOW06_ae4_rms = rms(GOW06_ae4_data);
GOW07_ae4_rms = rms(GOW07_ae4_data);
GOW08_ae4_rms = rms(GOW08_ae4_data);
GOW09_ae4_rms = rms(GOW09_ae4_data);
GOW10_ae4_rms = rms(GOW10_ae4_data);
GOW101_ae4_rms = rms(GOW101_ae4_data);
GOW102_ae4_rms = rms(GOW102_ae4_data);
GOW103_ae4_rms = rms(GOW103_ae4_data);
GOW104_ae4_rms = rms(GOW104_ae4_data);
GOW105_ae4_rms = rms(GOW105_ae4_data);
GOW106_ae4_rms = rms(GOW106_ae4_data);
GOW107_ae4_rms = rms(GOW107_ae4_data);

%Make lists of acoustic measurements
GOW_ae1_data = [GOW01_ae1_data; GOW02_ae1_data; GOW03_ae1_data; GOW04_ae1_data;
GOW05_ae1_data; GOW06_ae1_data; GOW07_ae1_data; GOW08_ae1_data; GOW09_ae1_data;
GOW10_ae1_data; GOW101_ae1_data; GOW102_ae1_data; GOW103_ae1_data;
GOW104_ae1_data; GOW105_ae1_data; GOW106_ae1_data; GOW107_ae1_data;];
GOW_ae3_data = [GOW01_ae3_data; GOW02_ae3_data; GOW03_ae3_data; GOW04_ae3_data;
GOW05_ae3_data; GOW06_ae3_data; GOW07_ae3_data; GOW08_ae3_data; GOW09_ae3_data;
GOW10_ae3_data; GOW101_ae3_data; GOW102_ae3_data; GOW103_ae3_data;
GOW104_ae3_data; GOW105_ae3_data; GOW106_ae3_data; GOW107_ae3_data;];
GOW_ae4_data = [GOW01_ae4_data; GOW02_ae4_data; GOW03_ae4_data; GOW04_ae4_data;
GOW05_ae4_data; GOW06_ae4_data; GOW07_ae4_data; GOW08_ae4_data; GOW09_ae4_data;
GOW10_ae4_data; GOW101_ae4_data; GOW102_ae4_data; GOW103_ae4_data;
GOW104_ae4_data; GOW105_ae4_data; GOW106_ae4_data; GOW107_ae4_data;];
GOW_ae1_rms = [GOW01_ae1_rms GOW02_ae1_rms GOW03_ae1_rms GOW04_ae1_rms
GOW05_ae1_rms GOW06_ae1_rms GOW07_ae1_rms GOW08_ae1_rms GOW09_ae1_rms
GOW10_ae1_rms GOW101_ae1_rms GOW102_ae1_rms GOW103_ae1_rms GOW104_ae1_rms
GOW105_ae1_rms GOW106_ae1_rms GOW107_ae1_rms];
GOW_ae3_rms = [GOW01_ae3_rms GOW02_ae3_rms GOW03_ae3_rms GOW04_ae3_rms
GOW05_ae3_rms GOW06_ae3_rms GOW07_ae3_rms GOW08_ae3_rms GOW09_ae3_rms
GOW10_ae3_rms GOW101_ae3_rms GOW102_ae3_rms GOW103_ae3_rms GOW104_ae3_rms
GOW105_ae3_rms GOW106_ae3_rms GOW107_ae3_rms];
GOW_ae4_rms = [GOW01_ae4_rms GOW02_ae4_rms GOW03_ae4_rms GOW04_ae4_rms
GOW05_ae4_rms GOW06_ae4_rms GOW07_ae4_rms GOW08_ae4_rms GOW09_ae4_rms
GOW10_ae4_rms GOW101_ae4_rms GOW102_ae4_rms GOW103_ae4_rms GOW104_ae4_rms
GOW105_ae4_rms GOW106_ae4_rms GOW107_ae4_rms];

%Make lists of flow measurements
GOW_Wref_Qv = [GOW01.testPoint.watRef.q GOW02.testPoint.watRef.q
GOW03.testPoint.watRef.q GOW04.testPoint.watRef.q GOW05.testPoint.watRef.q
GOW06.testPoint.watRef.q GOW07.testPoint.watRef.q GOW08.testPoint.watRef.q
GOW09.testPoint.watRef.q GOW10.testPoint.watRef.q GOW101.testPoint.watRef.q
GOW102.testPoint.watRef.q GOW103.testPoint.watRef.q GOW104.testPoint.watRef.q
GOW105.testPoint.watRef.q GOW106.testPoint.watRef.q GOW107.testPoint.watRef.q];
```

```
GOW_Oref_Qv = [GOW01.testPoint.oilRef.q GOW02.testPoint.oilRef.q
GOW03.testPoint.oilRef.q GOW04.testPoint.oilRef.q GOW05.testPoint.oilRef.q
GOW06.testPoint.oilRef.q GOW07.testPoint.oilRef.q GOW08.testPoint.oilRef.q
GOW09.testPoint.oilRef.q GOW10.testPoint.oilRef.q GOW101.testPoint.oilRef.q
GOW102.testPoint.oilRef.q GOW103.testPoint.oilRef.q GOW104.testPoint.oilRef.q
GOW105.testPoint.oilRef.q GOW106.testPoint.oilRef.q GOW107.testPoint.oilRef.q];
GOW_Gref_Qv =
[10.60,10.39,10.53,10.17,10.00,9.97,10.52,10.05,9.76,10.54,14.71,25.89,31.35,20.63
,10.36,10.35,14.85];
%GOW_Gref_Qv = [GOW01.testPoint.gasRef.q GOW02.testPoint.gasRef.q
GOW03.testPoint.gasRef.q GOW04.testPoint.gasRef.q GOW05.testPoint.gasRef.q
GOW06.testPoint.gasRef.q GOW07.testPoint.gasRef.q GOW08.testPoint.gasRef.q
GOW09.testPoint.gasRef.q GOW10.testPoint.gasRef.q GOW102.testPoint.gasRef.q
GOW103.testPoint.gasRef.q GOW104.testPoint.gasRef.q GOW105.testPoint.gasRef.q
GOW106.testPoint.gasRef.q GOW107.testPoint.gasRef.q];

%Make lists of valve positions and pressure measurements
GOW_HIC = [GOW01.testPoint.HIC.pos GOW02.testPoint.HIC.pos GOW03.testPoint.HIC.pos
GOW04.testPoint.HIC.pos GOW05.testPoint.HIC.pos GOW06.testPoint.HIC.pos
GOW07.testPoint.HIC.pos GOW08.testPoint.HIC.pos GOW09.testPoint.HIC.pos
GOW10.testPoint.HIC.pos GOW101.testPoint.HIC.pos GOW102.testPoint.HIC.pos
GOW103.testPoint.HIC.pos GOW104.testPoint.HIC.pos GOW105.testPoint.HIC.pos
GOW106.testPoint.HIC.pos GOW107.testPoint.HIC.pos];
GOW_wikaDp1 = [GOW01.testPoint.Wika.dp1 GOW02.testPoint.Wika.dp1
GOW03.testPoint.Wika.dp1 GOW04.testPoint.Wika.dp1 GOW05.testPoint.Wika.dp1
GOW06.testPoint.Wika.dp1 GOW07.testPoint.Wika.dp1 GOW08.testPoint.Wika.dp1
GOW09.testPoint.Wika.dp1 GOW10.testPoint.Wika.dp1 GOW101.testPoint.Wika.dp1
GOW102.testPoint.Wika.dp1 GOW103.testPoint.Wika.dp1 GOW104.testPoint.Wika.dp1
GOW105.testPoint.Wika.dp1 GOW106.testPoint.Wika.dp1 GOW107.testPoint.Wika.dp1];
GOW_wikaDp2 = [GOW01.testPoint.Wika.dp2 GOW02.testPoint.Wika.dp2
GOW03.testPoint.Wika.dp2 GOW04.testPoint.Wika.dp2 GOW05.testPoint.Wika.dp2
GOW06.testPoint.Wika.dp2 GOW07.testPoint.Wika.dp2 GOW08.testPoint.Wika.dp2
GOW09.testPoint.Wika.dp2 GOW10.testPoint.Wika.dp2 GOW101.testPoint.Wika.dp2
GOW102.testPoint.Wika.dp2 GOW103.testPoint.Wika.dp2 GOW104.testPoint.Wika.dp2
GOW105.testPoint.Wika.dp2 GOW106.testPoint.Wika.dp2 GOW107.testPoint.Wika.dp2];
GOW_wikaDp3 = [GOW01.testPoint.Wika.dp3 GOW02.testPoint.Wika.dp3
GOW03.testPoint.Wika.dp3 GOW04.testPoint.Wika.dp3 GOW05.testPoint.Wika.dp3
GOW06.testPoint.Wika.dp3 GOW07.testPoint.Wika.dp3 GOW08.testPoint.Wika.dp3
GOW09.testPoint.Wika.dp3 GOW10.testPoint.Wika.dp3 GOW101.testPoint.Wika.dp3
GOW102.testPoint.Wika.dp3 GOW103.testPoint.Wika.dp3 GOW104.testPoint.Wika.dp3
GOW105.testPoint.Wika.dp3 GOW106.testPoint.Wika.dp3 GOW107.testPoint.Wika.dp3];
GOW_EmcoDp1 = [GOW01.testPoint.Emco.dp1 GOW02.testPoint.Emco.dp1
GOW03.testPoint.Emco.dp1 GOW04.testPoint.Emco.dp1 GOW05.testPoint.Emco.dp1
GOW06.testPoint.Emco.dp1 GOW07.testPoint.Emco.dp1 GOW08.testPoint.Emco.dp1
GOW09.testPoint.Emco.dp1 GOW10.testPoint.Emco.dp1 GOW101.testPoint.Emco.dp1
GOW102.testPoint.Emco.dp1 GOW103.testPoint.Emco.dp1 GOW104.testPoint.Emco.dp1
GOW105.testPoint.Emco.dp1 GOW106.testPoint.Emco.dp1 GOW107.testPoint.Emco.dp1];
GOW_EmcoDp2 = [GOW01.testPoint.Emco.dp2 GOW02.testPoint.Emco.dp2
GOW03.testPoint.Emco.dp2 GOW04.testPoint.Emco.dp2 GOW05.testPoint.Emco.dp2
GOW06.testPoint.Emco.dp2 GOW07.testPoint.Emco.dp2 GOW08.testPoint.Emco.dp2
GOW09.testPoint.Emco.dp2 GOW10.testPoint.Emco.dp2 GOW101.testPoint.Emco.dp2
GOW102.testPoint.Emco.dp2 GOW103.testPoint.Emco.dp2 GOW104.testPoint.Emco.dp2
GOW105.testPoint.Emco.dp2 GOW106.testPoint.Emco.dp2 GOW107.testPoint.Emco.dp2];
GOW_Po = [GOW01.testPoint.press.out GOW02.testPoint.press.out
GOW03.testPoint.press.out GOW04.testPoint.press.out GOW05.testPoint.press.out
GOW06.testPoint.press.out GOW07.testPoint.press.out GOW08.testPoint.press.out
GOW09.testPoint.press.out GOW10.testPoint.press.out GOW101.testPoint.press.out
```

```
GOW102.testPoint.press.out GOW103.testPoint.press.out GOW104.testPoint.press.out
GOW105.testPoint.press.out GOW106.testPoint.press.out GOW107.testPoint.press.out];
GOW_Pi = [GOW01.testPoint.press.in GOW02.testPoint.press.in
GOW03.testPoint.press.in GOW04.testPoint.press.in GOW05.testPoint.press.in
GOW06.testPoint.press.in GOW07.testPoint.press.in GOW08.testPoint.press.in
GOW09.testPoint.press.in GOW10.testPoint.press.in GOW101.testPoint.press.in
GOW102.testPoint.press.in GOW103.testPoint.press.in GOW104.testPoint.press.in
GOW105.testPoint.press.in GOW106.testPoint.press.in GOW107.testPoint.press.in];

%Make lists of density measurements
GOW_Gref_rho = [GOW01.testPoint.gasRef.rho GOW02.testPoint.gasRef.rho
GOW03.testPoint.gasRef.rho GOW04.testPoint.gasRef.rho GOW05.testPoint.gasRef.rho
GOW06.testPoint.gasRef.rho GOW07.testPoint.gasRef.rho GOW08.testPoint.gasRef.rho
GOW09.testPoint.gasRef.rho GOW10.testPoint.gasRef.rho GOW101.testPoint.gasRef.rho
GOW102.testPoint.gasRef.rho GOW103.testPoint.gasRef.rho
GOW104.testPoint.gasRef.rho GOW105.testPoint.gasRef.rho
GOW106.testPoint.gasRef.rho GOW107.testPoint.gasRef.rho];
GOW_Oref_rho = [GOW01.testPoint.oilRef.rho GOW02.testPoint.oilRef.rho
GOW03.testPoint.oilRef.rho GOW04.testPoint.oilRef.rho GOW05.testPoint.oilRef.rho
GOW06.testPoint.oilRef.rho GOW07.testPoint.oilRef.rho GOW08.testPoint.oilRef.rho
GOW09.testPoint.oilRef.rho GOW10.testPoint.oilRef.rho GOW101.testPoint.oilRef.rho
GOW102.testPoint.oilRef.rho GOW103.testPoint.oilRef.rho
GOW104.testPoint.oilRef.rho GOW105.testPoint.oilRef.rho
GOW106.testPoint.oilRef.rho GOW107.testPoint.oilRef.rho];
GOW_Wref_rho = [GOW01.testPoint.watRef.rho GOW02.testPoint.watRef.rho
GOW03.testPoint.watRef.rho GOW04.testPoint.watRef.rho GOW05.testPoint.watRef.rho
GOW06.testPoint.watRef.rho GOW07.testPoint.watRef.rho GOW08.testPoint.watRef.rho
GOW09.testPoint.watRef.rho GOW10.testPoint.watRef.rho GOW101.testPoint.watRef.rho
GOW102.testPoint.watRef.rho GOW103.testPoint.watRef.rho
GOW104.testPoint.watRef.rho GOW105.testPoint.watRef.rho
GOW106.testPoint.watRef.rho GOW107.testPoint.watRef.rho];
```

Appendix J

Preparing GW flow experiment data from dataset 1.

