TKP4580 - Specialization Project, Process Systems Engineering

Hybrid Modeling with Machine Learning and First Principles Models

Ding Nan

Submission date:	19.12.2019
Supervisors:	Johannes Jäschke
Co-supervisors:	Timur Bikmukhametov

NTNU
Norwegian University of
Science and Technology
Department of Chemical Engineering

Summary

As a popular, powerful tool, machine learning model (MLM) has a wide range of applications from computer versions to chemical process systems. However, the main drawback of machine learning algorithm is it is hard to explain the behavior behind its algorithm and it always approximates the input and output variables without providing insights about the system behavior. On the other hand, first principles model (FPM) which is based on physics behind the phenomena, is able to provide good insight of the system behavior. But it is hard to derive and requires empirical calibration to real systems. Since both MLM and FPM have their own corresponding advantages and disadvantages, the combination of these two models, which is called hybrid model, is the main modeling approach in this project. By building up electric submersible pump's (ESP) first principle model from ESP's head prediction procedures, the first principles model will be the main part of hybrid model to model multiphase flow in oil well and estimate oil production in oil and gas production system and futhur create accurate and explainable models by using prior knowledge of the process and measurement data. In order to study the abilities of different models, pure machine learning and hybrid model, and the effects of different datasets, this project will be implemented in different three cases.

Preface

This report was written as a part of the course TKP4580 Chemical Engineering, Specialization Project during autumn 2019 in the second year of the chemical engineering master degree at Norwegian University of Science and Technology. I would like to express my deep gratitude to my supervisor Associate Professor Johannes Jäschke for guidance, advice and valuable discussions during my work. At the meantime, I would like to sincerely appreciate my co-supervisor, Ph.D Candidate Timur Bikmukhametov, who really pushed me hard to learn and taught me how to think, provided me huge helps from guidance to advice during the summer vocation and whole semester as well as valuable feedback during when writing this report. Sincerely, I have learnt a lot from this specialisation project with your help. I am grateful to my parents who always support me to work hard and improve. I am also grateful to all my friends for giving me great mental supports. And I also need to thank Jithin Gopakumar and Haakon Svane provided me great interpretations on machine learning part.

Table of Contents

List of Tables

List of Figures

Abbreviations

ESP	=	Electric submersible pump
PVT	=	Pressure-volume-temperature
IPR	=	Inflow performance relationship
FPM	=	First principles model
MLM	=	Machine learning model

Chapter 1

Introduction

In this project, the implementation of hybrid model (grey box model) is the combination of first principle model with machine learning algorithms. The focus of the project is on modeling of multiphase flow and estimation of oil production in oil and gas production system. The main objective of the project is to build up precise and explainable hybrid model using prior knowledge of the process and data measurements from OLGA oil well model.

As a popular subset of artificial intelligence, the application of machine learning is becoming more and more widespread in chemical process model identification from computer vision applications such as high-level understanding from digital images or videos. Despite machine learning algorithms have a wide variety of applications due to its apparent and powerful advantages, it still have deficiencies when applied in chemical process systems. For example, the input and output variables of the chemical process system would be directly approximated by machine learning algorithm without doing or providing deeper insight into the real physical behavior of the process system. As a result, the shortage would occur as insufficient explanation of the behavior behind the its algorithm.

In order to complement the drawback of machine learning applications in chemical process systems, hybrid model, which is the combination of first principles model with machine learning, would be used in this project to explore its capabilities. In this project, the first principle model is derived from the head performance prediction of electric submersible pump and calculating oil production from ESP's first principle model to let it become an important feature of training dataset to set up hybrid model. Pump performance, however, is significantly affected by the presence of free gas or high-viscosity fluids. So, necessary head performance correction will be considered and implemented in this project, in order to get more reliable performance.

1.1 OLGA

In order to set up a oil well model, which is used to produce data such as pressure, temperature and flow rate for machine learning part of the hybrid model, OLGA dynamic multiphase flow simulator is applied in this project. Furthermore, MatrikonOPC explorer is also used to obtain service data from OLGA by setting SIMULATORMODEL as external and STOPATENDTIME as OFF. By coding in MATLAB can finally read data from OPC server to simulate the oil well production process.
The initial layout of oil well model without artificial lift is illustrated Figure 1.1 by using OLGA and layout values can be found from Table 1.1. As a $1500m$ vertical well, oil is pushed to the production surface through a single pipe which have 10 sections.

