Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Geoscience and Petroleum

Martin Pham

Using Acoustic Emission to monitor sand production

Master's thesis in MTPETR
Supervisor: Tor Berge Gjersvik
Co-supervisor: Sigbjørn Sangesland
June 2023

NTNU
Norwegian University of
Science and Technology

Martin Pham

Using Acoustic Emission to monitor sand production

Master's thesis in MTPETR
Supervisor: Tor Berge Gjersvik
Co-supervisor: Sigbjørn Sangesland
June 2023

Norwegian University of Science and Technology
Faculty of Engineering
Department of Geoscience and Petroleum

NTNU
Norwegian University of
Science and Technology

Abstract

This thesis focuses on improving the glass tube used in the sand injection system and developing a Python script for analyzing sand grain diameters. The goal is to enhance the performance and reliability of sand production monitoring and provide accurate measurements of sand grain sizes.

The study addresses challenges related to the fragility and breakage of glass tubes in sand injection applications. Issues such as skipping and irregular flow patterns are resolved, leading to improved sand flow and pumping efficiency.

Additionally, a Python script is developed for precise analysis of sand grain diameters. The script demonstrates excellent performance in analyzing pure sand mixes, providing valuable insights into sand mixture characterization. However, challenges arise when mixed sand sizes are introduced, resulting in higher error values. Future refinements and optimizations of the script are necessary to enhance accuracy and reliability in mixed sand scenarios.

In conclusion, this thesis addresses the challenges associated with glass tubes in sand injection systems and develops a Python script for accurate sand grain diameter analysis. Further refinements of the glass tube design and the Python script are recommended to improve their performance in mixed sand scenarios. Enhancing these aspects will optimize the efficiency and reliability of sand production monitoring, benefiting various industrial applications.

Sammendrag

Denne oppgaven tar for seg oppgraderinger av sandinjeksjons-systemet i flow-loop systemet, og utviklingen av et Python-program for å analysere kornstørrelser hos sand. Målet er å forbedre ytelsen og påliteligheten til monitorering av sandproduksjon og gi nøyaktige målinger av sandkornstørrelser.

Studien tar for seg utfordringer knyttet til skjørhet og brudd i glassrøret brukt i sandinjeksjons-systemet. Problemer som hopping og uregelmessig strømning løses, noe som fører til forbedret sandstrøm og pumpeeffektivitet.

I tillegg utvikles et Python-program for presis analyse av sandkornstørrelser. Programmet viser utmerket ytelse i analyse av rene sandblandinger og gir verdifulle resultater i karakterisering av sandblandinger. Imidlertid oppstår utfordringer når blandinger med forskjellige sandstørrelser introduseres, noe som resulterer i høyere feilmarginer. Fremtidige justeringer og optimaliseringer av programmet er nødvendig for å forbedre nøyaktigheten og påliteligheten i tilfeller med blandet sand.

For å konkludere, tar denne oppgaven for seg utfordringene knyttet til glassrør i sandinjeksjonssystemer og utviklingen av et Python-program for analyse av sandkornstørrelser. Videre forbedringer av glassrørets design og Python-programmet anbefales for å forbedre ytelsen i blandet sand. Å forbedre disse aspektene vil optimalisere effektiviteten og påliteligheten til monitorering av sand produksjon, til nytte for ulike industrielle bruksområder.

Acknowledgements

I would like to express my gratitude to my supervisors, Tor Berge Gjersvik and Sigbjørn Sangesland, for their invaluable guidance throughout the course of this research. I am truly grateful for their mentorship and the opportunity to work under their supervision, as well as the research topic.

I would like to express my gratitude to Igor Grishkov for his contributions to my research. Igor's guidance in his previous work have been instrumental in shaping the direction of my own study. His willingness to assist me with my inquiries has been helpful throughout this process, and I am truly fortunate to have had the opportunity to collaborate with him.

Lastly, I would also like to extend my appreciation to my lab supervisor, Noralf Vedvik, for his exceptional support and guidance throughout this research project. Noralf has played a vital role in ensuring the smooth operation of the equipment, assisting me with fitting and building parts, and providing valuable insights into various equipment-related queries. His expertise, and willingness to assist have been invaluable assets to the success of this study. I am sincerely grateful for his unwavering support and contributions to my research.

Table of contents

List of figures

List of tables

1 Introduction

Sand monitoring is a critical component of the petroleum industry due to the potential hazards and expenses associated with sand production. Equipment damage caused by sand, such as to pipeline bends, chokes, and valves, can be both dangerous and costly, highlighting the need for effective monitoring and surveillance. Various approaches, including unplanned shutdowns and well fluid sampling, can be utilized to detect sand production. In this project, our primary focus is on using sensors to monitor sand production and detect early signs of sand production.

The most advanced and promising method for sand monitoring today is non-intrusive Acoustic Emission systems, which are simple to install. These systems detect noise created by sand grains as they collide with the pipe wall following a pipe bend. The noise can be picked up by an acoustic sensor installed outside the pipe and converted into an electrical signal for computer analysis after sampling and filtering. However, these acoustic signals are not easy to analyze, as several factors can distort or alter the signals, making interpretation difficult.

2 Objectives and plan

This research aims to develop a model that can accurately estimate the concentration of sand in fluid flow by analyzing AE signals. Although there are existing models that have been commercially implemented in hydrocarbon fields, many are incomplete and unable to provide reliable quantitative results. Additionally, some models are proprietary, making it difficult to access the information needed for detailed research. Thus, it is scientifically attractive to conduct a thorough investigation of AE signals in order to build a more reliable model.

One of the main challenges faced is the high dependence of AE signals on various flow parameters and surrounding noise. The characteristics of the signal are influenced by factors such as sand concentration, the size and shape of sand particles, the velocity of the sand particles impacting the pipe wall, fluid flow properties such as flow regime, velocity, and viscosity, and pipe geometry (Gao et al., 2016). Other parameters that can affect the AE signal include gas/oil ratio (GOR), temperature, and the surrounding media.

Igor Grishkov's work in 2022 has already laid the foundation for this analysis. However, the primary objective of this thesis is to see if there is any correlation between the different AE readings for different grain sizes.

2.1 AE correlation between different sand grain sizes

The primary objective of this research project is to investigate the correlation between the acoustic emission (AE) readings obtained from different sizes of sand used in the flow loop. The study will involve several experimental phases.

First, the experiment will be conducted using the same grain sizes as those used in Igor Grishkov's previous study. This will serve as a control test to replicate his readings and establish a baseline for comparison.

Next, the experiment will involve the use of sand with higher grain sizes than those used in the control test. This will enable the examination of how the AE readings vary with increased grain size.

Following that, sand with lower grain sizes will be used in the experiment to assess the impact of smaller particles on the AE readings.

The subsequent phase of the study aims to explore any potential correlation between the AE readings obtained from different grain sizes. Specifically, the focus will be on examining if there is a relationship between the AE signals and the varying sand particle sizes.

Finally, a mixture of fine and coarse sand will be introduced to the flow loop. The objective is to determine if the correlation established in the previous phase can be utilized to estimate the percentage of fine and coarse sand in the mixture based on the AE readings.

Through these experiments, this research aims to gain insights into the relationship between sand grain sizes and AE signals, potentially providing valuable information for future studies in this field.

2.2 Upgrading sand injection tube:

An opportunity for enhancing the system lies in substituting the current PVC tube used for sand injection with a glass tube. While PVC may hold up against chemical exposure in the short term, opting for a different material such as glass may mitigate any plausible damage that could arise over the long haul. Igor Grishkov has already taken the initiative to purchase and receive the glass tube, yet it still remains pending installation and utilization. The goal is to be able to install the glass tube and make it function properly.

At present, the hose is secured with rubber straps, which, though practical, may not be the most visually pleasing resolution. By devising a more aesthetically pleasing manner to connect the hose, it might be possible to refine the system's overall appearance.

2.3 Gel type modification

During the short time working with Grishkov, he brought up a small but noteworthy issue he had encountered while mixing the sand particles with the gel. He discovered that the gel currently used required extensive mixing and sometimes had to be mixed for a prolonged period, which can be tedious and inefficient for larger-scale experiments. Furthermore, the sand suspension in the gel was not stable, which resulted in Grishkov having to rush the process of filling the sand injection tube and start the piston as soon as possible to prevent the sand from settling.

Although some may think that this is a minor issue, I firmly believe that it is worth exploring and finding a solution for. Therefore, I intend to spend the first period of my testing time during the semester focused on finding a new and improved type of gel that can address these issues. While the gel will most likely still be an alcogel, I plan to experiment with different brands of disinfectors to see if any of them can provide a more stable and efficient solution. This should not be a major problem, as there was a significant supply of disinfectors produced during the pandemic, and with the decrease in demand these days, they should be readily available and affordable to purchase. Ultimately, my goal is to find a gel that can make the mixing and suspension process more efficient and effective for future experiments.

2.4 Additional parameters to the AE model

The intricate process of monitoring sand concentration in a flow using AE (acoustic emissions) requires a deep understanding and careful examination. Although Grishkov's work has allowed us to explore various flow parameters and create a working model, there is still a considerable amount of work that needs to be done to enhance its accuracy and reliability.

One of the key areas we can focus on to improve the model is its accuracy. One way to achieve this is by utilizing computational fluid dynamics (CFD) simulations, which can provide us with a better understanding and prediction of the flow behavior. Additionally, we can also work on enhancing the separation system responsible for separating the solid particles from the fluid. This can be accomplished through the utilization of advanced filters or other cutting-edge separation technologies. Furthermore, we can perform repeated laboratory measurements to calibrate the model more effectively and ensure that it accurately predicts the sand concentration in the flow.

To enhance the model's accuracy and applicability, we need to expand the analysis to include more flow parameters. These can include density, viscosity, GOR (gas-oil ratio), and multiphase flow type, which all have an impact on the flow's behavior and the model's precision. Accounting for these parameters can produce a program that is better suited for real-world hydrocarbon fields.

Moreover, we must consider how different properties of solid particles affect the model.

Aside from grain size, we should explore the impact of grain angularity and density on the AE signals. Our studies have shown that signals can vary when using lighter or heavier grains. By comprehending how these properties affect the model, we can enhance its accuracy and reliability, leading to more precise predictions of sand concentration in a flow.

3 Literature Review

3.1 Why sand monitoring in oil and gas fields

Sand, as a naturally occurring granular material, is an inherent part of oil and gas production. During the extraction process, sand can enter the production stream and accumulate in various parts of the production equipment. Sand accumulation can lead to equipment erosion, corrosion, and failure, resulting in significant downtime and repair costs. Additionally, sand can pose significant safety risks for workers by causing fires and explosions in equipment.

To mitigate these risks, sand monitoring systems are installed to detect and measure sand accumulation in oil and gas production equipment. Sand monitoring systems use various technologies such as acoustic, pressure, and vibration sensors to detect and measure sand accumulation. The data collected by these sensors is then analyzed and used to make informed decisions on equipment maintenance, production optimization, and worker safety.

One of the critical benefits of sand monitoring systems is that they provide real-time data on sand accumulation in production equipment. This data allows operators to monitor equipment performance, detect issues before they become significant problems, and take corrective action promptly. This proactive approach to equipment management helps to minimize downtime and maintenance costs, ultimately improving production efficiency.

Moreover, sand monitoring systems help to ensure worker safety by detecting and alerting operators to potential sand-related safety hazards. With accurate data, operators can take appropriate safety measures, such as equipment shutdown, to prevent worker injuries or fatalities resulting from sand accumulation in equipment.

Sand monitoring systems are also crucial in optimizing production performance. By providing real-time data on sand accumulation, operators can identify opportunities to improve production efficiency. For instance, sand monitoring data can help operators optimize the rate of sand removal, thereby reducing the frequency of equipment shutdowns and maintenance.

Sand concentration in oil and gas flow can be monitored through a range of methods, not just AE measurements. For instance, a table comparing available sand monitoring options for the Siri Complex in the North Sea illustrates multiple approaches (Lee, Kasper,

& Quinn, 2017).

Sand Detection Method	**Total cost**	**Real-time**	**Workload (offshore)**	**Workload (onshore)**	**Sum** **1 = highest** **5 = lowest**
Acoustic Sand Detectors	2	5	4	3	14
Choke internal inspection	1	1	1	3	6
Flowline thickness monitoring	3	1	2	3	9
Seperator thermographic imaging	3	3	2	3	11
Wellhead sampling	2	2	1	1	5

Table 1: Siri Complex' Sand monitoring methods (Lee, Kasper, & Quinn, 2017)

Upon close examination, AE sand monitoring stands out as the optimal solution for this offshore field. AE monitoring provides real-time measurements with minimal offshore workload, making it a highly desirable option. The same comparison parameters could be applied to other offshore fields, further highlighting the benefits of AE monitoring. Moreover, AE sensors are non-intrusive, which makes them more advantageous than intrusive sensors in terms of safety, reliability, and installation and maintenance costs (Gao, et al., 2016).

3.2 Sensor placement

Sand monitoring sensor placement is a crucial aspect of sand monitoring systems in oil and gas fields. These sensors are typically installed at various locations in the production line, such as at the wellhead, separator, or downstream flowlines, to detect the presence of sand and measure its concentration. To ensure the effectiveness of the sand monitoring system, it is advisable to install it at a strategic location within the pipe where sand is most likely to collide with the wall. This typically occurs just downstream of a bend in the system. The sensor must be positioned in close proximity to the impact area, specifically on the outer side of the bend as depicted in Figure 3.1., to ensure accurate readings.

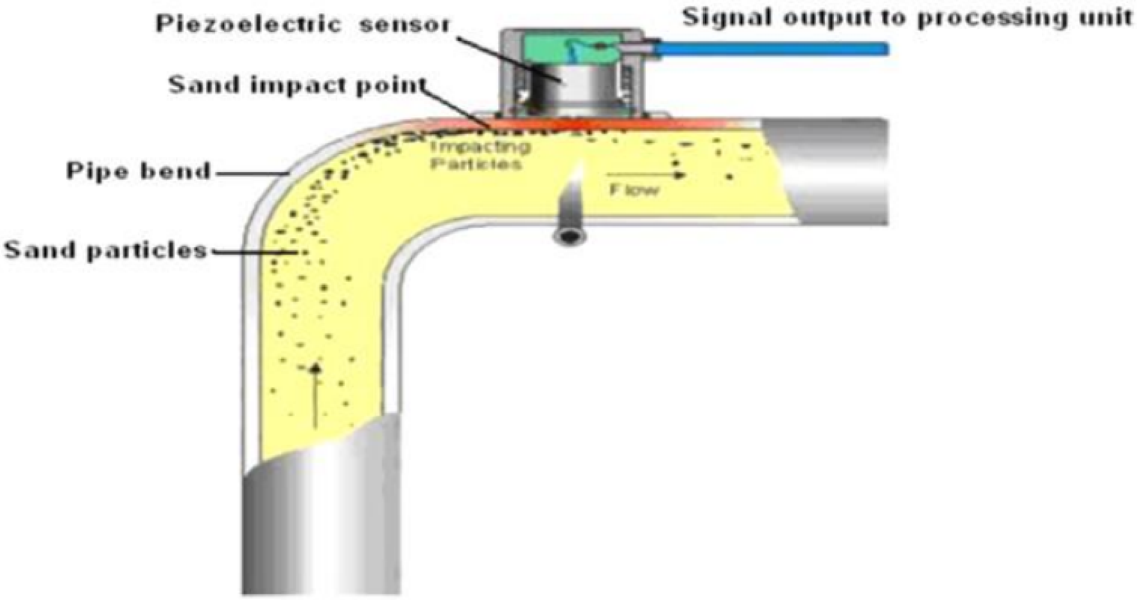


Figure 3.1: Passive non-intrusive ultrasonic sand detector (Emiliani et al., 2011)

The placement of sand monitoring sensors is critical as it determines the accuracy and reliability of the sand monitoring system. The sensors must be placed at locations where sand is likely to be present, such as in areas where there is a high flow rate or where the fluid changes direction or velocity. Additionally, the sensors should be placed in areas where they can be easily accessed for maintenance and cleaning.

The location of sand monitoring sensors also depends on the specific production process and the equipment used in the oil and gas field. For example, in subsea production systems, sand monitoring sensors may be placed at the wellhead or on the seabed to detect sand production and prevent damage to downstream equipment.

In onshore production systems, sand monitoring sensors may be placed at the separator to detect sand carryover from the production well. The sensors may also be placed in the flowlines to measure the sand concentration and detect any changes in sand production.

Proper placement of sand monitoring sensors is critical in maintaining the integrity of production equipment and preventing sand-related issues such as erosion, corrosion, and blockages. By detecting the presence of sand and measuring its concentration, sand monitoring systems can help operators take corrective action to prevent equipment damage and production downtime.

Alternative sensor placements have been proposed in other studies. In the case of popular wellhead sand monitoring systems, the ultrasonic signal is generated by particles impacting the inner side of the pipe wall, just after the bend where the sensor is located (Appalonov, Maslennikova, & Khasanov, 2021). A separate research study, which focused on gas flow, suggested placing the sensor slightly above the center of curvature. For laboratory installations, it's recommended to place the sensor approximately two-pipe diameter after pipe elbows, as per the guidelines proposed by Haugen et al. in 1995.

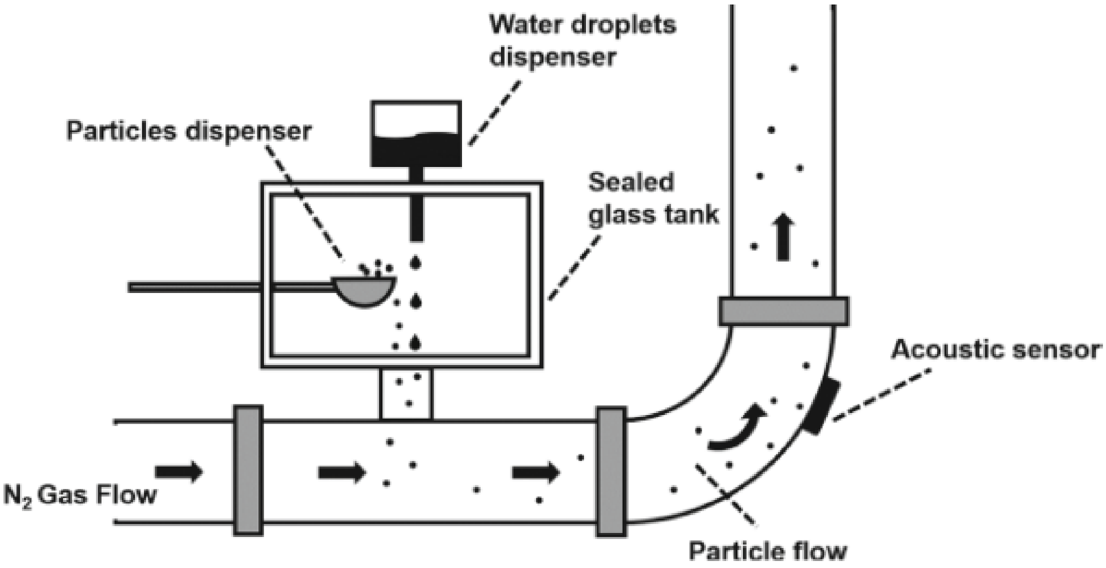


Figure 3.2: Acoustic sensor placement on the bend (Appalonov, Maslennikova, & Khasanov, 2021)

However, researchers should keep in mind that the sensor placement can be lowered for gas flow, as illustrated in Figure 3.2 of Appalonov, Maslennikova, & Khasanov's research. The variation in sensor placement recommendations underscores the importance of carefully assessing the specific needs and requirements of each individual installation when designing and implementing a sand monitoring system.

In summary, the placement of sand monitoring sensors in oil and gas fields is an essential aspect of sand monitoring systems. The sensors must be placed in areas where sand is likely to be present and where they can be easily accessed for maintenance and cleaning. Proper sensor placement can help operators prevent equipment damage and maintain the integrity of the production system.