```
clc;
clear all;

%Load dataset
GW30 = load('GW30.mat');
GW31 = load('GW31.mat');
GW32 = load('GW32.mat');
GW33 = load('GW33.mat');
GW34 = load('GW34.mat');
GW35 = load('GW35.mat');

%Choose samples
start = 7705601;
stop = 7707600;

%Samples
GW30_ae1_data = GW30.testPoint.data(1).vData.ch(1).data(start:stop);
GW31_ae1_data = GW31.testPoint.data(1).vData.ch(1).data(start:stop);
GW32_ae1_data = GW32.testPoint.data(1).vData.ch(1).data(start:stop);
GW33_ae1_data = GW33.testPoint.data(1).vData.ch(1).data(start:stop);
GW34_ae1_data = GW34.testPoint.data(1).vData.ch(1).data(start:stop);
GW35_ae1_data = GW35.testPoint.data(1).vData.ch(1).data(start:stop);
GW30_ae3_data = GW30.testPoint.data(1).vData.ch(3).data(start:stop);
GW31_ae3_data = GW31.testPoint.data(1).vData.ch(3).data(start:stop);
GW32_ae3_data = GW32.testPoint.data(1).vData.ch(3).data(start:stop);
GW33_ae3_data = GW33.testPoint.data(1).vData.ch(3).data(start:stop);
GW34_ae3_data = GW34.testPoint.data(1).vData.ch(3).data(start:stop);
GW35_ae3_data = GW35.testPoint.data(1).vData.ch(3).data(start:stop);
GW30_ae4_data = GW30.testPoint.data(1).vData.ch(4).data(start:stop);
GW31_ae4_data = GW31.testPoint.data(1).vData.ch(4).data(start:stop);
GW32_ae4_data = GW32.testPoint.data(1).vData.ch(4).data(start:stop);
GW33_ae4_data = GW33.testPoint.data(1).vData.ch(4).data(start:stop);
GW34_ae4_data = GW34.testPoint.data(1).vData.ch(4).data(start:stop);
GW35_ae4_data = GW35.testPoint.data(1).vData.ch(4).data(start:stop);

%Calculate RMS
GW30_ae1_rms = rms(GW30_ae1_data);
GW31_ae1_rms = rms(GW31_ae1_data);
GW32_ae1_rms = rms(GW32_ae1_data);
GW33_ae1_rms = rms(GW33_ae1_data);
GW34_ae1_rms = rms(GW34_ae1_data);
GW35_ae1_rms = rms(GW35_ae1_data);
GW30_ae3_rms = rms(GW30_ae3_data);
GW31_ae3_rms = rms(GW31_ae3_data);
GW32_ae3_rms = rms(GW32_ae3_data);
GW33_ae3_rms = rms(GW33_ae3_data);
GW34_ae3_rms = rms(GW34_ae3_data);
GW35_ae3_rms = rms(GW35_ae3_data);
GW30_ae4_rms = rms(GW30_ae4_data);
GW31_ae4_rms = rms(GW31_ae4_data);
GW32_ae4_rms = rms(GW32_ae4_data);
GW33_ae4_rms = rms(GW33_ae4_data);
```

```
GW34_ae4_rms = rms(GW34_ae4_data);
GW35_ae4_rms = rms(GW35_ae4_data);

%Make lists of acoustic measurements
GW_ae1_data = [GW30_ae1_data; GW31_ae1_data; GW32_ae1_data; GW33_ae1_data;
GW34_ae1_data; GW35_ae1_data];
GW_ae3_data = [GW30_ae3_data; GW31_ae3_data; GW32_ae3_data; GW33_ae3_data;
GW34_ae3_data; GW35_ae3_data];
GW_ae4_data = [GW30_ae4_data; GW31_ae4_data; GW32_ae4_data; GW33_ae4_data;
GW34_ae4_data; GW35_ae4_data];
GW_ae1_rms = [GW30_ae1_rms GW31_ae1_rms GW32_ae1_rms GW33_ae1_rms GW34_ae1_rms
GW35_ae1_rms];
GW_ae3_rms = [GW30_ae3_rms GW31_ae3_rms GW32_ae3_rms GW33_ae3_rms GW34_ae3_rms
GW35_ae3_rms];
GW_ae4_rms = [GW30_ae4_rms GW31_ae4_rms GW32_ae4_rms GW33_ae4_rms GW34_ae4_rms
GW35_ae4_rms];

%Make lists of flow measurements
GW_Wref_Qv = [GW30.testPoint.watRef.q GW31.testPoint.watRef.q
GW32.testPoint.watRef.q GW33.testPoint.watRef.q GW34.testPoint.watRef.q
GW35.testPoint.watRef.q];
GW_Gref_Qv = [30.34, 29.52, 31.49, 31.49, 30.11, 30.58];
%GW_Gref_Qv = [GW30.testPoint.gasRef.q GW31.testPoint.gasRef.q
GW32.testPoint.gasRef.q GW33.testPoint.gasRef.q GW34.testPoint.gasRef.q
GW35.testPoint.gasRef.q];


%Make lists of valve positions and pressure measurements
GW_HIC = [GW30.testPoint.HIC.pos GW31.testPoint.HIC.pos GW32.testPoint.HIC.pos
GW33.testPoint.HIC.pos GW34.testPoint.HIC.pos GW35.testPoint.HIC.pos];
GW_Pi = [GW30.testPoint.press.in GW31.testPoint.press.in GW32.testPoint.press.in
GW33.testPoint.press.in GW34.testPoint.press.in GW35.testPoint.press.in];
GW_Po = [GW30.testPoint.press.out GW31.testPoint.press.out
GW32.testPoint.press.out GW33.testPoint.press.out GW34.testPoint.press.out
GW35.testPoint.press.out];
GW_wikaDp1 = [GW30.testPoint.Wika.dp1 GW31.testPoint.Wika.dp1
GW32.testPoint.Wika.dp1 GW33.testPoint.Wika.dp1 GW34.testPoint.Wika.dp1
GW35.testPoint.Wika.dp1];
GW_wikaDp2 = [GW30.testPoint.Wika.dp2 GW31.testPoint.Wika.dp2
GW32.testPoint.Wika.dp2 GW33.testPoint.Wika.dp2 GW34.testPoint.Wika.dp2
GW35.testPoint.Wika.dp2];
GW_wikaDp3 = [GW30.testPoint.Wika.dp3 GW31.testPoint.Wika.dp3
GW32.testPoint.Wika.dp3 GW33.testPoint.Wika.dp3 GW34.testPoint.Wika.dp3
GW35.testPoint.Wika.dp3];
GW_EmcoDp1 = [GW30.testPoint.Emco.dp1 GW31.testPoint.Emco.dp1
GW32.testPoint.Emco.dp1 GW33.testPoint.Emco.dp1 GW34.testPoint.Emco.dp1
GW35.testPoint.Emco.dp1];
GW_EmcoDp2 = [GW30.testPoint.Emco.dp2 GW31.testPoint.Emco.dp2
GW32.testPoint.Emco.dp2 GW33.testPoint.Emco.dp2 GW34.testPoint.Emco.dp2
GW35.testPoint.Emco.dp2];

%Make lists of density measurements
GW_Gref_rho = [GW30.testPoint.gasRef.rho GW31.testPoint.gasRef.rho
GW32.testPoint.gasRef.rho GW33.testPoint.gasRef.rho GW34.testPoint.gasRef.rho
GW35.testPoint.gasRef.rho];
```

```
GW_Wref_rho = [GW30.testPoint.watRef.rho GW31.testPoint.watRef.rho
GW32.testPoint.watRef.rho GW33.testPoint.watRef.rho GW34.testPoint.watRef.rho
GW35.testPoint.watRef.rho];
```

Appendix K

Preparing GO flow experiment data from dataset 1.

```
clc;
clear all;

% %Load dataset
GO01 = load('GO01.mat');
GO01C = load('GO01C.mat');
GO02C = load('GO02C.mat');
GO03C = load('GO03C.mat');
GO04C = load('GO04C.mat');
GO05 = load('GO05.mat');
GO05C = load('GO05C.mat');
GO06 = load('GO06.mat');
GO09 = load('GO09.mat');
GO17 = load('GO17.mat');
GO18 = load('GO18.mat');
GO19 = load('GO19.mat');
GO22 = load('GO22.mat');
GOC11 = load('GOC11.mat');
GOC12 = load('GOC12.mat');
GOC13 = load('GOC13.mat');
GOC14 = load('GOC14.mat');
GOC15 = load('GOC15.mat');
GOC16 = load('GOC16.mat');
GOC17 = load('GOC17.mat');
GOC18 = load('GOC18.mat');
GOC19 = load('GOC19.mat');
GOC20 = load('GOC20.mat');

%Choose samples
start = 7705601;
stop = 7707600;

%Samples
GO01_ae1_data = GO01.testPoint.data(1).vData.ch(1).data(start:stop);
GO01C_ae1_data = GO01C.testPoint.data(1).vData.ch(1).data(start:stop);
GO02C_ae1_data = GO02C.testPoint.data(1).vData.ch(1).data(start:stop);
GO03C_ae1_data = GO03C.testPoint.data(1).vData.ch(1).data(start:stop);
GO04C_ae1_data = GO04C.testPoint.data(1).vData.ch(1).data(start:stop);
GO05_ae1_data = GO05.testPoint.data(1).vData.ch(1).data(start:stop);
GO05C_ae1_data = GO05C.testPoint.data(1).vData.ch(1).data(start:stop);
GO06_ae1_data = GO06.testPoint.data(1).vData.ch(1).data(start:stop);
GO09_ae1_data = GO09.testPoint.data(1).vData.ch(1).data(start:stop);
GO17_ae1_data = GO17.testPoint.data(1).vData.ch(1).data(start:stop);
GO18_ae1_data = GO18.testPoint.data(1).vData.ch(1).data(start:stop);
GO19_ae1_data = GO19.testPoint.data(1).vData.ch(1).data(start:stop);
GO22_ae1_data = GO22.testPoint.data(1).vData.ch(1).data(start:stop);
GOC11_ae1_data = GOC11.testPoint.data(1).vData.ch(1).data(start:stop);
GOC12_ae1_data = GOC12.testPoint.data(1).vData.ch(1).data(start:stop);
GOC13_ae1_data = GOC13.testPoint.data(1).vData.ch(1).data(start:stop);
GOC14_ae1_data = GOC14.testPoint.data(1).vData.ch(1).data(start:stop);
GOC15_ae1_data = GOC15.testPoint.data(1).vData.ch(1).data(start:stop);
GOC16_ae1_data = GOC16.testPoint.data(1).vData.ch(1).data(start:stop);
```