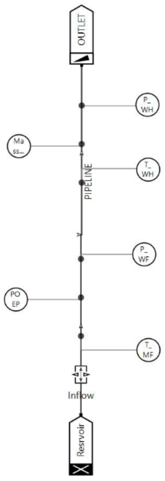


Figure 1.1: Initial oil well without artificial lift in OLGA

Branches	PIPLINE
No. of Pipes	1
No. of Sections	10
Diameter	0.12 [m]
Roughness	5x10$^{-5}[m]$
Length	1500 [m]
Elevation	1500 [m]

Table 1.1: Well layout

1.2 Electric Submersible Pump

When the natural drive energy of the reservoir is not strong sufficient to push the oil to the surface, or when the production rate is too low to be economic, artificial lift is employed to recover more production. So, artificial lift is used on oil wells to increase pressure within the reservoir and encourage oil to the surface.
In this project, the reason why have to use artificial lift is that the pressure of reservoir is not enough to produce oil along 1500 m vertical oil well to the surface. From figure 1.2, can see that the oil rate at wellhead without artificial lift is lower than 0 m^3/d except from beginning period and this phenomena leads to initial negative flow, and the steady state preprocessor can not be further converged in OLGA . So, it means the reservoir pressure, 100 bar, is not sufficient to encourage oil to the surface to get normal oil production. (Figure 1.2 is captured from simulator model in order to get a converged oil well model from OLGA)

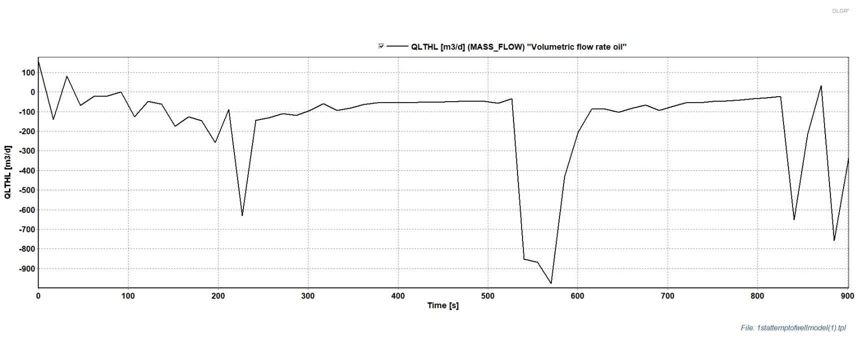


Figure 1.2: Oil flow rate without artificial lift [m$^3/d$]

So, after doing oil well simulation without artificial lift, one decision have to be made to get normal oil production, which is applying artificial lift method. In this project, artificial lift is implemented by electric submersible pump (ESP). As the second most widely used artificial lift method, electric submersible pump (ESP), which employs downhole centrifugal pump driven by a three phase, electric motor supplied with electric power via a cable run from the surface on the outside of the tubing, has a long application history in the oil and gas industry. The main purpose of ESP is to maintain and also increase production flow rate by converting kinetic energy to hydraulic pressure of hydrocarbon fluid. The pressure increment mechanism is achieved by the change of area from impeller intake to impeller discharge, which has an increasing trend, figure1.3. It is a practical illustration of Bernoulli's principle, which is explained as Eq.1.1 .

$$Areaincrease = Flowratedecrease = Pressureincrease \tag{1.1}$$

After implementing plenty of ESP pump models in OLGA, REDA H22500N version 1 is finally applied as the electic submersible pump in this project. By using this pump model

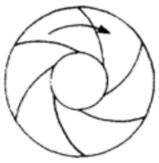


Figure 1.3: Impeller diagram

the steady state preprocessor is converged and the well successfully starts to produce oil to the wellhead. Several other pump models are also working for producing oil, however, due to the main propose of the project is not focused on the effects on oil production from diverse pump types, the study and analysis of different ESP models is not considered in this project. But, in fact, REDA H22500N version 1 has the best oil production performance in all pump models which are tested in this project. The pump details can be found from Table 1.3. The actual head performance curve from manufacture is shown in Figure 1.4 (Takacs, 2018).