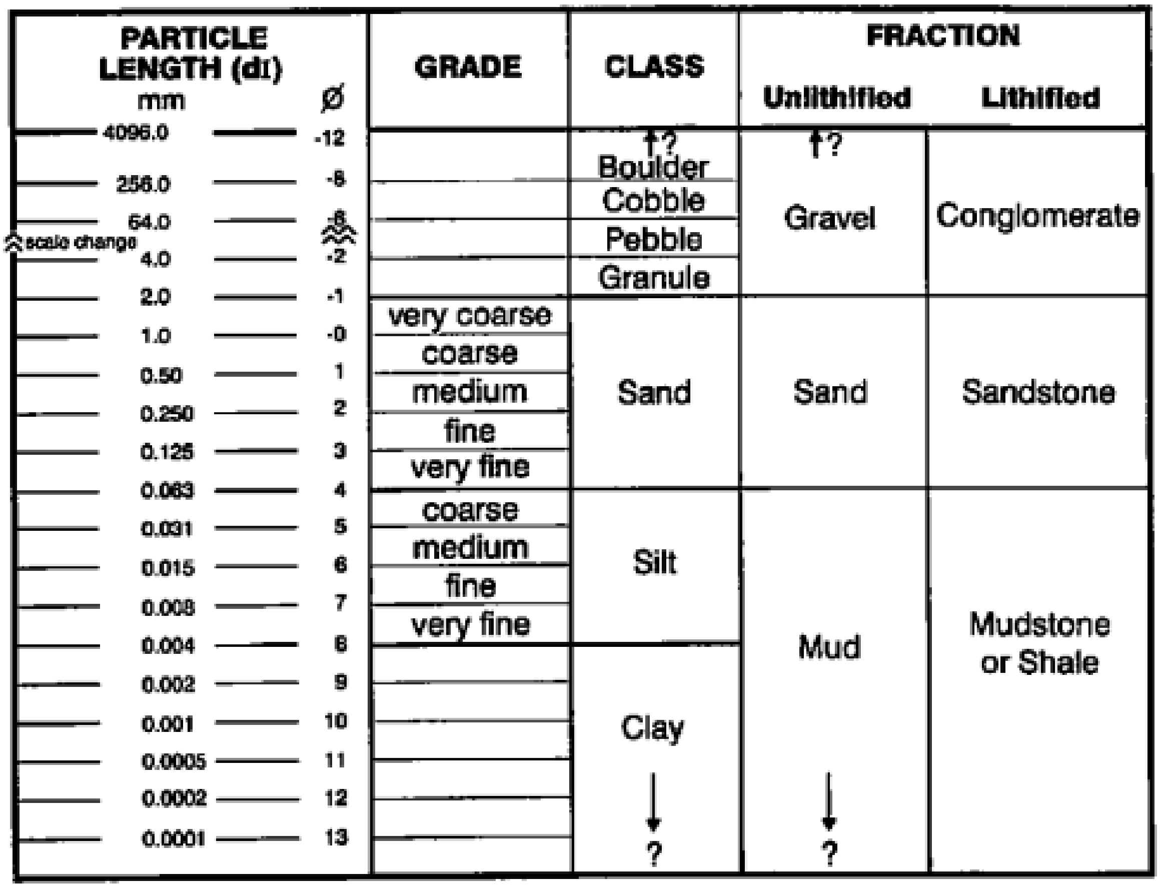
3.3 Sand grain size

The "Udden-Wentworth" scale for detrital particles is a widely accepted and practically standard scale used for communicating observations and deductions about sediment and sedimentary rocks. The scale recognizes three fractions: gravel (2 to 4096 mm), sand (1/16 to 2 mm), and mud (< 1/16 mm). The mud fraction is divided into silt and clay classes, while the gravel fraction is divided into granule, pebble, cobble, and boulder classes (Figure 3.3). To define the boundaries of the subclasses called 'grades', Udden used a logarithmic scale with 1 mm as the starting point. Coarser grade boundaries were established by progressive multiples of 2, while finer ones by progressive multiples of ½. Wentworth assigned the commonly used names for the sand grades, including very coarse sand (1 to 2 mm), coarse sand (1/2 to 1 mm), medium sand (1/4 to 1/2 mm), fine sand (1/8 to 1/4 mm), and very fine sand (1/16 to 1/8 mm). Additionally, Udden's silt grades are still in use, including coarse silt (1/16 to 1/32 mm), medium silt (1/32 to 1/64 mm), fine silt (1/64 to 1/128 mm), and very fine silt (1/128 to 1/256 mm). It is worth noting that the intermediate axial length of a grain is the one that determines classification due to the use of sieves for size analysis.

Terence Blair modified the Udden-Wentworth grain size scale by changing the boundaries between each size category and adding a few additional categories. The modified scale is commonly referred to as the "modified Udden-Wentworth scale" (Figure 3.4). The scale is based on a logarithmic progression, with each grade representing a doubling of the grain diameter. The scale ranges from -8 (clay) to 10 (boulders larger than 256 mm). The grades in between are silt (-7 to -5), sand (-4 to 3), gravel (-2 to 1), cobble (0), and small boulder (1 to 2).

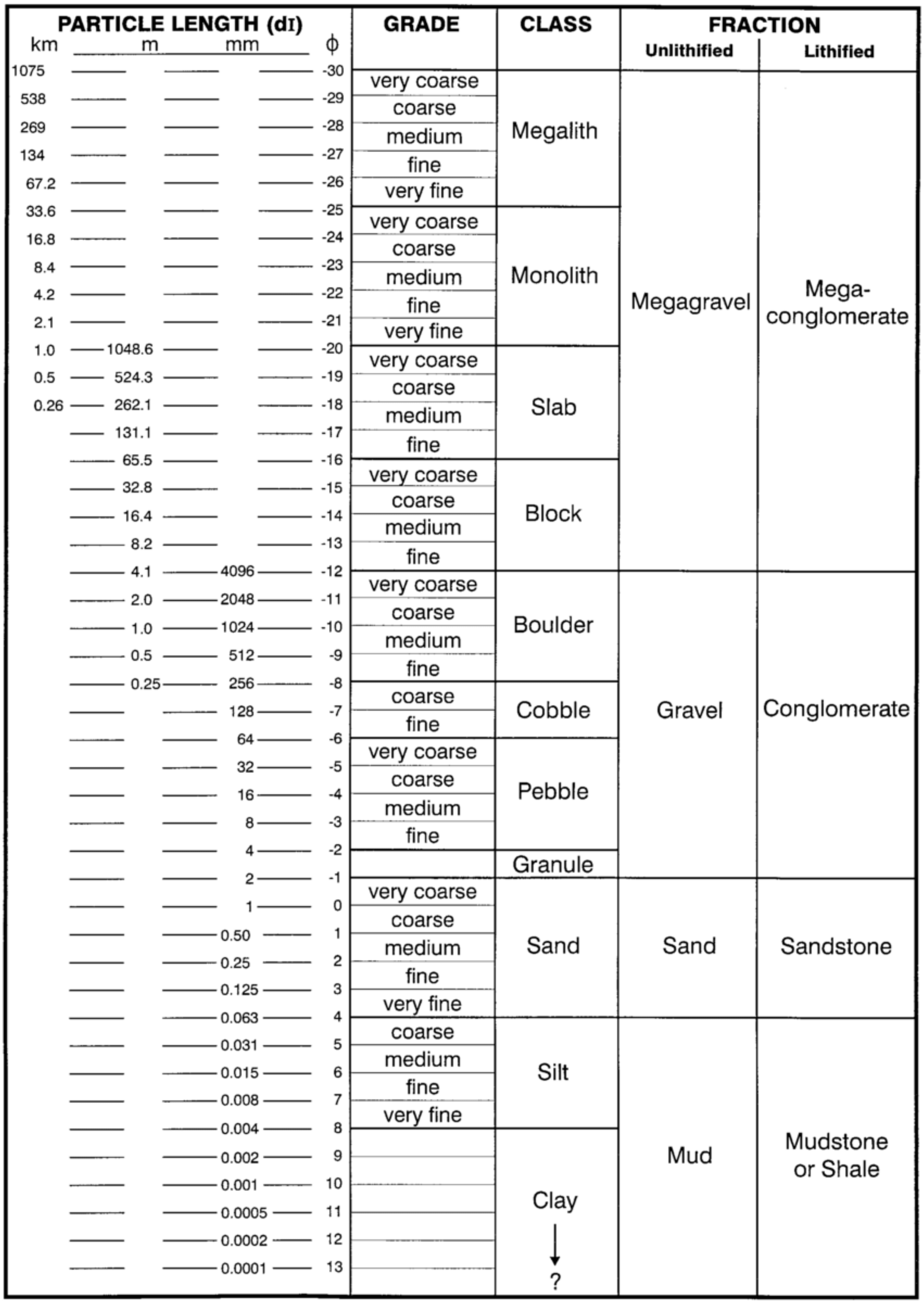
The modified Udden-Wentworth grain-size scale is commonly used in sedimentology, geology, and other earth sciences to describe the size distribution of sediments and rocks. It provides a standardized and objective way to communicate sediment characteristics and facilitates comparison between different sites and studies.

From the Udden-Wentsworth grain size scales, we see that the finer layers of sand have a particle length of from 0.006 mm (very fine sand) to 0.125 mm (fine sand). According to Mahmud, Leong and Lestarino (2019), "The average sand size varies from production well to another, even in the identical formation, however; typical sand particle sizes are within the range of 50–150 µm., which happens to be this same corresponding layer on the sedimentary grain-size scale.



PARTICLE LENGTH (dI) mm	Ø	GRADE	CLASS	FRACTION Unlithified	FRACTION Lithified
4096.0	-12		↑? Boulder	↑? Gravel	Conglomerate
256.0	-8		Cobble		
64.0	-6		Pebble		
scale change					
4.0	-2		Granule		
2.0	-1	very coarse	Sand	Sand	Sandstone
1.0	-0	coarse			
0.50	1	medium			
0.250	2	fine			
0.125	3	very fine			
0.063	4	coarse	Silt	Mud	Mudstone or Shale
0.031	5	medium			
0.015	6	fine			
0.008	7	very fine			
0.004	8		Clay		
0.002	9				
0.001	10				
0.0005	11				
0.0002	12				
0.0001	13		↓ ?	↓ ?	

Figure 3.3: Udden-Wentworth sedimentary grain-size scale (Blair & McPherson, 1949)



PARTICLE LENGTH (dI)				GRADE	CLASS	FRACTION	
km	m	mm	ϕ			Unlithified	Lithified
1075			-30	very coarse	Megalith	Megagravel	Mega-conglomerate
538			-29	coarse			
269			-28	medium			
134			-27	fine			
67.2			-26	very fine			
33.6			-25	very coarse	Monolith		
16.8			-24	coarse			
8.4			-23	medium			
4.2			-22	fine			
2.1			-21	very fine			
1.0	1048.6		-20	very coarse	Slab		
0.5	524.3		-19	coarse			
0.26	262.1		-18	medium			
	131.1		-17	fine			
	65.5		-16	very coarse	Block		
	32.8		-15	coarse			
	16.4		-14	medium			
	8.2		-13	fine			
	4.1	4096	-12	very coarse	Boulder	Gravel	Conglomerate
	2.0	2048	-11	coarse			
	1.0	1024	-10	medium			
	0.5	512	-9	fine			
	0.25	256	-8	coarse	Cobble		
		128	-7	fine			
		64	-6	very coarse	Pebble		
		32	-5	coarse			
		16	-4	medium			
		8	-3	fine			
		4	-2		Granule		
		2	-1	very coarse	Sand	Sand	Sandstone
		1	0	coarse			
		0.50	1	medium			
		0.25	2	fine			
		0.125	3	very fine			
		0.063	4	coarse	Silt	Mud	Mudstone or Shale
		0.031	5	medium			
		0.015	6	fine			
		0.008	7	very fine			
		0.004	8		Clay ↓ ?		
		0.002	9				
		0.001	10				
		0.0005	11				
		0.0002	12				
		0.0001	13				

Figure 3.4: Modified Udden-Wentworth sedimentary grain-size scale (Blair & McPherson, 1949)

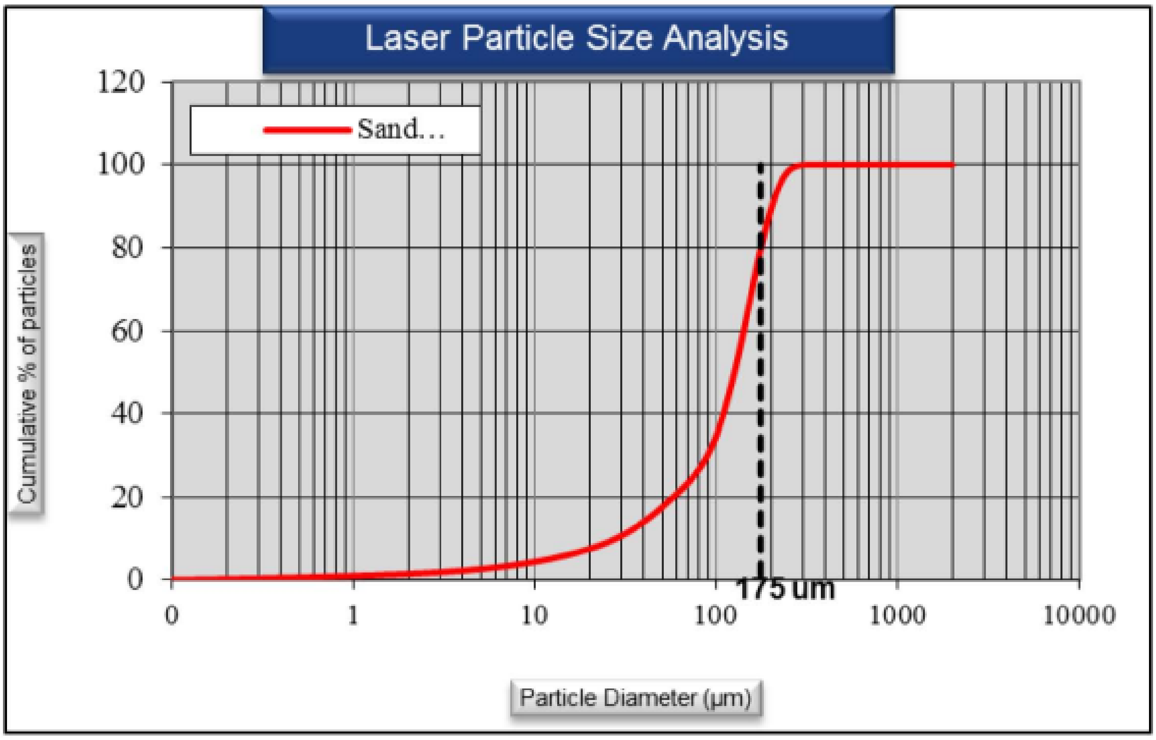


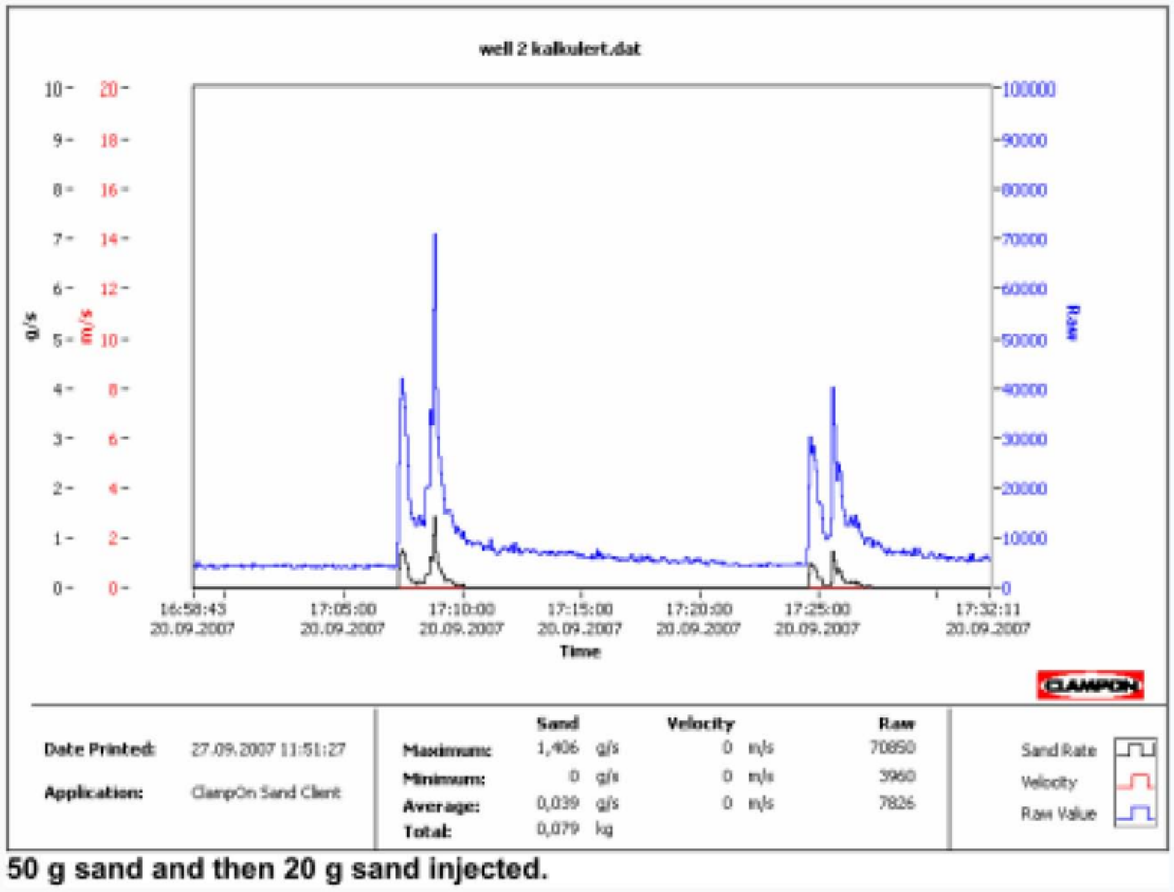
Figure 3.5: Laser Particle Sieve Analysis (LPSA) on separator samples (Lee, Kasper, & Quinn, 2017)

At the Siri Complex in 2015, a Laser Particle Sieve Analysis (LPSA) was carried out, and the results showed that about 80% of the particles were less than 175 microns. (Lee, Kasper, & Quinn, 2017). The LPSA is shown in Figure 3.5.

Based on the data from the LPSA, it is evident that while a significant number of particles fall within the 100-175 size range, we can also anticipate the presence of particles with even smaller diameters. Therefore, there is no defined lower limit to the size of grains that we may encounter.

It's hard to determine the minimum grain size we should expect, so instead, we need to focus on the sensitivity of the sensor we're using. To get an idea of what sensitivity we should aim for, we can look at a sensor already in use by ClampOn, which has been implemented in various fields, including the Brage field operated by OKEA.

Determining the sensitivity limit that is appropriate for specific field conditions can be a challenging task, especially in cases where the field is yet to be developed, and all available data is merely based on estimations and simulations. In a producing field, the sensor can be calibrated by intentionally injecting a certain amount of sand into a flowing well and checking what can be detected. However, this is not an option for undeveloped fields. Despite this limitation, insights can still be gained from data obtained from a test that involved injecting 50 and 20 grams of sand for 8 minutes each, resulting in an average sand rate of 0.04 g/s at a velocity of 12 m/s. This test can be seen in Figure 3.6



Date Printed:	27.09.2007 11:51:27
Application:	ClampOn Sand Client

	Sand		Velocity		Raw
Maximum:	1,406	g/s	0	m/s	70850
Minimum:	0	g/s	0	m/s	3960
Average:	0,039	g/s	0	m/s	7826
Total:	0,079	kg			

Sand Rate
Velocity
Raw Value

50 g sand and then 20 g sand injected.

Figure 3.6: Injection test (Haugsdal, 2017)

To some extent, laboratory tests have validated these findings (Figure 3.7). For instance, at a particle size of 35 microns, the sensor was able to detect 0.01 g/second at a velocity of 20 meters per second. However, it is unclear whether the highest sensitivity is achieved with the smallest particle size, as the sensitivity test program was only conducted for particles of 35 microns.