```
GOC17_ae1_data = GOC17.testPoint.data(1).vData.ch(1).data(start:stop);
GOC18_ae1_data = GOC18.testPoint.data(1).vData.ch(1).data(start:stop);
GOC19_ae1_data = GOC19.testPoint.data(1).vData.ch(1).data(start:stop);
GOC20_ae1_data = GOC20.testPoint.data(1).vData.ch(1).data(start:stop);
GO01_ae3_data = GO01.testPoint.data(1).vData.ch(3).data(start:stop);
GO01C_ae3_data = GO01C.testPoint.data(1).vData.ch(3).data(start:stop);
GO02C_ae3_data = GO02C.testPoint.data(1).vData.ch(3).data(start:stop);
GO03C_ae3_data = GO03C.testPoint.data(1).vData.ch(3).data(start:stop);
GO04C_ae3_data = GO04C.testPoint.data(1).vData.ch(3).data(start:stop);
GO05_ae3_data = GO05.testPoint.data(1).vData.ch(3).data(start:stop);
GO05C_ae3_data = GO05C.testPoint.data(1).vData.ch(3).data(start:stop);
GO06_ae3_data = GO06.testPoint.data(1).vData.ch(3).data(start:stop);
GO09_ae3_data = GO09.testPoint.data(1).vData.ch(3).data(start:stop);
GO17_ae3_data = GO17.testPoint.data(1).vData.ch(3).data(start:stop);
GO18_ae3_data = GO18.testPoint.data(1).vData.ch(3).data(start:stop);
GO19_ae3_data = GO19.testPoint.data(1).vData.ch(3).data(start:stop);
GO22_ae3_data = GO22.testPoint.data(1).vData.ch(3).data(start:stop);
GOC11_ae3_data = GOC11.testPoint.data(1).vData.ch(3).data(start:stop);
GOC12_ae3_data = GOC12.testPoint.data(1).vData.ch(3).data(start:stop);
GOC13_ae3_data = GOC13.testPoint.data(1).vData.ch(3).data(start:stop);
GOC14_ae3_data = GOC14.testPoint.data(1).vData.ch(3).data(start:stop);
GOC15_ae3_data = GOC15.testPoint.data(1).vData.ch(3).data(start:stop);
GOC16_ae3_data = GOC16.testPoint.data(1).vData.ch(3).data(start:stop);
GOC17_ae3_data = GOC17.testPoint.data(1).vData.ch(3).data(start:stop);
GOC18_ae3_data = GOC18.testPoint.data(1).vData.ch(3).data(start:stop);
GOC19_ae3_data = GOC19.testPoint.data(1).vData.ch(3).data(start:stop);
GOC20_ae3_data = GOC20.testPoint.data(1).vData.ch(3).data(start:stop);
GO01_ae4_data = GO01.testPoint.data(1).vData.ch(4).data(start:stop);
GO01C_ae4_data = GO01C.testPoint.data(1).vData.ch(4).data(start:stop);
GO02C_ae4_data = GO02C.testPoint.data(1).vData.ch(4).data(start:stop);
GO03C_ae4_data = GO03C.testPoint.data(1).vData.ch(4).data(start:stop);
GO04C_ae4_data = GO04C.testPoint.data(1).vData.ch(4).data(start:stop);
GO05_ae4_data = GO05.testPoint.data(1).vData.ch(4).data(start:stop);
GO05C_ae4_data = GO05C.testPoint.data(1).vData.ch(4).data(start:stop);
GO06_ae4_data = GO06.testPoint.data(1).vData.ch(4).data(start:stop);
GO09_ae4_data = GO09.testPoint.data(1).vData.ch(4).data(start:stop);
GO17_ae4_data = GO17.testPoint.data(1).vData.ch(4).data(start:stop);
GO18_ae4_data = GO18.testPoint.data(1).vData.ch(4).data(start:stop);
GO19_ae4_data = GO19.testPoint.data(1).vData.ch(4).data(start:stop);
GO22_ae4_data = GO22.testPoint.data(1).vData.ch(4).data(start:stop);
GOC11_ae4_data = GOC11.testPoint.data(1).vData.ch(4).data(start:stop);
GOC12_ae4_data = GOC12.testPoint.data(1).vData.ch(4).data(start:stop);
GOC13_ae4_data = GOC13.testPoint.data(1).vData.ch(4).data(start:stop);
GOC14_ae4_data = GOC14.testPoint.data(1).vData.ch(4).data(start:stop);
GOC15_ae4_data = GOC15.testPoint.data(1).vData.ch(4).data(start:stop);
GOC16_ae4_data = GOC16.testPoint.data(1).vData.ch(4).data(start:stop);
GOC17_ae4_data = GOC17.testPoint.data(1).vData.ch(4).data(start:stop);
GOC18_ae4_data = GOC18.testPoint.data(1).vData.ch(4).data(start:stop);
GOC19_ae4_data = GOC19.testPoint.data(1).vData.ch(4).data(start:stop);
GOC20_ae4_data = GOC20.testPoint.data(1).vData.ch(4).data(start:stop);

% %Calculate RMS
GO01_ae1_rms = rms(GO01_ae1_data);
GO01C_ae1_rms = rms(GO01C_ae1_data);
GO02C_ae1_rms = rms(GO02C_ae1_data);
GO03C_ae1_rms = rms(GO03C_ae1_data);
GO04C_ae1_rms = rms(GO04C_ae1_data);
```

```
GO05_ae1_rms = rms(GO05_ae1_data);
GO05C_ae1_rms = rms(GO05C_ae1_data);
GO06_ae1_rms = rms(GO06_ae1_data);
GO09_ae1_rms = rms(GO09_ae1_data);
GO17_ae1_rms = rms(GO17_ae1_data);
GO18_ae1_rms = rms(GO18_ae1_data);
GO19_ae1_rms = rms(GO19_ae1_data);
GO22_ae1_rms = rms(GO22_ae1_data);
GOC11_ae1_rms = rms(GOC11_ae1_data);
GOC12_ae1_rms = rms(GOC12_ae1_data);
GOC13_ae1_rms = rms(GOC13_ae1_data);
GOC14_ae1_rms = rms(GOC14_ae1_data);
GOC15_ae1_rms = rms(GOC15_ae1_data);
GOC16_ae1_rms = rms(GOC16_ae1_data);
GOC17_ae1_rms = rms(GOC17_ae1_data);
GOC18_ae1_rms = rms(GOC18_ae1_data);
GOC19_ae1_rms = rms(GOC19_ae1_data);
GOC20_ae1_rms = rms(GOC20_ae1_data);
GO01_ae3_rms = rms(GO01_ae3_data);
GO01C_ae3_rms = rms(GO01C_ae3_data);
GO02C_ae3_rms = rms(GO02C_ae3_data);
GO03C_ae3_rms = rms(GO03C_ae3_data);
GO04C_ae3_rms = rms(GO04C_ae3_data);
GO05_ae3_rms = rms(GO05_ae3_data);
GO05C_ae3_rms = rms(GO05C_ae3_data);
GO06_ae3_rms = rms(GO06_ae3_data);
GO09_ae3_rms = rms(GO09_ae3_data);
GO17_ae3_rms = rms(GO17_ae3_data);
GO18_ae3_rms = rms(GO18_ae3_data);
GO19_ae3_rms = rms(GO19_ae3_data);
GO22_ae3_rms = rms(GO22_ae3_data);
GOC11_ae3_rms = rms(GOC11_ae3_data);
GOC12_ae3_rms = rms(GOC12_ae3_data);
GOC13_ae3_rms = rms(GOC13_ae3_data);
GOC14_ae3_rms = rms(GOC14_ae3_data);
GOC15_ae3_rms = rms(GOC15_ae3_data);
GOC16_ae3_rms = rms(GOC16_ae3_data);
GOC17_ae3_rms = rms(GOC17_ae3_data);
GOC18_ae3_rms = rms(GOC18_ae3_data);
GOC19_ae3_rms = rms(GOC19_ae3_data);
GOC20_ae3_rms = rms(GOC20_ae3_data);
GO01_ae4_rms = rms(GO01_ae4_data);
GO01C_ae4_rms = rms(GO01C_ae4_data);
GO02C_ae4_rms = rms(GO02C_ae4_data);
GO03C_ae4_rms = rms(GO03C_ae4_data);
GO04C_ae4_rms = rms(GO04C_ae4_data);
GO05_ae4_rms = rms(GO05_ae4_data);
GO05C_ae4_rms = rms(GO05C_ae4_data);
GO06_ae4_rms = rms(GO06_ae4_data);
GO09_ae4_rms = rms(GO09_ae4_data);
GO17_ae4_rms = rms(GO17_ae4_data);
GO18_ae4_rms = rms(GO18_ae4_data);
GO19_ae4_rms = rms(GO19_ae4_data);
GO22_ae4_rms = rms(GO22_ae4_data);
GOC11_ae4_rms = rms(GOC11_ae4_data);
GOC12_ae4_rms = rms(GOC12_ae4_data);
GOC13_ae4_rms = rms(GOC13_ae4_data);
```

```
GOC14_ae4_rms = rms(GOC14_ae4_data);
GOC15_ae4_rms = rms(GOC15_ae4_data);
GOC16_ae4_rms = rms(GOC16_ae4_data);
GOC17_ae4_rms = rms(GOC17_ae4_data);
GOC18_ae4_rms = rms(GOC18_ae4_data);
GOC19_ae4_rms = rms(GOC19_ae4_data);
GOC20_ae4_rms = rms(GOC20_ae4_data);

%Make lists of acoustic measurements
GO_ae1_data = [GO01_ae1_data; GO01C_ae1_data; GO02C_ae1_data; GO03C_ae1_data;
GO04C_ae1_data; GO05_ae1_data; GO05C_ae1_data; GO06_ae1_data; GO09_ae1_data;
GO17_ae1_data; GO18_ae1_data; GO19_ae1_data; GO22_ae1_data; GOC11_ae1_data;
GOC12_ae1_data; GOC13_ae1_data; GOC14_ae1_data; GOC15_ae1_data; GOC16_ae1_data;
GOC17_ae1_data; GOC18_ae1_data; GOC19_ae1_data; GOC20_ae1_data];
GO_ae3_data = [GO01_ae3_data; GO01C_ae3_data; GO02C_ae3_data; GO03C_ae3_data;
GO04C_ae3_data; GO05_ae3_data; GO05C_ae3_data; GO06_ae3_data; GO09_ae3_data;
GO17_ae3_data; GO18_ae3_data; GO19_ae3_data; GO22_ae3_data; GOC11_ae3_data;
GOC12_ae3_data; GOC13_ae3_data; GOC14_ae3_data; GOC15_ae3_data; GOC16_ae3_data;
GOC17_ae3_data; GOC18_ae3_data; GOC19_ae3_data; GOC20_ae3_data];
GO_ae4_data = [GO01_ae4_data; GO01C_ae4_data; GO02C_ae4_data; GO03C_ae4_data;
GO04C_ae4_data; GO05_ae4_data; GO05C_ae4_data; GO06_ae4_data; GO09_ae4_data;
GO17_ae4_data; GO18_ae4_data; GO19_ae4_data; GO22_ae4_data; GOC11_ae4_data;
GOC12_ae4_data; GOC13_ae4_data; GOC14_ae4_data; GOC15_ae4_data; GOC16_ae4_data;
GOC17_ae4_data; GOC18_ae4_data; GOC19_ae4_data; GOC20_ae4_data];
GO_ae1_rms = [GO01_ae1_rms GO01C_ae1_rms GO02C_ae1_rms GO03C_ae1_rms GO04C_ae1_rms
GO05_ae1_rms GO05C_ae1_rms GO06_ae1_rms GO09_ae1_rms GO17_ae1_rms GO18_ae1_rms
GO19_ae1_rms GO22_ae1_rms GOC11_ae1_rms GOC12_ae1_rms GOC13_ae1_rms GOC14_ae1_rms
GOC15_ae1_rms GOC16_ae1_rms GOC17_ae1_rms GOC18_ae1_rms GOC19_ae1_rms
GOC20_ae1_rms];
GO_ae3_rms = [GO01_ae3_rms GO01C_ae3_rms GO02C_ae3_rms GO03C_ae3_rms GO04C_ae3_rms
GO05_ae3_rms GO05C_ae3_rms GO06_ae3_rms GO09_ae3_rms GO17_ae3_rms GO18_ae3_rms
GO19_ae3_rms GO22_ae3_rms GOC11_ae3_rms GOC12_ae3_rms GOC13_ae3_rms GOC14_ae3_rms
GOC15_ae3_rms GOC16_ae3_rms GOC17_ae3_rms GOC18_ae3_rms GOC19_ae3_rms
GOC20_ae3_rms];
GO_ae4_rms = [GO01_ae4_rms GO01C_ae4_rms GO02C_ae4_rms GO03C_ae4_rms GO04C_ae4_rms
GO05_ae4_rms GO05C_ae4_rms GO06_ae4_rms GO09_ae4_rms GO17_ae4_rms GO18_ae4_rms
GO19_ae4_rms GO22_ae4_rms GOC11_ae4_rms GOC12_ae4_rms GOC13_ae4_rms GOC14_ae4_rms
GOC15_ae4_rms GOC16_ae4_rms GOC17_ae4_rms GOC18_ae4_rms GOC19_ae4_rms
GOC20_ae4_rms];