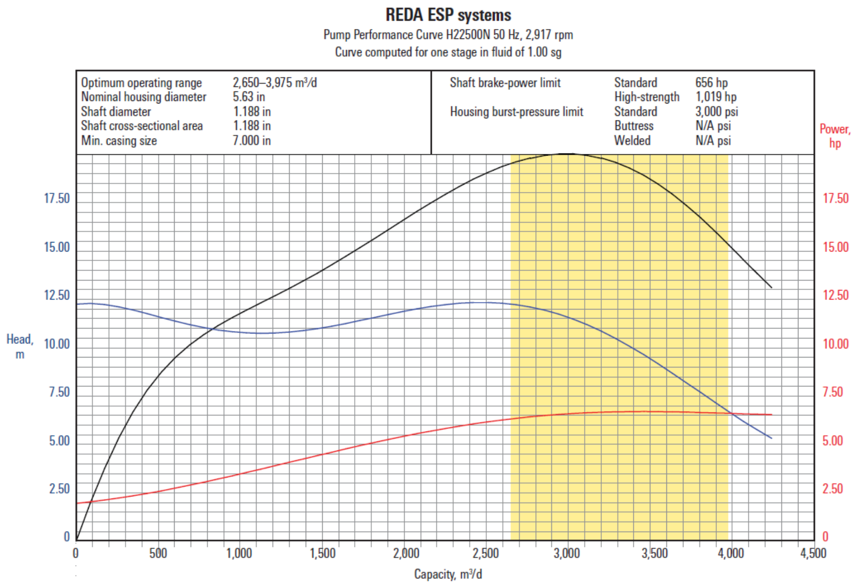


Figure 1.4: REDA H22500N, 50Hz, 2917rpm, performance curve

where the blue solid line indicates pump head, and the rectangular area filled by yellow implies best efficiency point area.

In order to produce oil normally, the pump rotational speed has to be controlled by using manual controller in OLGA, by defining and exposing the speed (SpeedSig) to ESP

Pump model	REDA H22500N version 1
No. of stages	26
Absolute position	200 [m]
Pump speed ω	3155-3500 [rpm]

Table 1.2: Pump details used in OLGA

block. By setting output signal setpoint, manual controller can control the pump speed. Manual controller output result and ESP rotational speed result are illustrated in figure1.5 and 1.6. When frequency is 50 Hz, the optimal pump rotational speed is 2917 rpm. But from OLGA operation experience, when REDA H22500N is in version 1, the frequency is 60 Hz and the optimal pump rotational speed is 3500 rpm, as shown in Figure 5.2 in appendix.

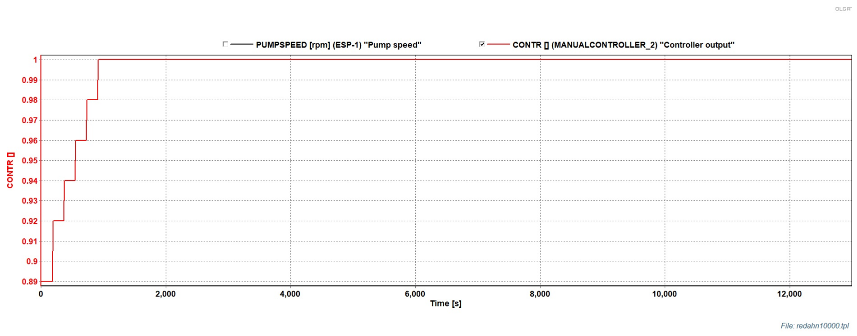


Figure 1.5: Manual controller output signal

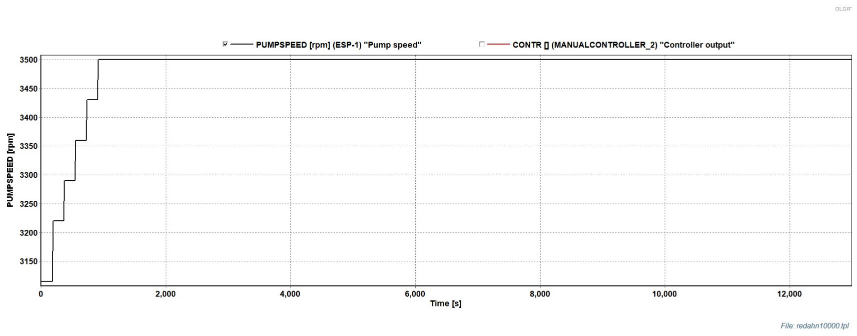


Figure 1.6: Pump rotational speed after control

The tidy version of oil well model with ESP as artificial lift is illustrated in Figure 1.7 and more specific version with all transmitters can be found from appendix 5.1.

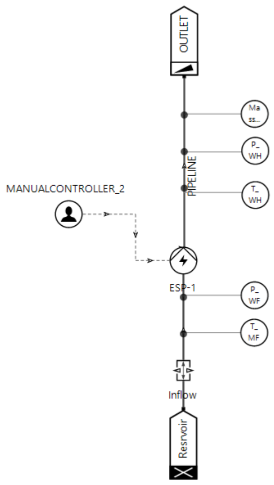


Figure 1.7: Oil well with ESP in OLGA

In order to obtain the first principle model of ESP, some important parameters like pump impeller geometric parameters are required. But since it is not possible to get the proper pump impeller geometry from pump manufacture, so in this project those parameters are collected from paper of Datong Sun (2006) about Single-Phase Model for Electric Submersible Pump (ESP) Head Performance. In Table 1.3, pump geometric values are presented.

