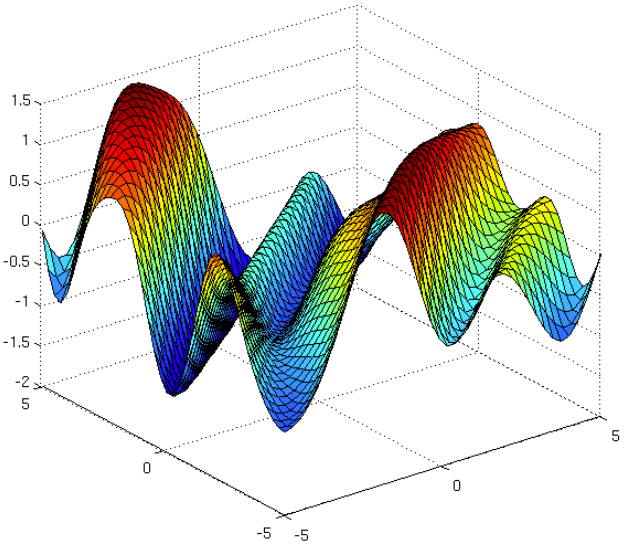
Gaussian Process Regression Models for Predicting Water Retention Curves

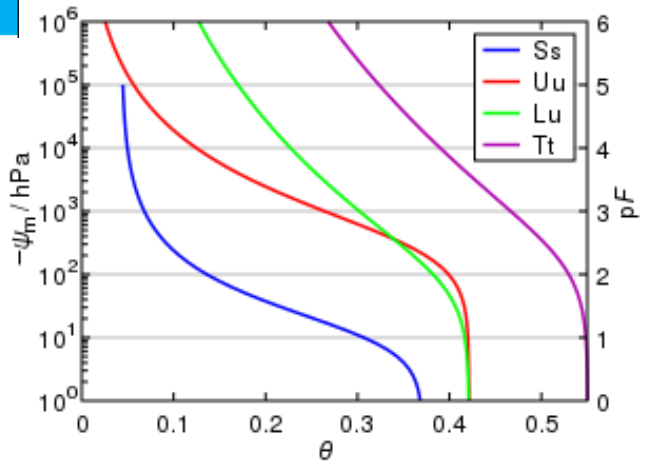
Application of Machine Learning Techniques for Modelling Uncertainty in Hydraulic Curves

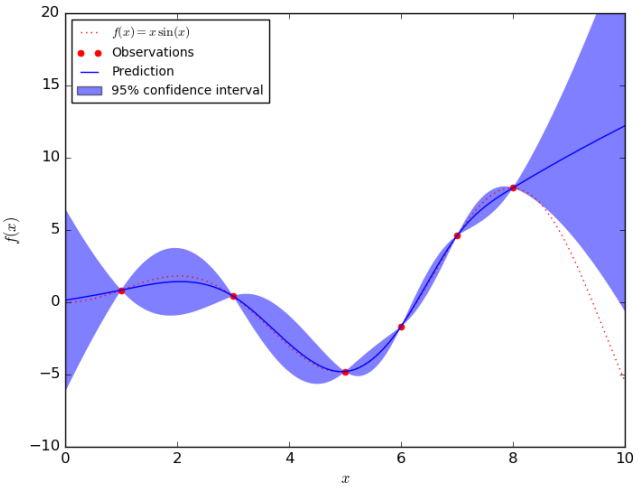
Burkan Yousef

CTB3000 – BSc Thesis

Delft University of Technology







TU Delft Delft University of Technology

Gaussian Process Regression Models for Predicting Water Retention Curves

Application of Machine Learning Techniques for Modelling Uncertainty in Hydraulic Curves

by

Burkan Yousef

CTB3000 - BSc Thesis Civil Engineering

Student number: 4387716

Supervisors:

1st:	Dr.ir. Gerrit Schoups
2nd:	Dr. Ronald van Nooyen

Faculty of Civil Engineering
Department of Water Management Research

April 30, 2019 / June 17, 2019

TU Delft

Abstract

An accurate representation of water retention curves is important for various reasons. Traditional models already exist for the representation of these curves, with one of them being the van Genuchten model. When soil parameters are available, the van Genuchten model can be used to plot water retention curves. However, when these soil parameters are not available, regression can be performed to estimate and predict the water retention curves. Commonly, the Non-Linear Least Squares regression method is used in combination with a certain water retention model. Problems arise for inhomogeneous soils as the traditional water retention models tend to break down.

To improve the representation of water retention curves, Gaussian Process regression will be implemented. This method will be combined with the Non-Linear Least Squares method to obtain new representations of water retention curves. These new curves are better in terms of curve fit and uncertainty, when compared to the traditional method. These comparisons can be made visually, by observing the plots and their confidence intervals, as well as quantitatively by computing the log-likelihoods of the different methods. When comparing the results of the log-likelihood computations for both methods, it follows that the value of the log-likelihood is greater for water retention data with correlated residuals. In the case where the residuals are uncorrelated, the log-likelihoods are equal for both methods and no improvements are observed.

Preface

Before you lies the thesis "Gaussian Process Regression Models for Predicting Water Retention Curves." It has been written to fulfil the graduation requirements of the Bachelor of Science Civil Engineering educational program at Delft University of Technology. My engagement in the researching- and writing process, lasted from April 30 to June 17. This research topic is consistent with my interests in mathematics and specifically regression analysis. I hope to learn more in this field during my masters in Econometrics, which I hope to start this September.

I would like to thank my supervisor Dr. Ir. Schoups, for his excellent guidance and critical feedback throughout this journey. His lectures on Water System Analysis have been of great help for understanding this topic. I would also like to thank my second supervisor Dr. Ronald van Nooyen, for his feedback and for his time on the co-assessment of this thesis report.

I hope you enjoy your reading.

Burkan Yousef

April 30, 2019

Contents

Abstract v

Preface vi

1 Introduction 1

1.1. Soil Moisture Content ... 2
1.2. Water Retention Curves ... 2
1.3. Conventional Methods ... 2
1.4. Non-Parametric Regression ... 3
1.5. Research Question ... 3

2 Water Retention Curves 5

2.1. Water Retention Models ... 6
2.1.1. Brooks-Corey Model ... 7
2.1.2. Van Genuchten Model ... 9
2.2. UNSODA ... 10

3 Non-Linear Regression 13

3.1. Non-linear Least-Squares ... 14
3.2. RETC ... 15
3.3. Optimize Curve Fit in Python ... 16

4 Gaussian Process Regression 17

4.1. Gaussian Process Theory ... 18
4.2. GPy Machine Learning Algorithm ... 23
4.3. GPy for Water Retention Curves ... 24

5 Results 25

5.1. Used Datasets ... 26
5.2. RETC Results ... 27
5.3. SciPy Curve Fit Results ... 29
5.4. GPy Results ... 31
5.5. Comparing Results from Least Squares and GPy ... 33

6 Discussion 37

7 Conclusion 39

A SciPy Curve Fit 41

B GPy for Gaussian Process Regression 46

C Log-likelihood Python Code 53

Bibliography 55

1

Introduction

Being able to determine the soil moisture content along the soil depth and understanding the soil water flow in various soils is key in improving the efficiency and performance in processes related to water management. Soil data is applied in useful and important processes, such as determining the optimal irrigation frequency as well as the irrigation amount, which allows for efficient crop growth. Another important application is the analysis of soil aggregate stability, where the resisting ability of soil aggregates is evaluated when exposed to external forces.

1.1. Soil Moisture Content

The soil moisture content, also denoted as θ, for a particular soil is determined by the volume of water that is contained in a soil. It is defined as the volume of the water in a soil sample over the total volume of the soil sample. The soil moisture content is closely related to the suction pressure that occurs in the soil pores and can have values between 0 and 1. In normal circumstances, water is attracted by soil particles and sucked into the soil pores. This creates a suction pressure as water is held by these pores against gravity. This phenomenon is called capillarity and is caused by adhesive as well as cohesive forces (Schoups, 2019). The relation between soil moisture content and suction pressure, characterises the water retaining ability of a soil layer. To empty a pore, a pressure needs to be applied that is greater than the capillary force. This again changes the moisture content of the particular soil. When a soil is fully saturated with water, the capillary force is equal to zero. However, when the pores are emptied, and the soil becomes drier, a larger suction occurs, meaning the capillary force increases.

1.2. Water Retention Curves

The relation between the water pressure and the moisture content can be represented through water retention curves, where the water pressure can be plotted against the moisture content. A conventional method to represent water retention curves, is by plotting the water pressure logarithmically instead of linearly. The y-axis ticks are then the pF values of the soil which is equal to the logarithm to base 10 of the water pressures in centimetres. pF curves can also be plotted with the x-axis and y-axis reversed. This gives a pF curve with the moisture content on the vertical axis and the logarithmic water pressure on the horizontal axis.

1.3. Conventional Methods

There are various methods to represent water retention curves. Conventional parametric models are the van Genuchten model and the Brooks-Corey model, in which characteristic soil parameters are used as the input of the model and the moisture content is the output (van Genuchten, Leij, & Yates, 1991). When these soil parameters are not known, it is not possible to obtain the moisture content directly and an alternative approach has to be implemented to represent the water retention curve. This alternative approach represents the water retention curves by performing traditional parametric non-linear regression analysis on the model through measured data points.

A useful tool that makes use of this principle, is the RETC (RETention Curve) computer program which analyses and predicts the soil water retention and the hydraulic properties for unsaturated soils (van Genuchten, Leij, & Yates, 1991). With RETC, a choice can be made between multiple models which can be used to perform an analysis. These parametric models are the Brooks-Corey model (1964), the van Genuchten model (1990), the lognormal distribution model of Kosugi (1996) and the dual-permeability model of Durner (1994). The RETC program takes known datapoints as input in order to perform its parametric non-linear regression analysis. Next to the water retention curve, the RETC program allows for the computation of the theoretical pore-size distribution models of Mualem (1976) and Burdine (1953), in order to predict the unsaturated hydraulic conductivity from observed soil water retention data (van Genuchten, Simunek, Leij, & Sejna, 1998).

While these conventional approaches work fairly well for homogeneous soils, it breaks down if the soil has a more composite structure as is often the case. Another common issue is the lack of data points where non-linear regression can be implemented on, which results in less accurate water retention curves. However, the main problem with the parametric non-linear regression method, is the lack of flexibility of the water retention curve fit. It forces the curve fit to be in a certain shape and therefore might lead to systematic errors when the plotted water retention data deviates from the used parametric model. These issues call for a more reliable and robust method for predicting water retention curves.

1.4. Non-Parametric Regression

In order to make better predictions and representations of water retention curves in less trivial soils, a new method needs to be investigated. This new method needs to have the property of being flexible to make better predictions of new data points. A regression method that satisfies this demand is non-parametric regression analysis. In contrast to parametric regression analysis, non-parametric regression analysis is based on either being distribution free or on having the distribution's parameters unspecified.

Gaussian Process regression is a well-known example of non-parametric regression. It is an interpolation method for which the values, that are interpolated between known values, are modelled by a Gaussian Process. This method is more commonly applied in situations where only little is known and fewer assumptions can be made, which makes it a more robust model (Winterstein, 2016).

Gaussian Processes have been applied in many branches of science. It was originally implemented in geostatistics, where observations for some spatially related points, either in 2D or in 3D, were collected and the values for the remaining unknown points are estimated. This process is also called kriging (Snelson, 2006). In the finance field, the Gaussian Process is used in combination with the Brownian Motion concept to model and predict financial markets (Anonymous, 2019). Lastly, the Gaussian process is commonly used in Machine Learning applications as has been researched prominently by Williams and Rasmussen (Rasmussen & Williams, 2006).

1.5. Research Question

As the Gaussian Process regression technique has both the properties of being robust and flexible, it might serve as a good solution to the water retention curve problem that occurs for composite soils with traditional parametric non-linear regression. This leads to the following research question:

> *Can the representation of water retention curves be improved with the help of Gaussian Processes, in terms of curve fitting and uncertainty analysis?*

The goal of this thesis is to assess the value of the more flexible non-parametric Gaussian Process Model for water retention curves. This will be done by representing water retention curves with the help of a Gaussian Process model and compare it to the traditional parametric soil water flow model from van Genuchten.

In Chapter 2, water retention curves will be discussed together with widely used models and data sources for the representation of these curves. In Chapter 3, traditional non-linear regression using the Least Squares method will be explained. Furthermore, different software possibilities, such as the RETC program and the Optimize.curve_fit

function of SciPy, will be discussed to obtain water retention curves, based on certain models, through the traditional non-linear regression. Chapter 4 will explain the use of the Gaussian Process model as a way to generate an optimal fit through the observations. In Chapter 5, the regression methods discussed in Chapters 3 & 4 will be applied on water retention data in order to obtain water retention curves. These methods and results will also be compared to each other in order to determine the added value of the Gaussian Process method on modelling water retention curves. In chapter 6 a general discussion on the research and results is presented and finally, a conclusion is drawn in Chapter 7.

2

Water Retention Curves

Water retention curves, or pF curves, allow for clear visualization and understanding of the relation between the soil moisture content and the corresponding water pressure or hydraulic head as shown in Figure 2.1. Over the years, different models have been introduced for the computation and representation of these pF curves. In this chapter, the most important and relevant models will be presented and explained further in detail. Also, the data source that is used for the analysis of water retention curves will be discussed along with its properties.

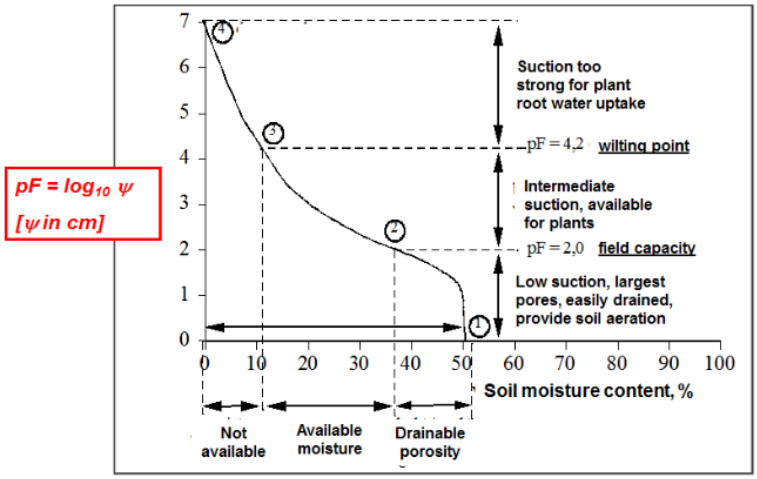


Figure 2.1: *Example of a water retention curve, also known as a pF curve, for a certain soil with the logarithm to base 10 of the water pressures on the vertical axis and the moisture content on the horizontal axis. Different pF values have different effects on the water uptake of plants. Generally, a pF value of 4.2 corresponds to the wilting point where the suction is too strong for the plant root to uptake water and a pF value of 2.0 corresponds to the field capacity which is the amount of soil moisture content held in the soil after excess water has drained away and the rate of downward movement has decreased. Source:* (Schoups, 2019)

2.1. Water Retention Models

Various water retention models already exist, including the Brooks-Corey model (1964), the van Genuchten model (1990), the lognormal distribution model of Kosugi (1996) and the dual-permeability model of Durner (1994) (van Genuchten, Simunek, Leij, & Sejna, 1998). Figure 2.2 presents the water retention curves for different models. This shows that models differ in smoothness of the curves as well as in accuracy and fit. In this thesis, only the van Genuchten model and the Brooks-Corey model will be discussed, as they are generally used the most in real life applications due to their simplicity and attractive mathematical characteristics (van Genuchten, Leij, & Yates, 1991).

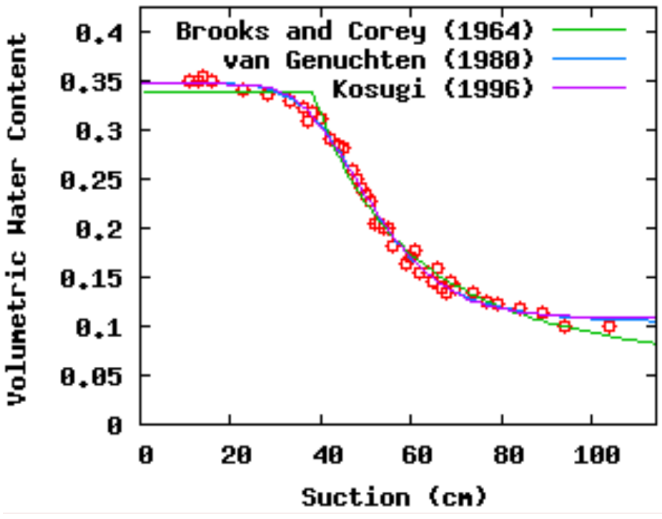


Figure 2.2: *Different models fitted to measured water retention data. The red circles represent the measurements. Source:* (Seki, 2007)

2.1.1. Brooks-Corey Model

The Brooks-Corey model (1964), is one of the most popular models for representing water retention curves. It describes the course of the moisture content as a function of the water pressure head in a soil.

$$\theta = \begin{cases} \theta_r + (\theta_s - \theta_r)(\alpha|h|)^{-\lambda}, & (\alpha h > 1) \\ \theta_s, & (\alpha h \leq 1) \end{cases} \quad (1)$$

In equation 1, the Brooks-Corey function is given. The Brooks-Corey function is a conditional function, meaning its function value depends on the value of the product of $\boldsymbol{\alpha}$ and $\boldsymbol{h}$, where $\boldsymbol{\alpha}$ is an empirical parameter with its unit defined as one over the length or (cm^{-1}) and $\boldsymbol{h}$ denotes the soil water pressure head in cm. The inverse of $\boldsymbol{\alpha}$ is equal to the air entry value, which is the suction value that must be exceeded before air recedes into the soil pores. Furthermore, the residual moisture content is denoted by $\boldsymbol{\theta_r}$. It specifies the maximum amount of water in a soil that will not contribute to liquid flow because of blockage from the flow paths or strong adsorption onto the solid phase (Luckner, van Genuchten, & Nielsen, 1989). It is the moisture content in a soil at which the change in moisture content approaches zero due to the increase of the water pressure head $\boldsymbol{h}$. The residual moisture content is not necessarily the smallest possible moisture content in a soil, as the soil can dry out to moisture contents lower than the residual moisture content in extremely dry regions. The saturated moisture content, or satiated moisture content, is denoted by $\boldsymbol{\theta_s}$. This is defined as the maximum soil moisture content. This maximum soil moisture content is 5% – 10% smaller than the soil porosity, due to trapped air in the soil pores that is unable to escape. Finally, $\boldsymbol{\lambda}$ is the pore-size distribution parameter. This parameter influences the shape of the water retention curve by affecting the slope of the curve.