%Make lists of flow measurements
GO_Oref_Qv = [GO01.testPoint.oilRef.q GO01C.testPoint.oilRef.q
GO02C.testPoint.oilRef.q GO03C.testPoint.oilRef.q GO04C.testPoint.oilRef.q
GO05.testPoint.oilRef.q GO05C.testPoint.oilRef.q GO06.testPoint.oilRef.q
GO09.testPoint.oilRef.q GO17.testPoint.oilRef.q GO18.testPoint.oilRef.q
GO19.testPoint.oilRef.q  GO22.testPoint.oilRef.q  GOC11.testPoint.oilRef.q
GOC12.testPoint.oilRef.q  GOC13.testPoint.oilRef.q  GOC14.testPoint.oilRef.q
GOC15.testPoint.oilRef.q  GOC16.testPoint.oilRef.q  GOC17.testPoint.oilRef.q
GOC18.testPoint.oilRef.q  GOC19.testPoint.oilRef.q  GOC20.testPoint.oilRef.q];
GO_Gref_Qv = [10.48, 54.00, 47.31, 43.00, 33.02, 50.55, 24.55, 99.06, 95.70,
83.64, 86.51, 98.42, 101.72, 5.30, 4.74, 16.34, 15.58, 20.43, 20.05, 20.18, 40.61,
40.81, 41.72];
GO_Wref_Qv = [GO01.testPoint.watRef.q  GO01C.testPoint.watRef.q
GO02C.testPoint.watRef.q GO03C.testPoint.watRef.q GO04C.testPoint.watRef.q
GO05.testPoint.watRef.q GO05C.testPoint.watRef.q GO06.testPoint.watRef.q
GO09.testPoint.watRef.q GO17.testPoint.watRef.q GO18.testPoint.watRef.q
GO19.testPoint.watRef.q  GO22.testPoint.watRef.q  GOC11.testPoint.watRef.q
```

```
GOC12.testPoint.watRef.q  GOC13.testPoint.watRef.q  GOC14.testPoint.watRef.q
GOC15.testPoint.watRef.q  GOC16.testPoint.watRef.q  GOC17.testPoint.watRef.q
GOC18.testPoint.watRef.q  GOC19.testPoint.watRef.q  GOC20.testPoint.watRef.q];

%GO_Gref_Qv = [GO01.gasRef.q GO01C.gasRef.q GO02C.gasRef.q GO03C.gasRef.q
GO04C.gasRef.q GO05.gasRef.q GO05C.gasRef.q GO06.gasRef.q GO09.gasRef.q
GO17.gasRef.q GO18.gasRef.q GO19.gasRef.q GO22.gasRef.q GOC11.gasRef.q
GOC12.gasRef.q GOC13.gasRef.q GOC14.gasRef.q GOC15.gasRef.q GOC16.gasRef.q
GOC17.gasRef.q GOC18.gasRef.q GOC19.gasRef.q GOC20.gasRef.q]

%Make lists of valve positions and pressure measurements
GO_HIC = [GO01.testPoint.HIC.pos GO01C.testPoint.HIC.pos GO02C.testPoint.HIC.pos
GO03C.testPoint.HIC.pos GO04C.testPoint.HIC.pos GO05.testPoint.HIC.pos
GO05C.testPoint.HIC.pos GO06.testPoint.HIC.pos GO09.testPoint.HIC.pos
GO17.testPoint.HIC.pos GO18.testPoint.HIC.pos GO19.testPoint.HIC.pos
GO22.testPoint.HIC.pos GOC11.testPoint.HIC.pos GOC12.testPoint.HIC.pos
GOC13.testPoint.HIC.pos GOC14.testPoint.HIC.pos GOC15.testPoint.HIC.pos
GOC16.testPoint.HIC.pos GOC17.testPoint.HIC.pos GOC18.testPoint.HIC.pos
GOC19.testPoint.HIC.pos GOC20.testPoint.HIC.pos];
GO_Pi = [GO01.testPoint.press.in GO01C.testPoint.press.in GO02C.testPoint.press.in
GO03C.testPoint.press.in GO04C.testPoint.press.in GO05.testPoint.press.in
GO05C.testPoint.press.in GO06.testPoint.press.in GO09.testPoint.press.in
GO17.testPoint.press.in GO18.testPoint.press.in GO19.testPoint.press.in
GO22.testPoint.press.in GOC11.testPoint.press.in GOC12.testPoint.press.in
GOC13.testPoint.press.in GOC14.testPoint.press.in GOC15.testPoint.press.in
GOC16.testPoint.press.in GOC17.testPoint.press.in GOC18.testPoint.press.in
GOC19.testPoint.press.in GOC20.testPoint.press.in];
GO_Po = [GO01.testPoint.press.out GO01C.testPoint.press.out
GO02C.testPoint.press.out GO03C.testPoint.press.out GO04C.testPoint.press.out
GO05.testPoint.press.out GO05C.testPoint.press.out GO06.testPoint.press.out
GO09.testPoint.press.out GO17.testPoint.press.out GO18.testPoint.press.out
GO19.testPoint.press.out GO22.testPoint.press.out GOC11.testPoint.press.out
GOC12.testPoint.press.out GOC13.testPoint.press.out GOC14.testPoint.press.out
GOC15.testPoint.press.out GOC16.testPoint.press.out GOC17.testPoint.press.out
GOC18.testPoint.press.out GOC19.testPoint.press.out GOC20.testPoint.press.out];
GO_wikaDp1 = [GO01.testPoint.Wika.dp1 GO01C.testPoint.Wika.dp1
GO02C.testPoint.Wika.dp1 GO03C.testPoint.Wika.dp1 GO04C.testPoint.Wika.dp1
GO05.testPoint.Wika.dp1 GO05C.testPoint.Wika.dp1 GO06.testPoint.Wika.dp1
GO09.testPoint.Wika.dp1 GO17.testPoint.Wika.dp1 GO18.testPoint.Wika.dp1
GO19.testPoint.Wika.dp1 GO22.testPoint.Wika.dp1 GOC11.testPoint.Wika.dp1
GOC12.testPoint.Wika.dp1 GOC13.testPoint.Wika.dp1 GOC14.testPoint.Wika.dp1
GOC15.testPoint.Wika.dp1 GOC16.testPoint.Wika.dp1 GOC17.testPoint.Wika.dp1
GOC18.testPoint.Wika.dp1 GOC19.testPoint.Wika.dp1 GOC20.testPoint.Wika.dp1];
GO_wikaDp2 = [GO01.testPoint.Wika.dp2 GO01C.testPoint.Wika.dp1
GO02C.testPoint.Wika.dp2 GO03C.testPoint.Wika.dp2 GO04C.testPoint.Wika.dp2
GO05.testPoint.Wika.dp2 GO05C.testPoint.Wika.dp2 GO06.testPoint.Wika.dp2
GO09.testPoint.Wika.dp2 GO17.testPoint.Wika.dp2 GO18.testPoint.Wika.dp2
GO19.testPoint.Wika.dp2 GO22.testPoint.Wika.dp2 GOC11.testPoint.Wika.dp2
GOC12.testPoint.Wika.dp2 GOC13.testPoint.Wika.dp2 GOC14.testPoint.Wika.dp2
GOC15.testPoint.Wika.dp2 GOC16.testPoint.Wika.dp2 GOC17.testPoint.Wika.dp2
GOC18.testPoint.Wika.dp2 GOC19.testPoint.Wika.dp2 GOC20.testPoint.Wika.dp2];
GO_wikaDp3 = [GO01.testPoint.Wika.dp3 GO01C.testPoint.Wika.dp1
GO02C.testPoint.Wika.dp3 GO03C.testPoint.Wika.dp3 GO04C.testPoint.Wika.dp3
GO05.testPoint.Wika.dp3 GO05C.testPoint.Wika.dp3 GO06.testPoint.Wika.dp3
GO09.testPoint.Wika.dp3 GO17.testPoint.Wika.dp3 GO18.testPoint.Wika.dp3
GO19.testPoint.Wika.dp3 GO22.testPoint.Wika.dp3 GOC11.testPoint.Wika.dp3
GOC12.testPoint.Wika.dp3 GOC13.testPoint.Wika.dp3 GOC14.testPoint.Wika.dp3
```

```
GOC15.testPoint.Wika.dp3 GOC16.testPoint.Wika.dp3 GOC17.testPoint.Wika.dp3
GOC18.testPoint.Wika.dp3 GOC19.testPoint.Wika.dp3 GOC20.testPoint.Wika.dp3];
GO_EmcoDp1 = [GO01.testPoint.Emco.dp1 GO01C.testPoint.Wika.dp1
GO02C.testPoint.Emco.dp1 GO03C.testPoint.Emco.dp1 GO04C.testPoint.Emco.dp1
GO05.testPoint.Emco.dp1 GO05C.testPoint.Emco.dp1 GO06.testPoint.Emco.dp1
GO09.testPoint.Emco.dp1 GO17.testPoint.Emco.dp1 GO18.testPoint.Emco.dp1
GO19.testPoint.Emco.dp1 GO22.testPoint.Emco.dp1 GOC11.testPoint.Emco.dp1
GOC12.testPoint.Emco.dp1 GOC13.testPoint.Emco.dp1 GOC14.testPoint.Emco.dp1
GOC15.testPoint.Emco.dp1 GOC16.testPoint.Emco.dp1 GOC17.testPoint.Emco.dp1
GOC18.testPoint.Emco.dp1 GOC19.testPoint.Emco.dp1 GOC20.testPoint.Emco.dp1];

GO_EmcoDp2 = [GO01.testPoint.Emco.dp2 GO01C.testPoint.Wika.dp1
GO02C.testPoint.Emco.dp2 GO03C.testPoint.Emco.dp2 GO04C.testPoint.Emco.dp2
GO05.testPoint.Emco.dp2 GO05C.testPoint.Emco.dp2 GO06.testPoint.Emco.dp2
GO09.testPoint.Emco.dp2 GO17.testPoint.Emco.dp2 GO18.testPoint.Emco.dp2
GO19.testPoint.Emco.dp2 GO22.testPoint.Emco.dp2 GOC11.testPoint.Emco.dp2
GOC12.testPoint.Emco.dp2 GOC13.testPoint.Emco.dp2 GOC14.testPoint.Emco.dp2
GOC15.testPoint.Emco.dp2 GOC16.testPoint.Emco.dp2 GOC17.testPoint.Emco.dp2
GOC18.testPoint.Emco.dp2 GOC19.testPoint.Emco.dp2 GOC20.testPoint.Emco.dp2];
GO_Krohn3Rho = [GO01.testPoint.Krohne.rho GO01C.testPoint.Krohne.rho
GO02C.testPoint.Krohne.rho GO03C.testPoint.Krohne.rho GO04C.testPoint.Krohne.rho
GO05.testPoint.Krohne.rho GO05C.testPoint.Krohne.rho GO06.testPoint.Emco.dp2
GO09.testPoint.Krohne.rho GO17.testPoint.Krohne.rho GO18.testPoint.Krohne.rho
GO19.testPoint.Krohne.rho GO22.testPoint.Krohne.rho GOC11.testPoint.Krohne.rho
GOC12.testPoint.Krohne.rho GOC13.testPoint.Krohne.rho GOC14.testPoint.Krohne.rho
GOC15.testPoint.Krohne.rho GOC16.testPoint.Krohne.rho GOC17.testPoint.Krohne.rho
GOC18.testPoint.Krohne.rho GOC19.testPoint.Krohne.rho GOC20.testPoint.Krohne.rho];

GO_Krohn3Rho = [GO01.testPoint.Krohne.rho GO01C.testPoint.Krohne.rho
GO02C.testPoint.Krohne.rho GO03C.testPoint.Krohne.rho GO04C.testPoint.Krohne.rho
GO05.testPoint.Krohne.rho GO05C.testPoint.Krohne.rho GO06.testPoint.Emco.dp2
GO09.testPoint.Krohne.rho GO17.testPoint.Krohne.rho GO18.testPoint.Krohne.rho
GO19.testPoint.Krohne.rho GO22.testPoint.Krohne.rho GOC11.testPoint.Krohne.rho
GOC12.testPoint.Krohne.rho GOC13.testPoint.Krohne.rho GOC14.testPoint.Krohne.rho
GOC15.testPoint.Krohne.rho GOC16.testPoint.Krohne.rho GOC17.testPoint.Krohne.rho
GOC18.testPoint.Krohne.rho GOC19.testPoint.Krohne.rho GOC20.testPoint.Krohne.rho];