However, it's important to note that these tests were conducted using dry gas, and we know that when dealing with wet gas, the noise level can be higher, which leads to a lower sensitivity. To account for this, the sensitivity was also tested in wet gas, and the results, as shown in Figure 3.8, revealed a significant decrease in sensitivity due to stronger background noise. Therefore, we should keep in mind that not all sand particles

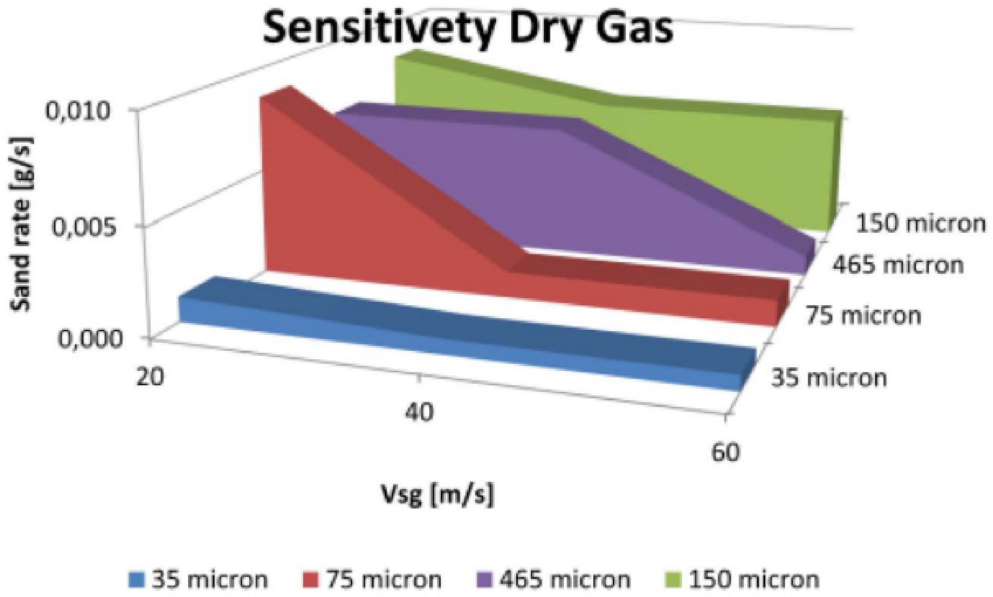


Figure 3.7: Sensitivity, dry gas (Haugsdal, 2017)

can be detected by the sensor.

In terms of particle size range, our research has shown that there is an upper limit of approximately 800 microns, but no lower limit. However, it's reasonable to assume that we should expect solid particles around 200 microns in size from a sandstone reservoir.

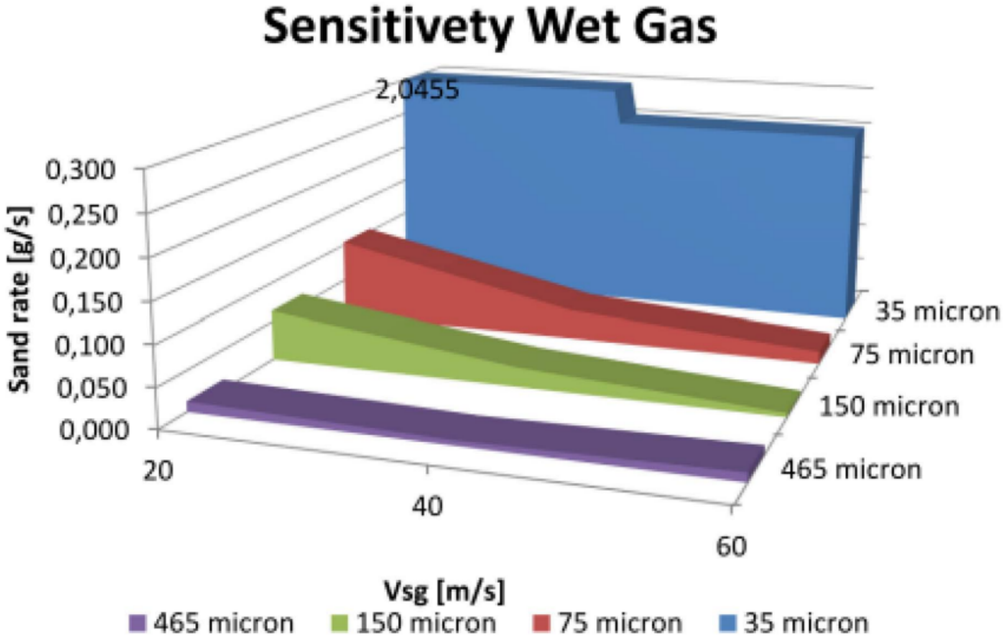


Figure 3.8: Sensitivity, wet gas (Haugsdal, 2017)

3.4 Processing Acoustic Emission signals

In theory, sand particles in a flow stream should generate an acoustic signal that is distinguishable from the background noise level. The characteristics of the signal depend on several factors, including the concentration of sand in the flow, the size and angularity of the sand particles, the velocity at which they hit the pipe wall, the properties of the fluid flow (such as regime, velocity, and viscosity), and the geometry of the pipe. Equation 3.1 is a formula that already exists for calculating the sand rate based on data obtained from non-invasive acoustic sensors.(Ibrahim and Haugsdal, 2008)

$$SandRate = (\frac{Signal - Zero}{Step})^{trend} \tag{3.1}$$

where:

$Signal$ = raw output from sand sensor

$Zero$ = background noise

$Step$ and $trend$ = factors that have been determined experimentally, and they are particularly sensitive to flow velocity and the size of sand particles. These values are obtained from the calibration process. (Ibrahim and Haugsdal, 2008).

A variation of this equation is used at the Siri Complex monitoring. (Lee, Kasper & Quinn, 2017). Here, it is defined by:

$$Sand\ Rate\ (g/s) = \frac{Raw\ value - Zero\ value}{Step\ value} \tag{3.2}$$

The sand rate is calculated by the acoustic sand detector software and is reported in g/s. The raw value is the measured energy from the detector, and the zero value is the background noise. The step value, which is energy/g, is obtained from standard step tables provided by the vendor and depends on the fluid velocity, whether the fluid is gas or oil, and the Gas Oil Ratio (GOR). The velocity of the fluids is calculated from performance curves, and the step value is assigned based on the standard step table. Personalizing the step table through sand injection testing is possible but was not done in the Siri Complex due to a lack of suitable injection points. (Lee, Kasper & Quinn, 2017).

There have been some concerns about the accuracy of the sand rate calculation using Equation 3.2 because it needs to be calibrated to the flow conditions and re-calibrated if those conditions change. Additionally, the processed signal from the piezoelectric crystal isn't the direct output voltage, but instead is the Root Mean Square (RMS) value. To address these concerns, a new equation for calculating the sand mass rate was proposed. (Gao, et al., 2016).

$$(S_{pRMS}^2 - S_{bRMS}^2)^{\frac{1}{2}} = C\sqrt{m_t}\frac{1}{2}mv^2 \tag{3.3}$$

where:

S_{pRMS} = Root Mean Square (RMS) value of the raw input

S_{bRMS} = RMS value of the background noise

m_t = sanding mass rate

C = calibration constant

m = representative particle mass found by calculating a representative particle diameter

v = representative particle impact velocity from a multiphase erosion model.

In order to use Equation 3.3 for calculating sand mass rate, it is necessary to know both the particle mass and velocity. However, accurately measuring the size of sand particles and tracking their impact velocity can be challenging. To make the calculation simpler, it is assumed that all particles have the same mass and velocity, but this assumption is not entirely realistic and may lead to questionable results.

Gao et. al, 2016, therefore proposed another way to calculate the sand rate, proposed as:

$$m_t = \frac{S_p - Sb}{K(\frac{Q}{A})^2} \tag{3.4}$$

Where:

m_t = mass rate of sand.

S_p = amplitude of the raw signal produced by the sand particles.

S_b = amplitude of the background noise.

K = proportionality constant, which equals $\frac{ck}{2}$.

Q = volumetric flow rate of the fluid.

A = cross-sectional area of the pipe.

We simplified Eq.3.3 to Eq.3.4 to reduce the number of inputs. However, Eq.3.4 can also be considered a variation of the first equation, Eq.3.1, with the *trend* variable set to 1. As a result, for this thesis, we will use Eq. 3.1 as the main equation.

4 Choice of method and equipment

4.1 Flow Loop

The experiments will mainly be done at the Flow Loop (Figure 4.1) in the basement of Petroleumsteknisk senter (PTS), NTNU.

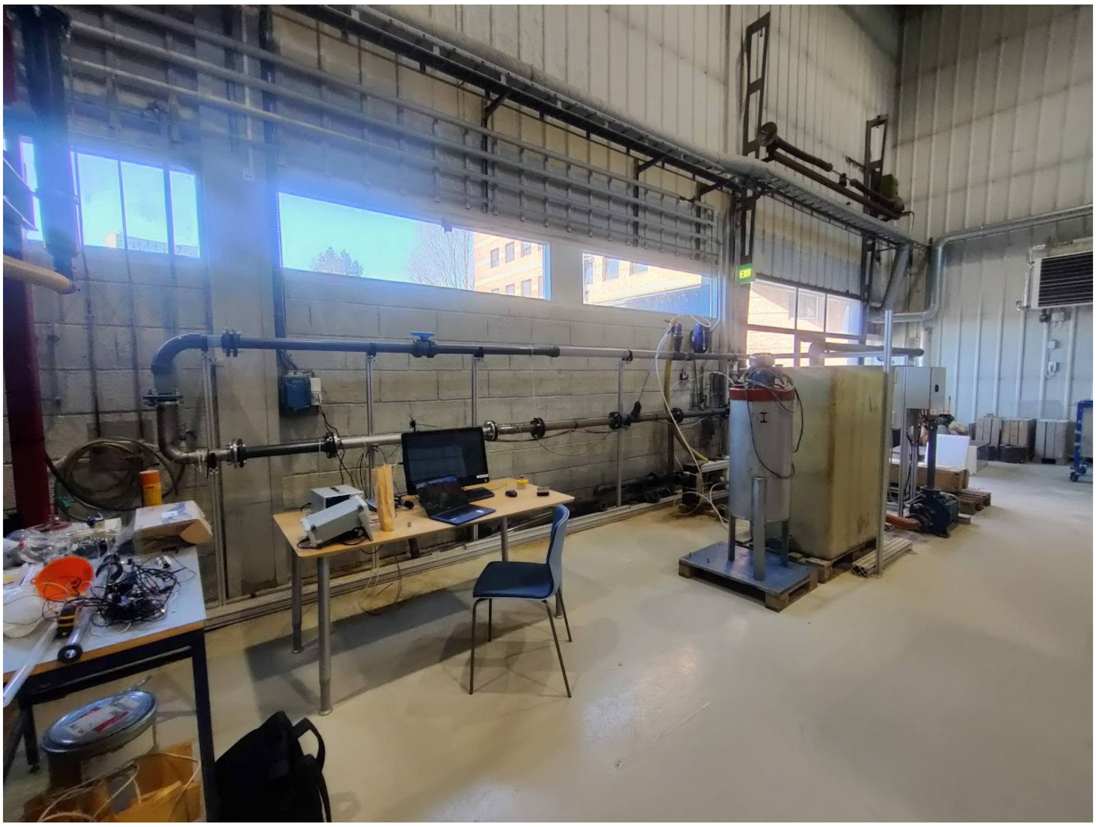


Figure 4.1: Flow loop at PTS

As this is a pretty large and complex-looking system, we'll be highlighting the most important parts of the setup.

In our experiments, water was used as the only medium for carrying sand. We were able to control the velocity of water flow by using a pump through a control station. To ensure accurate flow rates, we calibrated the pump output frequencies to their corresponding flow rates before conducting experiments (Grishkov, 2022). This calibration process was carried out using a multiphase meter by KROHNE, which is installed close to the pump discharge. (Figure 4.2).

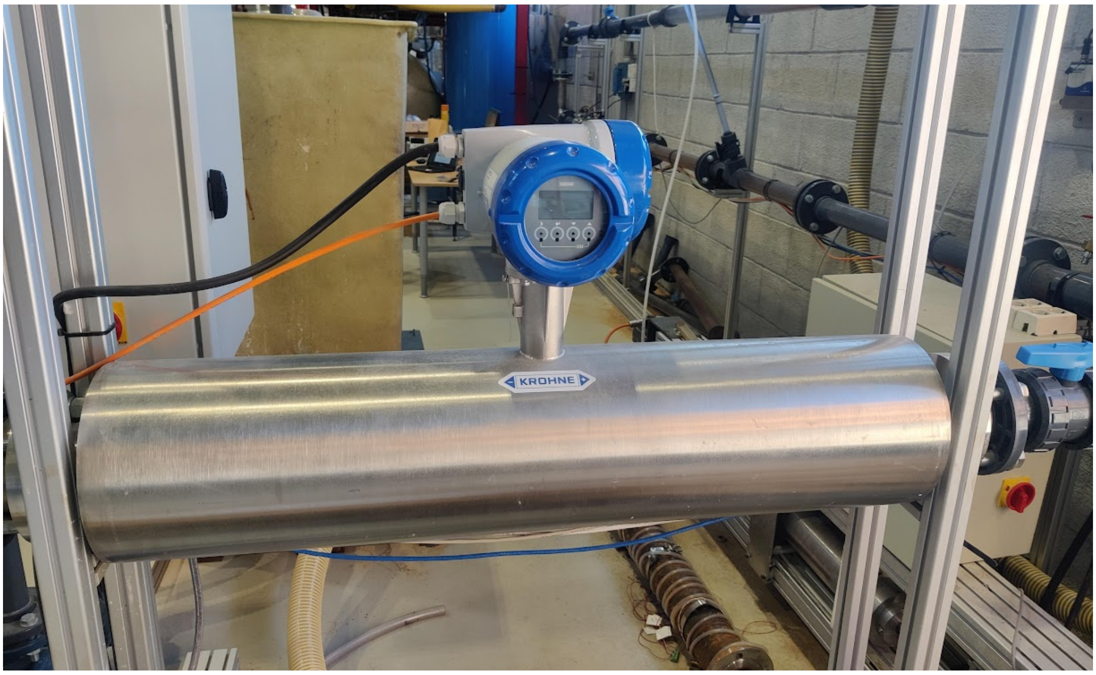


Figure 4.2: KROHNE multiphase meter

Grishkov conducted measurements at different pump frequencies and calculated the average flow rate for each. The results are presented in Table 2 showing the flow rate versus pump frequency. In order to simplify the data, Grishkov rounded the average flow rates to a simplified value. Since the same pump will be used in my experiments, I have decided to use the same simplified values as Grishkov.

Pump frequency, Hz	**Average recorded flow rate, l/min**	**Simplified flow rate, l/min**
6	100.00	100
7	131.75	130
8	163.00	165
9	193.25	195
10	224.50	225
11	250.75	250
12	276.50	275
13	300.75	300
14	324.75	325
15	348.50	350
16	372.50	375
17	396.67	400
18	420.25	420

Table 2: Liquid flow rates corresponding to pump frequencies. (Grishkov, 2022)

4.2 Sand separation system

The sand separation system plays a vital role in the flow loop system, enabling the effective separation of sand particles from the water stream. This system consists of a carefully designed setup where water and sand from the flow loop are directed into a container configuration for the purpose of separation.

The container, with a height of approximately 1.5 meters, houses a smaller bucket within its structure. Upon exiting the pipe, the water and sand mixture is channeled downward into the smaller bucket. Due to the disparity in density, the sand particles settle at the bottom of the bucket, while the water begins to fill the available space.

As the water level within the bucket rises, a critical threshold is reached when it overflows into the surrounding container. This overflow marks the successful separation of water from the sand, as the sand particles accumulate at the bottom of the bucket, while the excess water flows out and is collected in the container.

The sand separation system offers an efficient mechanism for the segregation of sand particles from the water stream. By leveraging the difference in density between sand and water, the system effectively separates and collects the sand, ensuring that it remains confined to the bottom of the bucket. Meanwhile, the water, free of sand particles, is collected in the bottom for further use within the flow loop system, or thrown away if it gets too muddy/brown.

This sand separation method, although simple in design, plays a crucial role in maintaining the integrity and efficiency of the flow loop system. It ensures that sand particles are effectively removed from the water stream, minimizing potential issues such as clogging and damage to downstream equipment. The successful separation of sand and water is an essential step in ensuring the reliable operation and longevity of the overall system.

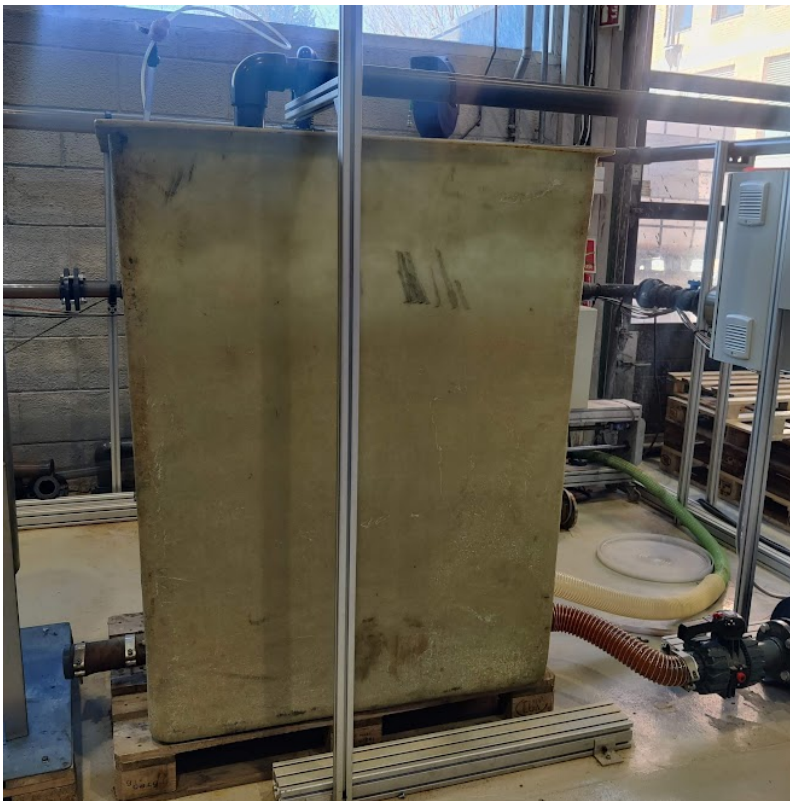


Figure 4.3: Sand separation system

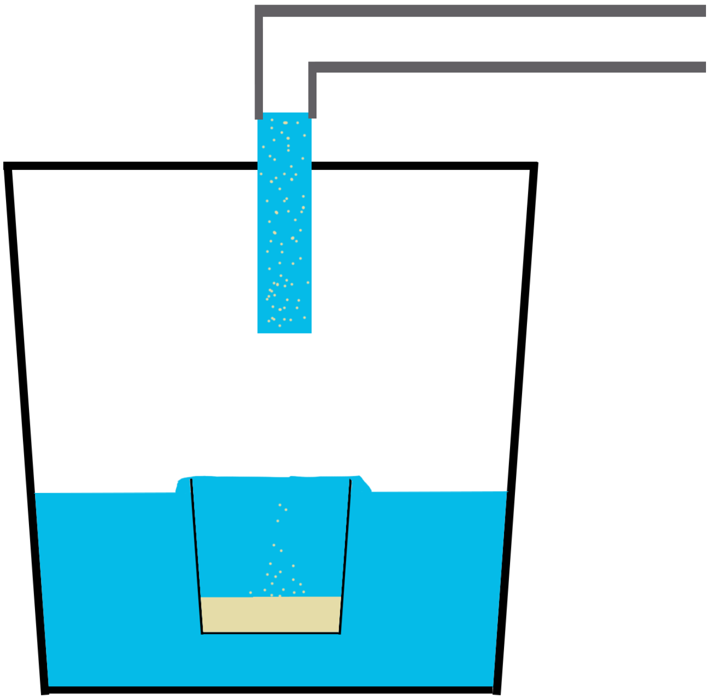


Figure 4.4: Sand separation system inner mechanics, drawn figure

4.3 Sand injection system

The sand injection system shown in Figure 4.5 is a crucial component of the Flow Loop system. It allows us to inject a gel containing suspended sand particles into the Flow Loop for circulation, enabling us to study the resulting signals.