Figure 2.3 shows the mean values of Brooks-Corey soil water retention parameters for different types of soils. It can be noticed that these soil types have different pore-size distribution index values. This is reflected in the water retention curves for different soils as can be seen in Figure 2.4. Initially, the soil sample is fully saturated and the moisture content is at its maximum for its particular soil type. As the pressure increases, the moisture content decreases. However, the rate at which the moisture content decreases is higher for sand than for clay loam. This is due to the fact that the pore-size distribution index value for sand is greater than for clay.

Soil Texture	Porosity	Hydraulic Conductivity (m/d)	Irreducible Water Content	Displacement Pressure Head* (m)	Pore Size Distribution Index
Clay	0.38 (0.09)	0.048 (0.10)	0.068 (0.034)	1.25 (1.88)	0.09 (0.09)
Clay loam	0.41 (0.09)	0.062 (0.17)	0.095 (0.010)	0.53 (0.42)	0.31 (0.09)
Loam	0.43 (0.10)	0.25 (0.44)	0.078(0.013)	0.28 (0.16)	0.56 (0.11)
Loamy sand	0.41 (0.09)	3.5 (2.7)	0.057 (0.015)	0.081 (0.028)	1.28 (0.27)
Silt	0.46 (0.11)	0.060 (0.079)	0.034 (0.010)	0.62 (0.27)	0.37 (0.05)
Silty loam	0.45 (0.08)	0.11 (0.30)	0.067 (0.015)	0.50 (0.30)	0.41 (0.12)
Silty clay	0.36 (0.07)	0.0048 (0.0260)	0.070 (0.023)	2.0 (2.0)	0.09 (0.06)
Silty clay loam	0.43 (0.07)	0.017 (0.046)	0.089 (0.009)	1.0 (0.6)	0.23 (0.06)
Sand	0.43 (0.07)	7.1 (3.7)	0.045 (0.010)	0.069 (0.014)	1.68 (0.29)
Sandy clay	0.38 (0.05)	0.029 (0.067)	0.100 (0.013)	0.37 (0.23)	0.23 (0.10)
Sandy clay loam	0.39 (0.07)	0.31 (0.66)	0.100 (0.006)	0.17 (0.11)	0.48 (0.13)
Sandy loam	0.41 (0.09)	1.1 (1.4)	0.065 (0.017)	0.13 (0.066)	0.89 (0.17)

Figure 2.3: *Brooks-Corey soil water retention parameters for different soil types. These parameters include porosity, hydraulic conductivity, residual moisture content, air-entry pressure or displacement pressure and pore-size distribution index. Source:* (Schoups, 2019)

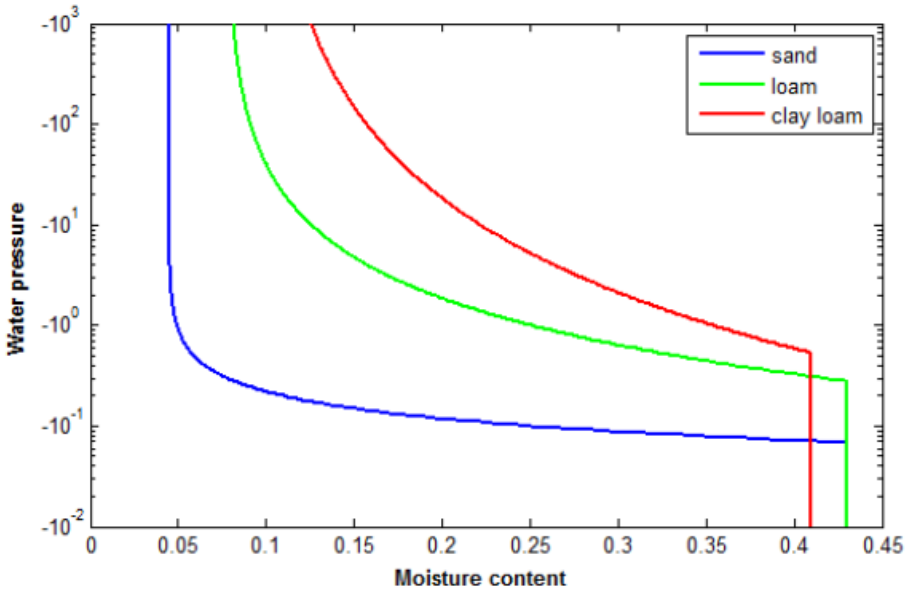


Figure 2.4: *Brooks-Corey model for water retention curves for sand, loam and clay loam. The water pressure is plotted logarithmically. The moisture content only starts to decrease when the water pressure exceeds the suction pressure of the soils. The value of the residual moisture content of a particular soil is approximately equal to the value of the vertical asymptote of the water retention curve. This shows that when the residual moisture content is reached, it remains fairly constant even though pressure is increased. Source:* (Schoups, 2019)

2.1.2. Van Genuchten Model

The van Genuchten model came many years after the Brooks-Corey model and describes the same phenomena only in a different form and with some different parameters. The main reason for a change in the equation form is to improve the representation of water retention curves when they near saturation. The van Genuchten equation can be considered a smooth function, meaning it can be continuously differentiated. Many other models have been proposed as well, but they were mathematically too complicated to be implemented in predictive pore-size distribution models for the hydraulic conductivity. The attractive mathematical properties of the van Genuchten model, together with the relatively smoothness of the function, is what makes the van Genuchten model a valuable model that has been used widely. An important difference with the Brooks-Corey model is that it has only one expression for the moisture content, unlike the conditional function that is the Brooks-Corey function. For the analysis of the different regression methods in this thesis, only the van Genuchten model will be considered for parametric non-linear regression.

In general, the effective degree of saturation or reduced moisture content, is the proportion of the prevailing suction that actually contributes to the effective stress. This can be expressed as in equation 2.

$$S_e = \frac{\theta - \theta_r}{\theta_s - \theta_r} \tag{2}$$

with S_e being the effective degree of saturation. The van Genuchten model gives another expression for the effective degree of saturation as can be seen in equation 3.

$$S_e = \frac{1}{\left[1 + (\alpha|h|)^n\right]^m} \tag{3}$$

By inserting equation 2 to into equation 3, an expression is derived for the moisture content as a function of the water pressure head, shown in equation 4. This expression is known as the van Genuchten equation.

$$\theta(h) = \theta_r + \frac{\theta_s - \theta_r}{\left[1 + (\alpha|h|)^n\right]^m} \tag{4}$$

The parameters α, **n** & m are empirical constants that influence the shape of the water retention curve. The constants **n** & m can be related to each other for simplicity reasons. Different models have been set up to describe the relation between the two parameters. Therefore, the relation between **n** & m, depends on which conductivity model is being used. The Burdine conductivity model (1953) presents the relation as $m = 1 - 2/n$ while the Mualem conductivity model (1976) describes this as $m = 1 - 1/n$. The Mualem conductivity model has proved itself to perform the best for most soils. For this reason, the Mualem conductivity model will be applied in further analysis of the van Genuchten model. Ultimately, by applying the Mualem conductivity model to the van Genuchten

water retention model, the modified van Genuchten equation is generated as can be seen in equation 5 (van Genuchten, Leij, & Yates, 1991).

$$\theta(h) = \theta_r + \frac{\theta_s - \theta_r}{\left[1 + (\alpha|h|)^n\right]^{1-1/n}} \quad (5)$$

where:

θ(h) *is the moisture content as a function of the water pressure head [-]*

$\boldsymbol{\theta_r}$ *is the residual moisture content [-]*

$\boldsymbol{\theta_s}$ *is the saturated moisture content [-]*

$\boldsymbol{\alpha}$ *is related to the inverse of the air entry suction [cm^{-1}], with* $\boldsymbol{\alpha} > 0$

h *is the water pressure head [cm], and*

n *is a measure of the pore-size distribution [-], with* $\boldsymbol{n} > 1$

2.2. UNSODA

Water retention curves can be represented by either knowing the soil parameters of a certain model, or by having some observed data which can be used to fit a model through. The UNsaturated SOil hydraulic DAtabase, or UNSODA, is a database that contains unsaturated hydraulic data and other soil properties. This database contains measured soil water retention, hydraulic conductivity, and water diffusivity data, as well as pedological information of some 790 soil samples from around the world (National Agriculture Library, 2015). The database can be accessed with Microsoft Access. It contains various relations between the unsaturated soil moisture content and another property, such as the conductivity, the diffusivity and the water pressure head. For this thesis, only the relation between the moisture content and the water pressure head will be considered, as only this relation is relevant for water retention curves.

A distinction can be made in field drying, field wetting, lab drying and lab wetting. Drying implies that the analysis on a particular soil sample is saturated initially and the measurements are done while draining the soil sample. Wetting involves the soil sample to be drained initially. Measurements are done while wetting the soil sample. This process can be executed in the lab under controlled conditions on a relatively small soil sample, or in the field on an actual soil layer (Rao, 2011).

Figure 2.5 shows a setup for determining the moisture content of a soil as a function of the pressure. When the pressure is increased, the moisture content in the soil decreases as air enters the soil pores and pushes the water out. The water is subsequently collected and measured at discrete pressure values. Every pressure value corresponds to a certain moisture content. This results in a dataset with a certain pressure head and a certain moisture content.

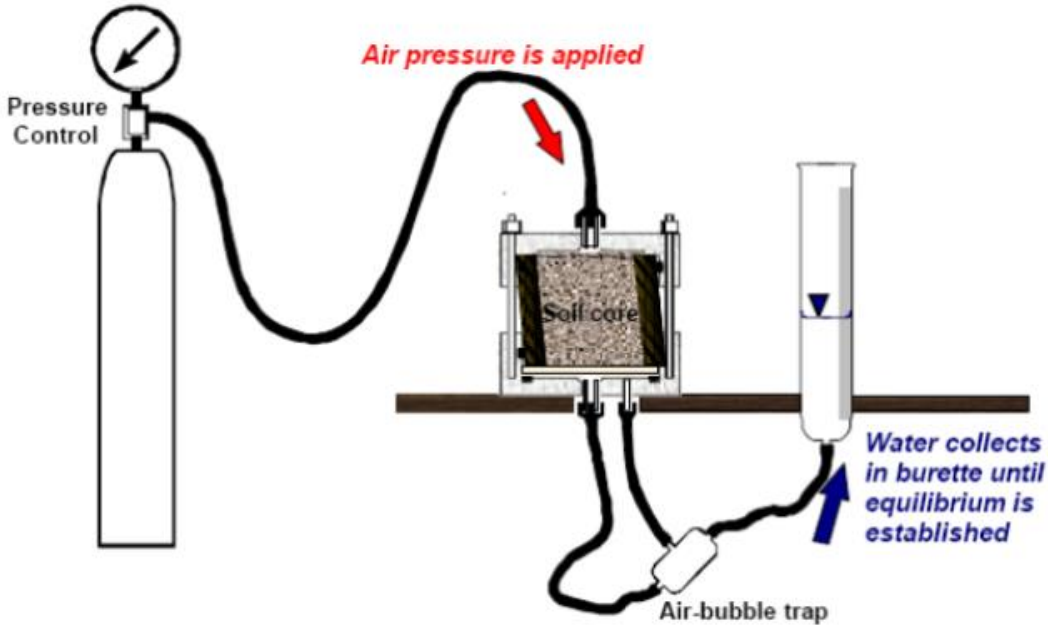


Figure 2.5: *Lab wetting setup. Air pressure is applied to a saturated soil sample. Water that is draining out of the soil sample is collected in a burette. Source:* (Schoups, 2019)

UNSODA contains a large number of (moisture content vs pressure head) measurements done under varying circumstances as previously mentioned. However, some measurement techniques have little or no data available. This is the case for the field wetting technique. No data is available for this technique. The field drying technique however contains 2621 datapoints. For measurements done under laboratory conditions, more datapoints are available. Lab wetting contains 528 datapoints while lab drying contains 8066 datapoints. A statistical analysis is presented in the form of a pie chart in Figure 2.6.

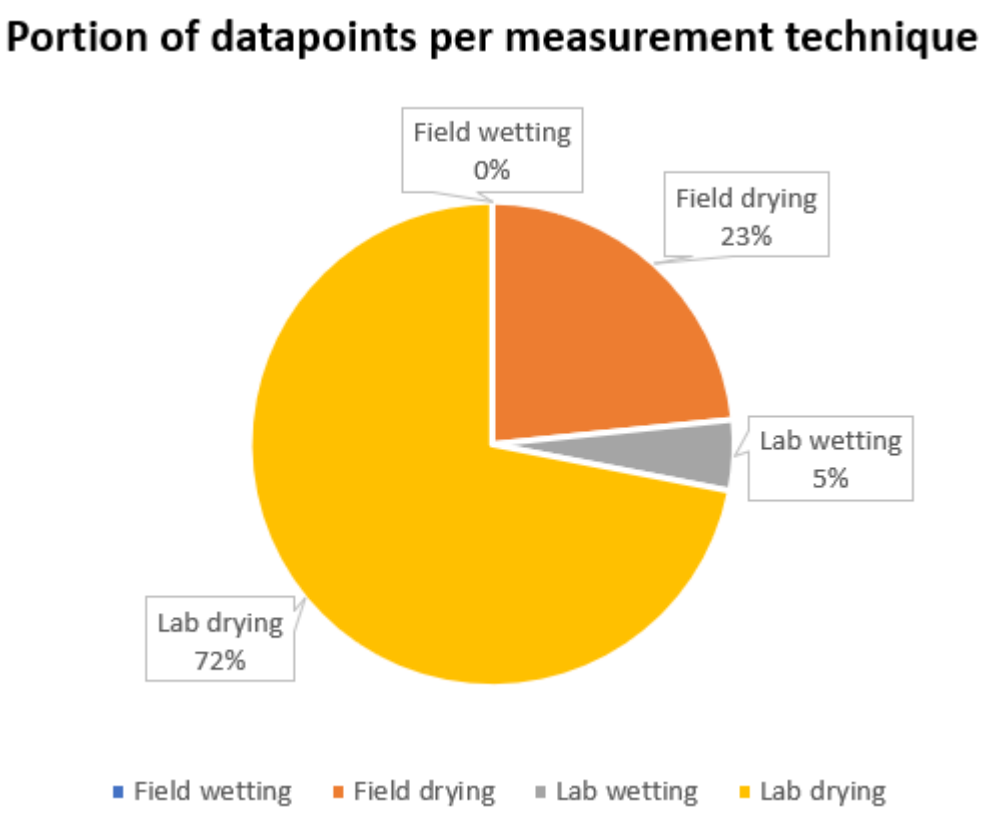


Figure 2.6: *Different measurement techniques with a corresponding portion of available datapoints. Each datapoint is a certain moisture content with a corresponding pressure head. Field wetting contains no data while lab drying contains a large amount of data. Retrieved from the UNSODA database.*

3

Non-Linear Regression

Non-linear regression is a common technique in representing water retention curves. Water retention curves can already be modelled by using the existing models of van Genuchten or Brooks-Corey. However, these models require soil hydraulic parameters as an input. As these soil parameters might not always be available, it is necessary that these models and their model parameters are estimated. This can be done by performing measurements on soil samples while draining or wetting the soil to obtain water retention data points. By applying non-linear regression to these measured water retention data points, the van Genuchten model and the Brooks-Corey model can be approximated by predicting the unknown soil parameters as accurate as possible.

3.1. Non-linear Least-Squares

The principle behind the RETC algorithm is the Non-Linear Least Squares regression method. This method can also be implemented in Python which is useful for the analysis of water retention curves with the use of Gaussian Process regression. This allows for a qualitative comparison between the Non-Linear Least Squares regression method and the Gaussian Process regression method.

Non-linear Least Squares is a form of Least Squares analysis which is used for fitting observations with a non-linear model. The Least Squares method is based on the principle of minimizing the sum of the squares of the residual, where the residuals are defined as the difference between the observed values and the fitted values of the model (Boyd, 2016). The residuals r_i can therefore be denoted as:

$$r_i = y_i - f(x_i, \boldsymbol{\beta}) \tag{6}$$

where x_i and y_i are the x and y value of datapoint i and $\boldsymbol{\beta}$ is the vector that contains the model parameters that need to be estimated. The Least Squares method minimizes the sum of these squared residual for every datapoint which means minimizing:

$$\sum_{i=1}^{n} {r_i}^2 = \sum_{i=1}^{n} |y_i - f(x_i, \boldsymbol{\beta})|^2 \tag{7}$$

For linear models, this minimization and therefore the parameter estimation can be performed by applying the closed form of the parameter estimation equation, which is given by equation 8 (Stansbury, 2012).

$$\widehat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T \boldsymbol{y} \tag{8}$$

where the estimated parameters vector is denoted by $\hat{\boldsymbol{\beta}}$, the $\boldsymbol{X}$ matrix denotes the design matrix and the $\boldsymbol{y}$ vector contains the observed values of the data points.

However, this closed form is not always applicable for non-linear models. The parameters can be estimated by starting with an initial guess for the parameters and iteratively adjusting the parameters while computing the sum of the residuals. The best parameters for the non-linear model are the parameters which give the smallest value for the sum of the squares. These parameters can subsequently be used to plot the optimal model through the datapoints.

The Non-Linear Least Squares regression method is based on some assumptions. The residuals (or errors) are normally distributed with:

1. $\boldsymbol{\mu = 0}$
2. $\boldsymbol{\sigma^2}$ **= constant**
3. ***no correlation between the errors***

where $\boldsymbol{\mu}$ is the mean or expected value and $\boldsymbol{\sigma^2}$ is the variance of the normal distribution. This can also be denoted as following:

$$X \sim \mathcal{N}(0, \boldsymbol{\sigma^2}) \tag{9}$$

where the residuals are denoted as the random variable X.