GO_4mStraight = [];
GO_Ti = [];
GO_To = [];


%Make lists of density measurements
GO_Gref_rho = [GO01.testPoint.gasRef.rho GO01C.testPoint.gasRef.rho
GO02C.testPoint.gasRef.rho GO03C.testPoint.gasRef.rho GO04C.testPoint.gasRef.rho
GO05.testPoint.gasRef.rho GO05C.testPoint.gasRef.rho GO06.testPoint.gasRef.rho
GO09.testPoint.gasRef.rho GO17.testPoint.gasRef.rho GO18.testPoint.gasRef.rho
GO19.testPoint.gasRef.rho GO22.testPoint.gasRef.rho GOC11.testPoint.gasRef.rho
GOC12.testPoint.gasRef.rho GOC13.testPoint.gasRef.rho GOC14.testPoint.gasRef.rho
GOC15.testPoint.gasRef.rho GOC16.testPoint.gasRef.rho GOC17.testPoint.gasRef.rho
GOC18.testPoint.gasRef.rho GOC19.testPoint.gasRef.rho GOC20.testPoint.gasRef.rho];
GO_Oref_rho = [GO01.testPoint.oilRef.rho GO01C.testPoint.oilRef.rho
GO02C.testPoint.oilRef.rho GO03C.testPoint.oilRef.rho GO04C.testPoint.oilRef.rho
GO05.testPoint.oilRef.rho GO05C.testPoint.oilRef.rho GO06.testPoint.oilRef.rho
GO09.testPoint.oilRef.rho GO17.testPoint.oilRef.rho GO18.testPoint.oilRef.rho
GO19.testPoint.oilRef.rho GO22.testPoint.oilRef.rho GOC11.testPoint.oilRef.rho
GOC12.testPoint.oilRef.rho GOC13.testPoint.oilRef.rho GOC14.testPoint.oilRef.rho
```

```
GOC15.testPoint.oilRef.rho GOC16.testPoint.oilRef.rho GOC17.testPoint.oilRef.rho
GOC18.testPoint.oilRef.rho GOC19.testPoint.oilRef.rho GOC20.testPoint.oilRef.rho];
GO_Wref_rho

%Make list of all data
dataset1_ae = [GO_ae1_rms zeros(length(GO_ae1_rms)) GO_ae3_rms GO_ae4_rms
GO_Oref_Qv ];
```

Appendix L

Preparing oil flow experiment data from dataset 1.

```
clc;
clear all;

% %Load dataset
OC01 = load('OC01.mat');
OC02 = load('OC02.mat');
OC03 = load('OC03.mat');
OC04 = load('OC04.mat');
OC07 = load('OC07.mat');
OC08 = load('OC08.mat');
OC09 = load('OC09.mat');
OC10 = load('OC10.mat');
OC11 = load('OC11.mat');
OC12 = load('OC12.mat');
OC13 = load('OC13.mat');
OC14 = load('OC14.mat');
OC16 = load('OC16.mat');
OC17 = load('OC17.mat');
OC18 = load('OC18.mat');

%Choose samples
start = 7705601;
stop = 7707600;

%Samples
OC01_ae1_data = OC01.testPoint.data(1).vData.ch(1).data(start:stop);
OC02_ae1_data = OC02.testPoint.data(1).vData.ch(1).data(start:stop);
OC03_ae1_data = OC03.testPoint.data(1).vData.ch(1).data(start:stop);
OC04_ae1_data = OC04.testPoint.data(1).vData.ch(1).data(start:stop);
OC07_ae1_data = OC07.testPoint.data(1).vData.ch(1).data(start:stop);
OC08_ae1_data = OC08.testPoint.data(1).vData.ch(1).data(start:stop);
OC09_ae1_data = OC09.testPoint.data(1).vData.ch(1).data(start:stop);
OC10_ae1_data = OC10.testPoint.data(1).vData.ch(1).data(start:stop);
OC11_ae1_data = OC11.testPoint.data(1).vData.ch(1).data(start:stop);
OC12_ae1_data = OC12.testPoint.data(1).vData.ch(1).data(start:stop);
OC13_ae1_data = OC13.testPoint.data(1).vData.ch(1).data(start:stop);
OC14_ae1_data = OC14.testPoint.data(1).vData.ch(1).data(start:stop);
OC16_ae1_data = OC16.testPoint.data(1).vData.ch(1).data(start:stop);
OC17_ae1_data = OC17.testPoint.data(1).vData.ch(1).data(start:stop);
OC18_ae1_data = OC18.testPoint.data(1).vData.ch(1).data(start:stop);
OC01_ae3_data = OC01.testPoint.data(1).vData.ch(3).data(start:stop);
OC02_ae3_data = OC02.testPoint.data(1).vData.ch(3).data(start:stop);
OC03_ae3_data = OC03.testPoint.data(1).vData.ch(3).data(start:stop);
OC04_ae3_data = OC04.testPoint.data(1).vData.ch(3).data(start:stop);
OC07_ae3_data = OC07.testPoint.data(1).vData.ch(3).data(start:stop);
OC08_ae3_data = OC08.testPoint.data(1).vData.ch(3).data(start:stop);
OC09_ae3_data = OC09.testPoint.data(1).vData.ch(3).data(start:stop);
OC10_ae3_data = OC10.testPoint.data(1).vData.ch(3).data(start:stop);
OC11_ae3_data = OC11.testPoint.data(1).vData.ch(3).data(start:stop);
OC12_ae3_data = OC12.testPoint.data(1).vData.ch(3).data(start:stop);
OC13_ae3_data = OC13.testPoint.data(1).vData.ch(3).data(start:stop);
OC14_ae3_data = OC14.testPoint.data(1).vData.ch(3).data(start:stop);
```

```
OC16_ae3_data = OC16.testPoint.data(1).vData.ch(3).data(start:stop);
OC17_ae3_data = OC17.testPoint.data(1).vData.ch(3).data(start:stop);
OC18_ae3_data = OC18.testPoint.data(1).vData.ch(3).data(start:stop);
OC01_ae4_data = OC01.testPoint.data(1).vData.ch(4).data(start:stop);
OC02_ae4_data = OC02.testPoint.data(1).vData.ch(4).data(start:stop);
OC03_ae4_data = OC03.testPoint.data(1).vData.ch(4).data(start:stop);
OC04_ae4_data = OC04.testPoint.data(1).vData.ch(4).data(start:stop);
OC07_ae4_data = OC07.testPoint.data(1).vData.ch(4).data(start:stop);
OC08_ae4_data = OC08.testPoint.data(1).vData.ch(4).data(start:stop);
OC09_ae4_data = OC09.testPoint.data(1).vData.ch(4).data(start:stop);
OC10_ae4_data = OC10.testPoint.data(1).vData.ch(4).data(start:stop);
OC11_ae4_data = OC11.testPoint.data(1).vData.ch(4).data(start:stop);
OC12_ae4_data = OC12.testPoint.data(1).vData.ch(4).data(start:stop);
OC13_ae4_data = OC13.testPoint.data(1).vData.ch(4).data(start:stop);
OC14_ae4_data = OC14.testPoint.data(1).vData.ch(4).data(start:stop);
OC16_ae4_data = OC16.testPoint.data(1).vData.ch(4).data(start:stop);
OC17_ae4_data = OC17.testPoint.data(1).vData.ch(4).data(start:stop);
OC18_ae4_data = OC18.testPoint.data(1).vData.ch(4).data(start:stop);

% %Calculate RMS
OC01_ae1_rms = rms(OC01_ae1_data);
OC02_ae1_rms = rms(OC02_ae1_data);
OC03_ae1_rms = rms(OC03_ae1_data);
OC04_ae1_rms = rms(OC04_ae1_data);
OC07_ae1_rms = rms(OC07_ae1_data);
OC08_ae1_rms = rms(OC08_ae1_data);
OC09_ae1_rms = rms(OC09_ae1_data);
OC10_ae1_rms = rms(OC10_ae1_data);
OC11_ae1_rms = rms(OC11_ae1_data);
OC12_ae1_rms = rms(OC12_ae1_data);
OC13_ae1_rms = rms(OC13_ae1_data);
OC14_ae1_rms = rms(OC14_ae1_data);
OC16_ae1_rms = rms(OC16_ae1_data);
OC17_ae1_rms = rms(OC17_ae1_data);
OC18_ae1_rms = rms(OC18_ae1_data);
OC01_ae3_rms = rms(OC01_ae3_data);
OC02_ae3_rms = rms(OC02_ae3_data);
OC03_ae3_rms = rms(OC03_ae3_data);
OC04_ae3_rms = rms(OC04_ae3_data);
OC07_ae3_rms = rms(OC07_ae3_data);
OC08_ae3_rms = rms(OC08_ae3_data);
OC09_ae3_rms = rms(OC09_ae3_data);
OC10_ae3_rms = rms(OC10_ae3_data);
OC11_ae3_rms = rms(OC11_ae3_data);
OC12_ae3_rms = rms(OC12_ae3_data);
OC13_ae3_rms = rms(OC13_ae3_data);
OC14_ae3_rms = rms(OC14_ae3_data);
OC16_ae3_rms = rms(OC16_ae3_data);
OC17_ae3_rms = rms(OC17_ae3_data);
OC18_ae3_rms = rms(OC18_ae3_data);
OC01_ae4_rms = rms(OC01_ae4_data);
OC02_ae4_rms = rms(OC02_ae4_data);
OC03_ae4_rms = rms(OC03_ae4_data);
OC04_ae4_rms = rms(OC04_ae4_data);
OC07_ae4_rms = rms(OC07_ae4_data);
OC08_ae4_rms = rms(OC08_ae4_data);
OC09_ae4_rms = rms(OC09_ae4_data);
```

```
OC10_ae4_rms = rms(OC10_ae4_data);
OC11_ae4_rms = rms(OC11_ae4_data);
OC12_ae4_rms = rms(OC12_ae4_data);
OC13_ae4_rms = rms(OC13_ae4_data);
OC14_ae4_rms = rms(OC14_ae4_data);
OC16_ae4_rms = rms(OC16_ae4_data);
OC17_ae4_rms = rms(OC17_ae4_data);
OC18_ae4_rms = rms(OC18_ae4_data);

%Make lists of acoustic measurements
O_ae1_rms = [OC01_ae1_rms OC02_ae1_rms OC03_ae1_rms OC04_ae1_rms OC07_ae1_rms
OC08_ae1_rms OC09_ae1_rms OC10_ae1_rms OC11_ae1_rms OC12_ae1_rms OC13_ae1_rms
OC14_ae1_rms OC16_ae1_rms OC17_ae1_rms OC18_ae1_rms];
O_ae3_rms = [OC01_ae3_rms OC02_ae3_rms OC03_ae3_rms OC04_ae3_rms OC07_ae3_rms
OC08_ae3_rms OC09_ae3_rms OC10_ae3_rms OC11_ae3_rms OC12_ae3_rms OC13_ae3_rms
OC14_ae3_rms OC16_ae3_rms OC17_ae3_rms OC18_ae3_rms];
O_ae4_rms = [OC01_ae4_rms OC02_ae4_rms OC03_ae4_rms OC04_ae4_rms OC07_ae4_rms
OC08_ae4_rms OC09_ae4_rms OC10_ae4_rms OC11_ae4_rms OC12_ae4_rms OC13_ae4_rms
OC14_ae4_rms OC16_ae4_rms OC17_ae4_rms OC18_ae4_rms];

%Make lists of flow measurements
O_Oref_Qv = [OC01.testPoint.oilRef.q OC02.testPoint.oilRef.q
OC03.testPoint.oilRef.q OC04.testPoint.oilRef.q OC07.testPoint.oilRef.q
OC08.testPoint.oilRef.q OC09.testPoint.oilRef.q OC10.testPoint.oilRef.q
OC11.testPoint.oilRef.q OC12.testPoint.oilRef.q OC13.testPoint.oilRef.q
OC14.testPoint.oilRef.q OC16.testPoint.oilRef.q OC17.testPoint.oilRef.q
OC18.testPoint.oilRef.q];