Impeller intake blade angle β_1	38°
Impeller discharge blade angle β_2	23°
Impeller channel height h	0.01 [m]
Impeller entrance radius r_1	0.029 [m]
Impeller discharge radius r_2	0.048 [m]
Channel wall roughness ϵ	1x$10^{-4}[m]$

Table 1.3: Pump impeller geometries

Chapter 2

Modeling Approach

The oil field model is already built up by using OLGA dynamic multiphase flow simulator, which is stated and done in Chapter 1. Like introduced in Chapter 1, the modeling of oil well in this project is focused on oil field with single production well. However, the typical oil and gas production system is composed of plenty of wells which are connected to the flow line and have the ability to carry field production from reservoir to separator. As a simplification, choke and separator are neglected in this project, due to one of the main objective of this project is focused on oil production with electric submersible pump (ESP). So, it means the production flow rate can be manipulated by ESP, since the one of the goals of this project is to produce oil as much as possible .

After finishing oil well model establishment in OLGA and getting data from MATLAB simulation, the first principle model of ESP, which is used to calculate oil production rate by using physical properties behind the production system, will be obtained and further used in hybrid modeling part. Gaussian process regression for pure machine learning case and hybrid modeling cases are also needed in modeling part.

2.1 First Principles Model

In general, process models can be developed by using either knowledge-based or data-based methodologies (Low Soon Tiong, 1997). As a knowledge-based model, constructed using physical knowledge of the process, first principles model can provide good explanations and insight of the system behaviour, but it also have disadvantages which would affect its efficiencies in process modeling. For example, FPM is hard to derive, so it is expensive or even impossible to obtain. And it often requires empirical calibration to real systems. As a representative of data-based model, the disadvantage of black-box model is it always do not capture the physical reality and they are usually valid only in a limited range, however these models are much less expensive to obtain (Qiang Xiong, 2001).

So, hybrid model or so-called grey- box model would be a great combination method to expend advantages of both sides in chemical process systems.

2.1.1 Fluid properties model

In order to produce output data such as temperature and pressure from field model in OLGA, the model requires pre-generated pressure-volume-temperature (PVT) data which contains fluid properties under given conditions. In this project, PVT data file is given as three phase fluid properties. Some of the properties like density and viscosity are represented as follow.

density.jpg

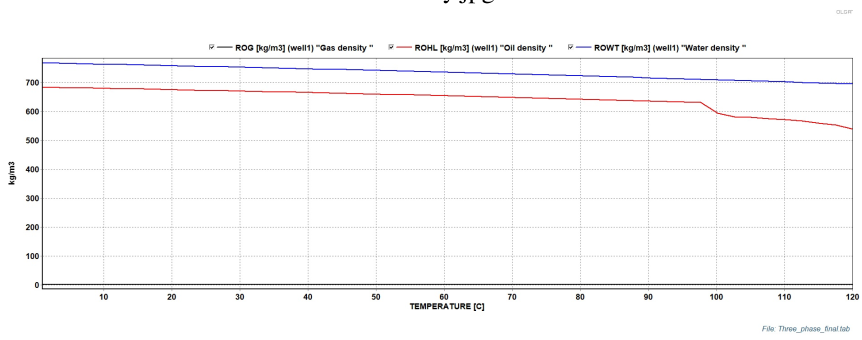


Figure 2.1: Fluid properties: density

viscosity.jpg

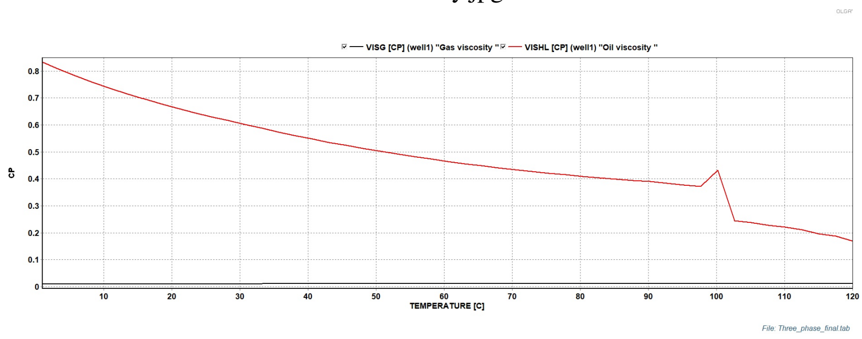


Figure 2.2: Fluid properties: viscosity

2.1.2 Production system model

After getting inspirations from the liturate of Timur Bikmukhametov (January 2020, 106487), the production system model in this project is classified into following two sub parts, reservior inflow model, thermal-hydraulic model and ESP model.