The variance σ^2 can be determined by calculating the Mean Squared Error or MSE. The MSE is defined as the mean of the squares of the residuals/errors. Equation 10 presents the mathematical from of the MSE.

$$\sigma^2 = \text{MSE} = \frac{1}{n}\sum_{i=1}^{n} {r_i}^2 = \frac{1}{n}\sum_{i=1}^{n} |y_i - f(x_i, \boldsymbol{\beta})|^2 \qquad (10)$$

When the MSE is computed, the 95% confidence interval can be computed and plotted with optimal fit as well . This interval is an estimate for the true value of a model. There is 95% certainty that a certain function value of the model is within that interval. For a normal distribution, the 95% interval approximates twice the standard deviation $\boldsymbol{\sigma}$, where the standard deviation is equal to the square root of the variance $\boldsymbol{\sigma^2}$. The equation for the 95% interval is shown in equation 10. Figure 3.1 shows the 95% confidence interval on a normal distribution with mean 0.

$$95\%_{interval} \approx \pm 2 * \sigma = \pm\, 2 * \sqrt{\text{MSE}} = \pm 2 * \sqrt{\frac{1}{n}\sum_{i=1}^{n} {r_i}^2} = \pm 2 * \sqrt{\frac{1}{n}\sum_{i=1}^{n} |y_i - f(x_i, \boldsymbol{\beta})|^2} \qquad (11)$$

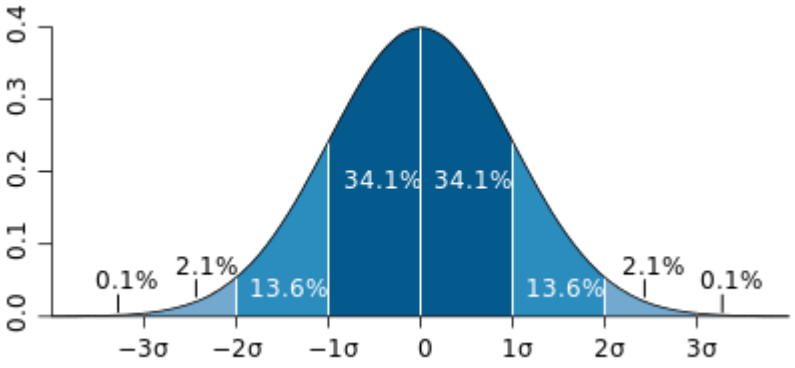


Figure 3.1: *A normal distribution with mean 0 and standard deviation σ. The 95% confidence interval is the area between -2σ and 2σ. Source:* (Toews, 2007)

3.2. RETC

The Retention Curve (RETC) computer program is a commonly used program, based on the Least Squares method, for analysing the hydraulic properties of various unsaturated soils (van Genuchten, Leij, & Yates, 1991).

In order to predict water retention curves through RETC, some information is needed. First, the type of model for the water retention curve is needed. The models that can be applied are the Brooks-Corey model (1964), the van Genuchten model (1990), the lognormal distribution model of Kosugi (1996) and the dual-permeability model of Durner

(1994). As already stated, the focus will be on the Brooks-Corey model and the van Genuchten model which have been discussed in Sections (2.1.1) and (2.1.2), respectively. It is necessary for the van Genuchten water retention model that a relation between the parameters m and n is chosen. The two options are the Burdine assumption which states that $m = 1 - 2/n$ and the Mualem assumption which states that $m = 1 - 1/n$. These assumptions have an effect on the shape of the water retention curve. Additionally, a conductivity model can be selected when certain soil properties need to be computed. These properties include the diffusivity and the conductivity, which then can be plotted.

After specifying the preferred model and its settings, an initial estimate is done for the soil parameters that will be estimated with this program. These parameters are θ_r, θ_s, α, n and K_s. The definition of the first four parameters can be found equation 5. The soil parameter K_s, which is defined as the saturated conductivity, quantifies the ability of a certain soil to conduct water when it is subjected to a hydraulic gradient. A higher saturated conductivity corresponds to an easier movement of water through soil pores of a saturated soil. These initial values can be estimated simultaneously in RETC by choosing a soil type with already appointed estimations for these soil parameters. A more advanced method is offered by RETC by means of neural network prediction. When soil ratios and other input parameters are known, a more accurate initial estimation of the soil hydraulic parameters is performed.

Finally, the water retention data is inputted by inserting the water pressure head and the corresponding moisture content and the outputs can be evaluated. The RETC program produces an ASCI file and a graph as outputs. The ASCI file contains the numerical data of the performed analysis. Most importantly, it presents the final results of the soil parameters estimation which were estimated through Non-Linear Least Squares analysis.

3.3. Optimize Curve Fit in Python

The SciPy package in Python contains the Optimize.curve_fit function which uses non-linear regression to fit a function to observed datapoints. First, a model is defined with unknown parameters that need to be estimated. It takes the observed x and y datapoints as input in the form of an array. Finally, the best parameters are determined by iteration.

For the water retention curves, the van Genuchten model with the Mualem assumption is defined in Python with the soil parameters unknown as shown in equation 5. The unknown parameters that need to be estimated are θ_r, θ_s, α and n. For each parameters, initial values and boundaries are defined. The boundaries are necessary for the fitted model to estimate realistic parameters. θ_r is the residual moisture content which can theoretically only take values between 0 and 1. In practice however, the value for the residual moisture content is generally slightly larger than 0 and never equal to 1. These values can differ for different soil types. The same boundaries apply for the saturated moisture content θ_s, which is between 0 and 1 as well. Furthermore, α is required to be larger than 0. This is due to the definition of parameter α, which is defined as the inverse of the air entry value. In the case that α is smaller than 0, it is implied that the air entry value is negative, which is physically impossible. Lastly, the parameter n has the property of being larger than 1 as it the based on the Mualem assumption for the relation between the parameters m and n.

Finally, the optimal parameters can be extracted from the Optimize.curve_fit function. These optimal parameters can subsequently be used to plot the water retention curve. Additionally, equation 11 can afterwards be used to compute and plot the 95% interval around the optimal fit.

4

Gaussian Process Regression

The lack of flexibility of traditional water retention regression methods leads to the search of more flexible methods. With Gaussian Process regression having these characteristics, it might serve as a possible solution to the problem of models breaking down for composite soils. In this chapter, the theory behind Gaussian Processes will be discussed as well as the methodology for modelling water retention curves with the help of Gaussian Processes. Additionally, the GPy package in Python for Gaussian Process regression will be discussed.

4.1. Gaussian Process Theory

A stochastic process is a collection of random variables that is indexed by time or space. When such a process is Gaussian, it is called a Gaussian Process and it can therefore be defined as a collection of random variables where every collection of these variables has a multivariate normal distribution (Rasmussen & Williams, 2006). Gaussian Process regression is a data interpolation method for which the interpolated values are modelled by a Gaussian Process. This type of regression results in a mean fit line that attempts to optimally fit through observables. Additionally, it also results in a confidence interval around this mean fit that can fluctuate dependent on the certainty of the Gaussian Process on the mean function output value.

A Gaussian Process is completely defined by its mean function and covariance function. For a one-dimensional normal distribution, a mean value and a variance can be determined. The mean value is equal to the expected value and the variance is related to the deviation of this expected value. For a multivariate Gaussian distribution, this mean function is a vector in k dimensions that contains the mean values of k random variables. This means that in a bivariate case where there are two random variables, the mean function vector contains two values which are the mean values of each normal distribution of each random variable. Instead of a variance, the Gaussian Process has a covariance function which represents the correlation between the variables. This covariance function is also called the kernel function. The three assumptions made for the Non-Linear Least Squares method in Chapter 3 are not made for the Gaussian Processes. Instead, the following three assumptions are made.

The residuals (or errors) are normally distributed with:

1. $\mu = 0$
2. $\sigma^2 \neq$ *constant*
3. ***correlation exists between the errors***

When the Gaussian Process is denoted as $f(\mathbf{x})$, the mean function of the Gaussian Process is denoted as $m(\mathbf{x})$, the kernel function is defined as $k(\mathbf{x}, \mathbf{x}')$ where $\mathbf{x}$ and $\mathbf{x}'$ denote the locations of some input points, and the following can be stated:

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \tag{12}$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}\big[\big(\mathrm{f}(\mathrm{x}) - m(\mathbf{x})\big)\big(\mathrm{f}(\mathbf{x}') - m(\mathbf{x}')\big)\big] \tag{13}$$

Equations 12 and 13 present the expressions for the mean function and the kernel function for the Gaussian Process, respectively. The symbol $\mathbb{E}$ denotes the expected value of the argument in between the brackets. The Gaussian Process denoted by $f(\mathbf{x})$ can be defined as following:

$$f(\mathbf{x}) \sim \mathcal{GP}\big(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\big) \tag{14}$$

which states that the stochastic process is Gaussian distributed with a mean function $m(\mathbf{x})$ and a kernel function $k(\mathbf{x}, \mathbf{x}')$. It is common to take the mean function to zero for notational simplicity. This means that the behaviour of the Gaussian Process is completely

defined by the covariance or kernel function. By implementing this assumption in equation 14, the following simplified expression for the Gaussian Process is obtained:

$$f(\mathbf{x}) \sim \mathcal{GP}\big(\mathbf{0}, k(\mathbf{x}, \mathbf{x}')\big) \tag{15}$$

The assumptions made for the Least Squares method are not made for the Gaussian Process method. Even though that the mean is set to zero for notational simplicity, it is not necessary to do so. Furthermore, the variance is not a fixed value which is the case for the Least Squares method.

There exist many different kernel functions, such as the Periodic kernel, the Linear kernel, the Matérn kernel and the Squared Exponential kernel (Duvenaud, 2014). Kernels can even be combined through addition or multiplication. In this thesis, only the Squared Exponential kernel will be discussed as it is the default kernel for Gaussian Processes (Duvenaud, 2014). The Squared Exponential (SE) kernel, also called the Radial Basic Function (RBF) kernel or Gaussian kernel, is a covariance function that specifies the correlation between random variables. Equation 16 presents this kernel function.

$$k_{SE}(\mathbf{x}, \mathbf{x}') = \sigma^2 \, e^{-\frac{(|\mathbf{x} - \mathbf{x}'|)^2}{2\ell^2}} \tag{16}$$

For the Squared Exponential kernel function, as shown in equation 16, there are two parameters, namely the output variance $\boldsymbol{\sigma^2}$ and the length scale $\boldsymbol{\ell}$. These parameters are also called hyperparameters (Rasmussen & Williams, 2006). The value of the output variance determines the deviation of the function from the expected value. Similar to the variance of a univariate normal distribution, it is basically a scale factor. The characteristic length scale determines how much two points at location $\mathbf{x}$ and $\mathbf{x}'$ can influence each other when they have a certain distance between them. It influences the smoothness of the function. While small length scales result in quick changing function values and therefore rough functions, large length scales result in slow changing smooth functions. In Figure 4.1, the effect of the length scale on the smoothness of the fit can be seen, with the larger length scale corresponding to the smoother function. Common methods to determine the hyperparameters of the kernel are the maximization of the marginal log-likelihood or the cross validation method (Rasmussen & Williams, 2006). In Section 4.2, optimization of the hyperparameters with the use of the marginal log-likelihood in python, is discussed.

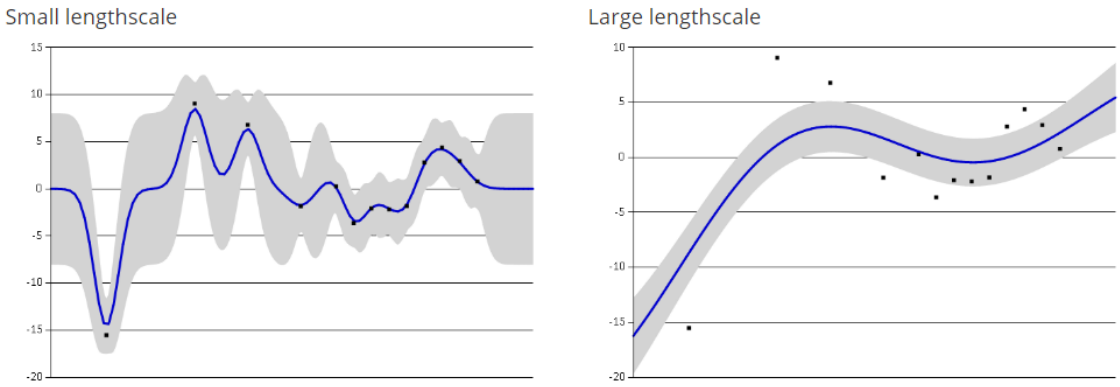


Figure 4.1: *Two fits through datapoints. The left plot shows the fit with a small length scale and the right plot shows the fit with a larger length scale. Source:* (Evelinag, 2014)

The SE kernel also possesses the input distance argument, denoted by $\mathbf{x} - \mathbf{x}'$. This is the distance between the x-coordinates of two input points. The output variance together with the length scale and the distance between two input points determine the value of the Squared Exponential kernel function and therefore influence the output function. Generally speaking, when two input points are close to each other, it is expected that their corresponding output points are close to each other as well.

For a clear understanding of the kernel function presented in equation 16, two extreme scenarios can be viewed. In the first scenario, the locations of two input datapoints are similar to each other which means that $\mathbf{x} - \mathbf{x}' = 0$. This results in the output variance multiplied by the exponential of zero which is equal to $\boldsymbol{\sigma^2}$. In the second scenario where two input datapoints are infinitely far apart from each other, the argument becomes the exponential of negative infinity which approaches to 0. This indicates that a covariance value of 0 corresponds to no correlation between these points and a covariance function value equal to $\boldsymbol{\sigma^2}$ corresponds to a high correlation between these points. The covariance function value usually lies between these two extreme values.

The kernel function implies a distribution over functions. The reason for this is that a function can be treated as an infinite long vector with its values being equal to every single input point. A distribution can be assigned to this infinite long vector which results in a multivariate distribution. Samples can be drawn from this distribution by selecting a finite set of input points stored in vector $\boldsymbol{X}$ and setting up the covariance matrix by inserting the entries of this vector $\boldsymbol{X}$ in the kernel function at $\mathbf{x}$ and $\mathbf{x}'$ in equation 16. The resulting n x n covariance matrix is presented in equation 17, with n being the amount of known input training points. In short, the $K(\boldsymbol{X},\boldsymbol{X})$ matrix is obtained after implementing the covariance function to n input training points $\boldsymbol{X}$.

$$K(\boldsymbol{X},\boldsymbol{X}) = \begin{bmatrix} k_{SE}(X_1, X_1) & \cdots & k_{SE}(X_1, X_n) \\ \vdots & \ddots & \vdots \\ k_{SE}(X_n, X_1) & \cdots & k_{SE}(X_n, X_n) \end{bmatrix} \tag{17}$$

With the obtained covariance matrix, a random Gaussian vector sample $\mathbf{f}$ can be generated, which is normally distributed with a mean vector equal to zero and a covariance matrix, as shown in equation 18.

$$\mathbf{f} \sim \mathcal{N}\big(\mathbf{0}, K(\boldsymbol{X},\boldsymbol{X})\big) \tag{18}$$

This vector sample contains the values from the Gaussian Process that correspond to the input values. The Gaussian vector can subsequently be plotted against these input values to obtain the prior, which is the probability distribution before any actual data has been introduced and is solely defined by the kernel function. More of these samples can be drawn and plotted as can be seen in Figure 4.2.

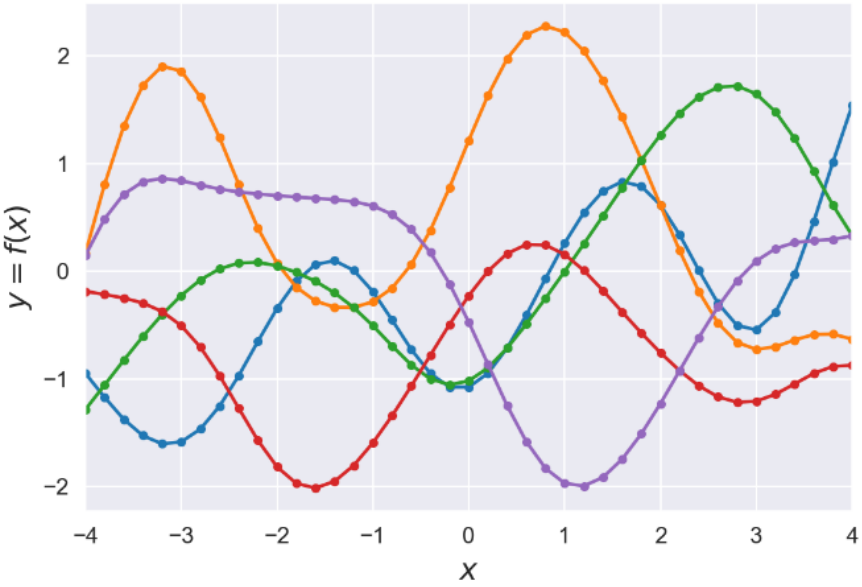


Figure 4.2: An example of *5 random samples drawn from a Gaussian Process with a Squared Exponential kernel function. Source:* (Roelants, 2019)

After defining the prior, a posterior can be created when some data is given. The posterior is the conditional probability distribution after data has been considered. It can be used to estimate the expected value and the probability of this expected value. For the prior, we already had n observed data points ($\boldsymbol{X}$, $\mathbf{f}$) which are the training points. Vector $\boldsymbol{X}$ contains the input data of these training points and vector $\mathbf{f}$ contains the corresponding function values. Predictions can now be made for n_* data points which are also called the test points ($\boldsymbol{X}_*$, $\mathbf{f}_*$). The subscript asterisk denotes that the points are test points which are points that are unknown and need to be estimated. Next to the test points, there are also training points which are known points that are used to estimate the test points. The training points are denoted without an asterisk. This means that the values in vector $\mathbf{f}_*$ will be predicted given the training data ($\boldsymbol{X}$, $\mathbf{f}$) and the input variable $\boldsymbol{X}_*$ which corresponds to the test points or the points that will be predicted. In mathematical notation, this can be stated as $\boldsymbol{p}(\mathbf{f}_*|\mathbf{f}, \boldsymbol{X}, \boldsymbol{X}_*)$ which translates to the probability density of the predictions $\mathbf{f}_*$ given $\mathbf{f}$, $\boldsymbol{X}$ and $\boldsymbol{X}_*$. Since $\mathbf{f}$ and $\mathbf{f}_*$ both come from the same function, it can be stated that they are jointly Gaussian which means that they must have a multivariate normal distribution (Roelants, 2019). This is shown in equation 19.

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(\boldsymbol{X},\boldsymbol{X}) + \sigma_n{}^2 I & K(\boldsymbol{X},\boldsymbol{X}_*) \\ K(\boldsymbol{X}_*,\boldsymbol{X}) & K(\boldsymbol{X}_*,\boldsymbol{X}_*) \end{bmatrix}\right) \tag{19}$$

where $K(\boldsymbol{X}, \boldsymbol{X})$ denotes the n x n matrix of the covariances evaluated at the training points and obtained by applying the kernel function to these points. Similarly, $K(\boldsymbol{X}, \boldsymbol{X}_*)$ denotes the n x n_* matrix of the covariances evaluated at all pairs of training and test points. The matrices $K(\boldsymbol{X}_*, \boldsymbol{X})$ and $K(\boldsymbol{X}_*, \boldsymbol{X}_*)$ follow this same logic. The noise variance is denoted by $\boldsymbol{\sigma_n{}^2}$. This noise variance represents the noisy observations that are realistically present and that need to be taken into account. In the case of water retention curves, it represents the accuracy at which the moisture content can be measured based on the given data. It is written in the form of an n x n identity matrix.