%Make lists of valve positions and pressure measurements
O_HIC = [OC01.testPoint.HIC.pos OC02.testPoint.HIC.pos OC03.testPoint.HIC.pos
OC04.testPoint.HIC.pos OC07.testPoint.HIC.pos OC08.testPoint.HIC.pos
OC09.testPoint.HIC.pos OC10.testPoint.HIC.pos OC11.testPoint.HIC.pos
OC12.testPoint.HIC.pos OC13.testPoint.HIC.pos OC14.testPoint.HIC.pos
OC16.testPoint.HIC.pos OC17.testPoint.HIC.pos OC18.testPoint.HIC.pos];
O_Pi = [OC01.testPoint.press.in OC02.testPoint.press.in OC03.testPoint.press.in
OC04.testPoint.press.in OC07.testPoint.press.in OC08.testPoint.press.in
OC09.testPoint.press.in OC10.testPoint.press.in OC11.testPoint.press.in
OC12.testPoint.press.in OC13.testPoint.press.in OC14.testPoint.press.in
OC16.testPoint.press.in OC17.testPoint.press.in OC18.testPoint.press.in];
O_Po = [OC01.testPoint.press.out OC02.testPoint.press.out OC03.testPoint.press.out
OC04.testPoint.press.out OC07.testPoint.press.out OC08.testPoint.press.out
OC09.testPoint.press.out OC10.testPoint.press.out OC11.testPoint.press.out
OC12.testPoint.press.out OC13.testPoint.press.out OC14.testPoint.press.out
OC16.testPoint.press.out OC17.testPoint.press.out OC18.testPoint.press.out];
O_wikaDp1 = [OC01.testPoint.Wika.dp1 OC02.testPoint.Wika.dp1
OC03.testPoint.Wika.dp1 OC04.testPoint.Wika.dp1 OC07.testPoint.Wika.dp1
OC08.testPoint.Wika.dp1 OC09.testPoint.Wika.dp1 OC10.testPoint.Wika.dp1
OC11.testPoint.Wika.dp1 OC12.testPoint.Wika.dp1 OC13.testPoint.Wika.dp1
OC14.testPoint.Wika.dp1 OC16.testPoint.Wika.dp1 OC17.testPoint.Wika.dp1
OC18.testPoint.Wika.dp1];
O_wikaDp2 = [OC01.testPoint.Wika.dp2 OC02.testPoint.Wika.dp1
OC03.testPoint.Wika.dp2 OC04.testPoint.Wika.dp2 OC07.testPoint.Wika.dp2
OC08.testPoint.Wika.dp2 OC09.testPoint.Wika.dp2 OC10.testPoint.Wika.dp2
OC11.testPoint.Wika.dp2 OC12.testPoint.Wika.dp2 OC13.testPoint.Wika.dp2
OC14.testPoint.Wika.dp2 OC16.testPoint.Wika.dp2 OC17.testPoint.Wika.dp2
OC18.testPoint.Wika.dp2];
```

```
O_wikaDp3 = [OC01.testPoint.Wika.dp3 OC02.testPoint.Wika.dp1
OC03.testPoint.Wika.dp3 OC04.testPoint.Wika.dp3 OC07.testPoint.Wika.dp3
OC08.testPoint.Wika.dp3 OC09.testPoint.Wika.dp3 OC10.testPoint.Wika.dp3
OC11.testPoint.Wika.dp3 OC12.testPoint.Wika.dp3 OC13.testPoint.Wika.dp3
OC14.testPoint.Wika.dp3 OC16.testPoint.Wika.dp3 OC17.testPoint.Wika.dp3
OC18.testPoint.Wika.dp3];
O_EmcoDp1 = [OC01.testPoint.Emco.dp1 OC02.testPoint.Wika.dp1
OC03.testPoint.Emco.dp1 OC04.testPoint.Emco.dp1 OC07.testPoint.Emco.dp1
OC08.testPoint.Emco.dp1 OC09.testPoint.Emco.dp1 OC10.testPoint.Emco.dp1
OC11.testPoint.Emco.dp1 OC12.testPoint.Emco.dp1 OC13.testPoint.Emco.dp1
OC14.testPoint.Emco.dp1 OC16.testPoint.Emco.dp1 OC17.testPoint.Emco.dp1
OC18.testPoint.Emco.dp1];
O_EmcoDp2 = [OC01.testPoint.Emco.dp2 OC02.testPoint.Wika.dp1
OC03.testPoint.Emco.dp2 OC04.testPoint.Emco.dp2 OC07.testPoint.Emco.dp2
OC08.testPoint.Emco.dp2 OC09.testPoint.Emco.dp2 OC10.testPoint.Emco.dp2
OC11.testPoint.Emco.dp2 OC12.testPoint.Emco.dp2 OC13.testPoint.Emco.dp2
OC14.testPoint.Emco.dp2 OC16.testPoint.Emco.dp2 OC17.testPoint.Emco.dp2
OC18.testPoint.Emco.dp2];
```

Appendix M

Preparing OW flow experiment data from dataset 1.

```
clc;
clear all;

%Load dataset
OW01 = load('OW01.mat');
OW02 = load('OW02.mat');
OW03 = load('OW03.mat');
OW04 = load('OW04.mat');
OW05 = load('OW05.mat');
OW06 = load('OW06.mat');
OW17 = load('OW17.mat');
OW19 = load('OW19.mat');
OW22 = load('OW22.mat');
OW23 = load('OW23.mat');
OWC10 = load('OWC10.mat');
OWC11 = load('OWC11.mat');
OWC12= load('OWC12.mat');
OWC13= load('OWC13.mat');
OWC14= load('OWC14.mat');
OWC15= load('OWC15.mat');
OWC16= load('OWC16.mat');
OWC17= load('OWC17.mat');
OWC18 = load('OWC18.mat');
OWC19 = load('OWC19.mat');
OWC20= load('OWC20.mat');

%Choose samples
start = 7705601;
stop = 7707600;

%Samples
OW01_ae1_data = OW01.testPoint.data(1).vData.ch(1).data(start:stop);
OW02_ae1_data = OW02.testPoint.data(1).vData.ch(1).data(start:stop);
OW03_ae1_data = OW03.testPoint.data(1).vData.ch(1).data(start:stop);
OW04_ae1_data = OW04.testPoint.data(1).vData.ch(1).data(start:stop);
OW05_ae1_data = OW05.testPoint.data(1).vData.ch(1).data(start:stop);
OW06_ae1_data = OW06.testPoint.data(1).vData.ch(1).data(start:stop);
OW17_ae1_data = OW17.testPoint.data(1).vData.ch(1).data(start:stop);
OW19_ae1_data = OW19.testPoint.data(1).vData.ch(1).data(start:stop);
OW22_ae1_data = OW22.testPoint.data(1).vData.ch(1).data(start:stop);
OW23_ae1_data = OW23.testPoint.data(1).vData.ch(1).data(start:stop);
OWC10_ae1_data = OWC10.testPoint.data(1).vData.ch(1).data(start:stop);
OWC11_ae1_data = OWC11.testPoint.data(1).vData.ch(1).data(start:stop);
OWC12_ae1_data = OWC12.testPoint.data(1).vData.ch(1).data(start:stop);
OWC13_ae1_data = OWC13.testPoint.data(1).vData.ch(1).data(start:stop);
OWC14_ae1_data = OWC14.testPoint.data(1).vData.ch(1).data(start:stop);
OWC15_ae1_data = OWC15.testPoint.data(1).vData.ch(1).data(start:stop);
OWC16_ae1_data = OWC16.testPoint.data(1).vData.ch(1).data(start:stop);
OWC17_ae1_data = OWC17.testPoint.data(1).vData.ch(1).data(start:stop);
OWC18_ae1_data = OWC18.testPoint.data(1).vData.ch(1).data(start:stop);
OWC19_ae1_data = OWC19.testPoint.data(1).vData.ch(1).data(start:stop);
OWC20_ae1_data = OWC20.testPoint.data(1).vData.ch(1).data(start:stop);
```

```
OW01_ae3_data = OW01.testPoint.data(1).vData.ch(3).data(start:stop);
OW02_ae3_data = OW02.testPoint.data(1).vData.ch(3).data(start:stop);
OW03_ae3_data = OW03.testPoint.data(1).vData.ch(3).data(start:stop);
OW04_ae3_data = OW04.testPoint.data(1).vData.ch(3).data(start:stop);
OW05_ae3_data = OW05.testPoint.data(1).vData.ch(3).data(start:stop);
OW06_ae3_data = OW06.testPoint.data(1).vData.ch(3).data(start:stop);
OW17_ae3_data = OW17.testPoint.data(1).vData.ch(3).data(start:stop);
OW19_ae3_data = OW19.testPoint.data(1).vData.ch(3).data(start:stop);
OW22_ae3_data = OW22.testPoint.data(1).vData.ch(3).data(start:stop);
OW23_ae3_data = OW23.testPoint.data(1).vData.ch(3).data(start:stop);
OWC10_ae3_data = OWC10.testPoint.data(1).vData.ch(3).data(start:stop);
OWC11_ae3_data = OWC11.testPoint.data(1).vData.ch(3).data(start:stop);
OWC12_ae3_data = OWC12.testPoint.data(1).vData.ch(3).data(start:stop);
OWC13_ae3_data = OWC13.testPoint.data(1).vData.ch(3).data(start:stop);
OWC14_ae3_data = OWC14.testPoint.data(1).vData.ch(3).data(start:stop);
OWC15_ae3_data = OWC15.testPoint.data(1).vData.ch(3).data(start:stop);
OWC16_ae3_data = OWC16.testPoint.data(1).vData.ch(3).data(start:stop);
OWC17_ae3_data = OWC17.testPoint.data(1).vData.ch(3).data(start:stop);
OWC18_ae3_data = OWC18.testPoint.data(1).vData.ch(3).data(start:stop);
OWC19_ae3_data = OWC19.testPoint.data(1).vData.ch(3).data(start:stop);
OWC20_ae3_data = OWC20.testPoint.data(1).vData.ch(3).data(start:stop);
OW01_ae4_data = OW01.testPoint.data(1).vData.ch(4).data(start:stop);
OW02_ae4_data = OW02.testPoint.data(1).vData.ch(4).data(start:stop);
OW03_ae4_data = OW03.testPoint.data(1).vData.ch(4).data(start:stop);
OW04_ae4_data = OW04.testPoint.data(1).vData.ch(4).data(start:stop);
OW05_ae4_data = OW05.testPoint.data(1).vData.ch(4).data(start:stop);
OW06_ae4_data = OW06.testPoint.data(1).vData.ch(4).data(start:stop);
OW17_ae4_data = OW17.testPoint.data(1).vData.ch(4).data(start:stop);
OW19_ae4_data = OW19.testPoint.data(1).vData.ch(4).data(start:stop);
OW22_ae4_data = OW22.testPoint.data(1).vData.ch(4).data(start:stop);
OW23_ae4_data = OW23.testPoint.data(1).vData.ch(4).data(start:stop);
OWC10_ae4_data = OWC10.testPoint.data(1).vData.ch(4).data(start:stop);
OWC11_ae4_data = OWC11.testPoint.data(1).vData.ch(4).data(start:stop);
OWC12_ae4_data = OWC12.testPoint.data(1).vData.ch(4).data(start:stop);
OWC13_ae4_data = OWC13.testPoint.data(1).vData.ch(4).data(start:stop);
OWC14_ae4_data = OWC14.testPoint.data(1).vData.ch(4).data(start:stop);
OWC15_ae4_data = OWC15.testPoint.data(1).vData.ch(4).data(start:stop);
OWC16_ae4_data = OWC16.testPoint.data(1).vData.ch(4).data(start:stop);
OWC17_ae4_data = OWC17.testPoint.data(1).vData.ch(4).data(start:stop);
OWC18_ae4_data = OWC18.testPoint.data(1).vData.ch(4).data(start:stop);
OWC19_ae4_data = OWC19.testPoint.data(1).vData.ch(4).data(start:stop);
OWC20_ae4_data = OWC20.testPoint.data(1).vData.ch(4).data(start:stop);

%Calculate RMS
OW01_ae1_rms = rms(OW01_ae1_data);
OW02_ae1_rms = rms(OW02_ae1_data);
OW03_ae1_rms = rms(OW03_ae1_data);
OW04_ae1_rms = rms(OW04_ae1_data);
OW05_ae1_rms = rms(OW05_ae1_data);
OW06_ae1_rms = rms(OW06_ae1_data);
OW17_ae1_rms = rms(OW17_ae1_data);
OW19_ae1_rms = rms(OW19_ae1_data);
OW22_ae1_rms = rms(OW22_ae1_data);
OW23_ae1_rms = rms(OW23_ae1_data);
OWC10_ae1_rms = rms(OWC10_ae1_data);
OWC11_ae1_rms = rms(OWC11_ae1_data);
OWC12_ae1_rms = rms(OWC12_ae1_data);
```