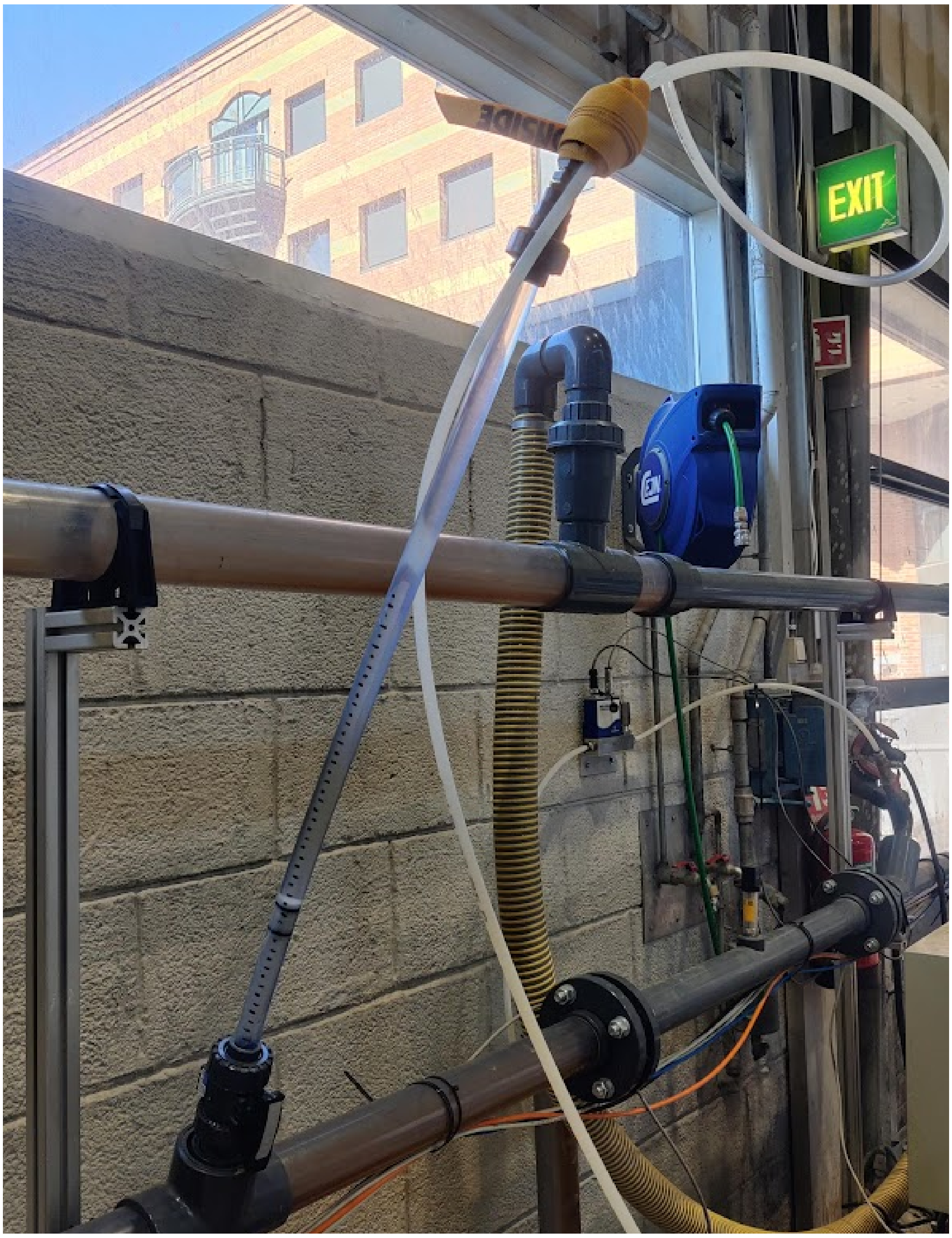
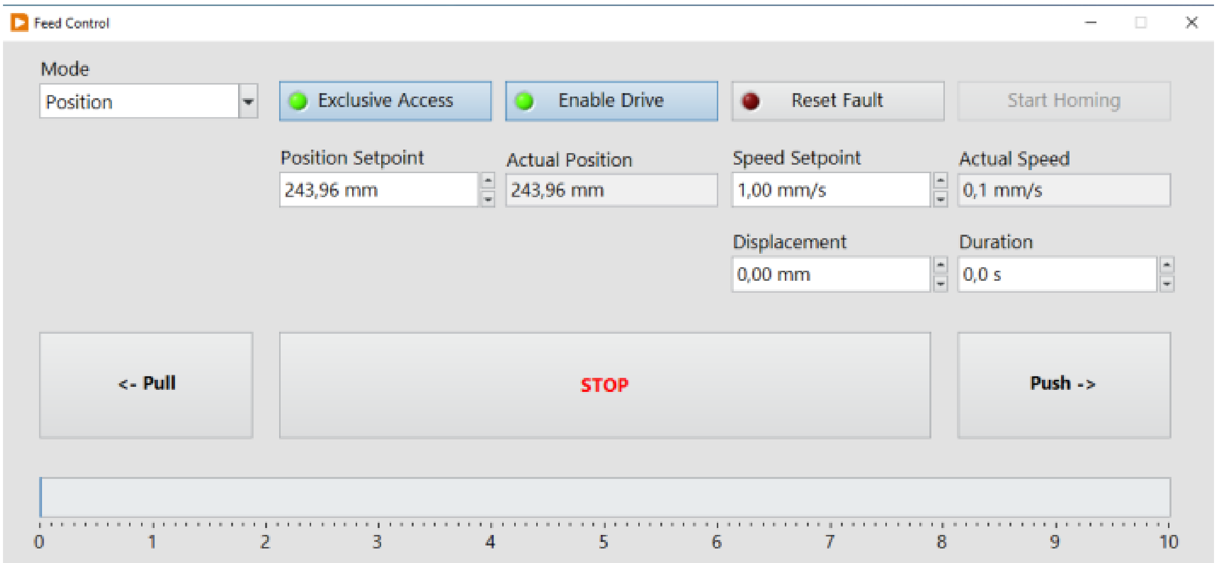


Figure 4.5: Sand injection system

The NTNU Flow Loop system currently uses a PVC pipe with a piston to inject the gel with suspended sand grains into the system. The piston is powered by water from a hose connected on top and is controlled by a program running on the same PC used to record the acoustic signals. Figure 4.6 displays the user interface for controlling the piston, where different parameters can be chosen, with the duration being the most critical parameter. The injection system was designed and developed by Igor Grishkov in 2021, and used in his research.



Mode: Position

Position Setpoint: 243,96 mm

Actual Position: 243,96 mm

Speed Setpoint: 1,00 mm/s

Actual Speed: 0,1 mm/s

Displacement: 0,00 mm

Duration: 0,0 s

Figure 4.6: Sand injection piston control interface (Pham, 2022)

The sand injection system will contain sand grains suspended in alcogel (Pham, 2022). The reason for using an alcogel is because of the main characteristics - it contains alcohol instead of water as the main ingredient, which allows for near-instantaneous dissolution in high-turbulence water flows. The gel is a non-Newtonian liquid with extremely high viscosity at low shear rates, which enables us to suspend sand particles in the gel for extended periods. Although there are some limitations on the mixing velocity and the volume fraction of sand, they are not critical for our objectives. Additionally, sand can be added to the system more quickly through a funnel directly to the water intake, which is a simpler approach than using the sand injection system. This method can also be utilized when investigating parameters other than sand concentration.

4.4 Acoustic Emission measurement equipment

To accurately capture the signals generated by the sand particles in the Flow Loop, it is essential to use a sensor capable of detecting the relevant frequency range. The frequency of the waves produced by the sand particles hitting the metal pipe wall is within the range of 100 to 500 kHz, as reported in a study by Brown et al. (2000). Although there are several types of sensors available, for the Flow Loop, we have chosen the PK15I sensor from Mistras.

One of the reasons for selecting this sensor is its integral preamplifier, which can amplify the signal and enhance the accuracy of the measurements. Furthermore, the PK15I sensor operates within the frequency range of 100-450 kHz, with a resonant frequency of 150 kHz, making it an ideal candidate for detecting the waves generated by the sand in the Flow Loop.

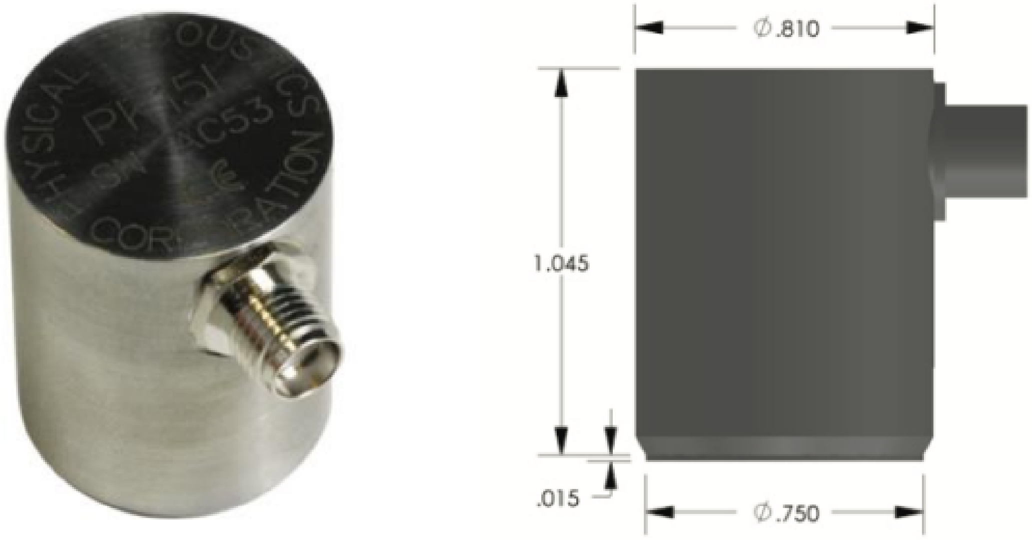


Figure 4.7: PK15I sensor from Mistars (Mistars website, 2022)

To convert the analog signals from the sensor into digital signals, we require an oscilloscope that can be connected to the sensor. This oscilloscope needs to meet certain criteria, and we must examine a few parameters such as the sample rate and resolution.

The signal frequencies that are anticipated are usually less than 1 MHz. According to the Nyquist theorem, a sample rate of at least twice the frequency is necessary, which means that a 2 MHz sample rate will suffice.

We've decided to use the PicoScope 4262. This oscilloscope has a 5 MHz sampling frequency, so more than enough for our research. The PicoScope 4262 also has a 16 bit resolution, also more than sufficient enough.

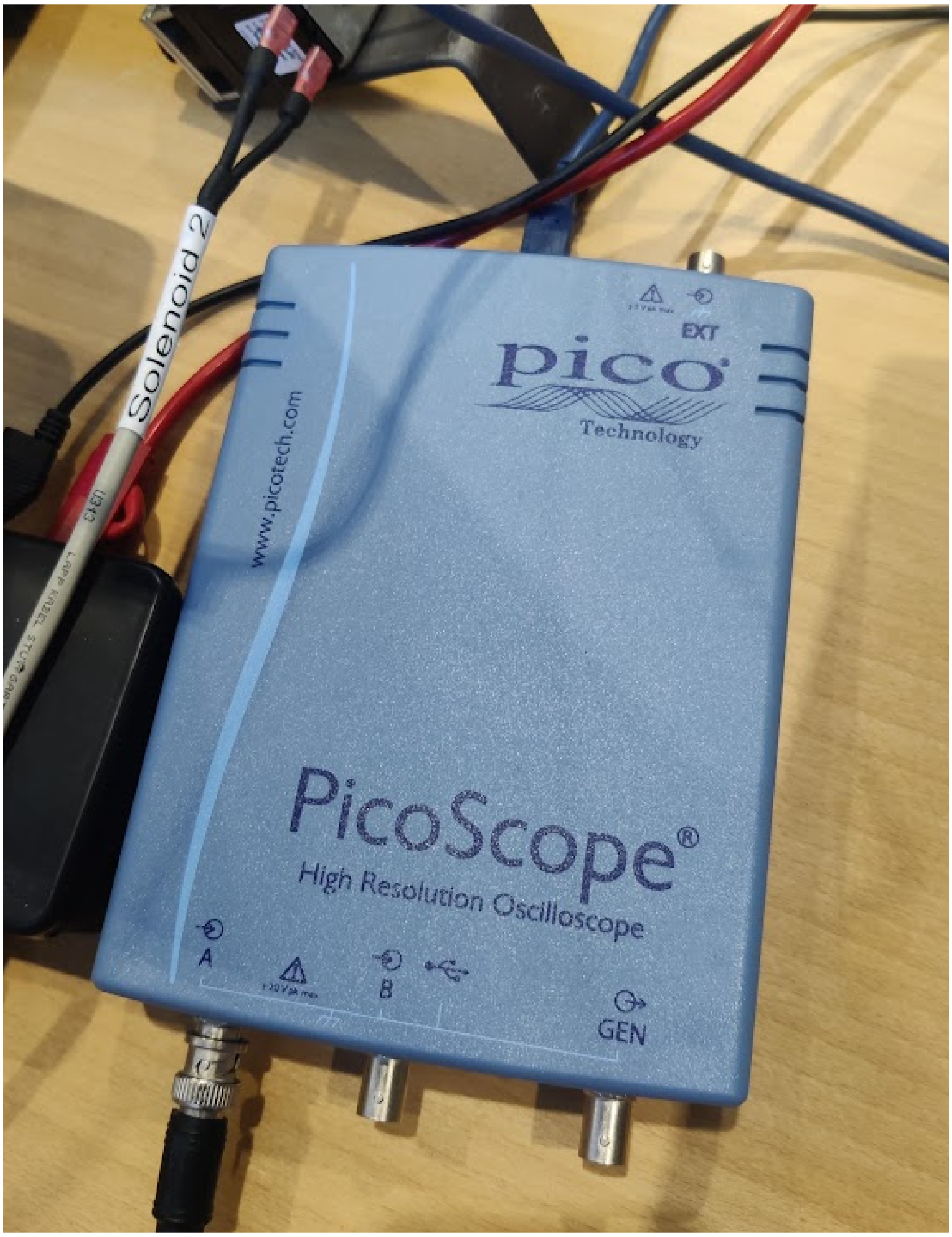


Figure 4.8: PicoScope 4262 Oscilloscope

Grishkov utilized Pico Technology's open-source GitHub page which contains extensive Python and MATLAB code templates for their oscilloscopes. Using this foundation, Grishkov developed a program that captures the signal as files consisting of 1 second of measurements with a 5 MHz sampling frequency.

In order to establish a connection between the sensor and the oscilloscope, a power/signal connection is required to provide power to the sensor and transfer the signal to the oscilloscope. Grishkov created this connection himself, using the scheme provided by Mistars (Figure 4.9).

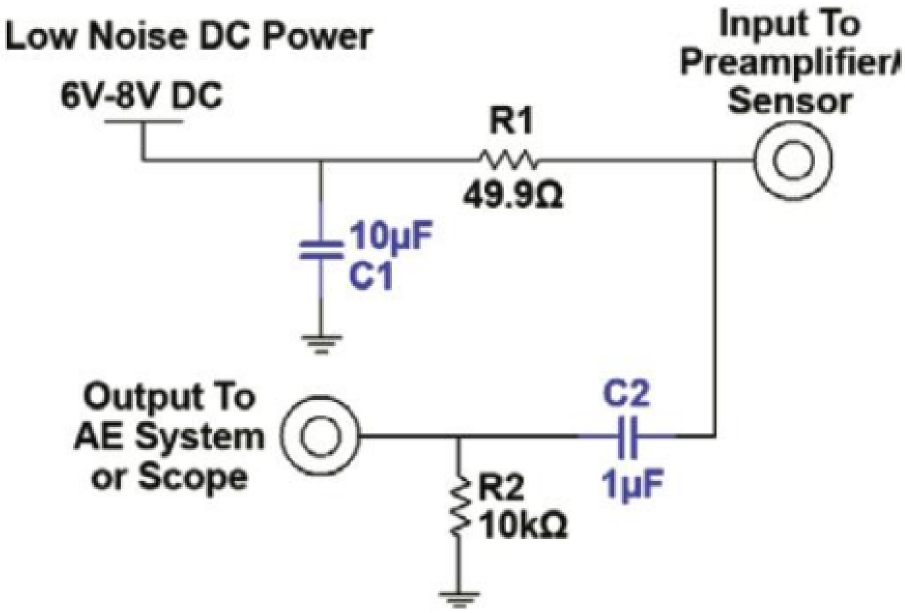


Figure 4.9: Sensor and Oscilloscope power/signal scheme

5 Sand distribution modelling

Igor Grishkov's sand distribution modelling will be briefly discussed in this chapter. Since I will be using the same model, the theory behind it remains unchanged, and therefore this chapter serves as a summary of the research conducted by Grishkov.

5.1 Flow Regime

Igor Grishkov's research starts by investigating the main flow regime, which requires calculating the Reynolds number for all possible flow velocities using equation 5.2. The parameters needed for Reynolds number calculations include the flow velocity, internal pipe diameter, fluid density, and fluid viscosity. For the purposes of the research, the investigation is centered on the test section where the internal pipe diameter is 72.1 mm, water density is assumed as 1000 kg/m^3, and water viscosity is 1 mPa·s.

$$Re = \frac{v \cdot d \cdot \rho}{\mu} \tag{5.1}$$

In order to determine flow rate, fluid velocity needs to be known. The experiments are conducted using a Flow Loop that allows for pump speeds ranging from 6 to 18 Hz. However, it has been decided to build the model for pump speeds between 9 to 18 Hz to avoid unnecessary complexity of sand injection at lower rates since the pump cannot take sand grains through direct injection below 9 Hz. With all parameters in place, the critical Reynolds number determining the transition between laminar and turbulent flow can be estimated as 2000, but for fully turbulent flow, it must be significantly larger by a factor of 10-50. Therefore, fully turbulent flow is in the range of 20,000 to 100,000.

From the Table 3, it is evident that the minimum flow rate during the research has a Reynolds number of 57,393, which is already within the turbulent flow regime. As a result, all further calculations will be performed for the turbulent flow regime. By using this approach, the research ensures that the model is kept as general as possible while also obtaining the necessary scale of Reynolds number to determine the flow regime.

Grishkov states that in order to determine the suspension of sand gains in a flowing system, it is necessary to construct a flow velocity profile. He notes that since the sensor is located after the bend, it is essential to use Computational Fluid Dynamics (CFD) to

Pump frequency, Hz	**Flow rate, l/min**	**Flow rate, m3/s**	**Flow velocity, m/s**	**Re**
9	195	0.0033	0.796	57393
10	225	0.0038	0.918	66223
11	250	0.0042	1.021	73581
12	275	0.0046	1.123	80939
13	300	0.0050	1.225	88297
14	325	0.0054	1.327	95655
15	350	0.0058	1.429	103013
16	375	0.0063	1.531	110371
17	400	0.0067	1.633	117729
18	420	0.0070	1.715	123615

Table 3: Calculation of Reynolds number for Flow Loop flow regime (Grishkov, 2022)

accurately predict flow lines. However, such calculations are resource-intensive and not ideal for continuous measurements in real-time. Therefore, Grishkov suggests considering a straight pipe segment to determine the type of flow rate, and then comparing the results with real recordings to estimate the relevance of the modelling. To calculate the velocity profile in the straight pipe segment, Grishkov used the Power Law Profile in Pipes (Sellens, 2022).

$$\frac{U(r)}{U_{MAX}} = (1-\frac{r}{R})^{\frac{1}{n}} = (1-\frac{r}{R})^{\frac{1}{7}} \tag{5.2}$$

$$\frac{1}{n} = \sqrt{f}, f = 0.02$$

Where "U(R)" is a flow velocity, "r" distance from pipe center. "U_{max}" is maximal flow velocity at cross-section middle. "R" denotes the pipe radius and "f" is the friction factor, which can be accurately calculated if the surface roughness of the pipe wall is known. It is worth noting that in fully turbulent flow, the friction factor is close to 0.02, which justifies the assumption made in this equation.

Grishkov also calculated and plotted the velocity profiles in MATLAB, in order to calculate the sand grain suspension in the flow. This is show in Figure 5.1.