Ultimately, the following Gaussian Process regression equations can be deducted by using the Gaussian Identities (Rasmussen & Williams, 2006).

$$\mathbf{f}_* \mid \boldsymbol{X}, \mathbf{f}, \boldsymbol{X}_* \sim \mathcal{N}\left(\bar{\mathbf{f}}_*, cov(\mathbf{f}_*)\right) \tag{20}$$

$$mean(\mathbf{f}_*) = \bar{\mathbf{f}}_* = K(\boldsymbol{X}_*, \boldsymbol{X})[K(\boldsymbol{X}, \boldsymbol{X}) + {\sigma_n}^2 I]^{-1}\mathbf{f} \tag{21}$$

$$cov(\mathbf{f}_*) = K(\boldsymbol{X}_*, \boldsymbol{X}_*) - K(\boldsymbol{X}_*, \boldsymbol{X})[K(\boldsymbol{X}, \boldsymbol{X}) + {\sigma_n}^2 I]^{-1} K(\boldsymbol{X}, \boldsymbol{X}_*) \tag{22}$$

In equation 21, the mean is denoted by $\bar{\mathbf{f}}_*$ which is used to plot the fit through the data points. The covariance of the mean is presented in equation 22. This can be used to plot the confidence interval around the mean fit for a certain model. An example can be found in Figure 4.3, which shows the optimal mean fit through some datapoints together with a confidence interval. It can be noticed that this confidence interval decreases in size when the density of the different datapoints is higher. This is because there is more certainty around these points that the true value lies in a certain interval, which is intuitively correct as well. When there are no datapoints nearby, the size of the confidence interval increases as there is less certainty. Thus, when there is few data available to confirm the predicted mean fit, the degree of certainty that this mean fit is true decreases.

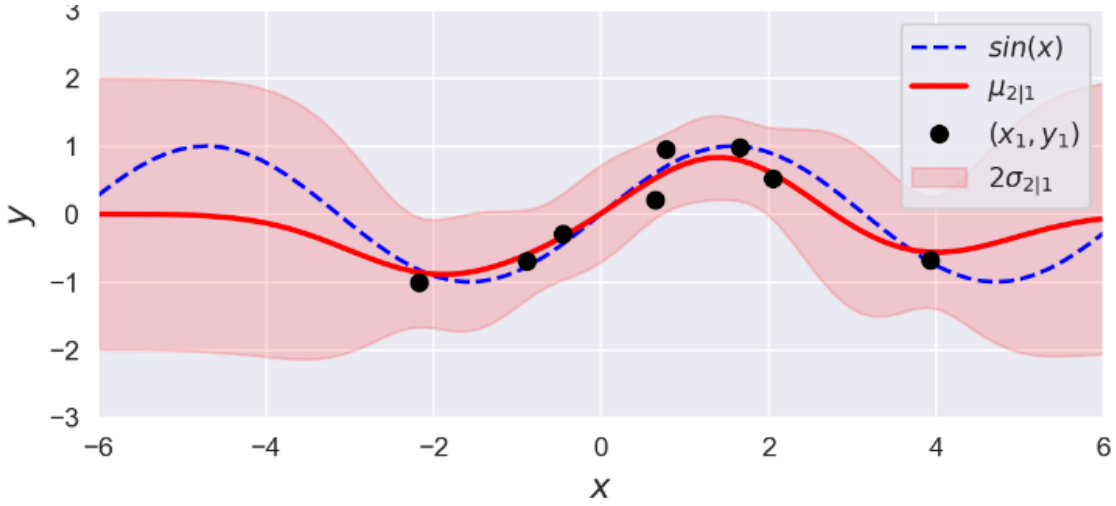


Figure 4.3: *The posterior for a number of datapoints in the form of a sinusoid with some noise. The red line represents the optimal mean fit through the observed data points. The pink interval represents the confidence interval equal to twice the standard deviation. The size of the interval fluctuates throughout. The black dots represent the observed data through which GP regression is performed upon.* ***Source:*** (Roelants, 2019)

4.2. GPy Machine Learning Algorithm

GPy is a Gaussian Process framework written in Python that can be used for the application of Gaussian Process regression (Sheffield ML Group, 2015). It was developed by the Sheffield Machine Learning group under the Department of Computer Science of the University of Sheffield. It can be downloaded as a ZIP file and subsequently imported in a Python notebook.

First, some training data is needed together with an interval where the data is plotted on. Next, a kernel function needs to be defined that will be used for the regression model. GPy offers a wide range of kernels, including but not limited to the Squared Exponential kernel, the Periodic kernel and the Matérn kernel. Together with the kernel, the initial length scale and variance for the chosen kernel can be determined. The Gaussian Process regression model can thereafter be extracted.

GPy also allows for optimization of the model through the marginal log-likelihood method which is based on the principle of choosing the model parameters that maximize the likelihood function. The likelihood function represents how likely particular parameter values are, given some measured data. After this optimization process, new model parameters are defined that usually result in a better fit and a smaller confidence interval. It also prints the value of the marginal log-likelihood, which can be compared to its value before optimization. A higher marginal log-likelihood corresponds to an improvement in fit and confidence interval. In Figure 4.4, an example can be found that shows the effects of optimization in GPy on the predicted model for a number of datapoints.

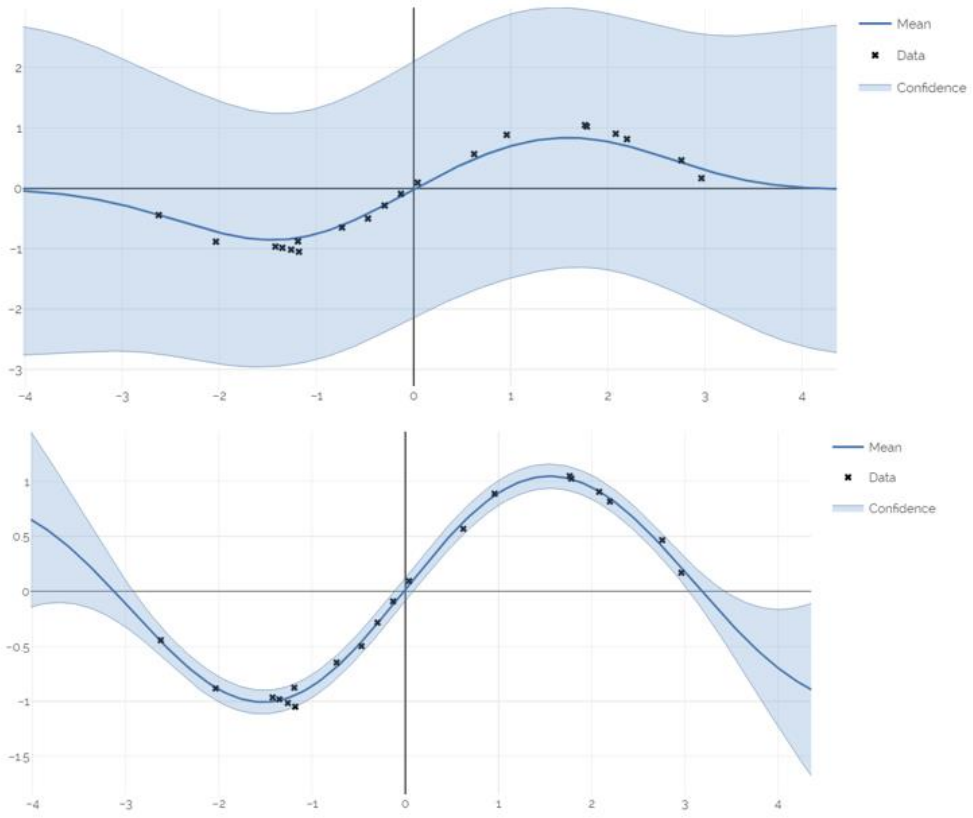


Figure 4.4: *The upper plot shows the GP regression before optimization of the model parameters σ^2 & ℓ. The lower plot represents the GP regression after optimization. Source:* (Bailey, 2016)

4.3. GPy for Water Retention Curves

Fitting water retention data with the Least Squares method leads to residuals which are defined as the difference between the fitted line and the actual observed datapoints. This is presented in equation 6. When these residuals show some sort of pattern, it might be interesting to fit a Gaussian Process model through these points. The flexibility of the Gaussian Process regression model usually results in an accurate fit of these residuals as it does not follow a certain model or form. When this fitted line is added to the fitted water retention model described in Section (3.3), it might lead to an improvement of the fit and confidence interval.

The benefits of combining Gaussian Processes and Non-Linear Least Squares for the representation of water retention curves, can be assessed both visually and quantitatively. When the model fits better and smoother through the datapoints and when the confidence interval is smaller, it can be stated that the regression model indeed improved. Quantitatively, this can be analysed by computing the log-likelihood of the fit before and after the implementation of Gaussian Processes. Equation 22 presents the equation for computing the log-likelihood in the scenario where only the Non-Linear Least Squares method is considered.

$$log\text{-}likelihood = \sum_{i=1}^{n} \log\left(\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x_i-\mu}{\sigma}\right)^2}\right) \tag{22}$$

where log denotes the natural logarithm and:
$\boldsymbol{\sigma}$ is the standard deviation which can be calculated from equation 10
$\mathbf{x}$ is the function value of an observed datapoint
$\boldsymbol{\mu}$ is the mean of model, represented by the fitted line

This can be computed in Python with the norm.logpdf function from the SciPy Stats package. When implementing the Gaussian Process regression model, the marginal log-likelihood can be automatically computed with GPy.

The log-likelihood values for the two models can then be compared to each other to determine whether the model actually improved. As already stated, a higher log-likelihood corresponds to a better fit. This means that the log-likelihood of the model where Gaussian Processes are implemented should be higher than the log-likelihood of the Least Squares model in order to state that with the help of Gaussian Processes, water retention curves improve indeed. This will be the focus in Chapter 5, where Gaussian Processes for water retention curves will be evaluated on the basis of datasets extracted from the UNSODA soil database.

5

Results

For the evaluation of Gaussian Processes for water retention curves, comparisons need to be made with the traditional approaches of representing water retention curves. This needs to be done on the basis of actual water retention data in order to be able to draw a correct conclusion. The datasets that will be used are originating from the UNSODA soil hydraulic database as discussed in Section (2.2). With the use of these datasets, water retention curves can be plotted and ultimately compared for each regression model.

5.1. Used Datasets

The UNSODA database contains many datasets in various categories as is illustrated in Figure 2.6. Many datasets are numerically fairly similar to each other. The following four datasets, presented in Tables 1 up until 4, were retrieved from the UNSODA database and will be used to analyse the different regression models for the water retention curves. These four datasets have been chosen carefully to be dissimilar to each other, so that a qualitative analysis can be performed on different types and shapes of water retention curves.

Table 1
Dataset 1: 25 Measurements for Moisture Content and Pressure Head.

No.	Pressure head [cm]	Moisture content [-]	No.	Pressure head [cm]	Moisture content [-]
1	1	0.578	14	200	0.545
2	2	0.577	15	300	0.537
3	3	0.576	16	500	0.524
4	5	0.574	17	700	0.513
5	7	0.573	18	1000	0.500
6	10	0.572	19	1500	0.482
7	15	0.570	20	2000	0.468
8	20	0.568	21	3000	0.448
9	30	0.566	22	5000	0.425
10	50	0.562	23	7000	0.414
11	70	0.559	24	10000	0.396
12	100	0.555	25	15000	0.378
13	150	0.549			

Note: Measurements done under laboratory drying conditions. Retrieved from the UNSODA database.

Table 2
Dataset 2: 13 Measurements for Moisture Content and Pressure Head.

No.	Pressure head [cm]	Moisture content [-]	No.	Pressure head [cm]	Moisture content [-]
1	1	0.447	8	345	0.319
2	5	0.434	9	690	0.308
3	10	0.417	10	2000	0.299
4	20	0.398	11	5000	0.297
5	40	0.378	12	10000	0.296
6	80	0.362	13	15000	0.294
7	160	0.335			

Note: Measurements done under laboratory drying conditions. Retrieved from the UNSODA database.

Table 3
Dataset 3: 9 Measurements for Moisture Content and Pressure Head.

No.	Pressure head [cm]	Moisture content [-]	No.	Pressure head [cm]	Moisture content [-]
1	1	0.474	6	500	0.417
2	2	0.469	7	1000	0.410
3	19	0.431	8	5000	0.396
4	48	0.427	9	15500	0.385
5	200	0.423			

Note: Measurements done under laboratory drying conditions. Retrieved from the UNSODA database.

Table 4
Dataset 4: Measurements for Moisture Content and Pressure Head.

No.	Pressure head [cm]	Moisture content [-]	No.	Pressure head [cm]	Moisture content [-]
1	5	0.355	9	300	0.297
2	10	0.354	10	350	0.282
3	20	0.353	11	500	0.271
4	30	0.350	12	1000	0.308
5	40	0.344	13	2000	0.281
6	50	0.344	14	4000	0.281
7	100	0.321	15	15000	0.268
8	200	0.301			

Note: Measurements done under laboratory drying conditions. Retrieved from the UNSODA database.

5.2. RETC Results

The datasets from Section 5.1 can be used in the RETC program to obtain the soil hydraulic parameters for a particular model and to plot this model as discussed in Section (3.2). The estimated values of the soil parameters for the datasets 1 up until 4 can be seen in Figure 5.1. The values of these soil parameters are subsequently used for plotting the water retention curve. These plots can be represented in different ways based on the scale that is used for the horizontal axis and for the vertical axis. The water retention curves for the datasets 1 up until 4 are presented in Figure 5.2.

The plots in Figures 5.2 show some poor water retention curves, even though the points are fairly well fitted. Figure 5.1a shows a table with the estimated soil parameters for the plot in Figure 5.2a. Here, the estimation for θ_r seems to be missing. This can be explained by looking at Figure 5.2a, where it can be seen that the gradient of the curve becomes larger in absolute value when the pressure head increases. This means that the moisture content decreases significantly when only little extra water pressure is occurring. Due to the fact that a soil will remain at least some moisture inside its pores, which is equal to the residual moisture content, a water retention curve should asymptotically approach this residual moisture content. As the plot in Figure 5.2a does not do this, no residual

moisture content value can be determined. This means that the plotted water retention curve in Figure 5.2a is unrealistic and therefore inaccurate.

In Figure 5.2c, the plot does not look like a typical water retention curve as well. This is partly caused by an incorrect estimation of $\boldsymbol{\alpha}$, which is unusually large as can be seen in Figure 5.1c. Due to the specification of the water retention curve model, the RETC program still attempts to fit a curve through the points by estimating the soil parameters for the model even though that these estimated parameters are not always physically possible. The estimated value for $\boldsymbol{\theta_r}$ seems to be missing for plot c as well. This can be explained by looking at the plot in Figure 5.2c. The fitted retention curve in this figure shows that the curve does not asymptotically approach any value when the water pressure increases as it keeps decreasing with a constant rate. Similar to plot a, this explains why the value for the residual moisture content $\boldsymbol{\theta_r}$ is missing. This goes to show that when too little data is available or when the data deviates from the norm, an adequate water retention curve is difficult to be obtained with the RETC program.

a

```
Nonlinear least-squares analysis: final results
================================================
                                          95% Confidence limits
Variable     Value     S.E.Coeff.    T-Value     Lower       Upper
ThetaS       .57234      .00101       568.88      .5702       .5744
Alpha        .00299      .00024        12.62      .0025       .0035
n           1.10697      .00299       370.69     1.1008      1.1132
```

b

```
Nonlinear least-squares analysis: final results
================================================
                                          95% Confidence limits
Variable     Value     S.E.Coeff.    T-Value     Lower       Upper
ThetaR       .28481      .00436        65.26      .2749       .2947
ThetaS       .44600      .00370       120.61      .4376       .4544
Alpha        .08158      .01523         5.36      .0471       .1160
n           1.43930      .05278        27.27     1.3199      1.5587
```

c

```
Nonlinear least-squares analysis: final results
================================================
                                          95% Confidence limits
Variable     Value     S.E.Coeff.    T-Value     Lower       Upper
ThetaS       .55417    41.76184          .01  -101.6274   102.7357
Alpha    2804.91137************          .00**********************
n           1.02069      .00263       388.15     1.0143      1.0271
```

d

```
Nonlinear least-squares analysis: final results
================================================
                                          95% Confidence limits
Variable     Value     S.E.Coeff.    T-Value     Lower       Upper
ThetaR       .27966      .00582        48.05      .2669       .2925
ThetaS       .35458      .00602        58.86      .3413       .3678
Alpha        .01314      .00466         2.82      .0029       .0234
n           2.31588      .60682         3.82      .9803      3.6515
```

Figure 5.1: *In RETC estimated hydraulic soil parameters for dataset 1 (**a**), dataset 2 (**b**), dataset 3 (**c**) and dataset 4 (**d**). The used water retention curve model for all datasets is the van Genuchten model in combination with the Mualem assumption for the relation between **m** and **n**. The final estimates for the soil parameters are stated under 'Value'.*

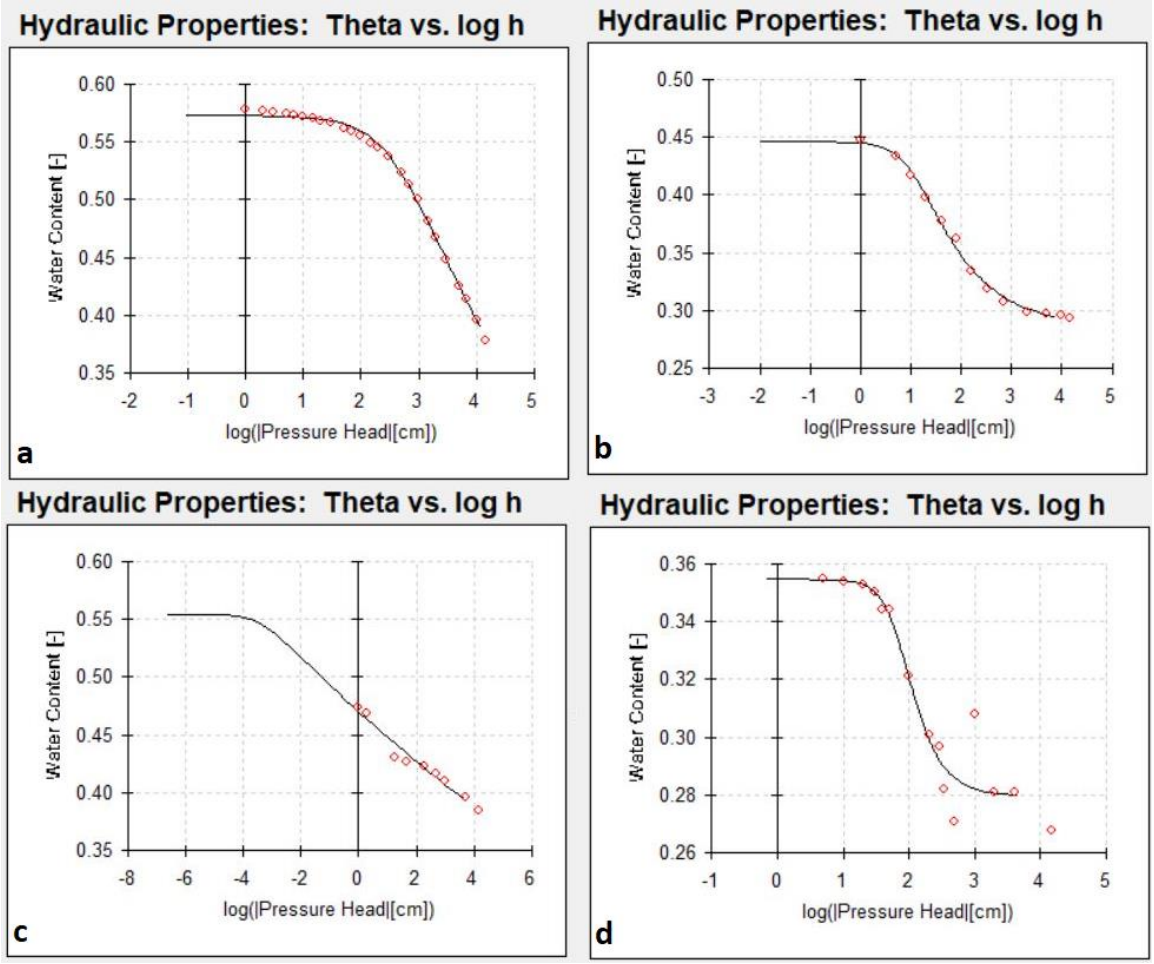


Figure 5.2: *In RETC fitted water retention curves for dataset 1 (**a**), dataset 2 (**b**), dataset 3 (**c**) and dataset 4 (**d**), represented by the black line. The red circles indicate the measured water retention data points. The used water retention curve model for all datasets is the van Genuchten model in combination with the Mualem assumption for the relation between **m** and **n**.*

5.3. SciPy Curve Fit Results

The same principle behind the RETC program can be implemented in Python as discussed in Section (3.3). First, the optimal soil parameters can be extracted from the Optimize.curve_fit function. These optimal parameters can subsequently be used to plot the water retention curve. This process is done for the datasets 1 up until 4, which can be found in Tables 1 up until 4, respectively. The estimated soil parameters are presented in Table 5. The plots for these datasets are presented in Figure 5.3. The Python code for the Non-Linear Least Squares regression method using SciPy Curve Fit can be found in Appendix A.