```
OWC13_ae1_rms = rms(OWC13_ae1_data);
OWC14_ae1_rms = rms(OWC14_ae1_data);
OWC15_ae1_rms = rms(OWC15_ae1_data);
OWC16_ae1_rms = rms(OWC16_ae1_data);
OWC17_ae1_rms = rms(OWC17_ae1_data);
OWC18_ae1_rms = rms(OWC18_ae1_data);
OWC19_ae1_rms = rms(OWC19_ae1_data);
OWC20_ae1_rms = rms(OWC20_ae1_data);
OW01_ae3_rms = rms(OW01_ae3_data);
OW02_ae3_rms = rms(OW02_ae3_data);
OW03_ae3_rms = rms(OW03_ae3_data);
OW04_ae3_rms = rms(OW04_ae3_data);
OW05_ae3_rms = rms(OW05_ae3_data);
OW06_ae3_rms = rms(OW06_ae3_data);
OW17_ae3_rms = rms(OW17_ae3_data);
OW19_ae3_rms = rms(OW19_ae3_data);
OW22_ae3_rms = rms(OW22_ae3_data);
OW23_ae3_rms = rms(OW23_ae3_data);
OWC10_ae3_rms = rms(OWC10_ae3_data);
OWC11_ae3_rms = rms(OWC11_ae3_data);
OWC12_ae3_rms = rms(OWC12_ae3_data);
OWC13_ae3_rms = rms(OWC13_ae3_data);
OWC14_ae3_rms = rms(OWC14_ae3_data);
OWC15_ae3_rms = rms(OWC15_ae3_data);
OWC16_ae3_rms = rms(OWC16_ae3_data);
OWC17_ae3_rms = rms(OWC17_ae3_data);
OWC18_ae3_rms = rms(OWC18_ae3_data);
OWC19_ae3_rms = rms(OWC19_ae3_data);
OWC20_ae3_rms = rms(OWC20_ae3_data);
OW01_ae4_rms = rms(OW01_ae4_data);
OW02_ae4_rms = rms(OW02_ae4_data);
OW03_ae4_rms = rms(OW03_ae4_data);
OW04_ae4_rms = rms(OW04_ae4_data);
OW05_ae4_rms = rms(OW05_ae4_data);
OW06_ae4_rms = rms(OW06_ae4_data);
OW17_ae4_rms = rms(OW17_ae4_data);
OW19_ae4_rms = rms(OW19_ae4_data);
OW22_ae4_rms = rms(OW22_ae4_data);
OW23_ae4_rms = rms(OW23_ae4_data);
OWC10_ae4_rms = rms(OWC10_ae4_data);
OWC11_ae4_rms = rms(OWC11_ae4_data);
OWC12_ae4_rms = rms(OWC12_ae4_data);
OWC13_ae4_rms = rms(OWC13_ae4_data);
OWC14_ae4_rms = rms(OWC14_ae4_data);
OWC15_ae4_rms = rms(OWC15_ae4_data);
OWC16_ae4_rms = rms(OWC16_ae4_data);
OWC17_ae4_rms = rms(OWC17_ae4_data);
OWC18_ae4_rms = rms(OWC18_ae4_data);
OWC19_ae4_rms = rms(OWC19_ae4_data);
OWC20_ae4_rms = rms(OWC20_ae4_data);

%Make lists of acoustic measurements
OW_ae1_data = [OW01_ae1_data; OW02_ae1_data; OW03_ae1_data; OW04_ae1_data;
OW05_ae1_data; OW06_ae1_data; OW17_ae1_data; OW19_ae1_data; OW22_ae1_data;
OW23_ae1_data; OWC10_ae1_data; OWC11_ae1_data; OWC12_ae1_data; OWC13_ae1_data;
OWC14_ae1_data; OWC15_ae1_data; OWC16_ae1_data; OWC17_ae1_data; OWC18_ae1_data;
OWC19_ae1_data; OWC20_ae1_data];
```

```
OW_ae3_data = [OW01_ae3_data; OW02_ae3_data; OW03_ae3_data; OW04_ae3_data;
OW05_ae3_data; OW06_ae3_data; OW17_ae3_data; OW19_ae3_data; OW22_ae3_data;
OW23_ae3_data; OWC10_ae3_data; OWC11_ae3_data; OWC12_ae3_data; OWC13_ae3_data;
OWC14_ae3_data; OWC15_ae3_data; OWC16_ae3_data; OWC17_ae3_data; OWC18_ae3_data;
OWC19_ae3_data; OWC20_ae3_data];
OW_ae4_data = [OW01_ae4_data; OW02_ae4_data; OW03_ae4_data; OW04_ae4_data;
OW05_ae4_data; OW06_ae4_data; OW17_ae4_data; OW19_ae4_data; OW22_ae4_data;
OW23_ae4_data; OWC10_ae4_data; OWC11_ae4_data; OWC12_ae4_data; OWC13_ae4_data;
OWC14_ae4_data; OWC15_ae4_data; OWC16_ae4_data; OWC17_ae4_data; OWC18_ae4_data;
OWC19_ae4_data; OWC20_ae4_data];
OW_ae1_rms = [OW01_ae1_rms OW02_ae1_rms OW03_ae1_rms OW04_ae1_rms OW05_ae1_rms
OW06_ae1_rms OW17_ae1_rms OW19_ae1_rms OW22_ae1_rms OW23_ae1_rms OWC10_ae1_rms
OWC11_ae1_rms OWC12_ae1_rms OWC13_ae1_rms OWC14_ae1_rms OWC15_ae1_rms
OWC16_ae1_rms OWC17_ae1_rms OWC18_ae1_rms OWC19_ae1_rms OWC20_ae1_rms];
OW_ae3_rms = [OW01_ae3_rms OW02_ae3_rms OW03_ae3_rms OW04_ae3_rms OW05_ae3_rms
OW06_ae3_rms OW17_ae3_rms OW19_ae3_rms OW22_ae3_rms OW23_ae3_rms OWC10_ae3_rms
OWC11_ae3_rms OWC12_ae3_rms OWC13_ae3_rms OWC14_ae3_rms OWC15_ae3_rms
OWC16_ae3_rms OWC17_ae3_rms OWC18_ae3_rms OWC19_ae3_rms OWC20_ae3_rms];
OW_ae4_rms = [OW01_ae4_rms OW02_ae4_rms OW03_ae4_rms OW04_ae4_rms OW05_ae4_rms
OW06_ae4_rms OW17_ae4_rms OW19_ae4_rms OW22_ae4_rms OW23_ae4_rms OWC10_ae4_rms
OWC11_ae4_rms OWC12_ae4_rms OWC13_ae4_rms OWC14_ae4_rms OWC15_ae4_rms
OWC16_ae4_rms OWC17_ae4_rms OWC18_ae4_rms OWC19_ae4_rms OWC20_ae4_rms];

%Make lists of flow measurements
OW_Wref_Qv = [OW01.testPoint.watRef.q  OW02.testPoint.watRef.q
OW03.testPoint.watRef.q  OW04.testPoint.watRef.q  OW05.testPoint.watRef.q
OW06.testPoint.watRef.q  OW17.testPoint.watRef.q OW19.testPoint.watRef.q
OW22.testPoint.watRef.q OW23.testPoint.watRef.q OWC10.testPoint.watRef.q
OWC11.testPoint.watRef.q OWC12.testPoint.watRef.q OWC13.testPoint.watRef.q
OWC14.testPoint.watRef.q OWC15.testPoint.watRef.q OWC16.testPoint.watRef.q
OWC17.testPoint.watRef.q OWC18.testPoint.watRef.q OWC19.testPoint.watRef.q
OWC20.testPoint.watRef.q];
OW_Oref_Qv = [OW01.testPoint.oilRef.q OW02.testPoint.oilRef.q
OW03.testPoint.oilRef.q OW04.testPoint.oilRef.q OW05.testPoint.oilRef.q
OW06.testPoint.oilRef.q OW17.testPoint.oilRef.q OW19.testPoint.oilRef.q
OW22.testPoint.oilRef.q OW23.testPoint.oilRef.q OWC10.testPoint.oilRef.q
OWC11.testPoint.oilRef.q OWC12.testPoint.oilRef.q OWC13.testPoint.oilRef.q
OWC14.testPoint.oilRef.q OWC15.testPoint.oilRef.q OWC16.testPoint.oilRef.q
OWC17.testPoint.oilRef.q OWC18.testPoint.oilRef.q OWC19.testPoint.oilRef.q
OWC20.testPoint.oilRef.q];
OW_Gref_Qv = [OW01.testPoint.gasRef.q OW02.testPoint.gasRef.q
OW03.testPoint.gasRef.q OW04.testPoint.gasRef.q OW05.testPoint.gasRef.q
OW06.testPoint.gasRef.q OW17.testPoint.gasRef.q OW19.testPoint.gasRef.q
OW22.testPoint.gasRef.q OW23.testPoint.gasRef.q OWC10.testPoint.gasRef.q
OWC11.testPoint.gasRef.q OWC12.testPoint.gasRef.q OWC13.testPoint.gasRef.q
OWC14.testPoint.gasRef.q OWC15.testPoint.gasRef.q OWC16.testPoint.gasRef.q
OWC17.testPoint.gasRef.q OWC18.testPoint.gasRef.q OWC19.testPoint.gasRef.q
OWC20.testPoint.gasRef.q];

%Make lists of valve positions and pressure measurements
OW_HIC = [OW01.testPoint.HIC.pos OW02.testPoint.HIC.pos OW03.testPoint.HIC.pos
OW04.testPoint.HIC.pos OW05.testPoint.HIC.pos OW06.testPoint.HIC.pos
OW17.testPoint.HIC.pos OW19.testPoint.HIC.pos OW22.testPoint.HIC.pos
OW23.testPoint.HIC.pos OWC10.testPoint.HIC.pos OWC11.testPoint.HIC.pos
OWC12.testPoint.HIC.pos OWC13.testPoint.HIC.pos OWC14.testPoint.HIC.pos
OWC15.testPoint.HIC.pos OWC16.testPoint.HIC.pos OWC17.testPoint.HIC.pos
OWC18.testPoint.HIC.pos OWC19.testPoint.HIC.pos OWC20.testPoint.HIC.pos];
```