Based on Grishkov's observations from Figure 5.1, we can see that the flow velocity experiences a sharp change near the pipe wall as expected in a turbulent regime. Also, there is an unnatural change in the center of the pipe, which can be attributed to the Power Law equation. While applying another method for calculating velocity profiles

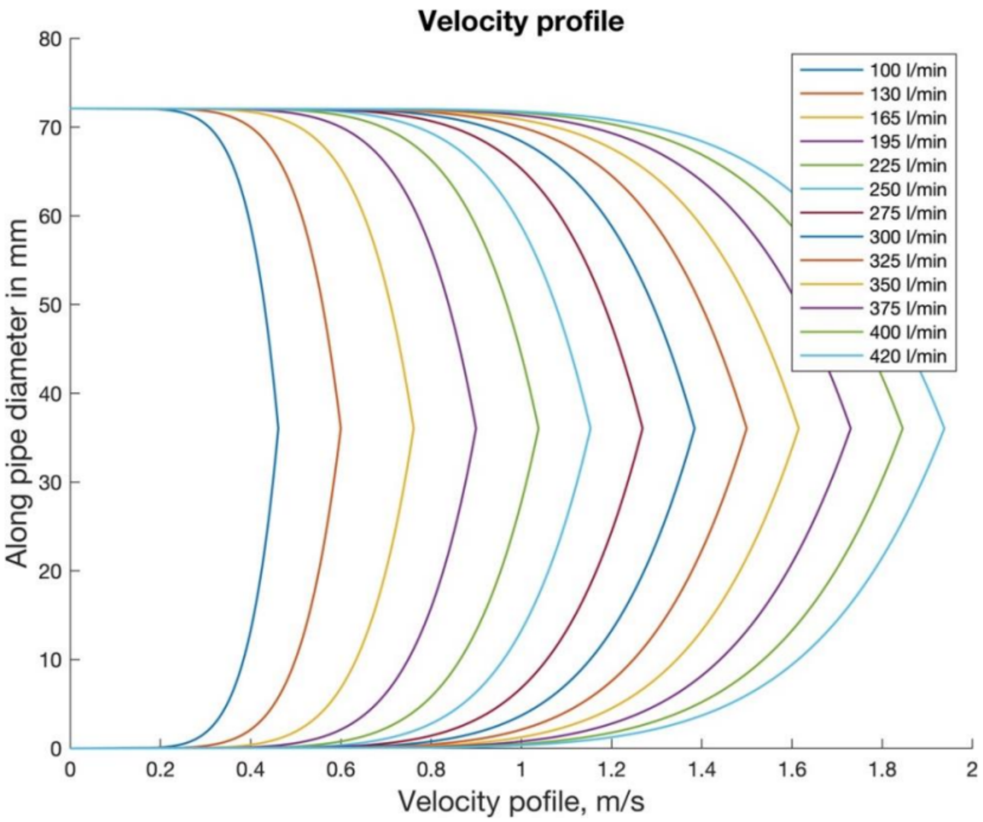


Figure 5.1: Flow Loop velocity profiles (Grishkov, 2022)

can potentially address this issue, the simplicity of the Power Law equation is crucial in ensuring that the final model can perform in real-time. As such, this weakness of the Power Law equation is accepted to maintain the model's computational efficiency.

5.2 Settling velocity of sands

In order to determine the sand suspension profile, another crucial parameter Grishkov considered is the grain settling velocity. When a particle sinks at a constant velocity, the forces acting on the particle must be balanced. Specifically, the gravitational force acting on the particle must be equal to the frictional force, resulting in the flow balance equation outlined in Figure 5.2.

Assuming the particle to be spherical, the volume and cross-sectional area can be related to the particle diameter as $V_s = \frac{\pi d^3}{6}$ and $A_s = \frac{\pi d^2}{4}$, respectively. By equating the gravity force with the friction force, we can obtain an expression for the settling velocity as follows (Asheim, 2021):

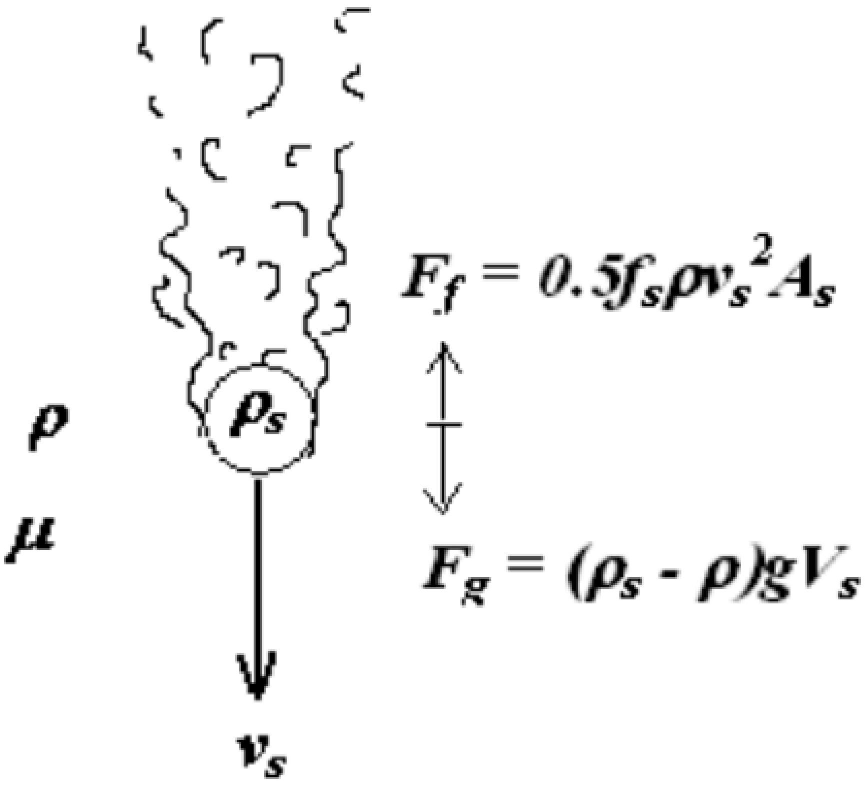


Figure 5.2: Solid particle falling in continuous gas/liquid (Asheim, 2021)

$$v_s = \left(\frac{4}{3}\frac{gd}{f_s}\right)^{0.5}\left(\frac{\rho_s - \rho}{\rho}\right)^{0.5} \tag{5.3}$$

ρ = density of the fluid
ρ_s = density of particle
d = diameter of the particle
f_s = friction factor between the particle and the surrounding fluid

Friction factor depends on the Reynolds number of the border layer between the particle and fluid:

$$Re_s = \frac{\rho v_s d}{\mu} \tag{5.4}$$

μ = viscosity of the surrounding fluid

Correlation for spherical particles can also be expressed analytically:

Reynolds number	**Friction Factor**	**Flow Ratio**
$500 < Re_s < 2 \cdot 10^5$	$f_s = 0.44$	Turbulent boundary layer
$2 < Re_s < 500$	$f_s = 18.5\ Re_s^{-0.6}$	Transition
$10^{-5} < Re_s < 2$	$f_s = 24\ Re_s^{-1}$	Laminar boundary layer

Table 4: Correlation for spherical particles

Using iterations in MATLAB on Equation 5.3 and Equation 5.4, the settling velocity was found. For sand with a diameter of 200 microns, the settling velocity was calculated to 0.0247 m/s.

5.3 Flow sand distribution

Once the fluid velocity profile and grain settling velocity are determined, the sand distribution profile can be constructed using Mixing Length Theory (Fredsøe, 2019). The main equation governing the sand flow through a small area perpendicular to the flow is given by:

$$c \cdot w + v_t \frac{dc}{dy} = 0 \tag{5.5}$$

where c is the sand concentration, w is the settling velocity of the grain, v_t is the eddy viscosity, and y is the coordinate along the pipe diameter.

The Eddy viscosity, v_t, is determined as:

$$v_t = l(y)^2 \left| \frac{dU}{dy} \right| \tag{5.6}$$

where U is the flow velocity and $l(y)$ is the mixing length depending on the coordinate.

Assuming that l varies linearly with the distance y from the wall, we can express it as:

$$l(y) = k \cdot y \tag{5.7}$$

where k is von Kármán’s constant, typically 0.40.

6 Digital signal processing (DSP)

6.1 Sensitivity

In our research, we analyze the acoustic signals generated by sand particles striking the pipe wall, which primarily consist of high frequencies. These signals vary depending on the particle size, with larger grains producing audible sound while smaller grains yield frequencies exceeding 100 kHz. We employ the PK15I sensor, designed for measurements between 100 and 450 kHz, with a resonant frequency around 150 kHz. However, it's essential to consider the sensor's sensitivity curve, which exhibits higher amplitudes around 150 kHz and lower sensitivity for frequencies between 200 and 450 kHz. To address this, we utilize the high-resolution PicoScope 4262a oscilloscope with a sampling frequency of 5 MHz and 16 bits resolution, enabling us to capture and investigate even weak signals effectively. These findings contribute to our understanding of sand suspension dynamics in fluid flow systems.

6.2 Signal processing

In the research conducted by Igor Grishkov, a Python program was developed to control the oscilloscope used in the laboratory. The program was programmed to record data at a sampling rate of 5 MHz, ensuring that all essential information was captured in the raw recordings. Grishkov designed the program to extract as much information as possible from the recordings.

Considering the sensor's limited sensitivity to frequencies above 450 kHz, Grishkov made the decision to downsample the recordings by a factor of 4. This downsampling allowed for the preservation of the important signal components while reducing the data size and computational requirements.

To further refine the recordings, Grishkov employed MATLAB's "Filter Designer" tool to design a finite impulse response (FIR) filter. The chosen filter had a linear phase response, ensuring that the filtered signal maintained its temporal integrity. By applying this filter to the downsampled data, unwanted frequencies were effectively removed, resulting in a clean and focused signal suitable for subsequent analysis and modeling.

These processing steps, meticulously performed by Grishkov as outlined in his master thesis, were essential in preparing the recorded data for further investigation in the laboratory.

6.3 Power Spectral Density

In order to analyze the characteristics of the signal obtained from the sand grains, the focus of the signal processing is on the power and frequency analysis. To achieve this, the next step involves computing the Power Spectral Density (PSD), which provides information about the distribution of power across different frequencies.

Since the signal produced by the sand grains is stochastic or random in nature, it requires analysis methods suitable for such signals. One approach is to convert the stochastic signal into the frequency domain using a periodogram, which is essentially the Fourier transform of the signal's autocorrelation function. However, this method can result in a representation with strong oscillations and uncertainty.

To obtain a clearer representation of the PSD, a decision was made to utilize the Bartlett method. This method involves dividing the signal, denoted as x[n], into "K" non-overlapping segments of length "M". By applying this method, the PSD can be estimated more accurately and with reduced uncertainty (Werner, 2021).

This approach, based on previous literature and implemented by Igor Grishkov, allows for a comprehensive analysis of the power distribution and frequency characteristics of the signal obtained from the sand grains.

6.4 Spectrogram analysis

Currently, all the necessary processing steps have been completed to generate the Power Spectral Density (PSD) representation for each recorded file. To analyze the sand concentration in the fluid flow, a spectrogram is constructed by building a PSD for each one-second recording segment. Considering the focus on the weak signal within the 200 kHz to 450 kHz frequency range, a logarithmic scale is used to represent the amplitude.

In Figure 6.1, a spectrogram is presented for a pump frequency of 15 Hz and sand grains with diameters ranging from 0.4 to 0.8 mm. The sand was directly added to the water intake, with approximately 20 ml (equivalent to around 34 g) injected into the system after a small amount of sand grains had been recirculating in the Flow Loop. The spectrogram clearly shows a significant increase in power density along the investigated frequencies when the added sand passes the sensor (around the 20th to 30th second). Furthermore, a rise in power density is observed after the majority of the sand has passed through the sensor, likely due to delayed grains or increased recirculation.

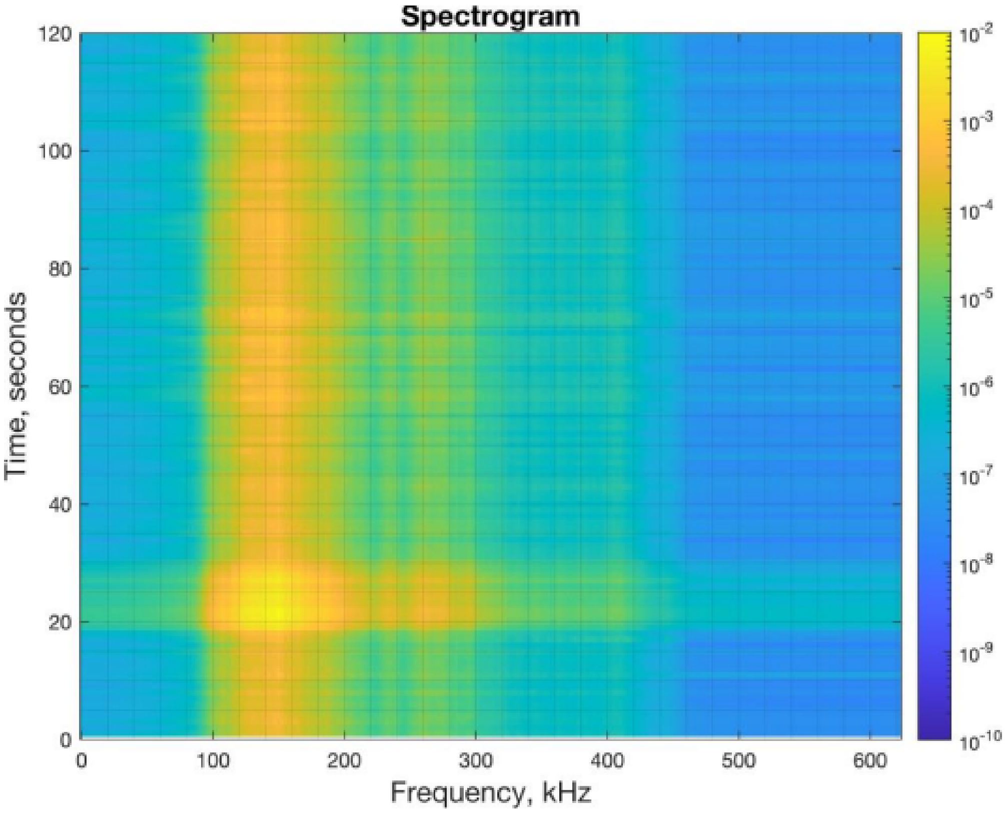


Figure 6.1: Pump frequency Spectrogram at 15 Hz, sand grain diameter: 0.4-0.8 mm (Grishkov, 2022)

A closer examination of the PSD reveals specific frequency bands of interest. From the literature review, it is known that focusing on representative frequency bands can aid in spectrum analysis. In this case, clear responses are observed around the resonant frequency of approximately 150 kHz, as well as around 265 kHz and 405 kHz. These intervals are deemed suitable for representing frequency bands. For further analysis, the average power density within each representing frequency band is calculated and then multiplied by appropriate coefficients (103, 104, and 105) to compensate for sensor sensitivity issues and achieve a unified scale for all three bands.

These resulting curves provide valuable insights into the passage of the added sand through the sensor and indicate the presence of energy from re-circulation before the sand arrival.

However, it cannot be guaranteed that these frequency bands are completely free from background noise. Based on the literature, it is understood that higher frequencies are less affected by background noise in the sand grain impingement signal. Therefore, the third frequency band from 400 kHz to 410 kHz appears to be the most promising, while the other two bands (140 kHz to 160 kHz and 260 kHz to 270 kHz) also contain relevant information, as depicted in Figure 6.2.

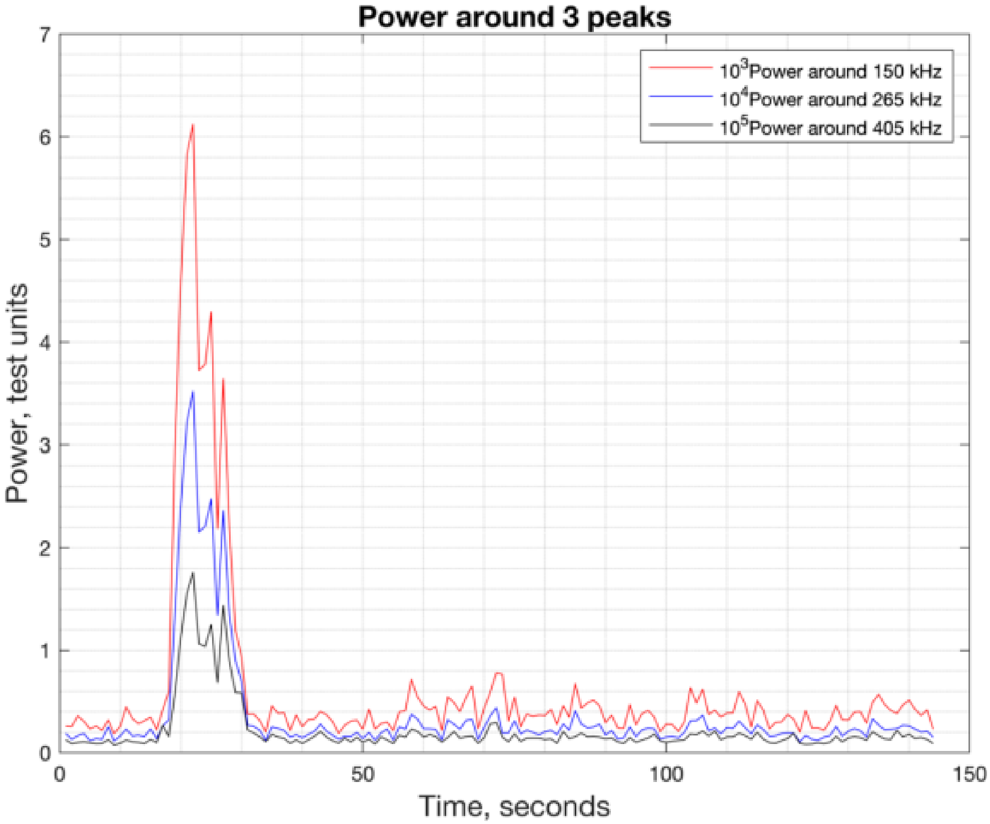


Figure 6.2: Average power density in three intervals, recorded with same specifications as in Figure 6.1 (Grishkov, 2022)

7 System improvements

7.1 Sand injection pumping tube

As mentioned earlier, one of the objectives was to improve the current sand injection system, and a glass tube was suggested. However, one key concern with glass tubes is their inconsistent rounding, stemming from the inherent variability in the glass manufacturing process. While glass tubes can be produced with a high degree of precision, attaining absolute circularity can be difficult to achieve consistently.

The manufacturing of glass tubes involves various steps, including glass melting, forming, and shaping processes. During these processes, the molten glass is typically shaped into a tube by extrusion or blowing methods. (Firth, 2013) However, factors such as temperature variations, the viscosity of the molten glass, and the shaping technique employed can introduce slight irregularities in the tube's cross-sectional shape.

Even with careful control and monitoring of the manufacturing parameters, it is challenging to eliminate all imperfections and variations in the tube's roundness. The cooling and solidification processes can further contribute to the formation of slight distortions in the tube's shape.

In contrast, PVC tubes undergo a more controlled manufacturing process, resulting in uniform and consistent internal surfaces. This smoothness facilitates a predictable and stable flow of sand, enhancing the performance of the sand injection pump. The consistent internal diameter of PVC tubes also minimizes the risk of blockages or clogging, ensuring reliable operation.

Durability is another crucial factor to consider. Glass, being brittle, is prone to breakage and cracking when subjected to mechanical stress or impact. In sand injection pumps, the constant movement of abrasive sand particles exacerbates this vulnerability, increasing the likelihood of tube failure. This fragility not only compromises system reliability but also poses safety hazards to operators and the surrounding environment.

On the other hand, in sand injection pump applications, the compatibility of PVC tubes with certain chemicals, like alcogel, may raise concerns due to the potential degradation of the plastic over time.

PVC is known to be susceptible to chemical attack by various solvents, including some alcohols. Exposure to these solvents can result in swelling, softening, or even dissolution of the PVC material. With alcogel, the alcohol component can cause a similar reaction, compromising the structural integrity of the PVC tube.

The breakdown of PVC caused by alcogel can lead to issues in our application. The degraded PVC material can create a rough and uneven internal surface, disrupting sand flow and reducing pump efficiency. Moreover, the weakened PVC may be prone to cracking or failure, posing safety risks and increasing maintenance needs.

In cases where alcogel is used in the tubing, alternative materials should be considered to ensure long-term reliability and performance of the sand injection pump system. Chemically resistant plastics like polypropylene (PP) or polyethylene (PE), as well as corrosion-resistant metals like stainless steel, can be viable options. The choice of material depends on specific application requirements, cost considerations, availability, and ease of installation.

In conclusion, PVC tubes offer several advantages over glass tubes in sand injection pump applications, however when alcogel compatibility is a concern, alternative materials with better chemical resistance should be considered to maintain the system's stability and performance.

7.1.1 Glass pipe testing - Initial testing

In an attempt to explore the viability of using glass tubes instead of the plastic tube, an experimental investigation was conducted to compare the use of glass and PVC tubes. The experiment sought to establish a sand injection pump system using a glass tube and evaluate its performance in terms of flow rate, pressure drop, and pumping efficiency under diverse operational conditions.

Left behind from previous attempts of Igor Grishkov, was a glass tube that he had acquired. Therefore the first test was done with this leftover glass tube, with the same piston and o-rings as Grishkov had used. However, the experiment encountered a significant challenge as the piston within the glass tube experienced sticking issues, and because of this began to skip/jump alongside the tube instead of smoothly moving along.

This resulted in irregular flow patterns and diminished pumping efficiency. Upon careful examination, it became apparent that the inconsistent wall structure of the glass tube was the underlying cause of this problem. The uneven internal surface of the glass tube led to resistance and obstruction, impeding the smooth movement of the piston.