Reservior inflow model

The reservoir inflow model is usually represented by an Inflow Performance Relationship (IPR) model which defines the well production rate as a function of pressure difference at reservoir and bottomhole conditions (Timur Bikmukhametov, January 2020, 106487).
The simplest approach to describe the inflow performance of oil wells is the use of the productivity index, PI, concept (Takacs, 2018).

$$q = PI \times (P_R - P_{wf}) \tag{2.1}$$

where q is liquid rate, P_R is reservoir pressure, P_{wf} is bottom hle pressure and PI is productivity index.
In practice, however, oil wells with artificial lift would experience the degradation of bottomhole pressures and it would be lower than bubblepoint pressure. Thus, there is a free gas phase present in the reservoir near the wellbore, and the assumptions that were used to develop the PI equation are no longer valid (Takacs, 2018). In order to correct the above mentioned effect, Vogel's IPR correlation is selected and used in this project as an IPR in OLGA.

$$\frac{q}{q_{max}} = 1 - 0.2\frac{P_{wf}}{P_R} - 0.8(\frac{P_{wf}}{P_R})^2 \tag{2.2}$$

where q is production rate at bottomhole pressure, q is maximum production rate.

ESP's first principle model

After directly choosing Vogel's IPR correlation in OLGA, the first principle model which would be used in this project is ESP's first principle model. The main approach of getting ESP's first principle model can be specified as follows.(1) Compute Euler head. When the fluids enter the impeller without pre-rotation Euler head can be simplified into ideal Euler head. (2) Compute head loss caused by friction effect.(3) Compute shock loss based on pure water pump performance. (4) Compute accumulated ESP head performance before correction. (5) Implement viscosity correction. (6) Implement free gas correction. (7) Compute actual ESP head performance. Effect of losses in the head curve of a centrifugal pump is illustrated in Figure 2.3 (Tatiane Silva Vieira a, March 2015).

Thus, the actual pump head is the result after subtracting all the head losses from Euler head, which is given by

$$H = H_e - \sum H_{loss} \tag{2.3}$$

where H_{loss} is head loss in an ESP pump. In this project, head loss caused by friction and shock are considered.

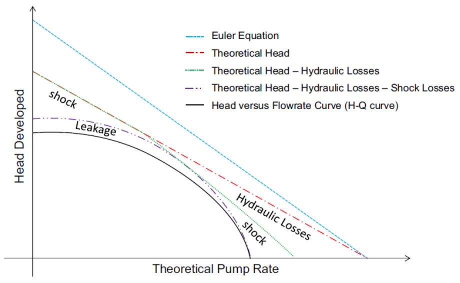


Figure 2.3: Pump head after deducting losses

Euler head: (Datong Sun, 2006)

$$He = \frac{\omega^2}{g}(r_2^2 - r_1^2) - \frac{Q\omega}{2\pi gh}(\frac{1}{\tan\beta_2} - \frac{1}{\tan\beta_1}) \tag{2.4}$$

Generally, the ESP head prediction starts from Euler head equations. Euler head is obtained based on the ideal assumptions about incompressible and frictionless fluids and infinite blades etc. So, Euler head is also the maximum head which ESP can develop. In this project Euler head is used as an estimation model to initially estimate the ideal boosting pressure of ESP.

When the fluids enter the impeller without pre-rotation, Euler head can be simplified into ideal Euler head.
Ideal Euler head: (Jianjun Zhu, 2019)

$$He_{ideal} = \frac{\omega^2 r_2^2}{g} - \frac{Q\omega}{2\pi gh\tan\beta_2} \tag{2.5}$$

Head loss caused by friction: (Jianjun Zhu, 2019)
When fluids flow along impeller channels , fluid friction would cause head degradation. The friction losses in the impeller can be expressed as:

$$H_f = \frac{fQ^2}{8g\pi^2 Dh^2 \sin^3\beta_m} \times \frac{r_2 - r_1}{r_1 r_2} \tag{2.6}$$

where, D is the hydraulic diameter of impeller channel, h is the channel height, β_m is impeller average blade angle. The calculation details can be referred to Appendix 5.0.2.

The friction factor used in a straight, stationary pipe with a circular cross section is not applicable to ESP impeller channels. An ESP channel has a rectangular cross section, is curved, and its impeller rotates during operation (Datong Sun, 2006). So, the consideration of friction factor effects is necessary when setting up ESP first principle model. In order to calculate the friction factor f in friction loss Eq.2.6, the hydraulic diameter is needed. So, in this project, assume ESP impeller channel has near-rectangular cross section. The shape of ESP channel cross section is illustrated in Figure 2.4 (Datong Sun, 2006).