The plots in Figure 5.3 are fairly similar to the RETC plots in Figure 5.2 as the same method and technique is used for fitting a model through the observed datapoints. Similar to the RETC program, the water retention model also breaks down for datasets 1 & 3, as can be seen in both the plots and the soil parameters. However, the estimated parameters are deviating slightly from the parameters that were estimated in the RETC program. An explanation for this is that different initial values are used for the RETC program and the

Python script. Furthermore, the number of iterations is also different for the two methods. In general, the water retention curves of the two methods are fairly similar. For practical reasons however, the Python scripts will be used instead of RETC for the comparison between the Non-Linear Least Squares method described in Chapter 3 and the Gaussian Process regression method described in Chapter 4.

Table 5
Estimated Soil Parameters obtained with the SciPy Optimize.curve_fit function in Python for Datasets 1 up until 4.

Dataset	θ_r [-]	θ_s [-]	α [L^{-1}]	n[-]
1	0.0	0.5723	0.003	1.107
2	0.252	0.4543	0.1875	1.23
3	0.248	0.4768	1.0	1.053
4	0.247	0.3629	0.0348	1.338

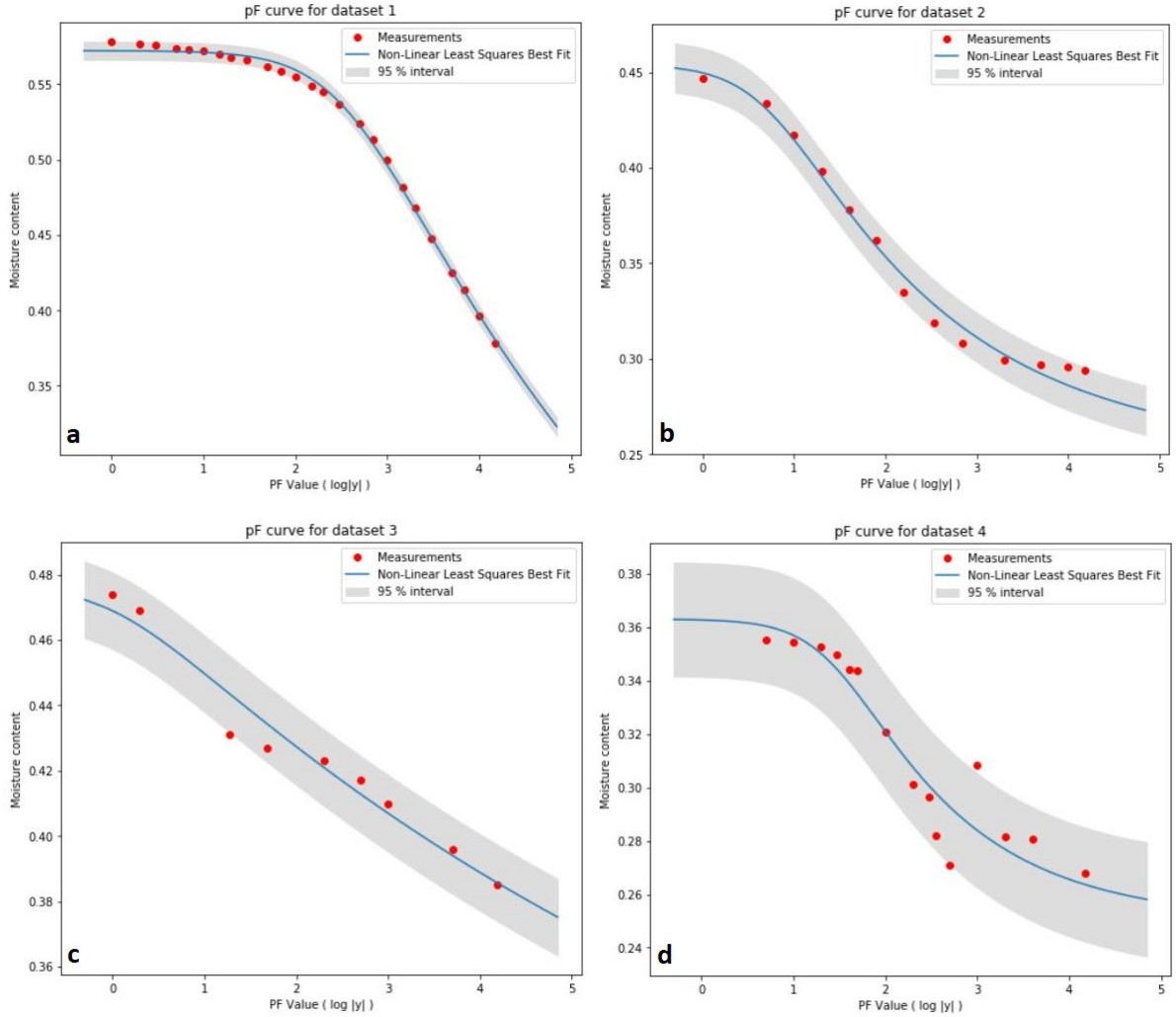


Figure 5.3: *Water retention curves, with estimated parameters with the use of Python's SciPy package and function Optimize.curve_fit for dataset 1 (**a**), dataset 2 (**b**), dataset 3 (**c**) and dataset 4 (**d**). A 95% confidence interval has been plotted around the fitted functions which is defined as twice the standard deviation. The blue line represents the fitted function for the optimal parameters. The red dots represent the measured water retention data points. The functions have been plotted with a logarithmic scale for the horizontal axis.*

5.4. GPy Results

In Section (4.3), the method for representing water retention curves with the help of Gaussian Processes is described. This will be done with the GPy framework in Python. The Python code that is used for Section 5.4 can be found in Appendix B.

First, the residuals of the Non-Linear Least Squares fit described in Section 3.3 need to be computed. Figure 5.4 shows the plots of these residuals for datasets 1 up until 4. Subsequently, Gaussian Process regression can be performed on these residual datapoints in order to obtain a fit through these points. This is illustrated in Figure 5.5. Notice that in plot d in Figure 5.5, no fit is obtained through the datapoints. The mean fit remains at zero throughout. The reason for this is that the errors are too random and no structure or pattern is recognized in contrast to plots a, b and c. This also leads to a larger confidence interval as there is weak correlation between the datapoints.

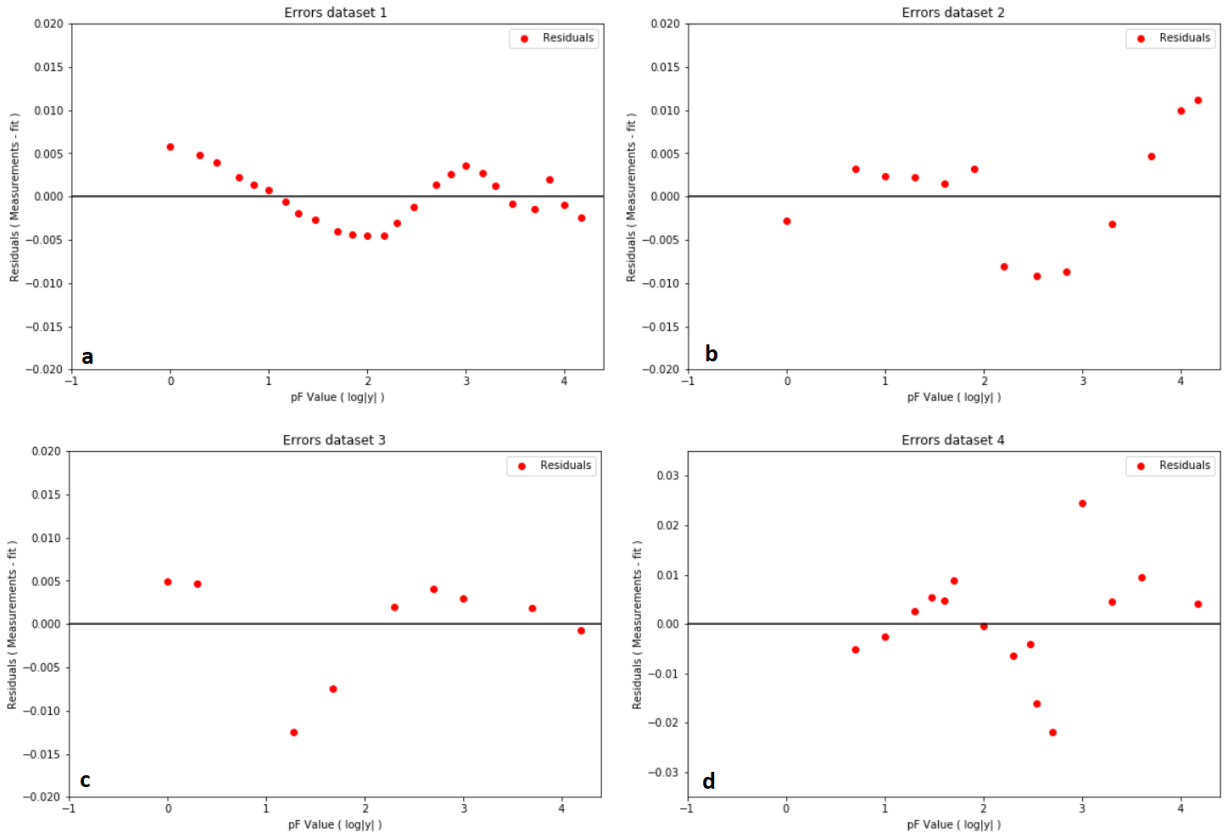


Figure 5.4: *Plots of the residuals for dataset 1 (**a**), dataset 2 (**b**), dataset 3 (**c**) and dataset 4 (**d**). The black horizontal line represents the zero line. Residuals above this line represent a larger value of the measurements than the Least Squares fit and vice versa.*

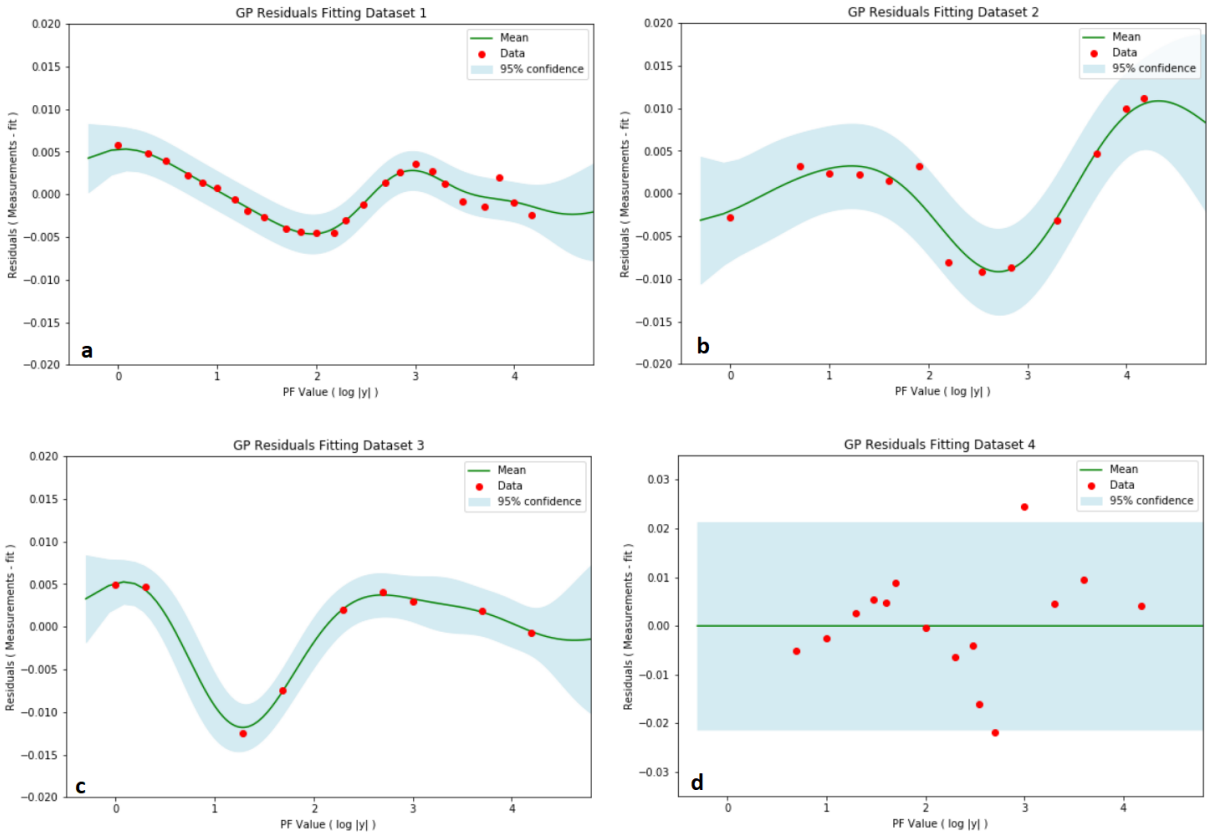


Figure 5.5: *Plots of the Gaussian Process fits, from GPy, through the residuals for dataset 1 (**a**), dataset 2 (**b**), dataset 3 (**c**) and dataset 4 (**d**). The red circles represent the residual datapoints and the green line represents the optimal mean fit through these points. The light blue area around the mean fit represents the 95% confidence interval.*

For the final step, the mean fits from Figure 5.5 will be added to the mean fits in figure 5.3 for each dataset. This will result in new water retention fits with different confidence intervals as illustrated in Figure 5.6. Furthermore, Table 6 presents the Gaussian Process model parameters σ^2, ℓ and σ_n^2 for the four datasets after optimization, as discussed in Section (4.2). Here, σ^2 represents the Squared Exponential Kernel variance, ℓ represents the Squared Exponential Kernel length scale and σ_n^2 represents the Gaussian noise variance which is the measurement error. These optimized model parameters correspond to the plots illustrated in Figure 5.6.

Table 6
Optimized model parameters extracted from GPy for Datasets 1 up until 4.

Dataset	σ^2	ℓ	σ_n^2
1	9.79e-06	0.51	1.0e-06
2	4.84e-05	0.75	4.48e-06
3	3.15e-05	0.58	1.0e-06
4	1.56e-12	12.96	1.1e-04

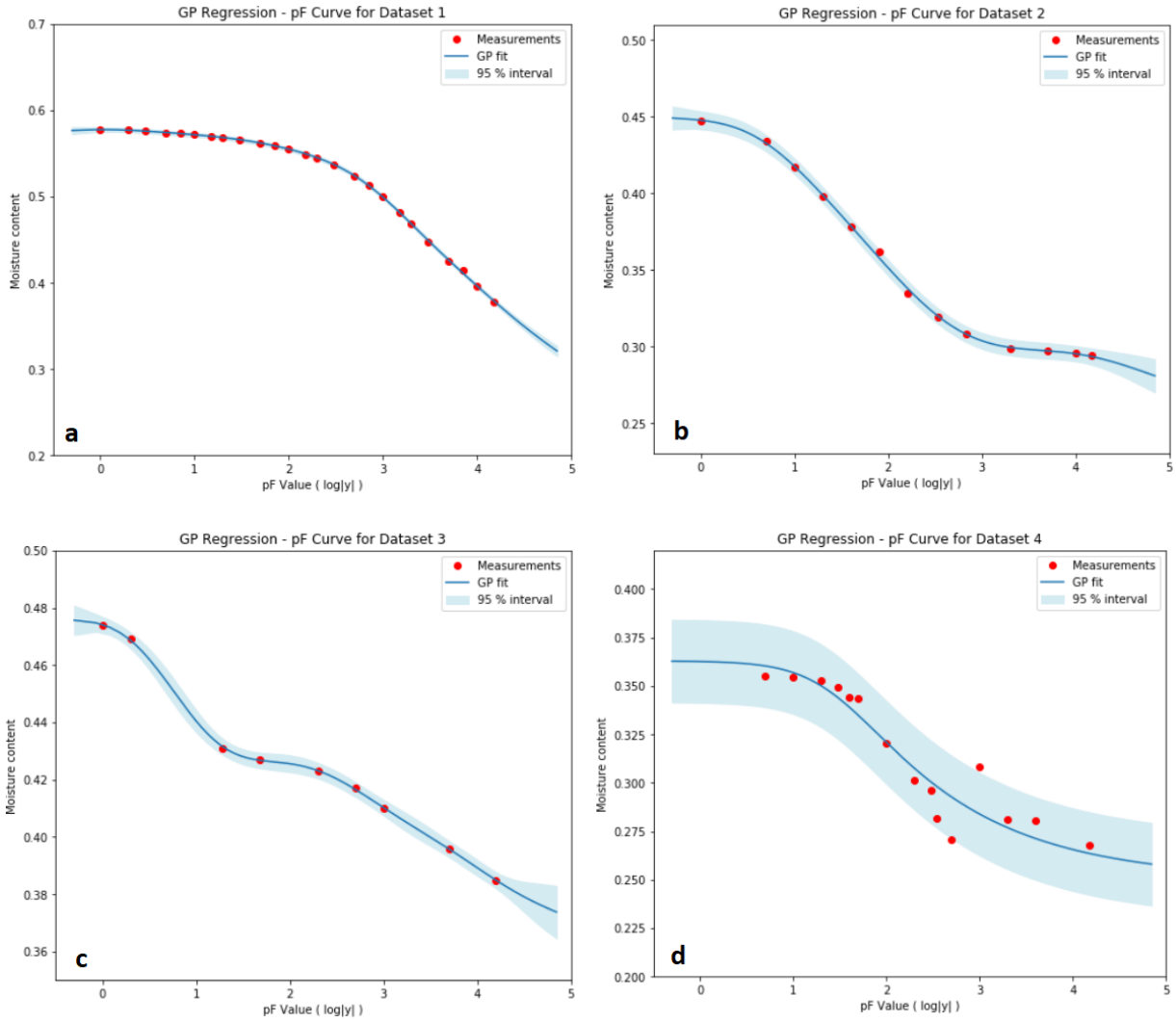


Figure 5.6: *Water retention curves estimated with the combination of Python's Optimize.curve_fit function and the GPy framework for Gaussian Process regression analysis, for dataset 1 (**a**), dataset 2 (**b**), dataset 3 (**c**) and dataset 4 (**d**). A 95% confidence interval has been plotted around the fitted functions. The blue line represents the fitted line. The red dots represent the measured water retention data points. The functions have been plotted with a logarithmic scale for the horizontal axis.*

5.5. Comparing Results from Least Squares and GPy

Now that the water retention curves have been computed and plotted for both methods, it is possible to compare them in order to draw a conclusion on whether water retention curves can be improved with the help of Gaussian Processes. Figure 5.7 up until 5.10 present the water retention curves for both methods next to each other for datasets 1 up until 4, respectively.