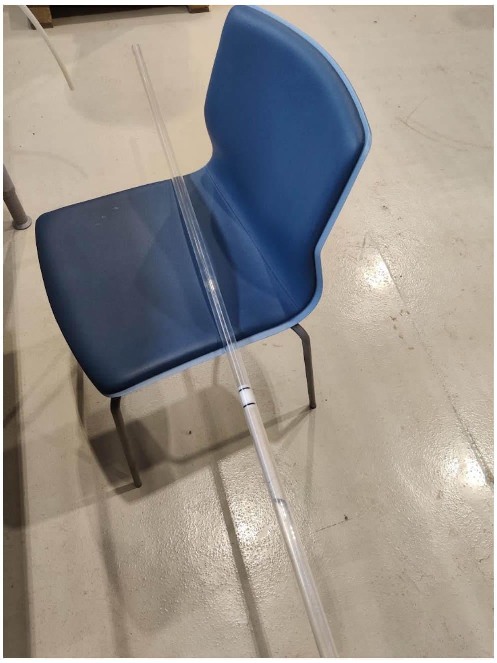


Figure 7.1: Old glass tube

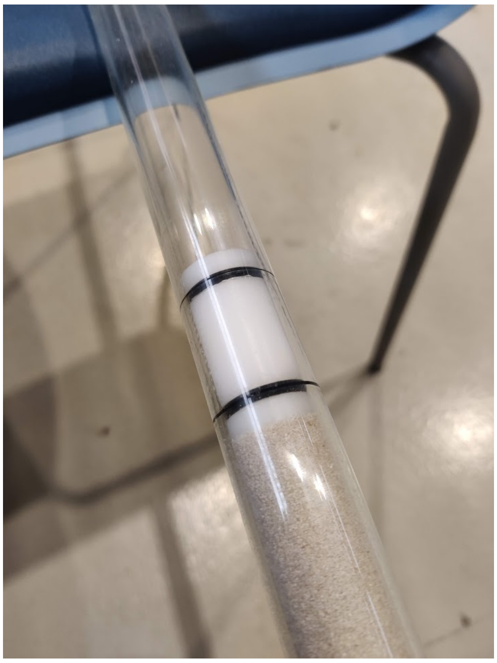


Figure 7.2: Closeup of old glass tube with piston and o-rings

7.1.2 Glass pipe testing - New tube, piston and o-ring

After conducting initial tests, I sought advice from Noralf Vedvik. Vedvik suggested exploring different glass tubes to find one with fewer imperfections and better performance. While I encountered some setbacks, including the breakage of a few tubes, Vedvik continued until he identified a glass tube where the piston slid smoothly instead of jumping.

To further optimize the glass tube's performance, I conducted additional testing with various pistons and replaced the o-rings. Once these modifications were complete, the glass tube was ready for practical application. To ensure secure installation, epoxy was carefully applied around the top and bottom sections of the tube, and flow loop clamps were attached as shown in Figure 7.4. The new tube was then ready for use after the epoxy dried.

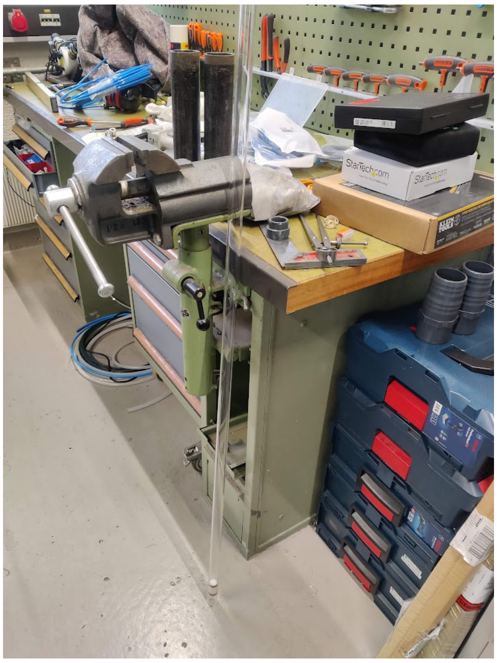


Figure 7.3: New glass tube

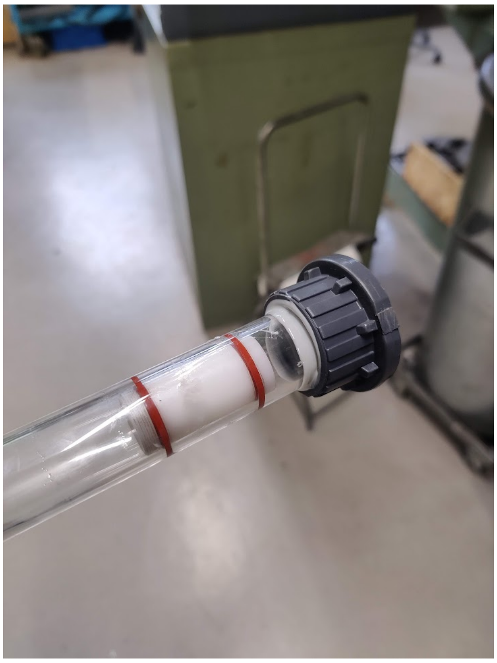


Figure 7.4: Closeup of new glass tube with new piston and o-rings

8 Signal acquisition and analysis

8.1 Signal acquisition program

In order to capture the acoustic signals from the sensor, a Python script developed by Igor Grishkov was used. This Python file can be found attached in Appendix 2 - Python Sand Acquisition Program.

The Python code performs data acquisition using the PicoScope 4000 Series device. Here's a description of what the code does:

1. Importing necessary libraries and modules: The code begins by importing the required libraries and modules. These include "ctypes" for interacting with C libraries, "numpy" for numerical operations, "scipy.io" for saving data to MATLAB files, "scipy.signal" for signal processing functions, "keyboard" for keyboard input handling, "picosdk.ps4000" for accessing the PicoScope 4000 Series device, "matplotlib.pyplot" for plotting, and "time" for time-related operations.

2. Set parameters: The code sets parameters for the data acquisition. "FileSize" represents the length of each recording in milliseconds, and "SampFreq" defines the sampling frequency in kHz. These values can be adjusted based on the desired recording duration and sampling rate.

3. Create a specific subfolder for log files: The code checks if a subfolder named "logfiles" exists. If not, it creates the folder to store log files generated during the data acquisition process. This helps organize and separate the log files from other project files.

4. Open the PicoScope device: The code opens the PicoScope 4000 Series device by calling the "ps4000OpenUnit" function from the PicoSDK library. It returns a handle ("chandle") that is used for subsequent API calls.

5. Set up channel A: The code configures channel A of the PicoScope device using the "ps4000SetChannel" function. It enables channel A, selects DC coupling, and sets the voltage range to 2V. These settings can be modified to suit the specific measurement requirements.

6. Configure data collection parameters: The code determines the total number of samples to collect based on the "FileSize" and "SampFreq" parameters. It calculates the number

of buffers needed to capture the data by dividing the total number of samples by the size of each buffer.

7. Set up data buffers: The code sets up data buffers for collecting and storing the acquired data. It uses the "ps4000SetDataBuffers" function to assign buffers to channel A. The buffer size is determined by the "sizeOfOneBuffer" variable.

8. Set up streaming mode: The code configures the PicoScope device for streaming mode data acquisition. It specifies the sample interval in nanoseconds ("sampleInterval"), the sample units ("sampleUnits"), the maximum number of pre-trigger samples, the total number of samples to collect, the auto-stop behavior, and the downsample ratio.

9. Define a streaming callback function: The code defines a callback function named "streaming_callback". This function is called by the PicoScope driver when new data is available. Inside the callback function, the acquired data is copied from the driver's buffer ("bufferAMax") to a larger buffer ("bufferCompleteA") for further processing.

10. Initiate the data acquisition loop: The code enters an infinite acquisition loop using a "while True" loop. Within the loop, it waits for the callback function to be called and checks if new data is available. If data is not yet available, it sleeps for a short duration to avoid excessive CPU usage. The loop can be terminated by pressing the ESC key.

11. Save data to a MAT file: Once the acquisition loop is complete or terminated, the code saves the acquired data to a MATLAB file (MAT file) using the "scipy.io.savemat" function. The MAT file contains the complete data buffer ("bufferCompleteA"), the sample interval in nanoseconds, the selected voltage range, and the timestamp of when the acquisition started.

12. Stop and close the PicoScope device: After the data acquisition is finished, the code

calls the "ps4000Stop" function to stop the data acquisition and "ps4000CloseUnit" to close the PicoScope device. These functions release any resources allocated during the acquisition process.

13. Display status information: The code prints the status information obtained during execution, indicating the success or failure of each API call. This information helps to track the progress and diagnose any potential issues during data acquisition.

Overall, this code provides a framework for continuous data acquisition using a PicoScope 4000 Series device, including setting up the device, configuring the acquisition parameters, collecting data using streaming mode, and saving the acquired data to a MATLAB file.

8.2 Signal analysis

To extract meaningful data from the signals, Igor Grishkov had developed a MATLAB script during his own thesis research. Subsequently, he began working on a Python version of the script, which was eventually passed on to me. However, upon receiving the Python script, I encountered difficulties as it did not function properly. Thus, it became necessary for me to engage in a process of modification, building, and tinkering to ensure its successful execution. This Python file can be found attached in Appendix 3 - Python Sand Analysis Program.

1. Importing Libraries: The code starts by importing various libraries such as "numpy", "matplotlib.pyplot", "scipy.io", "scipy.signal", "tkinter", "glob", "os.path", "time", and "threading". These libraries provide functionalities for scientific computing, data visualization, file operations, user interface, time-related operations, and threading.

2. Setting Parameters: Several parameters are defined, such as "FlowRate", "Sand", "fs", "lengthSamp", "UpdateRate", "M", "zero1", "zero3", and "Vrange". These parameters control aspects such as fluid flow rate, sand grain diameter, sampling frequency, recording length, data update frequency, number of points in one periodogram, and voltage range.

3. Approximate Pump Velocities: Some comments that suggest the pump velocities corresponding to specific frequencies. However, this section is commented out and doesn't affect the code execution. This is used as a reference for the parameter "FlowRate", mentioned earlier.

4. Loop Preparation: Several arrays ("SandSizeAv", "SandProdAv", "PAV1", "PAV3", "SP", "SZ", "POW1", "POW3", "SandEst") are initialized with zeros to store data.

5. Loading Filter Coefficients: Used to load filter coefficients from a file. This filtercoefficient-file was made with MATLAB's filterDesigner, and is attached in Appendix 4. However, this step was not used, and as such, there is no filtercoefficient.

6. Creating Control Buttons: A GUI application is created using the "tkinter" library. Three buttons ("myButton1", "myButton2", "myButton3") are defined with corresponding click event functions ("myClick1", "myClick2", "myClick3"). These buttons allow the user to control the script by stopping it, calibrating sand, and getting zero values.

7. Infinite Loop for Data Processing: This is where the most important part happens, and the part I worked mostly on. Therefore this explaination will be a little more in-depth.

- Before anything, the folder with all the MATLAB logfiles are loaded. Then, the latest file is removed.

- An infinite loop is started using "while True" to continuously process data.

- The script prints "Working" as an indication of progress.

- Inside the loop, there is a nested loop that averages and updates main monitoring values ("SandSizeAv", "SandProdAv", "PAV1", "PAV3") based on the specified update rate ("l").

- The loop loads data from the first MATLAB file ("first_file") in the logfiles-folder, using "scipy.io.loadmat".

- The loaded data is processed and analyzed, including filtering, calculating periodograms, plotting, and performing frequency bands analysis.

- The power values in specific frequency zones are calculated ("power1" and "power3").

- Amplification coefficients ("amp") are determined based on the sand grain diameter ("Sand").

- The power of the scaled signal and sand production are calculated ("POWER" and "SandProd").

- The main values and time check are updated.

- The current file that has just been loaded and calculated on, is then removed, and the loop continues until the all the files have been loaded, or the user cancels.

- After the loop finishes, an array of data is created ("sandsize", "sandproduction", "SP", "SZ", "POW1").

9 Current results

9.1 System improvements - Glass sand injection tube

The new and improved sand injection system proved to be highly effective in fulfilling its intended purpose. The responsive behavior of the piston within the system demonstrated the successful implementation of the push-and-pull mechanics controlled through the sand loop interface. While concerns regarding skipping or jumping of the piston were prominent during the initial stages, the current glass tube exhibited minimal noticeable effects of this issue. Interestingly, we observed that these effects were more prevalent within specific sections of the tube, approximately 10-15 cm from both ends.

In our pursuit of an optimal solution, one possibility was to cut the glass tube to remove the imperfect portions. However, upon careful consideration, we deemed it impractical to undertake this approach. The risk of potential glass breakage, as experienced during previous attempts with other glass pipes, outweighed the potential benefits. Additionally, the time investment required for such a precise operation was not worthwhile our time. Therefore, we opted to keep the glass tube as it was.

However, this decision did introduce some challenges when reconnecting the sand injection system between each round of tests. Given the considerable height of the glass pipe, which extended to nearly two meters, a long hose was necessary to connect the piston drive and pressurize the injection system using water. This resulted in careful attention to ensuring the hose contained only water and no air, in order to maintain consistent pressure.

Refilling the lengthy glass tube with our alcogel and sand mixture proved to be an problematic and time-consuming task. Inverting the glass tube presented a considerable challenge as it needed to remain connected to the pressure hose throughout the process. The task of refilling required me to ascend a chair each time, carefully handle the tube, and carefully pour the mixture. Subsequently, reattaching the hose to the flow loop required me to carefully flip the glass tube back to its original position without any spills.

However, with practice and repeated attempts, I gradually developed proficiency in maneuvering through these steps. What initially posed as a significant inconvenience gradually transformed into a mere irritation, as I refined my technique and navigated the process more smoothly.

9.2 Signal correlations of different sand grains

9.2.1 Non-mixed sand diameter tests

The flow loop inlet pump ran at 30% of 50Hz, meaning 15Hz or 345 l/min, for all the tests.

The initial phase of the experimentation involved conducting five tests using sand particles ranging from 710 to 850 microns in size, with an average diameter of approximately 780 microns. Each test consisted of mixing 30 grams of sand with an estimated amount of alcogel, followed by thorough mixing for a few minutes to ensure proper suspension of the sand particles within the gel. However, due to the challenges posed by the length of the glass tube, as previously discussed, difficulties arose in transferring the entire mixture into the tube. Some spillage occurred, and a portion of the mixture remained in the pouring glass. As an estimate, it is suspected that approximately 7 grams out of the total 30 grams were not successfully transferred, leaving a total weight of 23 grams for subsequent calculations.

The second phase involved five tests using finer sand particles, this time ranging from 210 to 295 microns, with an average diameter of approximately 253 microns. Similar to the first test, each test was made by mixing 30 grams of sand with an arbitrary amount of alcogel, enough to disperse the sand. However, with finer sand, it was still difficult to get all of the mixture into the tube, similarly to the first test. Anyhow, it was a bit easier, and it is estimated that approximately 5 grams out of the total 30 grams were not successfully transferred, leaving 25 grams of injected mass.

Table 5 provides an overview of the conducted tests, indicating the measured mass in grams, and the corresponding error in percentage. The results demonstrate that for medium-sized sands, the average error was 29.66%, while for finer sands, it was 27.244%. Remarkably, these error values are quite similar, suggesting that the program does not exhibit a significant preference or difficulty in handling one sand type over the other.

The obtained average error values serve as a benchmark for comparing the performance of our Python algorithm when applied to pure sand sizes versus mixed sand sizes. By utilizing these baseline values, we can assess the effectiveness and accuracy of the algorithm in differentiating between the two scenarios and evaluate any variations in the error rates.

Test number	Sand type	Injected mass, g	Measured mass, g	Error
1	Medium size sand with grain distribution from 710 to 850 microns, with an average diameter around 780 microns.	30 (23 after estimated loss during transfer into injection tube)	25.05	8.533%
2			13.69	50.749%
3			15.62	38.219%
4			19.11	18.475%
5			16.60	32.323%
			Average error:	**29.660%**
1	Fine sand with grain size distribution from 210 to 295 microns, with an average diameter around 253 microns	30 (25 after estimated loss during transfer into injection tube)	20.28	20.848%
2			20.25	20.995%
3			17.02	37.982%
4			20.48	19.877%
5			17.28	36.518%
			Average error:	**27.244%**

Table 5: Error estimation for tests 1-5 for only medium and fine sand, respectively

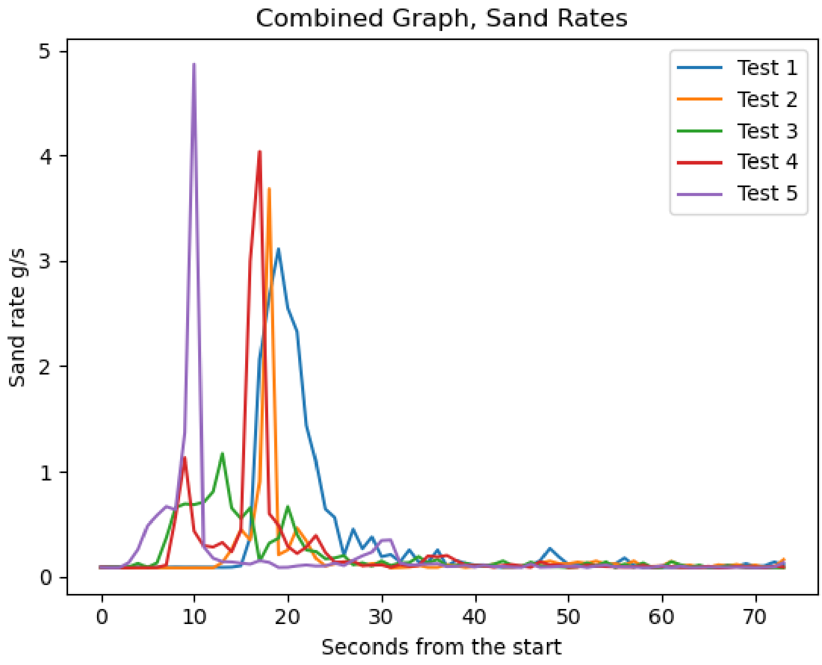


Figure 9.1: Sand Production, Medium sand

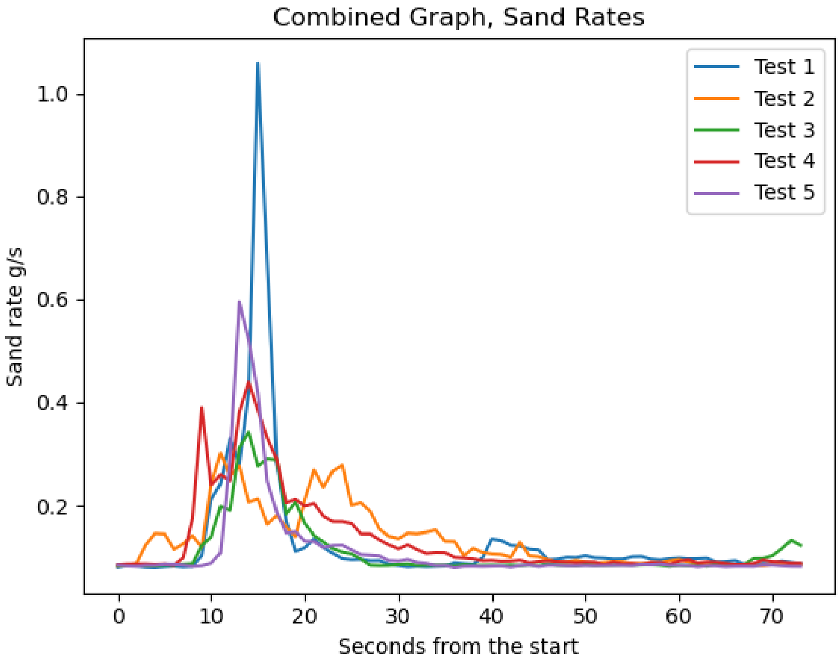


Figure 9.2: Sand Production, Fine sand

9.2.2 Mixed sand diameter tests

9.2.2.1 50%-50% grain size mix

After conducting the initial tests with pure fine or medium sand, we proceeded to perform an additional set of five tests using a mixture consisting of 50% fine sand and 50% medium sand. To ensure accurate proportions, meticulous care was taken in weighing the sand types to achieve as close to equal parts as possible. Given the average grain diameter of 253 microns for fine sand and 780 microns for medium sand, an average grain size of 516 microns was calculated as the representative diameter used in the Python program for data processing.

During the pouring process, an estimated 3 grams of the mixture did not successfully enter the sand injection tube. Consequently, the injected mass considered for subsequent calculations was adjusted to 27 grams.