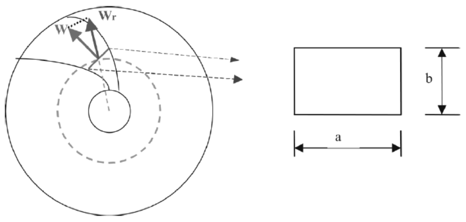


Figure 2.4: Shape of ESP channel cross section

The friction factor based on Churchill correlation incorporates the friction loss in modeling ESP boosting pressure, which can be written as:

$$f = F_\gamma F_\beta F_\omega f_c \tag{2.7}$$

where F_γ is shape effect correction factor, F_β is curvature effect correction factor,F_ω is rotational speed effect correction factor and f_c is Churchill correlation, respectively. The calculation details for F_γ, F_β ,F_ω and f_c can be referred to Appendix 5.0.2 and 5.0.3.
Shock loss: (Datong Sun, 2006)
Shock loss is mainly caused by the mismatch of fluid flow and the impeller channel angle at impeller inlet. In this project, the calculation of shock loss is obtained from the Datong Sun and Mauricio Prado (2006). They calculated the shock loss for pure water using the head difference between the pure water and frictional head model. Thus, the water shock loss at certain rotational speed can be written as:

$$\Delta H_{shock,water,base} = aQ_l^2 + bQ_l + c \tag{2.8}$$

where a, b and c are called as base rotational speed. For different rotational speeds, shock loss can be expressed as follow, which is following the affinity law assumption.

$$\Delta H_{shock,l} = (\frac{\omega_{impeller}}{\omega_{impeller,base}})^2[a(Q_l\frac{\omega_{impeller,base}}{\omega_{impeller}})^2 + b(Q_l\frac{\omega_{impeller,base}}{\omega_{impeller}}) + c] \tag{2.9}$$

After obtaining head losses caused by friction and shock loss, actual head developed by ESP can be written as:

$$H = H_e - H_f - \Delta H_{shock,l} \tag{2.10}$$

In general, ESP is characterized under water condition and the water performance curve is provided by manufacturers. Hydrocarbon fluids properties, however, are very different from water and significantly effect and alter the pump performance. Such effects are generally caused by high viscosity of hydrocarbon fluids and gas flow involvement etc.

Head loss viscosity correction:
Effects of liquid viscosity on pump performance: As one of the properties that characterises fluids, the viscosity of fluids flowing inside the pump would affect the performance of the pump. In practice, if the pumped fluid viscosity differs significantly from viscosity of water, which is used to obtain pump performance curve, then actual pump performance would be reduced and differ from published pump curve from manufactures. Then it would also reduce production rates of hydrocarbon as a result.
In order to correct different head performance caused by viscous pumped fluids, the hydraulic institute (HI) method is considered to be used for predicting pump head performance for viscous fluids. However, the HI method is just an approximation of pump head performance, because it does not take into account factors such as pump geometries and flow conditions etc. In the HI method, the estimation of head reduction can be obtained by applying correction factors for head to the performance of water (American national standards institute, 2010).

$$C_H = \frac{H_{vis}}{H_W} \tag{2.11}$$

where C_H is head correction factor, H_{vis} is head performance of viscous pumped fluid and H_W is water head. The calculation details can be referred to Appendix 5.0.4.

As an artificial lift method for high-flow-rate oil production, electrical submersible pumps' (ESP) performance surfers from gas entrainment, a frequently encountered phenomenon in ESPs. When it occurs, ESPs can experience moderate or severe head degradation accompanied with production rate reduction, gas locking and flow instabilities. However, free gas corrections, which presented in corresponding papers , are all focused on some specific pump types and lack of general correction information. Meanwhile, OLGA does not have free gas correction functions. So, in this project, free gas corrections are neglected.

2.2 Hybrid Model

Since both first principles model and black-box model have their own corresponding advantages and disadvantages, the combination of them which is called hybrid model or grey-box model can combine advantages and also complement shortages which they have. From a practical point of view, the grey-box modelling is a very convenient way to model nonlinear processes, since, the model structure can be derived from the first principles of mass and energy balances and the nonlinear characteristics of the process can be modelled as an empirical additive component.
In this project the hybrid model is the combination of the first principle model of ESP and machine learning black box model. And machine learning algorithm is focused on Gaussian process regression which will be introduced in the next part.

2.3 Gaussian Processes Regression

As a generic supervised learning method designed to solve regression problem, Gaussian Process Regression (GPR) is used in this project to solve problems in pure machine learning case and hybrid model case, respectively. A kernel based Gaussian Process Regression model that is equipped with a kernel function belongs to the wider area of machine learning (Rasmussen and Williams, 2006). Other machine learning methodologies such as neural network are also effective. However, in this project, due to the size of training dataset is not too big, the implementation of other machine learning algorithms will be computationally expensive. So, it means that GPR is able to work well on small datasets, implement a prediction model that is computationally inexpensive and have the ability to provide uncertainty measurements on the predictions.

The schematic diagram of GPR is illustrited in figure 2.5.