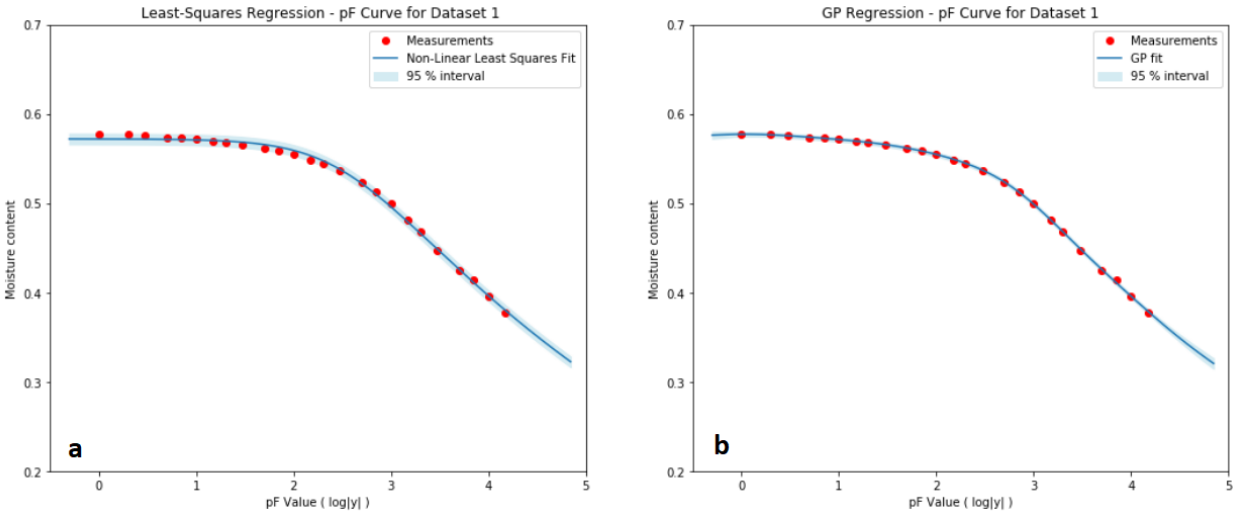


Figure 5.7: *Different water retention curves for dataset 1. Plot 5.7**a** represents the Non-Linear Least Squares method for representing water retention curves. Plot 5.7**b** represents the Non-Linear Least Squares method combined with the Gaussian Process regression method for representing water retention curves. A 95% confidence interval has been plotted around the fitted functions. The blue line represents the fitted line. The red dots represent the measured water retention data points. The functions have been plotted with a logarithmic scale for the horizontal axis.*

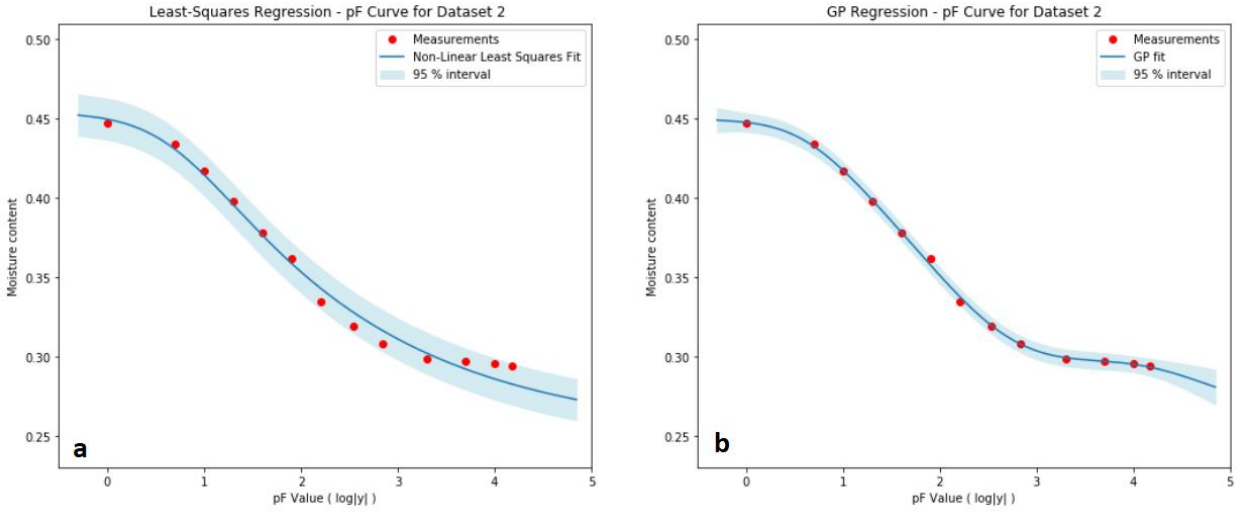


Figure 5.8: *Different water retention curves for dataset 2. Plot 5.8**a** represents the Non-Linear Least Squares method for representing water retention curves. Plot 5.8**b** represents the Non-Linear Least Squares method combined with the Gaussian Process regression method for representing water retention curves. A 95% confidence interval has been plotted around the fitted functions. The blue line represents the fitted line. The red dots represent the measured water retention data points. The functions have been plotted with a logarithmic scale for the horizontal axis.*

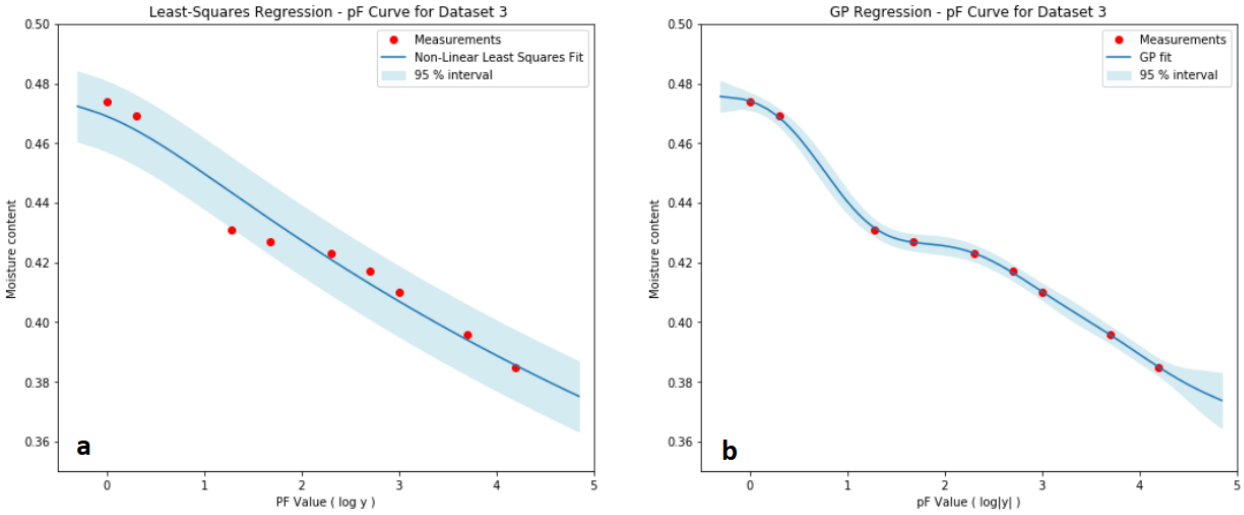


Figure 5.9: *Different water retention curves for dataset 3. Plot 5.9**a** represents the Non-Linear Least Squares method for representing water retention curves. Plot 5.9**b** represents the Non-Linear Least Squares method combined with the Gaussian Process regression method for representing water retention curves. A 95% confidence interval has been plotted around the fitted functions. The blue line represents the fitted line. The red dots represent the measured water retention data points. The functions have been plotted with a logarithmic scale for the horizontal axis.*

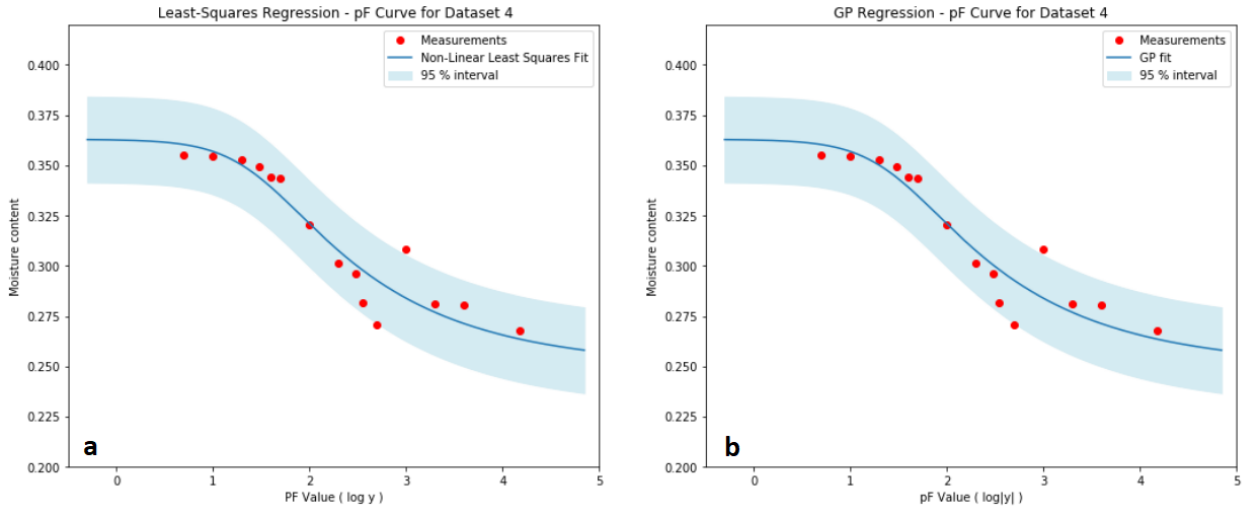


Figure 5.10: *Different water retention curves for dataset 4. Plot 5.10**a** represents the Non-Linear Least Squares method for representing water retention curves. Plot 5.10**b** represents the Non-Linear Least Squares method combined with the Gaussian Process regression method for representing water retention curves. A 95% confidence interval has been plotted around the fitted functions. The blue line represents the fitted line. The red dots represent the measured water retention data points. The functions have been plotted with a logarithmic scale for the horizontal axis.*

Figures 5.7 up until 5.10 allow for a clear visual comparison between the two water retention curve representation methods. For dataset 1, the implementation of Gaussian Processes did result visually in a better fit and a better confidence interval as can be seen in Figure 5.7. However, because the traditional method already resulted in a good fit, the implementation of Gaussian Processes did not result in major improvements. However, the water retention curve models did improve for datasets 2 and 3 as is illustrated in Figure 5.8 and 5.9. Figure 5.9a shows a bad representation of a water retention curve with too much uncertainty. When the Gaussian Process regression method is applied on this dataset, the curve significantly improves as is shown in Figure 5.9b. Also, the confidence interval is decreased in size which means that the uncertainty has decreased as well. Finally, dataset 4 can be evaluated which corresponds to Figure 5.10. The plots and the confidence intervals seem quite similar to each other. This was expected as the mean fit of the residuals was equal to zero. When this mean of zero is added to the Least Squares model, no changes are made. This means that when the residuals are weakly correlated, Gaussian Processes have no impact on the model, which is intuitively correct as Gaussian Processes depend on the correlation between training points as discussed in Chapter 4.

Next to the visual evaluation of the regression methods, a quantitative evaluation also exists. The models can be compared quantitatively by computing the log-likelihood of the fits as mentioned in Section 4.3. Table 7 presents the log-likelihoods of each model for the datasets 1 up until 4. The Python code that was used for the computation of these values can be found in Appendix C. It appears that for datasets 1 up until 3, the implementation of Gaussian Process regression, indeed improved the water retention curves, as the log-likelihood values for this model are higher which is shown in Table 7. For dataset 4, the log-likelihood did not increase but stayed the same. This means that the fit did not improve or worsen when Gaussian Process regression was implemented. This confirms the visual assessment made previously regarding Figure 5.10.

Table 7

Log-Likelihoods for the Non-Linear Least Squares Regression Model and the Gaussian Process Regression Model, for Datasets 1 up until 4.

Dataset	Log-Likelihood with only Least Squares	Log-Likelihood with Gaussian Processes
1	109.89	126.29
2	47.16	52.96
3	33.52	37.44
4	46.86	46.86

6

Discussion

For this research, Gaussian Process regression was performed on the residuals that resulted from Non-Linear Least Squares regression. Subsequently, the fit obtained from this method was added to the Non-Linear Least Squares fit in order to obtain a new fit together with a new confidence interval. It appears that in most cases, the implementation of Gaussian Process regression on the representation of water retention curves, is beneficial. The flexible property of the Gaussian Process model plays an important factor in the improvement of water retention curves.

A suggestion for further research, is to perform Gaussian Process regression on water retention data instead of the residuals that resulted from Least Squares fit. The water retention fit will then be completely determined by the Gaussian Process. An obstacle for this method is that the physical limitations of water retention curves are not considered. An example of a physical limitation is that the moisture content cannot increase when the pressure that pushes the water out of the soil pores also increases. This can however be solved by implementing constraints to the Gaussian Process regression model, which forces the fit to follow a logical path. A question that arises from this approach, is the accuracy and validity of such a fit when these constraints are implemented. This also needs to be investigated.

Furthermore, the analysis in this report was performed on four datasets. These datasets were retrieved by conducting experiments under laboratory drying conditions. Even though that these datasets represent a large portion of the water retention data available, it might be valuable to perform this analysis on a larger number of datasets and under different types of experimental conditions. A different soil database can be consulted as the UNSODA soil hydraulic database does not contain sufficient qualitative datasets for all the measurement techniques.

Finally, it would be interesting to make use of different models for the Non-Linear Least Squares method. In this report, the model that was used for the Non-Linear Least Squares method, is the van Genuchten model. This model was then combined with the Gaussian Process regression model in order to obtain a final fit. Different models that can be considered include the Brooks-Corey model and the lognormal distribution model of Kosugi, discussed in Section (2.1).

7

Conclusion

The aim of this report was to investigate whether water retention curves can be improved with the help of Gaussian Processes. Due to the breakdown of traditional models in certain situations, Gaussian Process regression analysis was performed to evaluate the improvement of water retention curve representations in terms of curve fitting and uncertainty analysis. This method was then compared to the Non-Linear Least Squares method in order to draw a valid conclusion on the effects of Gaussian Processes for water retention curves.

For this analysis, four datasets from the UNSODA soil hydraulic database have been used to plot the water retention curves for the two different methods. Next to the water retention curve fits, the 95% confidence intervals and the log-likelihoods of the two methods have been computed as well.

For three out of four datasets, the implementation of Gaussian Processes did improve the representation of water retention curves. This is visible in the pF curve plots, where the fitted lines pass through the points more accurately. The water retention curve corresponding to dataset 3, significantly improved as the shape of this curve represented a more realistic water retention curve. Furthermore, the 95% confidence intervals did improve as well for the first three datasets. This improvement presents itself in the form of a smaller confidence interval and thus less uncertainty of the mean fit, and in a fluctuating confidence interval. This fluctuating confidence interval shows that when more data is introduced around a certain point, the uncertainty decreases and vice versa.

For the fourth dataset, the implementation of Gaussian Processes had no effect on the representation of water retention curves. The reason for this, is that the residuals showed weak correlation and that the Gaussian Process regression model could not fit a useful line through these points.

The log-likelihoods of the two methods for the datasets confirm these evaluations. The log-likelihood for the first three datasets is greater for the method with Gaussian Processes, while the log-likelihood for the last dataset is equal for both methods. As the log-likelihood represents how good a certain fit is, it can be stated that the water retention curves for the first three datasets did improve and that the water retention curve for the last dataset remained the same.

It can be concluded that for datasets with some degree of correlation between the residuals, Gaussian Processes do improve the representation of water retention curves in terms of curve fitting and uncertainty analysis. This finding will contribute to improving the modelling of water flow through soils.

A

SciPy Curve Fit

Python code for Non-Linear Least Squares Regression with the use of the SciPy Optimize Curve Fit package, for datasets 1 up until 4.