Examining Table 6, it becomes evident that the observed error is considerably significant, ranging from 126.877% to a maximum of 138.558%. The calculated average error across all tests amounted to 133.621%. These high error values indicate notable deviations between the expected and measured results, suggesting potential limitations in the accuracy and precision of the Python algorithm when dealing with mixed sand sizes.

Test number	**Sand type**	**Injected mass, g**	**Measured mass, g**	**Error**
1	50% fine sand, with average diameter around 253 microns, 50% medium sand, with average diameter around 780 microns. Total average: 516 microns	30 (27 after estimated loss during transfer into injection tube)	6.04	126.877%
2			5.08	136.658%
3			5.12	136.239%
4			4.90	138.558%
5			5.75	129.771%
			Average error:	**133.621%**

Table 6: Error estimation for tests 1-5 for 50%-50% mix of fine + medium sand

9.2.2.2 75%-25% grain size mixes

In order to explore the performance limits of the Python code, additional tests were conducted using mixed sand compositions. Specifically, five tests were carried out with a mixture of 75% fine sand and 25% medium sand, while another five tests were performed with a composition of 75% medium sand and 25% fine sand. The average grain size diameters for these two cases were calculated to be approximately 385 microns and 648 microns, respectively.

Similar to previous experiments, an estimated 3 grams of the sand mixture did not make it into the injection tube during the pouring process. Consequently, the injected mass used for subsequent calculations was adjusted to 27 grams to account for this discrepancy.

Analyzing the results presented in Table 7, it becomes evident that the average error for the 75% fine sand mixture was 138.411%, exhibiting a relatively consistent range of individual error percentages across the five tests, spanning from 125.89% to 144.718%. In contrast, the 75% medium sand mixture displayed a lower average error of 108.836%, with a narrower range of individual error percentages, ranging from 104.483% to 111.419%.

Test number	**Sand type**	**Injected mass, g**	**Measured mass, g**	**Error**
1	75% fine sand, with average diameter around 253 microns, 25% medium sand, with average diameter around 780 microns. Total average: 385 microns	30 (27 after estimated loss during transfer into injection tube)	4.68	140.909%
2			4.33	144.718%
3			4.87	138.877%
4			6.14	125.890%
5			4.61	141.664%
			Average error:	**138.411%**
1	75% medium sand, with average diameter around 780 microns, 25% fine sand, with average diameter around 253 microns. Total average: 648 microns	30 (27 after estimated loss during transfer into injection tube)	7.97	108.836%
2			8.47	104.483%
3			7.86	109.811%
4			7.88	109.633%
5			7.68	111.419%
			Average error:	**108.836%**

Table 7: Error estimation for tests 1-5 for 75%-25% mix of fine + medium sand, and medium + fine sand, respectively

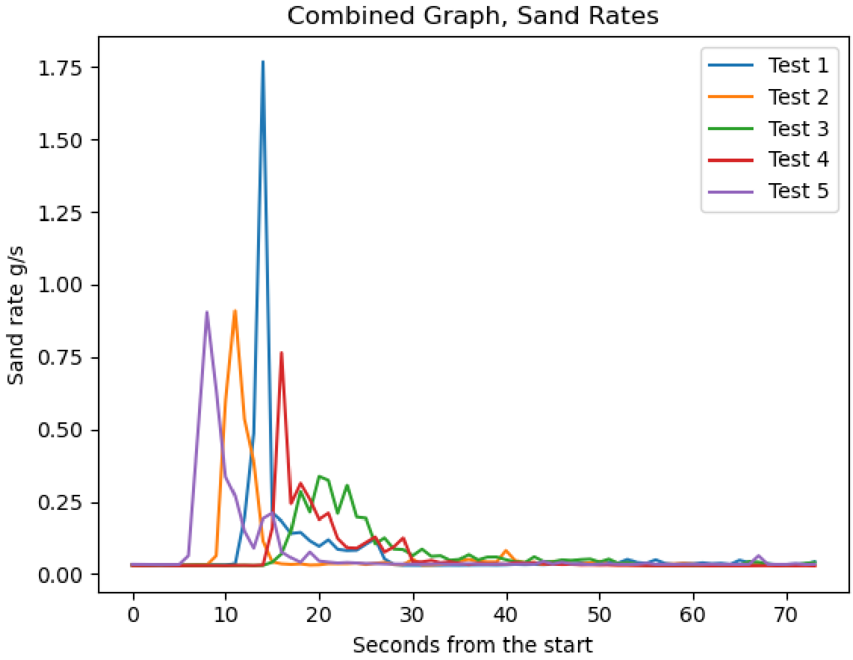


Figure 9.3: Sand Production, 50% fine sand, 50% medium sand

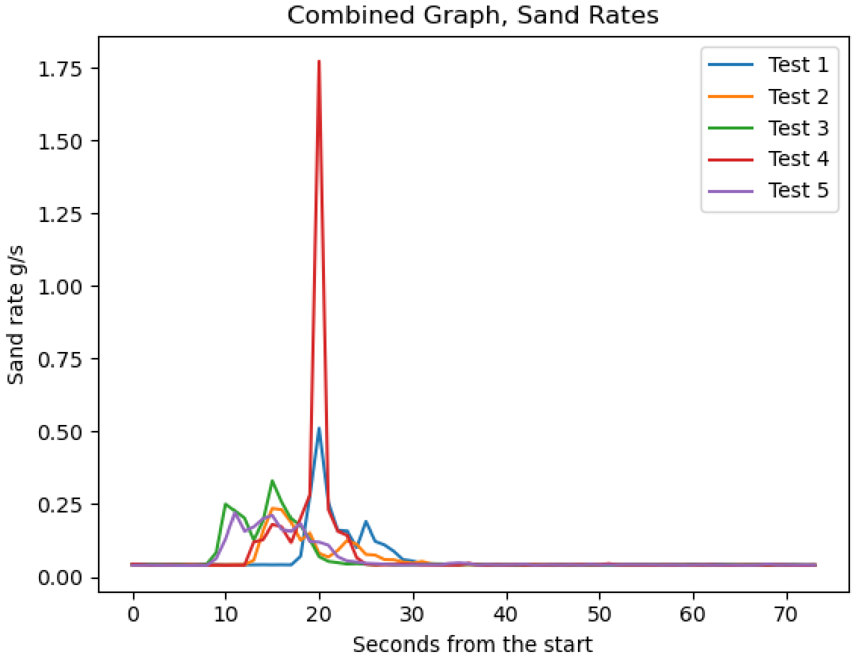


Figure 9.4: Sand Production, 75% fine sand, 25% medium sand

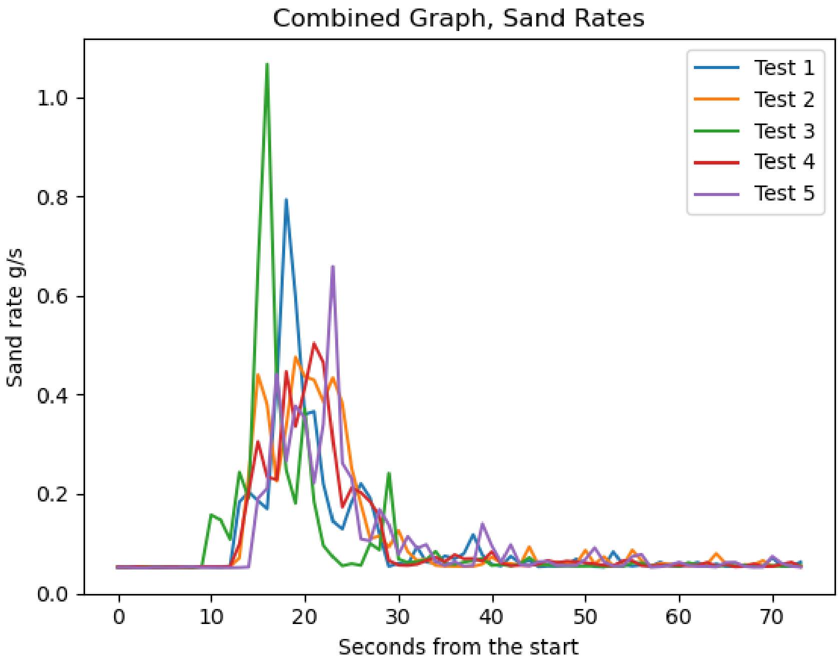


Figure 9.5: Sand Production, 75% medium sand, 25% fine sand

9.2.3 Conclusion

The substantial average error rates observed when using the Python script with mixed sand compositions compared to pure sand mixtures can be attributed to several factors. One contributing factor is the difference in grain sizes between the fine and medium sand components. As the Python script utilizes an average grain size value derived from the diameters of the individual sand types, this approximation may not accurately capture the actual behavior and interactions within the mixed sand system. The varied characteristics of fine and medium sand grains, such as their shape, surface texture, and packing tendencies, can lead to complex dynamics that are not adequately captured by a single average grain size.

These findings also suggest that the Python code faced greater challenges and yielded higher error rates when dealing with the 75% fine sand mixture compared to the 75% medium sand mixture. The discrepancies in error rates may be attributed to the variations in grain sizes, which can affect the flow behavior, packing density, and settling dynamics within the sand injection system. Additionally, differences in the distribution and arrangement of the sand particles, as well as the underlying assumptions and modeling approach of the Python code, could contribute to the observed variations in error rates.

Furthermore, the distribution and arrangement of the sand particles within the mixture can significantly influence the flow behavior and packing density. In the case of the mixed sand samples, achieving a perfectly homogeneous distribution of fine and medium sand particles can be challenging, leading to local variations and inconsistencies in the sand packing structure. These variations can introduce errors and discrepancies in the measured mass and subsequently affect the accuracy of the calculations performed by the Python script.

Another aspect to consider is the potential differences in settling and suspension characteristics between the fine and medium sand particles. Due to variations in particle size, shape, and density, the sedimentation and suspension behaviors of the sand particles can differ. This can result in uneven distribution and settling of the sand grains during the pouring process, further contributing to discrepancies between the expected and measured masses.

Lastly, it is essential to recognize the limitations and assumptions inherent in the Python script itself. The script uses simplifying assumptions or models that do not fully capture the complexities of mixed sand systems. Variations in particle size, shape, and composition can introduce additional challenges in accurately predicting the behavior and performance of the sand injection system, leading to higher error rates when working with mixed sands.

These results further emphasize the importance of understanding the limitations and performance characteristics of the Python code when applied to mixed sand compositions. Evaluating the system's response to varying ratios of fine and medium sand provides valuable insights for refining the code and optimizing its accuracy in predicting and analyzing sand injection processes.

Overall, the combination of differences in grain sizes, challenges in achieving homogeneous mixtures, variations in settling behavior, and limitations in the Python script's modeling approach collectively contribute to the observed higher average error rates when using the script with mixed sand compositions compared to pure sand mixtures.

10 Future work

10.1 Glass Tube Enhancement

One of the key findings of this research is the susceptibility of the glass tube to breakage and fragility issues. The experiments revealed that the glass tube's structural integrity is compromised, leading to sub-optimal performance and potential safety hazards. To overcome this challenge, it is recommended to explore alternative tube materials such as reinforced plastics, stainless steel, or corrosion-resistant metals. These materials offer superior strength and durability, ensuring the reliability and longevity of the sand injection system. Additionally, attention should be given to improving the design of the tube, particularly reinforcing vulnerable areas prone to breakage or implementing protective measures like rubber or silicone sleeves.

10.2 Python Algorithm Refinement

The Python algorithm used for detecting mixed sand grain diameters demonstrated limitations in accuracy during the experiments. It is essential to address these limitations to ensure reliable and precise results. Based on the insights gained from this research, several improvements can be made to the algorithm. Firstly, refining the data processing and analysis algorithms will enhance the program's ability to accurately detect and classify mixed sand grain diameters. This may involve optimizing the thresholds, filters, and calibration factors to better suit the characteristics of mixed sand samples. Secondly, implementing additional data validation and error handling mechanisms will improve the robustness of the algorithm and minimize inaccuracies or false readings. Through these refinements, the Python program can become a powerful tool for analyzing mixed sand grain data and provide valuable insights for further research.

11 Conclusion

This thesis has focused on two key aspects: the improvement of the glass tube used in sand injection systems and the development of a Python script for analyzing sand grain diameters.

The investigation into the glass tube revealed its potential for sand injection applications due to its transparency and chemical resistance. However, challenges were encountered, including fragility and breakage issues. Through careful modifications and refinements, the structural integrity of the glass tube was improved, ensuring its reliability and durability during sand injection operations. Addressing concerns such as skipping and irregular flow patterns, the performance of the glass tube was enhanced, resulting in more stable sand flow and increased pumping efficiency.

The development of the Python script for analyzing sand grain diameters proved to be a significant contribution. Building upon previous work, the script was refined and optimized to accurately detect and quantify sand grain sizes. The script demonstrated great performance when analyzing pure sand mixes, providing precise enough measurements. However, it encountered challenges when mixed sand sizes were introduced, as the error values increased significantly. This indicates the need for further improvements and fine-tuning of the script to enhance its accuracy and reliability in mixed sand scenarios.

In conclusion, this research has addressed the challenges associated with glass tubes in sand injection systems and has developed a Python script for analyzing sand grain diameters. The improvements made to the glass tube have enhanced its reliability, while the Python script has shown promising results for pure sand mixes. Future work should focus on refining the script to handle mixed sand sizes more effectively, reducing the associated error values. By continuing to improve these aspects, sand production monitoring can be further optimized, leading to advancements in various applications worldwide.

12 References

[1] Igor Grishkov. *Sand production monitoring based on Acoustic Emission.* Trondheim, Trøndelag, Norway: Norwegian University of Science and Technology., 2021.

[2] Igor Grishkov. *Sand production modeling based on Acoustic Emission.* Trondheim, Trøndelag, Norway: Norwegian University of Science and Technology., 2022.

[3] Martin Pham. *Using Acoustic Emission to monitor Sand production.* Trondheim, Trøndelag, Norway: Norwegian University of Science and Technology., 2022.

[4] P. Y. Lee, S. Kasper, and C. Quinn. *The 7 Sins of Managing Acoustic Sand Monitoring Systems.* 2017.

[5] G. Gao et al. *Sand rate model and data processing method for non-intrusive ultrasonic sand monitoring in flow pipeline.* 2016.

[6] C. Emiliani et al. *Improved sand management strategy: testing of sand monitors under controlled conditions.* 2011.

[7] A. Appalonov, Y. Maslennikova, and A. Khasanov. *Advanced Data Recognition Technique for Real-Time Sand Monitoring Systems.* 2021.

[8] Terence C. Blair and John G. McPherson. *Grain-Size and textural classification of coarse sedimentary particles.* Boulder, Colorado, USA Perth, Australia, 1949.

[9] Hisham B. Mahmud, Van H. Leong, and Yuli Lestariono. *Sand production: A smart control framework for risk mitigation.* Department of Petroleum Engineering, Curtin Malaysia, Malaysia., 2019.

[10] M. Ibrahim and T. Haugsdal. *Optimum procedures for calibrating acoustic sand detector, gas field case.* In Proceedings of the Canadian International Petroleum Conference/SPE Gas Technology Symposium 2008 Joint Conference, Calgary, Alberta, Canada, 17–19 June, Paper No. 2008–2025, 2008.

[11] B. McLaury. *A model to predict solid particle erosion in oilfield geometries.* Master's Thesis. Tulsa, Oklahoma, USA: Department of Mechanical Engineering, The University of Tulsa., 1993.

[12] R Sellens. *Power Law Profiles in Pipes.* Queen's University, 2022.

[13] H. Asheim. *Compendium for TPG4245 Production Wells.* Trondheim, Trøndelag, Norway: Norwegian University of Science and Technology, 2021.

[14] M. Firth. *Making Tubes & Sheets of Glass.* 2009. URL: https://mikegigi.com/tubeshet.htm#TUBES.

13 Appendix 1 - Risk assessment

To conduct a Risk Assessment, we use a matrix that divides all possible risks in three categories: Red – unacceptable risk, actions must be taken to reduce risk. Yellow – assessment area, actions must be considered. Green – acceptable risk, actions can be taken based on other criteria.

Consequence	(E) Catastrophic	🟨 -	🟥 -	🟥 -	🟥 -	🟥 -
	(D) Extensive	🟨 -	🟨 -	🟥 -	🟥 -	🟥 -
	(C) Moderate	🟨 C1 Electrical contact with water	🟨 C2 Flow Loop high pressure	🟨 C3 Injection system high pressure	🟥 -	🟥 -
	(B) Negligible	🟩 B1 Large water spill	🟩 B2 Small water spill	🟨 B3 Getting sand in eyes	🟨 -	🟥 -
	(A) Insignificant	🟩 A1 Ambient temperature out of equipment working range	🟩 A2 Loud noise from other activities	🟩 A3 Physical obstacles around installation	🟨 A4 Skin irritation because of gel	🟨 -
		(1) Rare	(2) Unlikely	(3) Possible	(4) Likely	(5) Almost certain
		Probability				

Table 8: Risk Assessment (Grishkov, 2022)

Based on the Risk Assessment matrix, it is evident that there are five potential risks identified in the yellow zone. Therefore, it is crucial to determine appropriate actions to mitigate these risks.

C1. Electrical contact with water: To minimize the risk of electrical contact with water, it is necessary to consistently wear protective rubber gloves while working. Additionally, all equipment should be waterproof and positioned at a safe distance from any water sources.

C2-C3. Over-pressurization of the sand injection system and Flow Loop: To address the risks associated with pump over-pressurization, it is essential to wear protective goggles at all times. Furthermore, it is crucial to exercise careful control over the valves of the installation to ensure the normal circulation of fluids.

B3. Potential eye injury from fine sand: Given that we are working with fine sand particles, it is important to wear protective goggles to prevent any particles from entering the eyes. Familiarity with a dedicated water tap for eye rinsing should also be ensured.

A4. Irritation from gel used for sand injection: The gel employed for sand injection may cause skin irritation. To mitigate this risk, it is necessary to consistently wear protective rubber gloves while handling the gel.

By implementing these recommended safety measures, the risks associated with electrical contact, over-pressurization, eye injury, and skin irritation can be significantly reduced, ensuring a safer working environment throughout the sand injection process.