```
OW_Pi = [OW01.testPoint.press.in OW02.testPoint.press.in OW03.testPoint.press.in
OW04.testPoint.press.in OW05.testPoint.press.in OW06.testPoint.press.in
OW17.testPoint.press.in OW19.testPoint.press.in OW22.testPoint.press.in
OW23.testPoint.press.in OWC10.testPoint.press.in OWC11.testPoint.press.in
OWC12.testPoint.press.in OWC13.testPoint.press.in OWC14.testPoint.press.in
OWC15.testPoint.press.in OWC16.testPoint.press.in OWC17.testPoint.press.in
OWC18.testPoint.press.in OWC19.testPoint.press.in OWC20.testPoint.press.in];
OW_Po = [OW01.testPoint.press.out OW02.testPoint.press.out
OW03.testPoint.press.out OW04.testPoint.press.out OW05.testPoint.press.out
OW06.testPoint.press.out OW17.testPoint.press.out OW19.testPoint.press.out
OW22.testPoint.press.out OW23.testPoint.press.out OWC10.testPoint.press.out
OWC11.testPoint.press.out OWC12.testPoint.press.out OWC13.testPoint.press.out
OWC14.testPoint.press.out OWC15.testPoint.press.out OWC16.testPoint.press.out
OWC17.testPoint.press.out OWC18.testPoint.press.out OWC19.testPoint.press.out
OWC20.testPoint.press.out];
OW_wikaDp1 = [OW01.testPoint.Wika.dp1 OW02.testPoint.Wika.dp1
OW03.testPoint.Wika.dp1 OW04.testPoint.Wika.dp1 OW05.testPoint.Wika.dp1
OW06.testPoint.Wika.dp1 OW17.testPoint.Wika.dp1 OW19.testPoint.Wika.dp1
OW22.testPoint.Wika.dp1 OW23.testPoint.Wika.dp1 OWC10.testPoint.Wika.dp1
OWC11.testPoint.Wika.dp1 OWC12.testPoint.Wika.dp1 OWC13.testPoint.Wika.dp1
OWC14.testPoint.Wika.dp1 OWC15.testPoint.Wika.dp1 OWC16.testPoint.Wika.dp1
OWC17.testPoint.Wika.dp1 OWC18.testPoint.Wika.dp1 OWC19.testPoint.Wika.dp1
OWC20.testPoint.Wika.dp1];
OW_wikaDp2 = [OW01.testPoint.Wika.dp2 OW02.testPoint.Wika.dp2
OW03.testPoint.Wika.dp2 OW04.testPoint.Wika.dp2 OW05.testPoint.Wika.dp2
OW06.testPoint.Wika.dp2 OW17.testPoint.Wika.dp2 OW19.testPoint.Wika.dp2
OW22.testPoint.Wika.dp2 OW23.testPoint.Wika.dp2 OWC10.testPoint.Wika.dp2
OWC11.testPoint.Wika.dp2 OWC12.testPoint.Wika.dp2 OWC13.testPoint.Wika.dp2
OWC14.testPoint.Wika.dp2 OWC15.testPoint.Wika.dp2 OWC16.testPoint.Wika.dp2
OWC17.testPoint.Wika.dp2 OWC18.testPoint.Wika.dp2 OWC19.testPoint.Wika.dp2
OWC20.testPoint.Wika.dp2];
OW_wikaDp3 = [OW01.testPoint.Wika.dp3 OW02.testPoint.Wika.dp3
OW03.testPoint.Wika.dp3 OW04.testPoint.Wika.dp3 OW05.testPoint.Wika.dp3
OW06.testPoint.Wika.dp3 OW17.testPoint.Wika.dp3 OW19.testPoint.Wika.dp3
OW22.testPoint.Wika.dp3 OW23.testPoint.Wika.dp3 OWC10.testPoint.Wika.dp3
OWC11.testPoint.Wika.dp3 OWC12.testPoint.Wika.dp3 OWC13.testPoint.Wika.dp3
OWC14.testPoint.Wika.dp3 OWC15.testPoint.Wika.dp3 OWC16.testPoint.Wika.dp3
OWC17.testPoint.Wika.dp3 OWC18.testPoint.Wika.dp3 OWC19.testPoint.Wika.dp3
OWC20.testPoint.Wika.dp3];
OW_EmcoDp1 = [OW01.testPoint.Emco.dp1 OW02.testPoint.Emco.dp1
OW03.testPoint.Emco.dp1 OW04.testPoint.Emco.dp1 OW05.testPoint.Emco.dp1
OW06.testPoint.Emco.dp1 OW17.testPoint.Emco.dp1 OW19.testPoint.Emco.dp1
OW22.testPoint.Emco.dp1 OW23.testPoint.Emco.dp1 OWC10.testPoint.Emco.dp1
OWC11.testPoint.Emco.dp1 OWC12.testPoint.Emco.dp1 OWC13.testPoint.Emco.dp1
OWC14.testPoint.Emco.dp1 OWC15.testPoint.Emco.dp1 OWC16.testPoint.Emco.dp1
OWC17.testPoint.Emco.dp1 OWC18.testPoint.Emco.dp1 OWC19.testPoint.Emco.dp1
OWC20.testPoint.Emco.dp1];
OW_EmcoDp2 = [OW01.testPoint.Emco.dp2 OW02.testPoint.Emco.dp2
OW03.testPoint.Emco.dp2 OW04.testPoint.Emco.dp2 OW05.testPoint.Emco.dp2
OW06.testPoint.Emco.dp2 OW17.testPoint.Emco.dp2 OW19.testPoint.Emco.dp2
OW22.testPoint.Emco.dp2 OW23.testPoint.Emco.dp2 OWC10.testPoint.Emco.dp2
OWC11.testPoint.Emco.dp2 OWC12.testPoint.Emco.dp2 OWC13.testPoint.Emco.dp2
OWC14.testPoint.Emco.dp2 OWC15.testPoint.Emco.dp2 OWC16.testPoint.Emco.dp2
OWC17.testPoint.Emco.dp2 OWC18.testPoint.Emco.dp2 OWC19.testPoint.Emco.dp2
OWC20.testPoint.Emco.dp2];

%Make lists of density measurements
```

```
OW_Oref_rho = [OW01.testPoint.oilRef.rho OW02.testPoint.oilRef.rho
OW03.testPoint.oilRef.rho OW04.testPoint.oilRef.rho OW05.testPoint.oilRef.rho
OW06.testPoint.oilRef.rho OW17.testPoint.oilRef.rho OW19.testPoint.oilRef.rho
OW22.testPoint.oilRef.rho OW23.testPoint.oilRef.rho OWC10.testPoint.oilRef.rho
OWC11.testPoint.oilRef.rho OWC12.testPoint.oilRef.rho OWC13.testPoint.oilRef.rho
OWC14.testPoint.oilRef.rho OWC15.testPoint.oilRef.rho OWC16.testPoint.oilRef.rho
OWC17.testPoint.oilRef.rho OWC18.testPoint.oilRef.rho OWC19.testPoint.oilRef.rho
OWC20.testPoint.oilRef.rho];
OW_Wref_rho = [OW01.testPoint.watRef.rho OW02.testPoint.watRef.rho
OW03.testPoint.watRef.rho OW04.testPoint.watRef.rho OW05.testPoint.watRef.rho
OW06.testPoint.watRef.rho OW17.testPoint.watRef.rho OW19.testPoint.watRef.rho
OW22.testPoint.watRef.rho OW23.testPoint.watRef.rho OWC10.testPoint.watRef.rho
OWC11.testPoint.watRef.rho OWC12.testPoint.watRef.rho OWC13.testPoint.watRef.rho
OWC14.testPoint.watRef.rho OWC15.testPoint.watRef.rho OWC16.testPoint.watRef.rho
OWC17.testPoint.watRef.rho OWC18.testPoint.watRef.rho OWC19.testPoint.watRef.rho
OWC20.testPoint.watRef.rho];
```

Appendix N

Preparing water flow experiment data from dataset 1.

```
clc;
clear all;

% %Load dataset
W01 = load('W01.mat');
W02 = load('W02.mat');
W03 = load('W03.mat');
W08 = load('W08.mat');
W09 = load('W09.mat');
W10 = load('W10.mat');
W11 = load('W11.mat');
W12 = load('W12.mat');


%Choose samples
start = 7705601;
stop = 7707600;

%Samples
W01_ae1_data = W01.testPoint.data(1).vData.ch(1).data(start:stop);
W02_ae1_data = W02.testPoint.data(1).vData.ch(1).data(start:stop);
W03_ae1_data = W03.testPoint.data(1).vData.ch(1).data(start:stop);
W08_ae1_data = W08.testPoint.data(1).vData.ch(1).data(start:stop);
W09_ae1_data = W09.testPoint.data(1).vData.ch(1).data(start:stop);
W10_ae1_data = W10.testPoint.data(1).vData.ch(1).data(start:stop);
W11_ae1_data = W11.testPoint.data(1).vData.ch(1).data(start:stop);
W12_ae1_data = W12.testPoint.data(1).vData.ch(1).data(start:stop);
W01_ae3_data = W01.testPoint.data(1).vData.ch(3).data(start:stop);
W02_ae3_data = W02.testPoint.data(1).vData.ch(3).data(start:stop);
W03_ae3_data = W03.testPoint.data(1).vData.ch(3).data(start:stop);
W08_ae3_data = W08.testPoint.data(1).vData.ch(3).data(start:stop);
W09_ae3_data = W09.testPoint.data(1).vData.ch(3).data(start:stop);
W10_ae3_data = W10.testPoint.data(1).vData.ch(3).data(start:stop);
W11_ae3_data = W11.testPoint.data(1).vData.ch(3).data(start:stop);
W12_ae3_data = W12.testPoint.data(1).vData.ch(3).data(start:stop);
W01_ae4_data = W01.testPoint.data(1).vData.ch(4).data(start:stop);
W02_ae4_data = W02.testPoint.data(1).vData.ch(4).data(start:stop);
W03_ae4_data = W03.testPoint.data(1).vData.ch(4).data(start:stop);
W08_ae4_data = W08.testPoint.data(1).vData.ch(4).data(start:stop);
W09_ae4_data = W09.testPoint.data(1).vData.ch(4).data(start:stop);
W10_ae4_data = W10.testPoint.data(1).vData.ch(4).data(start:stop);
W11_ae4_data = W11.testPoint.data(1).vData.ch(4).data(start:stop);
W12_ae4_data = W12.testPoint.data(1).vData.ch(4).data(start:stop);

% %Calculate RMS
W01_ae1_rms = rms(W01_ae1_data);
W02_ae1_rms = rms(W02_ae1_data);
W03_ae1_rms = rms(W03_ae1_data);
W08_ae1_rms = rms(W08_ae1_data);
W09_ae1_rms = rms(W09_ae1_data);
W10_ae1_rms = rms(W10_ae1_data);
W11_ae1_rms = rms(W11_ae1_data);
```

```
W12_ae1_rms = rms(W12_ae1_data);
W01_ae3_rms = rms(W01_ae3_data);
W02_ae3_rms = rms(W02_ae3_data);
W03_ae3_rms = rms(W03_ae3_data);
W08_ae3_rms = rms(W08_ae3_data);
W09_ae3_rms = rms(W09_ae3_data);
W10_ae3_rms = rms(W10_ae3_data);
W11_ae3_rms = rms(W11_ae3_data);
W12_ae3_rms = rms(W12_ae3_data);
W01_ae4_rms = rms(W01_ae4_data);
W02_ae4_rms = rms(W02_ae4_data);
W03_ae4_rms = rms(W03_ae4_data);
W08_ae4_rms = rms(W08_ae4_data);
W09_ae4_rms = rms(W09_ae4_data);
W10_ae4_rms = rms(W10_ae4_data);
W11_ae4_rms = rms(W11_ae4_data);
W12_ae4_rms = rms(W12_ae4_data);


%Make lists of acoustic measurements
W_ae1_rms = [W01_ae1_rms W02_ae1_rms W03_ae1_rms W08_ae1_rms W09_ae1_rms
W10_ae1_rms W11_ae1_rms W12_ae1_rms];
W_ae3_rms = [W01_ae3_rms W02_ae3_rms W03_ae3_rms W08_ae3_rms W09_ae3_rms
W10_ae3_rms W11_ae3_rms W12_ae3_rms];
W_ae4_rms = [W01_ae4_rms W02_ae4_rms W03_ae4_rms W08_ae4_rms W09_ae4_rms
W10_ae4_rms W11_ae4_rms W12_ae4_rms];

%Make lists of flow measurements
W_Wref_Qv = [W01.testPoint.watRef.q W02.testPoint.watRef.q W03.testPoint.watRef.q
W08.testPoint.watRef.q W09.testPoint.oilRef.q W10.testPoint.watRef.q
W11.testPoint.watRef.q W12.testPoint.watRef.q];

%Make lists of valve positions and pressure measurements
W_HIC = [W01.testPoint.HIC.pos W02.testPoint.HIC.pos W03.testPoint.HIC.pos
W08.testPoint.HIC.pos W09.testPoint.HIC.pos W10.testPoint.HIC.pos
W11.testPoint.HIC.pos W12.testPoint.HIC.pos];
W_Pi = [W01.testPoint.press.in W02.testPoint.press.in W03.testPoint.press.in
W08.testPoint.press.in W09.testPoint.press.in W10.testPoint.press.in
W11.testPoint.press.in W12.testPoint.press.in];
W_Po = [W01.testPoint.press.out W02.testPoint.press.out W03.testPoint.press.out
W08.testPoint.press.out W09.testPoint.press.out W10.testPoint.press.out
W11.testPoint.press.out W12.testPoint.press.out];
W_wikaDp1 = [W01.testPoint.Wika.dp1 W02.testPoint.Wika.dp1 W03.testPoint.Wika.dp1
W08.testPoint.Wika.dp1 W09.testPoint.Wika.dp1 W10.testPoint.Wika.dp1
W11.testPoint.Wika.dp1 W12.testPoint.Wika.dp1];
W_wikaDp2 = [W01.testPoint.Wika.dp2 W02.testPoint.Wika.dp1 W03.testPoint.Wika.dp2
W08.testPoint.Wika.dp2 W09.testPoint.Wika.dp2 W10.testPoint.Wika.dp2
W11.testPoint.Wika.dp2 W12.testPoint.Wika.dp2];
W_wikaDp3 = [W01.testPoint.Wika.dp3 W02.testPoint.Wika.dp1 W03.testPoint.Wika.dp3
W08.testPoint.Wika.dp3 W09.testPoint.Wika.dp3 W10.testPoint.Wika.dp3
W11.testPoint.Wika.dp3 W12.testPoint.Wika.dp3];
W_EmcoDp1 = [W01.testPoint.Emco.dp1 W02.testPoint.Wika.dp1 W03.testPoint.Emco.dp1
W08.testPoint.Emco.dp1 W09.testPoint.Emco.dp1 W10.testPoint.Emco.dp1
W11.testPoint.Emco.dp1 W12.testPoint.Emco.dp1];
W_EmcoDp2 = [W01.testPoint.Emco.dp2 W02.testPoint.Wika.dp1 W03.testPoint.Emco.dp2
W08.testPoint.Emco.dp2 W09.testPoint.Emco.dp2 W10.testPoint.Emco.dp2
W11.testPoint.Emco.dp2 W12.testPoint.Emco.dp2];
```