Dataset 1

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
```

mean and std calculation

```
#mean for errors
def mean(x,y):
    sum_of_mean = 0
    for i in range(len(x)):
        sum_of_mean += y[i]
    mean = sum_of_mean/len(x)
    return mean

#std for errors
def std(x,y):
    var = 0
    for i in range(len(x)):
        var += ((y[i] - mean(x, y))**2)/len(x)
    std = np.sqrt(var)
    return std

#std for curvefit
def std2(x,y):
    var = 0
    for i in range(len(x)):
        e = y[i] - h1(y1, thetas1, thetar1, alpha1, n1) [ int(x[i]/70000*200000) ]
        var +=  (e**2) / len(x)
    std = np.sqrt(var)
    return std
```

Data: Lab drying

```
theta1 = [0.578, 0.577, 0.576, 0.574, 0.573, 0.572, 0.57, 0.568, 0.566, 0.562, 0.559, 0.555,
          0.549, 0.545, 0.537, 0.524, 0.513, 0.5, 0.482, 0.468, 0.448, 0.425, 0.414, 0.396, 0.378]
preshead1 = [1, 2, 3, 5, 7, 10, 15, 20, 30, 50, 70, 100, 150, 200, 300,
             500, 700, 1000, 1500, 2000, 3000, 5000, 7000, 10000, 15000]
```

Curvefit Dataset

```
def h1(y, thetas, thetar, alpha, n):
    thetaa = thetar + (thetas - thetar) / (1 + (alpha * np.abs(y))**n)**(1-1/n)
    return thetaa



popt1, pcov1 = curve_fit(h1, preshead1, theta1, p0 = [0.5, 0.1, 0.02, 2], bounds = ([0,0,0,1],[0.7, 0.25, 1, 5]))
thetas1, thetar1, alpha1, n1 = popt1

y1 = (np.linspace(0.5,70000, 200000))

fig1 = plt.figure(figsize=[18,7])

#normal plot
plt.subplot(1,2,1)
plt.plot((preshead1), theta1,'ro', label = 'Measurements');
plt.plot((y1), h1(y1, thetas1, thetar1, alpha1, n1), label = 'Non-Linear Least Squares Best Fit');
plt.title('Hydraulic curve with Data set 1')
plt.xlabel('Water pressure y [cm]');
plt.ylabel('Moisture content');
plt.legend()


#log plot
plt.subplot(1,2,2)
plt.plot(np.log10(preshead1), theta1,'ro', label = 'Measurements');
plt.plot(np.log10(y1),h1(y1, thetas1, thetar1, alpha1, n1),  label = 'Non-Linear Least Squares Best Fit');
plt.title('pF curve for dataset 1');
plt.ylabel('Moisture content');
plt.xlabel('PF Value ( log|y| )');

plt.gca().fill_between(np.log10(y1), h1(y1, thetas1, thetar1, alpha1, n1) - 2 * std2(preshead1, theta1),
                       h1(y1, thetas1, thetar1, alpha1, n1) + 2 * std2(preshead1, theta1),
                       label= '95 % interval', color = '#dddddd')

plt.legend()

print('θs =',round(thetas1,4), 'θr =',round(thetar1,4), 'α =',round(alpha1,4), 'n =',round(n1,3));
print('The standard deviation is:', round(std2(preshead1, theta1),6))
```

Dataset 2

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
```

mean and std calculation

```
#mean for errors
def mean(x,y):
    sum_of_mean = 0
    for i in range(len(x)):
        sum_of_mean += y[i]
    mean = sum_of_mean/len(x)
    return mean

#std for errors
def std(x,y):
    var = 0
    for i in range(len(x)):
        var += ((y[i] - mean(x, y))**2)/len(x)
    std = np.sqrt(var)
    return std

#std for curvefit
def std2(x,y):
    var = 0
    for i in range(len(x)):
        e = y[i] - h1(y1, thetas1, thetar1, alpha1, n1) [ int(x[i]/70000*200000) ]
        var +=  (e**2) / len(x)
    std = np.sqrt(var)
    return std
```

Data: Lab drying

```
theta1 =    [0.447, 0.434, 0.417, 0.398, 0.378, 0.362, 0.335, 0.319, 0.308,
            0.299, 0.297, 0.296, 0.294]
preshead1 = [1, 5, 10, 20, 40, 80, 160, 345, 690,
            2000, 5000, 10000, 15000]
```

Curvefit Dataset

```
def h1(y, thetas, thetar, alpha, n):
    thetaa = thetar + (thetas - thetar) / (1 + (alpha * np.abs(y))**n)**(1-1/n)
    return thetaa



popt1, pcov1 = curve_fit(h1, preshead1, theta1, p0 = [0.5, 0.1, 0.02, 2], bounds = ([0,0,0,1],[0.7, 0.25, 1, 5]))
thetas1, thetar1, alpha1, n1 = popt1

y1 = (np.linspace(0.5,70000, 200000))

fig1 = plt.figure(figsize=[18,7])
```

```
#normal plot
plt.subplot(1,2,1)
plt.plot((preshead1), theta1,'ro', label = 'Measurements');
plt.plot((y1), h1(y1, thetas1, thetar1, alpha1, n1), label = 'Non-Linear Least Squares Best Fit');
plt.title('Hydraulic curve with Data set 2')
plt.xlabel('Water pressure y [cm]');
plt.ylabel('Moisture content');
plt.legend()


#log plot
plt.subplot(1,2,2)
plt.plot(np.log10(preshead1), theta1,'ro', label = 'Measurements');
plt.plot(np.log10(y1),h1(y1, thetas1, thetar1, alpha1, n1),  label = 'Non-Linear Least Squares Best Fit');
plt.title('pF curve for dataset 2');
plt.ylabel('Moisture content');
plt.xlabel('PF Value ( log|y| )');

plt.gca().fill_between(np.log10(y1), h1(y1, thetas1, thetar1, alpha1, n1) - 2 * std2(preshead1, theta1),
                       h1(y1, thetas1, thetar1, alpha1, n1) + 2 * std2(preshead1, theta1),
                       label= '95 % interval', color = '#dddddd')

plt.legend()

print('θs =',round(thetas1,4), 'θr =',round(thetar1,4), 'α =',round(alpha1,4), 'n =',round(n1,3));
print('The standard deviation is:', round(std2(preshead1, theta1),6))
```

Dataset 3

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
```

mean and std calculation

```
#mean for errors
def mean(x,y):
    sum_of_mean = 0
    for i in range(len(x)):
        sum_of_mean += y[i]
    mean = sum_of_mean/len(x)
    return mean

#std for errors
def std(x,y):
    var = 0
    for i in range(len(x)):
        var += ((y[i] - mean(x, y))**2)/len(x)
    std = np.sqrt(var)
    return std
```

```
#std for curvefit
def std2(x,y):
    var = 0
    for i in range(len(x)):
        e = y[i] - h1(y1, thetas1, thetar1, alpha1, n1) [ int(x[i]/70000*200000) ]
        var +=  (e**2) / len(x)
    std = np.sqrt(var)
    return std
```

Data: Lab drying

```
theta1 = [0.474,0.469, 0.431,0.427,0.423,0.417,0.41,0.396,0.385]
preshead1 = [1,2,19,48,200,500,1000,5000,15500]
```

Curvefit Dataset

```
def h1(y, thetas, thetar, alpha, n):
    thetaa = thetar + (thetas - thetar) / (1 + (alpha * np.abs(y))**n)**(1-1/n)
    return thetaa



popt1, pcov1 = curve_fit(h1, preshead1, theta1, p0 = [0.5, 0.1, 0.02, 2], bounds = ([0,0,0,1],[0.7, 0.25, 1, 5]))
thetas1, thetar1, alpha1, n1 = popt1

y1 = (np.linspace(0.5,70000, 200000))

fig1 = plt.figure(figsize=[18,7])
```

```
#normal plot
plt.subplot(1,2,1)
plt.plot((preshead1), theta1,'ro', label = 'Measurements');
plt.plot((y1), h1(y1, thetas1, thetar1, alpha1, n1), label = 'Non-Linear Least Squares Best Fit');
plt.title('Hydraulic curve with Data set 1')
plt.xlabel('Water pressure y [cm]');
plt.ylabel('Moisture content');
plt.legend()


#log plot
plt.subplot(1,2,2)
plt.plot(np.log10(preshead1), theta1,'ro', label = 'Measurements');
plt.plot(np.log10(y1),h1(y1, thetas1, thetar1, alpha1, n1),  label = 'Non-Linear Least Squares Best Fit');
plt.title('pF curve for dataset 3');
plt.ylabel('Moisture content');
plt.xlabel('PF Value ( log |y| )');

plt.gca().fill_between(np.log10(y1), h1(y1, thetas1, thetar1, alpha1, n1) - 2 * std2(preshead1, theta1),
                       h1(y1, thetas1, thetar1, alpha1, n1) + 2 * std2(preshead1, theta1),
                       label= '95 % interval', color = '#dddddd')

plt.legend()

print('θs =',round(thetas1,4), 'θr =',round(thetar1,4), 'α =',round(alpha1,4), 'n =',round(n1,3));
print('The standard deviation is:', round(std2(preshead1, theta1),6))
```

Dataset 4

```
# Dataset 3import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
```

mean and std calculation

```
#mean for errors
def mean(x,y):
    sum_of_mean = 0
    for i in range(len(x)):
        sum_of_mean += y[i]
    mean = sum_of_mean/len(x)
    return mean

#std for errors
def std(x,y):
    var = 0
    for i in range(len(x)):
        var += ((y[i] - mean(x, y))**2)/len(x)
    std = np.sqrt(var)
    return std
```

```
#std for curvefit
def std2(x,y):
    var = 0
    for i in range(len(x)):
        e = y[i] - h1(y1, thetas1, thetar1, alpha1, n1) [ int(x[i]/70000*200000) ]
        var +=  (e**2) / len(x)
    std = np.sqrt(var)
    return std
```

Data: Lab drying

```
theta1 = [0.3552,0.3543,0.3526,0.3495,0.3439,0.3436,0.3206,0.3011,0.2964,0.2819,0.2708,0.3084,0.2814,0.2808,0.2677]
preshead1 = [5,10,20,30,40,50,100,200,300,350,500,1000,2000,4000,15000]
```

Curvefit Dataset

```
def h1(y, thetas, thetar, alpha, n):
    thetaa = thetar + (thetas - thetar) / (1 + (alpha * np.abs(y))**n)**(1-1/n)
    return thetaa



popt1, pcov1 = curve_fit(h1, preshead1, theta1, p0 = [0.5, 0.1, 0.02, 2], bounds = ([0,0,0,1],[0.7, 0.25, 1, 5]))
thetas1, thetar1, alpha1, n1 = popt1

y1 = (np.linspace(0.5,70000, 200000))

fig1 = plt.figure(figsize=[18,7])
```

```
#normal plot
plt.subplot(1,2,1)
plt.plot((preshead1), theta1,'ro', label = 'Measurements');
plt.plot((y1), h1(y1, thetas1, thetar1, alpha1, n1), label = 'Non-Linear Least Squares Best Fit');
plt.title('Hydraulic curve with Data set 1')
plt.xlabel('Water pressure y [cm]');
plt.ylabel('Moisture content');
plt.legend()


#log plot
plt.subplot(1,2,2)
plt.plot(np.log10(preshead1), theta1,'ro', label = 'Measurements');
plt.plot(np.log10(y1),h1(y1, thetas1, thetar1, alpha1, n1),  label = 'Non-Linear Least Squares Best Fit');
plt.title('pF curve for dataset 4');
plt.ylabel('Moisture content');
plt.xlabel('PF Value ( log |y| )');

plt.gca().fill_between(np.log10(y1), h1(y1, thetas1, thetar1, alpha1, n1) - 2 * std2(preshead1, theta1),
                       h1(y1, thetas1, thetar1, alpha1, n1) + 2 * std2(preshead1, theta1),
                       label= '95 % interval', color = '#dddddd')

plt.legend()

print('θs =',round(thetas1,4), 'θr =',round(thetar1,4), 'α =',round(alpha1,4), 'n =',round(n1,3));
print('The standard deviation is:', round(std2(preshead1, theta1),6))
```

B

GPy for Gaussian Process Regression

Python code for Gaussian Process Non-Linear Regression with the use of the GPy package, for datasets 1 up until 4.

Dataset 1

Residuals

```
# data points - best fit line

def residuals(preshead, thetas, thetar, alpha, n, theta):
    fit = np.zeros(len(preshead))
    for i in range(len(preshead)):
        fit[i] = h1(preshead[i], thetas, thetar, alpha, n)
    residuals = theta - fit
    return residuals

residuals1 = residuals(preshead1, thetas1, thetar1, alpha1, n1, theta1)


fig4 = plt.figure(figsize = [9,6])

plt.gca().fill_between(np.linspace(-1,4.5), mean(preshead1, residuals1) - 2 * std(preshead1, residuals1),
                       mean(preshead1, residuals1) + 2 * std(preshead1, residuals1),
                       label= '95 % interval', color = '#dddddd')

plt.plot(np.log10(preshead1), residuals1, 'ro', label = 'Residuals');

plt.axhline(mean(preshead1, residuals1), color = 'r', label = 'Mean of Errors');


plt.title('Errors dataset 1 ')
plt.xlabel('Water pressure y [cm]')
plt.ylabel('Residuals ( Measurements - fit )')
plt.axis([-1,4.4,-0.02,0.02])
plt.legend(loc = 0);
```

GP Process through error points

```
import GPy;
```

```
#kernel
kernel = GPy.kern.RBF(input_dim=1)
```

```
#Data
X = np.log10(preshead1).reshape(-1,1)
Y = residuals1.reshape(-1,1)
```

```
m = GPy.models.GPRegression(X,Y,kernel)
```

```
from IPython.display import display
m['.*Gaussian_noise.variance'].constrain_bounded(1e-6, 0.0005);
display(m)
m.optimize(optimizer='scg',messages=False, max_iters=1000);
m.optimize_restarts(num_restarts = 10);
display(m)
```

```
#Plotting
fig = m.plot(figsize=[8,5], plot_density=False, legend = True, plot_data = False);
plt.plot(X, Y, 'ro', label= 'Data');
plt.title('Gaussian Process Hydraulic Curve fitting');
plt.ylabel('Moisture content');
plt.xlabel('PF Value ( log y )');
plt.legend(loc = 0);
```

Extracting data from GPy

```
Xt = np.log10(y1).reshape(-1,1)
```

```
figgpy = plt.figure(figsize=[9,6])
#plt.plot(Xt,m.predict(Xt)[0] + 2*((m.predict(Xt)[1])**0.5))
plt.plot(Xt,m.predict(Xt)[0],'g' ,label = 'Mean')
#plt.plot(Xt,m.predict(Xt)[0] -2*((m.predict(Xt)[1])**0.5));

Xta = Xt.reshape(-1,)

plt.gca().fill_between(Xta, m.predict(Xt)[0].reshape(-1,) + (2*((m.predict(Xt)[1].reshape(-1,))**0.5)),
                       m.predict(Xt)[0].reshape(-1,) -(2*((m.predict(Xt)[1].reshape(-1,))**0.5)),
                       label= '95% confidence', color = '#d4ebf2');
plt.plot(X, Y, 'ro', label= 'Data');
plt.title('Gaussian Process error fitting');
plt.ylabel('Error moisture content');
plt.xlabel('PF Value ( log y )');
plt.legend(loc = 1);
```

Adding mean of errors to curvefit ¶

```
#log plot
figures1 = plt.figure(figsize=[18,7])

plt.subplot(1,2,2)
plt.plot(np.log10(preshead1), theta1,'ro', label = 'Measurements');
plt.plot(np.log10(y1),h1(y1, thetas1, thetar1, alpha1, n1) + m.predict(Xt)[0].reshape(-1,),
         label = 'GP fit');
plt.title('Gaussian Process regression model');
plt.ylabel('Moisture content');
plt.xlabel('PF Value ( log y )');

plt.gca().fill_between(np.log10(y1),
            (h1(y1, thetas1, thetar1, alpha1, n1) + m.predict(Xt)[0].reshape(-1,)) -2*((m.predict(Xt)[1].reshape(-1,))**0.5),
            (h1(y1, thetas1, thetar1, alpha1, n1) + m.predict(Xt)[0].reshape(-1,)) +2*((m.predict(Xt)[1].reshape(-1,))**0.5),
            label= '95 % interval', color = '#dddddd')
plt.axis([-0.5, np.log10(100000), 0.2, 0.7])
plt.legend();

plt.subplot(1,2,1)
#log plot
plt.plot(np.log10(preshead1), theta1,'ro', label = 'Measurements');
plt.plot(np.log10(y1),h1(y1, thetas1, thetar1, alpha1, n1),  label = 'Non-Linear Least Squares Best Fit');
plt.title('Ordinary Least-Squares non-linear regression model');
plt.ylabel('Moisture content');
plt.xlabel('PF Value ( log y )');

plt.gca().fill_between(np.log10(y1), h1(y1, thetas1, thetar1, alpha1, n1) - 2 * std2(preshead1, theta1),
                       h1(y1, thetas1, thetar1, alpha1, n1) + 2 * std2(preshead1, theta1),
                       label= '95 % interval', color = '#dddddd')
plt.axis([-0.5, np.log10(100000), 0.2, 0.7])
plt.legend();
```

Dataset 2

Residuals

```
# data points - best fit line

def residuals(preshead, thetas, thetar, alpha, n, theta):
    fit = np.zeros(len(preshead))
    for i in range(len(preshead)):
        fit[i] = h1(preshead[i], thetas, thetar, alpha, n)
    residuals = theta - fit
    return residuals

residuals1 = residuals(preshead1, thetas1, thetar1, alpha1, n1, theta1)


fig4 = plt.figure(figsize = [9,6])

plt.gca().fill_between(np.linspace(-1,4.5), mean(preshead1, residuals1) - 2 * std(preshead1, residuals1),
                       mean(preshead1, residuals1) + 2 * std(preshead1, residuals1),
                       label= '95 % interval', color = '#dddddd')

plt.plot(np.log10(preshead1), residuals1, 'ro', label = 'Residuals');

plt.axhline(mean(preshead1, residuals1), color = 'r', label = 'Mean of Errors');


plt.title('Errors dataset 1 ')
plt.xlabel('Water pressure y [cm]')
plt.ylabel('Residuals ( Measurements - fit )')
plt.axis([-1,4.4,-0.02,0.02])
plt.legend(loc = 0);
```

GP Process through error points

```
import GPy;
```

```
#kernel
kernel = GPy.kern.RBF(input_dim=1)
```

```
#Data
X = np.log10(preshead1).reshape(-1,1)
Y = residuals1.reshape(-1,1)
```

```
m = GPy.models.GPRegression(X,Y,kernel)
```

```
from IPython.display import display
m['.*Gaussian_noise.variance'].constrain_bounded(1e-6, 0.0005);
display(m)
m.optimize(optimizer='scg',messages=False, max_iters=1000);
m.optimize_restarts(num_restarts = 10);
display(m)
```

```
#Plotting

fig = m.plot(figsize=[8,5], plot_density=False, legend = True, plot_data = False);
plt.plot(X, Y, 'ro', label= 'Data');
plt.title('Gaussian Process Hydraulic Curve fitting');
plt.ylabel('Moisture content');
plt.xlabel('PF Value ( log y )');
plt.legend(loc = 0);
```

Extracting data from GPy