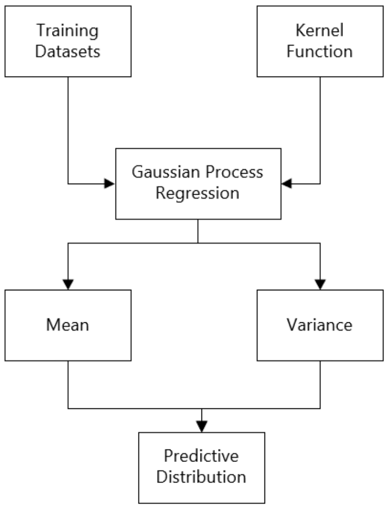


Figure 2.5: The schematic diagram of GPR

More particularly, the kernel based Gaussian Process Regression (GPR) is adopted for predicting the oil production rate within 24 months as doing well tests 7 hours per day in this project. Initially, instantiating a Gaussian Process model by defining kernel function and setting up Gaussian Process Regressor according to Algorithm 2.1 of Gaussian Processes for Machine Learning (GPML) by Rasmussen and Williams (Rasmussen and Williams, 2006). Second, the GPR model is fed with training data to fit to data using Maximum Likelihood Estimation of the parameters. Then predict the oil production rate by inserting test dataset into GPR model to get predictive distribution which is defined by the mean value together with the respective variance.

Since the objective of this project is to apply Gaussian process regression to predict oil production rate in pure machine learning and hybrid modeling cases and not to discuss and study the effect of diverse kernel functions to the prediction results, so kernel function is selected as basic RBF kernel.

Chapter 3

Case Studies

In order to more accurately predict oil production rate, the project is implemented into three different cases, respectively. The inspiration of these three cases is coming from different modeling approaches would give different modeling results. Applications of pure machine learning modeling and hybrid modeling would be executed in the following three cases. Comparisons and discussions would also be included in the following subsections. Furthermore, in pure machine learning model, different training and test datasets would be used to study the effects of diverse datasets on prediction accuracy.

The cases are tested on an ESP oil well with the following features:

Reservoir pressure	100	[bar]
Reservoir temperature	70	[C°]
Wellhead pressure	5	[bar]
Wellhead temperature	40	[C°]
Well depth	1500	[m]
Pump depth, absolute position	200	[m]
Pump REDA H22500N, stages	26	

Table 3.1: ESP oil well features

The main process of the three cases are composed of sending service data to OPC server and using MATLAB to run the simulation to read and get corresponding data of oil well which will be used to train the machine learning algorithm, which is Gaussian process regression in all three cases. The specific process of each case will be discussed in different case sections. Modeling approach overview can be found from Table 3.2.

Case	Modeling approach
Case one	Pure machine learning, GPR
Case two	Hybrid modeling with ESP's FPM
Case three	Pure machine learning, GPR

Table 3.2: Three cases' overview

3.1 Case One

3.1.1 Case one statement

The idea of case one is to use training dataset to train pure machine learning model and make predictions over test dataset. In this case, Gaussian process regression (GPR) is used as the machine learning algorithm. The total simulation period for collecting training dataset of oil well is set as 2 years or 24 months. The main process of case one can be illustrated as figure 3.1.

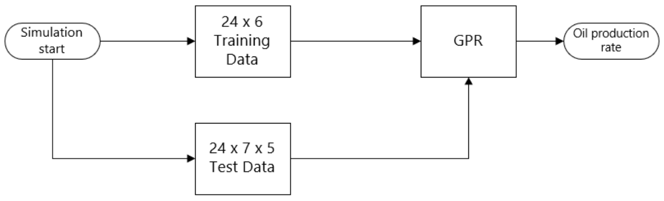


Figure 3.1: Process of case one

By running first simulation with 24 times well tests in 2 years period to get the training data set, which is composed of features (X) and target (y). Using training dataset to learn and train Gaussian process regression by fitting features and target. Then doing well test every 7 hours in 24 months to obtain test dataset. Then getting mean and standard deviation, which would provide predictive distribution, by implementing test dataset into Gaussian process prediction model. The final result will be discussed in results section, Chapter 4.1.

3.1.2 Data description

The details of training dataset and test dataset, which can be read from MATLAB simulation results, are illustrated in the following two Tables 3.3 and 3.4, respectively.

3.1.3 Implementation of case one

Using Python as the programming language to train Gaussian process regression by using uploaded 24×6 dimensions' training dataset and get prediction from uploaded 24×5 dimensions' test dataset. The coding details can be found from appendix 5.0.6.