14 Appendix 2 - Python Sand Acquisition Program

```python
# Copyright (C) 2018-2019 Pico Technology Ltd. See LICENSE file for terms.
#
# PS2000 Series (A API) STREAMING MODE EXAMPLE
# This example demonstrates how to call the ps4000 driver API functions in order to open a device, setup 2 channels and collects streamed data (1 buffer).
# This data is then plotted as mV against time in ns.

import ctypes
import numpy as np
import scipy.io
import scipy.signal
import keyboard
from picosdk.ps4000 import ps4000 as ps
import matplotlib.pyplot as plt
from picosdk.functions import adc2mV, assert_pico_ok
import time
import os



################################################
#Set parameters
FileSize=1000 #lenght of one recording milliseconds
SampFreq=1250 * 1000 #sampling frequency in kHz

#! changed to +-2 V range for Vallen systeme sensors
################################################


# ---------------- create a specific subfolder for logfiles if it does not already exist ---------------
# BEWARE: no protection of old logfiles, existing logfiles in the folder may be overwritten!!!!!!
if not os.path.exists("logfiles"):  # save logfiles to folder logfiles
    os.makedirs("logfiles")


# Create chandle and status ready for use
chandle = ctypes.c_int16()
```

```
status = {}

# Open PicoScope 4000 Series device
# Returns handle to chandle for use in future API functions
status["openunit"] = ps.ps4000OpenUnit(ctypes.byref(chandle))
assert_pico_ok(status["openunit"])


enabled = 1
disabled = 0
analogue_offset = 0.0

# tabell for å legge metadata om skop setup i mat fila som ren tekst
range=['PS4000_10MV',
'PS4000_20MV',
'PS4000_50MV',
'PS4000_100MV',
'PS4000_200MV',
'PS4000_500MV',
'PS4000_1V',
'PS4000_2V',
'PS4000_5V',
'PS4000_10V',
'PS4000_20V']


# Set up channel A
# handle = chandle
# channel = PS4000_CHANNEL_A = 0
# enabled = 1
# coupling type = PS4000_DC = 1
# range = PS4000_2V = 7
# SW experts often makes things complicated
# from datasheet: +-10 mV, +-20 mV, +-50 mV, +-100 mV, +-200 mV, +-500 mV, +-1 V, +-2 V, +-5 V, +-10 V, +-20 V
# the 'range' argument in the function call is an integer from 0 to 10 which corresponds to the ranges above
channel_range = ps.PS4000_RANGE['PS4000_2V']
status["setChA"] = ps.ps4000SetChannel(chandle,
                                        ps.PS4000_CHANNEL['PS4000_CHANNEL_A'],
                                        enabled,
                                        1,
                                        channel_range)
```

```
assert_pico_ok(status["setChA"])


# ---------------------------------------------------------
# The way this works is that a relatively small buffer in the hardware
# (number of samples is 'sizeOfOneBuffer')
# is filled numBuffersToCapture times to create one data capture

# Size of capture

# so this is the number of samples to collect to one datafile
totalSamples = int(FileSize * SampFreq/1000)
numBuffersToCapture = 10
sizeOfOneBuffer = int(totalSamples/numBuffersToCapture)



# declaration of storage space
# Create buffers ready for assigning pointers for data collection
bufferAMax = np.zeros(shape=sizeOfOneBuffer, dtype=np.int16)


memory_segment = 0

# Configuration of misc parameters for the driver
# Set data buffer location for data collection from channel A
# handle = chandle
# source = PS4000_CHANNEL_A = 0
# pointer to buffer max = ctypes.byref(bufferAMax)
# pointer to buffer min = ctypes.byref(bufferAMin)
# buffer length = maxSamples
# segment index = 0
# ratio mode = PS4000_RATIO_MODE_NONE = 0
status["setDataBuffersA"] = ps.ps4000SetDataBuffers(chandle,
                                                    ps.PS4000_CHANNEL['PS4000_CHANNEL_A'],
                                                    bufferAMax.ctypes.data_as(ctypes.POINTER(ctypes.c_int16)),
                                                    None,
                                                    sizeOfOneBuffer)
assert_pico_ok(status["setDataBuffersA"])


# Begin streaming mode:
```

```python
# set our sample periode in ns
#periodNanoSec=1/(SampFreq)*1000

periodSamp=int(1/SampFreq*1000000000)

sampleInterval = ctypes.c_int32(periodSamp)
sampleUnits = ps.PS4000_TIME_UNITS['PS4000_NS']

# this means 200ns sample intervall (5MHz sampling frequency)

# We are not triggering:
maxPreTriggerSamples = 0
#autoStopOnOff = 1   # this means autostop after collection of one buffer
autoStopOnOff = 0   # this means continue collection
# No downsampling:
downsampleRatio = 1
status["runStreaming"] = ps.ps4000RunStreaming(chandle,
                                               ctypes.byref(sampleInterval),
                                               sampleUnits,
                                               maxPreTriggerSamples,
                                               totalSamples,
                                               autoStopOnOff,
                                               downsampleRatio,
                                               sizeOfOneBuffer)
assert_pico_ok(status["runStreaming"])

actualSampleInterval = sampleInterval.value
actualSampleIntervalNs = actualSampleInterval

print("Capturing at sample interval %s ns" % actualSampleIntervalNs)

# We need a big buffer, not registered with the driver, to keep our complete capture in.
bufferCompleteA = np.zeros(shape=totalSamples, dtype=np.int16)
nextSample = 0
autoStopOuter = False
wasCalledBack = False


def streaming_callback(handle, noOfSamples, startIndex, overflow, triggerAt, triggered, autoStop, param):
    global nextSample, autoStopOuter, wasCalledBack
    wasCalledBack = True
    destEnd = nextSample + noOfSamples
    sourceEnd = startIndex + noOfSamples
    bufferCompleteA[nextSample:destEnd] = bufferAMax[startIndex:sourceEnd]
    nextSample += noOfSamples
    if autoStop:
        autoStopOuter = True


# Convert the python function into a C function pointer.
cFuncPtr = ps.StreamingReadyType(streaming_callback)

# ---------------------------------------------------------------
# this is our infinite acquisition loop which and saves data to file
# ---------------------------------------------------------------

my_counter=0    # just an index counter
while True:     # infinite loop
    tic=time.time()
    mystop=False
    start=time.time()   # read the system clock (Unix time, nuber of seconds since january 1, 1970)
    start_timestr=time.asctime() # human readable string
    nextSample = 0          # some initialization for the callback stuff
    autoStopOuter = False   # some initialization for the callback stuff
    wasCalledBack = False   # some initialization for the callback stuff
    # Fetch data from the driver in a loop, copying it out of the registered buffers and into our complete one.
    while nextSample < totalSamples and not autoStopOuter:
        wasCalledBack = False
        status["getStreamingLastestValues"] = ps.ps4000GetStreamingLatestValues(chandle, cFuncPtr, None)
        if not wasCalledBack:
            # If we weren't called back by the driver, this means no data is ready. Sleep for a short while before trying
            # again.
            time.sleep(0.01)
            if keyboard.is_pressed('esc'):  # if key 'q' is pressed
                print('You Pressed esc!')
                mystop=True
                break  # finishing the loop
    stop=time.time()    # read clock again
    my_counter +=1      # increment index
    #print("Done grabbing samples, in:%5.3f s" % (stop-start))

    if mystop:
        my_counter +=100000 # make sure we stop now

    matfile='logfiles/loopdata_%04d.mat' % my_counter
    print('dumping scopedata to mat file %s' % matfile)
    start=time.time()   # read the system clock (Unix time, nuber of seconds since january 1, 1970)
    start_timestr=time.asctime() # human readable string
    # save to matfile
    scipy.io.savemat(matfile, mdict={'a': bufferCompleteA, 'sample_int_ns': actualSampleIntervalNs,'range': range[channel_range], 't_stamp': start,'t_string': start_timestr})
    # the data saved to disk are in 16bit signed integer format.
    # to convert from sample count to floating point voltage:
    # take the signed integer value, multiply by voltage range (for example 2 for 2volts, 1 for 1 volt and so on),
    # and divide by 32767.
    #
    toc=time.time()
    print(toc-tic)
    #plt.cla()
    #plt.plot(bufferCompleteA[0:1000])
    #plt.title('QA plot, 1000 first samples of 5000000 samples')
    #plt.pause(0.05)

    # never collect more than 30 minutes of data
    if my_counter>30*60/FileSize*1000:
        #if my_counter>10:
        break

    # this could have been more clever ......
    print('press ESC key to exit acquisition loop')

# Stop the scope
# handle = chandle
status["stop"] = ps.ps4000Stop(chandle)
assert_pico_ok(status["stop"])

# Disconnect the scope
# handle = chandle
status["close"] = ps.ps4000CloseUnit(chandle)
assert_pico_ok(status["close"])
```

```

# Display status returns
print(status)
```

Listing 1: Python Sand Acquisition Program

15 Appendix 3 - Python Sand Analysis Program

```
##Sand Production estimation based on Acoustic Emission
import numpy as np
import matplotlib.pyplot as plt
import scipy.io
import scipy.signal
import tkinter as tk
import glob
import os.path
import time
from IPython.display import display, clear_output
import threading


####################PARAMETERS TO CHANGE
FlowRate = 345 #l/min, fluid flow rate
Sand = 780 #microns, Assumed sand grain diameter
fs = 1250000 #Hz, sampling frequency
lengthSamp = 1 #seconds, length of one recording
UpdateRate = 1 #seconds, data update frequency
M = 500 #number of points in one periodogram
zero1 = 0 #zero values around 150 kHz
zero3 = 0 #zero values around 405 kHz
Vrange=2 #Oscillascope voltage is from -2 to 2 V for low frequency Vallen
####################

#Approximate pump  velocities
#  9Hz=170  l/min     10Hz=200 l/min     11Hz=230 l/min
#  12Hz=260 l/min     13Hz=290 l/min     14Hz=320 l/min
#  15Hz=345 l/min     16Hz=370 l/min     17Hz=400 l/min
#  18Hz=440 l/min     -------- l/min     -------- l/min

#1.1 Loop preparation
l = UpdateRate / lengthSamp
SandSizeAv = np.zeros((int(l),1))
SandProdAv = np.zeros((int(l),1))
PAV1 = np.zeros((int(l),1))
PAV3 = np.zeros((int(l),1))

SP=np.array(0)
SZ = np.array(0)
```

```
POW1 = np.array(0)
POW3 = np.array(0)
SandEst = np.array(0)

#1.2 Loading of filter coefficient
###scipy.io.loadmat('FilterCoefficients1250.mat')
#dataFilter=np.load('/home/pi/Desktop/PythonProcessing/Filter1250kHz.npz
    ')
#Num = dataFilter['ba']
#Num=Num[0]

############################################
#2.1 Creating control buttons
global is_on_work
is_on_work = True
global is_on_zero
is_on_zero = False
global is_on_calib
is_on_calib = False

class App(threading.Thread):
    def __init__(self):
        threading.Thread.__init__(self)
        self.start()

    def run(self):
        self.root = tk.Tk()
        self.root.title("Buttons for sand")
        self.root.geometry("500x300")

        def myClick1():
            global is_on_work
            if is_on_work:
                myButtom1.config(text="Stopped")
                is_on_work = False
            else:
                myButtom1.config(text="Click to stop")
                is_on_work = True

        def myClick2():
            global is_on_calib
            if is_on_calib:
                myButtom2.config(text="Calibrate Sand")
                is_on_calib = False
            else:
                myButtom2.config(text="Calibrating Sand")
                is_on_calib = True

        def myClick3():
            global is_on_zero
            if is_on_zero:
                myButtom3.config(text="Get Zeros")
                is_on_zero = False
            else:
                myButtom3.config(text="Getting zeros")
                is_on_zero = True

        myButtom1=tk.Button(self.root, text="Click to stop" , padx=150, pady=20, command=myClick1, bg="red")
        myButtom1.pack()
        myButtom2=tk.Button(self.root, text="Calibrate Sand" , padx=150, pady=20, command=myClick2, bg="gold")
        myButtom2.pack()
        myButtom3=tk.Button(self.root, text="Get Zeros" , padx=150, pady=20, command=myClick3, bg="silver")
        myButtom3.pack()
        self.root.mainloop()

App()
print('control app is launched')
############################################

#2.2 Preparation for plotting

fig1 = plt.figure()
fig2 = plt.figure()
fig3 = plt.figure()
fig4 = plt.figure()
ax1 = fig1.add_subplot(111)
ax2 = fig2.add_subplot(111)
ax3 = fig3.add_subplot(111)
ax4 = fig4.add_subplot(111)


tic=time.time()
#2.2 Infinite loop for data processing

list_of_files = glob.glob(r'C:\Users\marti\Desktop\logfiles_grov5\*') #
```

```
   * means all if specific format then *.mat
latest_file = max(list_of_files, key=os.path.getctime)
list_of_files.remove(latest_file)

list_of_new_files = list_of_files

print("List of files:", list_of_new_files)


while True:

    print('working')
    if not is_on_work:
        print('Stopped by user')
        break

    # 2.2.1 Loop to average and update main monitoring values
    for j in np.arange(start=0, stop=1, step=1):

        first_file = min(list_of_new_files, key=os.path.getctime)

        loadMat=scipy.io.loadmat(first_file)
        a = loadMat['a']
        x = a * Vrange / (2 ** 15)
        #xfilt = scipy.signal.lfilter(Num, 1, x)

        xfilt=x; #no filtering
        N = a.shape[1]
        #M=N/K; #length of one periodogram for Bartlett
        K = int(N / M)
        r = int(M / 2)

        Xb=np.reshape(xfilt,(M,K), 'F')
        Xb=Xb.astype(float)

        Sxx_B=np.mean(np.transpose(1/M*pow(abs(np.fft.fft(Xb, axis=0)),
   2)), axis=0)
        S=Sxx_B[0:int(r)]

        #Plotting
        fb = np.arange(0, 0.5, 0.5/r)

        ax4.cla()
        ax4.plot(fb*fs/1000, S, color= "green")
        ax4.set_xlabel('Frequency, kHz')
        ax4.set_ylabel('Amplitude Bartlett')
        ax4.set_title('density spectrum')
        clear_output(wait = True)


        #2.2.3 Frequency bands analysis
        #frequency gaps
        p11 = 140000
        p12 = 160000
        p31 = 400000
        p32 = 410000

        #transfering to index numbers
        dev = fs / 2 / S.shape[0]
        p11 = int(p11 / dev)
        p12 = int(p12 / dev)
        p31 = int(p31 / dev)
        p32 = int(p32 / dev)
        #taking power values in zones of interest with scaling for PKI15 sensor
        power1 = np.mean(S[p11:p12+1]) * 10 ** 3
        power3 = np.mean(S[p31:p32+1]) * 10 ** 5
        #Substracting zero level
        power1 = power1 - zero1
        power3 = power3 - zero3


        #2.2.5 Amplification coefficient
        #Here we are setting main point on simplified amplification curve
        P4 = 1000
        P3 = 550
        P2 = 350
        P1 = 180
        if Sand >= P4:
            amp = 5
        else:
            if Sand >= P3:
                amp = 1 + (5 - 1) / (P4 - P3) * (Sand - P3)
            else:
                if Sand >= P2:
                    amp = 1
                else:
                    if Sand >= P1:
                        amp = 5 + (1 - 5) / (P2 - P1) * (Sand - P1)
                    else:
                        amp = 5
        #2.2.6. Power of the scaled signal and Sand Production
        #We update power of the signal with amplification coefficient
        POWER = power3 * amp * 1.1
        slope = - 3e-12 * Sand ** 3 + 3e-09 * Sand ** 2 + 2e-06 * Sand + 0.0033
        displacement = 4e-07 * Sand ** 2 - 8e-05 * Sand + 2.0294
        GSE = 10 ** (slope * FlowRate - displacement)
        SandProd = POWER / lengthSamp / GSE
        #2.2.6. Updating main values and time check
        SandSizeAv[int(j)] = Sand
        SandProdAv[int(j)] = SandProd
        PAV1[int(j)] = power1
        PAV3[int(j)] = power3
        j = j + 1

        print(first_file)
        print("Number of files found:", len(list_of_files))

        list_of_new_files.remove(min(list_of_new_files, key=os.path.getctime))

    #2.2.7 Array of data creation
    j = 0
    sandsize = np.mean(SandSizeAv)
    sandproduction = np.mean(SandProdAv)

    SP=np.append(SP, sandproduction)
    SZ=np.append(SZ, sandsize)
    POW1=np.append(POW1, np.mean(PAV1))
    POW3=np.append(POW3, np.mean(PAV3))

    toc=time.time()
    timeprocess = toc-tic

    if timeprocess < UpdateRate:
        timeSleep=UpdateRate - timeprocess
    else:
        timeSleep=0

    #Enter when you want the Sand production to count from and end
    t_start = 15
    t_end = 89

    #plotting
    ax1.cla()
    ax1.plot(SP[t_start:t_end], color= "brown")
    ax1.set_xlabel('seconds from the start')
    ax1.set_ylabel('Sand rate g/s')
    ax1.set_title('Sand production')
    clear_output(wait = True)

    ax2.cla()
    ax2.plot(POW1, color= "red")
    ax2.set_xlabel('seconds from the start')
    ax2.set_ylabel('Power around 150 kHz')
    ax2.set_title('Power around 150 kHz')
    clear_output(wait = True)

    ax3.cla()
    ax3.plot(POW3, color= "black")
    ax3.set_xlabel('seconds from the start')
    ax3.set_ylabel('Power around 405 kHz')
    ax3.set_title('Power around 405 kHz')
    clear_output(wait = True)

    plt.pause(timeSleep)
    print(timeprocess)
    print("Accumulated sand production:", sum(SP[t_start:t_end]))

    #print("Sand production period:", sum(SP[32:55]))

    if is_on_zero:
        print('Getting zero values by user comand')
        zero1 = zero1 + POW1[-1] / 2
        zero3 = zero3 + POW3[-1] / 2

    if is_on_calib:
        print('Calibration of grain diameter')
        GrainSearch = POW1[-1] / POW3[-1]
        Sand225 = 232.67 * np.log(GrainSearch) + 369
        Sand325 = 239.53 * np.log(GrainSearch) + 451.64
        Sand420 = 227.45 * np.log(GrainSearch) + 479.36
        if FlowRate >= 325:
            SandEst = Sand325 + (Sand420 - Sand325) / (420 - 325) * (
FlowRate - 325)
        else:
            SandEst = Sand225 + (Sand325 - Sand225) / (325 - 225) * (
FlowRate - 225)
        Sand = (8 * Sand + 2 * SandEst) / 10
        print(Sand)

 tic=time.time()
```

Listing 2: Python Sand Analysis Program

16 Appendix 4 - Python Plot Compiler

```python
import numpy as np
import matplotlib.pyplot as plt
import scipy.io
import scipy.signal
import tkinter as tk
import glob
import os.path
import time
from IPython.display import display, clear_output
import threading

from SandProcessing_new1 import SP as SP_1, t_start as t_start_1, t_end as t_end_1
from SandProcessing_new2 import SP as SP_2, t_start as t_start_2, t_end as t_end_2
from SandProcessing_new3 import SP as SP_3, t_start as t_start_3, t_end as t_end_3
from SandProcessing_new4 import SP as SP_4, t_start as t_start_4, t_end as t_end_4
from SandProcessing_new5 import SP as SP_5, t_start as t_start_5, t_end as t_end_5

# Create a figure and axis
fig, ax = plt.subplots()

# Plot the first graph
ax.plot(SP_1[t_start_1:t_end_1], label='Test 1')

# Plot the second graph
ax.plot(SP_2[t_start_2:t_end_2], label='Test 2')

# Plot the third graph
ax.plot(SP_3[t_start_3:t_end_3], label='Test 3')

# Plot the fourth graph
ax.plot(SP_4[t_start_4:t_end_4], label='Test 4')

# Plot the fifth graph
ax.plot(SP_5[t_start_5:t_end_5], label='Test 5')

# Add a legend
```

```python
ax.legend()

# Add labels and title
ax.set_xlabel('Seconds from the start')
ax.set_ylabel('Sand rate g/s')
ax.set_title('Combined Graph, Sand Rates')

# Display the graph
plt.show()
```

Listing 3: Python Sand Analysis Program

17 Appendix 4 - Filter Coefficients from MATLAB

The FIR coefficients were calculated in MATLAB, with the use of the tool “filterDesigner”.

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
-0.00146	-0.00466	0.00401	0.01637	0.00048	-0.01469	-0.00272	-0.00183	-0.00648	0.00474
11	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
0.00269	0.00081	0.00832	0.00111	0.00103	0.00397	-0.00586	-0.00191	-0.00235	-0.00920
21	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
0.00038	-0.00246	-0.00303	0.00854	0.00116	0.00528	0.01104	-0.00217	0.00472	0.00088
31	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**
-0.01212	-0.00037	-0.01026	-0.01257	0.00437	-0.00797	0.00375	0.01635	-0.00022	0.01843
41	**42**	**43**	**44**	**45**	**46**	**47**	**48**	**49**	**50**
0.01300	-0.00671	0.01242	-0.01291	-0.02140	-0.00036	-0.03252	-0.01128	0.00879	-0.01876
51	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**
0.03254	0.02909	0.00530	0.06389	0.00423	-0.01024	0.03199	-0.10133	-0.05704	-0.04397
61	**62**	**63**	**64**	**65**	**66**	**67**	**68**	**69**	**70**
-0.28585	0.09448	0.58220	0.09448	-0.28585	-0.04397	-0.05704	-0.10133	0.03199	-0.01024
71	**72**	**73**	**74**	**75**	**76**	**77**	**78**	**79**	**80**
0.00423	0.06389	0.00530	0.02909	0.03254	-0.01876	0.00879	-0.01128	-0.03252	-0.00036
81	**82**	**83**	**84**	**85**	**86**	**87**	**88**	**89**	**90**
-0.02140	-0.01291	0.01242	-0.00671	0.01300	0.01843	-0.00022	0.01635	0.00375	-0.00797
91	**92**	**93**	**94**	**95**	**96**	**97**	**98**	**99**	**100**
0.00437	-0.01257	-0.01026	-0.00037	-0.01212	0.00088	0.00472	-0.00217	0.01104	0.00528
101	**102**	**103**	**104**	**105**	**106**	**107**	**108**	**109**	**110**
0.00116	0.00854	-0.00303	-0.00246	0.000384	-0.00920	-0.00235	-0.00191	-0.00586	0.00397
111	**112**	**113**	**114**	**115**	**116**	**117**	**118**	**119**	**120**
0.00103	0.00111	0.00832	0.00081	0.00269	0.00474	-0.00648	-0.00183	-0.00272	-0.01469
121	**122**	**123**	**124**	**125**					
0.00048	0.01637	0.00401	-0.00466	-0.00146					

Table 9: Filter coefficients, from MATLAB's "FilterDesigner"

NTNU

Norwegian University of
Science and Technology