```
Xt = np.log10(y1).reshape(-1,1)
```

```
figgpy = plt.figure(figsize=[9,6])
#plt.plot(Xt,m.predict(Xt)[0] + 2*((m.predict(Xt)[1])**0.5))
plt.plot(Xt,m.predict(Xt)[0],'g' ,label = 'Mean')
#plt.plot(Xt,m.predict(Xt)[0] -2*((m.predict(Xt)[1])**0.5));

Xta = Xt.reshape(-1,)

plt.gca().fill_between(Xta, m.predict(Xt)[0].reshape(-1,) + (2*((m.predict(Xt)[1].reshape(-1,))**0.5)),
                       m.predict(Xt)[0].reshape(-1,) -(2*((m.predict(Xt)[1].reshape(-1,))**0.5)),
                       label= '95% confidence', color = '#d4ebf2');
plt.plot(X, Y, 'ro', label= 'Data');
plt.title('Gaussian Process error fitting');
plt.ylabel('Error moisture content');
plt.xlabel('PF Value ( log y )');
plt.legend(loc = 1);
```

Adding mean of errors to curvefit

```
#log plot
figures1 = plt.figure(figsize=[18,7])

plt.subplot(1,2,2)
plt.plot(np.log10(preshead1), theta1,'ro', label = 'Measurements');
plt.plot(np.log10(y1),h1(y1, thetas1, thetar1, alpha1, n1) + m.predict(Xt)[0].reshape(-1,),
         label = 'Fit');
plt.title('Gaussian Process regression model');
plt.ylabel('Moisture content');
plt.xlabel('PF Value ( log y )');

plt.gca().fill_between(np.log10(y1),
            (h1(y1, thetas1, thetar1, alpha1, n1) + m.predict(Xt)[0].reshape(-1,)) -2*((m.predict(Xt)[1].reshape(-1,))**0.5),
            (h1(y1, thetas1, thetar1, alpha1, n1) + m.predict(Xt)[0].reshape(-1,)) +2*((m.predict(Xt)[1].reshape(-1,))**0.5),
            label= '95 % interval', color = '#dddddd')
plt.axis([-0.5, np.log10(100000), 0.23, 0.51])
plt.legend();

plt.subplot(1,2,1)
#log plot
plt.plot(np.log10(preshead1), theta1,'ro', label = 'Measurements');
plt.plot(np.log10(y1),h1(y1, thetas1, thetar1, alpha1, n1),  label = 'Non-Linear Least Squares Best Fit');
plt.title('pF curve for dataset 2');
plt.ylabel('Moisture content');
plt.xlabel('PF Value ( log|y| )');

plt.gca().fill_between(np.log10(y1), h1(y1, thetas1, thetar1, alpha1, n1) - 2 * std2(preshead1, theta1),
                       h1(y1, thetas1, thetar1, alpha1, n1) + 2 * std2(preshead1, theta1),
                       label= '95 % interval', color = '#dddddd')
plt.axis([-0.5, np.log10(100000), 0.23, 0.51])
plt.legend();
```

Dataset 3

Residuals

```
# data points - best fit line

def residuals(preshead, thetas, thetar, alpha, n, theta):
    fit = np.zeros(len(preshead))
    for i in range(len(preshead)):
        fit[i] = h1(preshead[i], thetas, thetar, alpha, n)
    residuals = theta - fit
    return residuals

residuals1 = residuals(preshead1, thetas1, thetar1, alpha1, n1, theta1)


fig4 = plt.figure(figsize = [9,6])

plt.gca().fill_between(np.linspace(-1,4.5), mean(preshead1, residuals1) - 2 * std(preshead1, residuals1),
                       mean(preshead1, residuals1) + 2 * std(preshead1, residuals1),
                       label= '95 % interval', color = '#dddddd')

plt.plot(np.log10(preshead1), residuals1, 'ro', label = 'Residuals');

plt.axhline(mean(preshead1, residuals1), color = 'r', label = 'Mean of Errors');


plt.title('Errors dataset 3 ')
plt.xlabel('Water pressure y [cm]')
plt.ylabel('Residuals ( Measurements - fit )')
plt.axis([-1,4.4,-0.02,0.02])
plt.legend(loc = 0);
```

GP Process through error points

```
import GPy;
```

```
#kernel
kernel = GPy.kern.RBF(input_dim=1)
```

```
#Data
X = np.log10(preshead1).reshape(-1,1)
Y = residuals1.reshape(-1,1)
```

```
m = GPy.models.GPRegression(X,Y,kernel)
```

```
from IPython.display import display
m['.*Gaussian_noise.variance'].constrain_bounded(1e-6, 0.0005);
display(m)
m.optimize(optimizer='scg',messages=False, max_iters=1000);
m.optimize_restarts(num_restarts = 10);
display(m)
```

```
#Plotting

fig = m.plot(figsize=[8,5], plot_density=False, legend = True, plot_data = False);
plt.plot(X, Y, 'ro', label= 'Data');
plt.title('Gaussian Process Hydraulic Curve fitting');
plt.ylabel('Moisture content');
plt.xlabel('PF Value ( log y )');
plt.legend(loc = 0);
```

Extracting data from GPy

```
Xt = np.log10(y1).reshape(-1,1)
```

```
figgpy = plt.figure(figsize=[9,6])
#plt.plot(Xt,m.predict(Xt)[0] + 2*((m.predict(Xt)[1])**0.5))
plt.plot(Xt,m.predict(Xt)[0],'g' ,label = 'Mean')
#plt.plot(Xt,m.predict(Xt)[0] -2*((m.predict(Xt)[1])**0.5));

Xta = Xt.reshape(-1,)

plt.gca().fill_between(Xta, m.predict(Xt)[0].reshape(-1,) + (2*((m.predict(Xt)[1].reshape(-1,))**0.5)),
                       m.predict(Xt)[0].reshape(-1,) -(2*((m.predict(Xt)[1].reshape(-1,))**0.5)),
                       label= '95% confidence', color = '#d4ebf2');
plt.plot(X, Y, 'ro', label= 'Data');
plt.title('Gaussian Process error fitting');
plt.ylabel('Error moisture content');
plt.xlabel('PF Value ( log y )');
plt.legend(loc = 1);
```

Adding mean of errors to curvefit

```
#log plot
figures1 = plt.figure(figsize=[18,7])

plt.subplot(1,2,2)
plt.plot(np.log10(preshead1), theta1,'ro', label = 'Measurements');
plt.plot(np.log10(y1),h1(y1, thetas1, thetar1, alpha1, n1) + m.predict(Xt)[0].reshape(-1,),
         label = 'Fit');
plt.title('Gaussian Process regression model');
plt.ylabel('Moisture content');
plt.xlabel('PF Value ( log y )');

plt.gca().fill_between(np.log10(y1),
            (h1(y1, thetas1, thetar1, alpha1, n1) + m.predict(Xt)[0].reshape(-1,)) -2*((m.predict(Xt)[1].reshape(-1,))**0.5),
            (h1(y1, thetas1, thetar1, alpha1, n1) + m.predict(Xt)[0].reshape(-1,)) +2*((m.predict(Xt)[1].reshape(-1,))**0.5),
            label= '95 % interval', color = '#dddddd')
plt.axis([-0.5, 5, 0.35, 0.5])
plt.legend();

plt.subplot(1,2,1)
plt.plot(np.log10(preshead1), theta1,'ro', label = 'Measurements');
plt.plot(np.log10(y1),h1(y1, thetas1, thetar1, alpha1, n1),  label = 'Non-Linear Least Squares Best Fit');
plt.title('Ordinary Least-Squares non-linear regression model');
plt.ylabel('Moisture content');
plt.xlabel('PF Value ( log y )');

plt.gca().fill_between(np.log10(y1), h1(y1, thetas1, thetar1, alpha1, n1) - 2 * std2(preshead1, theta1),
                       h1(y1, thetas1, thetar1, alpha1, n1) + 2 * std2(preshead1, theta1),
                       label= '95 % interval', color = '#dddddd')
plt.axis([-0.5, 5, 0.35, 0.5])

plt.legend();
```

Dataset 4

Residuals (white noise errors)

```
# data points - best fit line

def residuals(preshead, thetas, thetar, alpha, n, theta):
    fit = np.zeros(len(preshead))
    for i in range(len(preshead)):
        fit[i] = h1(preshead[i], thetas, thetar, alpha, n)
    residuals = theta - fit
    return residuals

residuals1 = residuals(preshead1, thetas1, thetar1, alpha1, n1, theta1)


fig4 = plt.figure(figsize = [9,6])

plt.gca().fill_between(np.linspace(-1,4.5), mean(preshead1, residuals1) - 2 * std(preshead1, residuals1),
                       mean(preshead1, residuals1) + 2 * std(preshead1, residuals1),
                       label= '95 % interval', color = '#dddddd')

plt.plot(np.log10(preshead1), residuals1, 'ro', label = 'Residuals');

plt.axhline(mean(preshead1, residuals1), color = 'r', label = 'Mean of Errors');


plt.title('Errors dataset 1 ')
plt.xlabel('Water pressure y [cm]')
plt.ylabel('Residuals ( Measurements - fit )')
plt.axis([-1,4.4,-0.045,0.045])
plt.legend(loc = 0);
```

GP Process through error points

```
import GPy;
```

```
#kernel
kernel = GPy.kern.RBF(input_dim=1)
```

```
#Data
X = np.log10(preshead1).reshape(-1,1)
Y = residuals1.reshape(-1,1)
```

```
m = GPy.models.GPRegression(X,Y,kernel)
```

```
from IPython.display import display
m['.*Gaussian_noise.variance'].constrain_bounded(1e-6, 0.0005);
display(m)
m.optimize(optimizer='scg',messages=False, max_iters=1000);
m.optimize_restarts(num_restarts = 10);
display(m)
```

```
#Plotting

fig = m.plot(figsize=[8,5], plot_density=False, legend = True, plot_data = False);
plt.plot(X, Y, 'ro', label= 'Data');
plt.title('Gaussian Process Hydraulic Curve fitting');
plt.ylabel('Moisture content');
plt.xlabel('PF Value ( log y )');
plt.legend(loc = 0);
```

Extracting data from GPy

```
Xt = np.log10(y1).reshape(-1,1)
```

```
figgpy = plt.figure(figsize=[9,6])
#plt.plot(Xt,m.predict(Xt)[0] + 2*((m.predict(Xt)[1])**0.5))
plt.plot(Xt,m.predict(Xt)[0],'g' ,label = 'Mean')
#plt.plot(Xt,m.predict(Xt)[0] -2*((m.predict(Xt)[1])**0.5));

Xta = Xt.reshape(-1,)

plt.gca().fill_between(Xta, m.predict(Xt)[0].reshape(-1,) + (2*((m.predict(Xt)[1].reshape(-1,))**0.5)),
                       m.predict(Xt)[0].reshape(-1,) -(2*((m.predict(Xt)[1].reshape(-1,))**0.5)),
                       label= '95% confidence', color = '#d4ebf2');
plt.plot(X, Y, 'ro', label= 'Data');
plt.title('Gaussian Process error fitting');
plt.ylabel('Error moisture content');
plt.xlabel('PF Value ( log y )');
plt.legend(loc = 1);
```

Adding mean of errors to curvefit

```
#Log plot
figures1 = plt.figure(figsize=[18,7])

plt.subplot(1,2,2)
plt.plot(np.log10(preshead1), theta1,'ro', label = 'Measurements');
plt.plot(np.log10(y1),h1(y1, thetas1, thetar1, alpha1, n1) + m.predict(Xt)[0].reshape(-1,),
         label = 'Fit');
plt.title('Gaussian Process regression model');
plt.ylabel('Moisture content');
plt.xlabel('PF Value ( log y )');

plt.gca().fill_between(np.log10(y1),
            (h1(y1, thetas1, thetar1, alpha1, n1) + m.predict(Xt)[0].reshape(-1,)) -2*((m.predict(Xt)[1].reshape(-1,))**0.5),
            (h1(y1, thetas1, thetar1, alpha1, n1) + m.predict(Xt)[0].reshape(-1,)) +2*((m.predict(Xt)[1].reshape(-1,))**0.5),
            label= '95 % interval', color = '#dddddd')
plt.axis([-0.5,5,0.2,0.42])
plt.legend();

plt.subplot(1,2,1)
plt.plot(np.log10(preshead1), theta1,'ro', label = 'Measurements');
plt.plot(np.log10(y1),h1(y1, thetas1, thetar1, alpha1, n1),  label = 'Non-Linear Least Squares Best Fit');
plt.title('Ordinary Least-Squares non-linear regression model');
plt.ylabel('Moisture content');
plt.xlabel('PF Value ( log y )');

plt.gca().fill_between(np.log10(y1), h1(y1, thetas1, thetar1, alpha1, n1) - 2 * std2(preshead1, theta1),
                       h1(y1, thetas1, thetar1, alpha1, n1) + 2 * std2(preshead1, theta1),
                       label= '95 % interval', color = '#dddddd')
plt.axis([-0.5,5,0.2,0.42])
plt.legend();
```

C

Log-likelihood Python Code

Python code for computing the Log-likelihood of the Gaussian Process Regression model and the Non-Linear Least Squares Regression model, for datasets 1 up until 4.

Dataset 1

Log Likelihood

```
from scipy.stats import norm

loglikelihood = 0
for i in range(len(theta1)):
    loglikelihood += norm.logpdf(theta1[i], loc=h1(y1, thetas1, thetar1, alpha1, n1) [ int(preshead1[i]/70000*200000) ],
                  scale=std2(preshead1, theta1))
print("The log likelihood of the OLS method is:",loglikelihood)
print('"The log likelihood with the GP method is:', 126.29293273857694 ) #Read off of GPy display output
```

Dataset 2

Log Likelihood

```
from scipy.stats import norm

loglikelihood = 0
for i in range(len(theta1)):
    loglikelihood += norm.logpdf(theta1[i], loc=h1(y1, thetas1, thetar1, alpha1, n1) [ int(preshead1[i]/70000*200000) ],
                  scale=std2(preshead1, theta1))
print("The log likelihood of the OLS method is:",loglikelihood)
print('"The log likelihood with the GP method is:', 52.96257251449306 ) #Read off of GPy display output
```

Dataset 3

Log Likelihood

```
from scipy.stats import norm

loglikelihood = 0
for i in range(len(theta1)):
    loglikelihood += norm.logpdf(theta1[i], loc=h1(y1, thetas1, thetar1, alpha1, n1) [ int(preshead1[i]/70000*200000) ],
                     scale=std2(preshead1, theta1))
print("The log likelihood of the OLS method is:",loglikelihood)
print('"The log likelihood with the GP method is:', 37.439857174489134 ) #Read off of GPy display output
```

Dataset 4

Log Likelihood

```
from scipy.stats import norm

loglikelihood = 0
for i in range(len(theta1)):
    loglikelihood += norm.logpdf(theta1[i], loc=h1(y1, thetas1, thetar1, alpha1, n1) [ int(preshead1[i]/70000*200000) ],
                     scale=std2(preshead1, theta1))
print("The log likelihood of the OLS method is:",loglikelihood)
print('"The log likelihood with the GP method is:', 46.86584653565418 ) #Read off of GPy display output
```

Bibliography

Anonymous. (2019). Applications of Gaussian Processes in Finance. *International Conference on Learning Representations 2019.*

Bailey, K. (2016, August 6). *Gaussian Processes for Dummies*. Retrieved from http://katbailey.github.io/post/gaussian-processes-for-dummies/

Boyd, S. (2016, December). *Nonlinear Least Squares [Powerpoint lecture slides]*. (Stanford University) Retrieved from https://stanford.edu/class/ee103/lectures/nlls_slides.pdf

Duvenaud, D. (2014). *The Kernel Cookbook: Advice on Covariance functions*. Retrieved from https://www.cs.toronto.edu/~duvenaud/cookbook/

Evelinag. (2014). *Covariance functions*. Retrieved from http://evelinag.com/Ariadne/covarianceFunctions.html

Luckner, L., van Genuchten, M., & Nielsen, D. (1989, October). A consistent set of parametric models for two-phase flow of immiscible fluids in the subsurface. *Water Resources Research, 25*(10), 2187-2193. doi:10.1029/WR025i010p02187

National Agriculture Library. (2015, March). *UNSODA 2.0: Unsaturated Soil Hydraulic Database. Database and program for indirect methods of estimating unsaturated hydraulic properties*. Retrieved from https://data.nal.usda.gov/dataset/unsoda-20-unsaturated-soil-hydraulic-database-database-and-program-indirect-methods-estimating-unsaturated-hydraulic-properties

Rao, S. (2011). Wetting and Drying, Effect on Soil Physical Properties. In: Gliński J., Horabik J., Lipiec J. (eds) Encyclopedia of Agrophysics. Encyclopedia of Earth Sciences Series. Springer, Dordrecht.

Rasmussen, C., & Williams, C. (2006). *Gaussian Processes for Machine Learning.* Massachusetts, Cambridge: the MIT press.

Roelants, P. (2019, January 5). *Understanding Gaussian processes*. Retrieved from https://peterroelants.github.io/posts/gaussian-process-tutorial/

Schoups, G. (2019). *Water System Analysis: Soil Water Flow [Powerpoint slides]*. Retrieved from https://brightspace.tudelft.nl/d2l/le/content/126237/viewContent/1025320/View

Seki, K. (2007). *SWRC fit - a nonlinear fitting program with a water retention curve for soils having unimodal and bimodal pore structure.* Retrieved from http://swrcfit.sourceforge.net/

Sheffield ML Group. (2015). *GPy*. Retrieved from https://sheffieldml.github.io/GPy/

Snelson, E. (2006, October). Tutorial: Gaussian process models [Powerpoints slides]. London, United Kingdom.

Stansbury, D. (2012, September 1). *Derivation: Ordinary Least Squares Solution and Normal Equations*. Retrieved from https://theclevermachine.wordpress.com/2012/09/01/derivation-of-ols-normal-equations/

Toews, M. (2007). *Standard deviation diagram Wikipedia*. Retrieved from https://upload.wikimedia.org/wikipedia/commons/8/8c/Standard_deviation_diagram.svg

van Genuchten, M., Leij, F., & Yates, S. (1991). *The RETC Code for Quantifying the Hydraulic Functions.* California.

van Genuchten, M., Simunek, J., Leij, F., & Sejna, M. (1998, December). *PC Progress RETC Program.* Retrieved from https://www.pc-progress.com/en/Default.aspx?retc

Winterstein, D. (2016, May). *A Simple Intro to Gaussian Processes (a great data modelling tool)*. Retrieved from http://platypusinnovation.blogspot.com/2016/05/a-simple-intro-to-gaussian-processes.html