Well test implementation	every 1 month x 24	[month]
Training data, feature X_1	pump intake pressure	[bar]
Training data, feature X_2	pump discharge pressure	[bar]
Training data, feature X_3	well head pressure	[bar]
Training data, feature X_4	pump intake temperature	[C°]
Training data, feature X_5	well head temperature	[C°]
Training data, target y	well head oil production rate	[m^3/d]

Table 3.3: Case one training dataset

Well test implementation	every 7 hours x every 1 month x 24	[hours]
Test data, feature Xt_1	pump intake pressure	[bar]
Test data, feature Xt_2	pump discharge pressure	[bar]
Test data, feature Xt_3	well head pressure	[bar]
Test data, feature Xt_4	pump intake temperature	[C°]
Test data, feature Xt_5	well head temperature	[C°]

Table 3.4: Case one test dataset

3.2 Case Two

3.2.1 Case two statement

In order to make a comparison between case one, which is executed with pure machine learning black-box model, the hybrid model of oil well is implemented in case two. In this case, the hybrid model is the combination of ESP first principles model and Gaussian process regression. The main process can be illustrated as shown in figure 3.2.

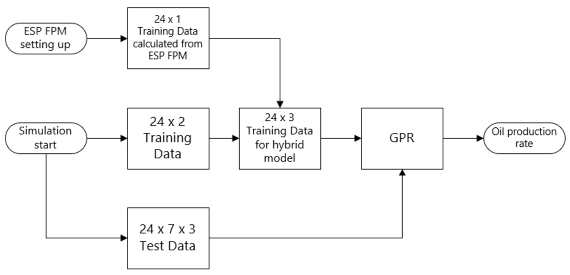


Figure 3.2: Process of case two

3.2.2 Data description

The details of training dataset and test dataset, which can be read from MATLAB simulation results, are illustrated in the following two Tables 3.5 and 3.6, respectively. The well test periods for training set and test set are identical with which are implemented in case one.

Well test implementation	every 1 month x 24	[month]
Training data, feature X_1	pump pressure difference	[bar]
Training data, feature X_2	pump intake temperature	[C°]

Table 3.5: Case two training dataset from simulation

Well test implementation	every 7 hours x every 1 month x 24	[hours]
Test data, feature Xt_1	pump pressure difference	[bar]
Test data, feature Xt_2	pump intake pressure	[bar]
Test data, feature Xt_3	oil rate through pump	[m$^3/d$]

Table 3.6: Case two test dataset from simulation

The training data will be used to train Gaussian process regression to set up machine learning black-box model. However, feature 3 of training data or oil production rate is the calculation result from ESP's first principles model. After setting up ESP's first principles model step by step, which is already explained in Chapter 2, the actual head performance curve of ESP can be obtained and oil production rate can also be fitted from the curve. As same as the process in case one, after getting the GPR, the test dataset can be used to predict oil production rate.

So, the training dataset for hybrid oil well modeling is shown in Table 3.7.

Well test implementation	every 1 month x 24	[month]
Training data, feature X_1	pump pressure difference	[bar]
Training data, feature X_2	pump intake temperature	[C°]
Training data, feature X_3	oil production rate, FPM result	[m$^3/d$]

Table 3.7: Case two training set

3.2.3 Implementation of case two

Using Python as the programming language to train Gaussian process regression by using 24×3 dimensions' training dataset, which is consist of calculated feature X_3 from ESP first principle model, and get prediction by applying uploaded 24×3 dimensions' test dataset. The coding details can be found from appendix 5.0.7.

3.3 Case There

3.3.1 Case three statement

The objective of building up case three is to study the effect on the pure machine learning result caused by applying different training and test dataset. Furthermore, compared with case one, case three still has a big difference in the prediction step. In case three the prediction of oil production rate is directly predicted from trained GPR without fitting test dataset to predict. Therefore, in this case the implementation process is not identical with which is applied in case one. In case three, the training data is directly read from MATLAB. And the main process is illustrated as follows in figure 3.3.

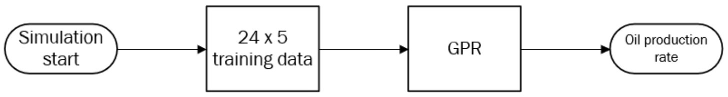


Figure 3.3: Process of case three

3.3.2 Data description

The details of training dataset, which can be read from MATLAB simulation results, are illustrated in the following Table 3.8. The well test periods for training set is identical with which are implemented in case one.

Well test implementation	every 1 month x 24	[month]
Training data, feature X_1	pump pressure difference	[bar]
Training data, feature X_2	pump intake temperature	[C°]
Training data, feature X_3	oil rate through pump	[m^3/d]
Training data, feature X_4	pump tubing pressure	[bar]
Training data, feature X_5	well head temperature	[C°]

Table 3.8: Case three training set

3.3.3 Implementation of case three

Using Python as the programming language to train Gaussian process regression by using 24×5 dimensions' training dataset, and get prediction directly from fitted GPR. The coding details can be found from appendix 5.0.8.

Chapter 4

Results

4.1 Results

The results of this project will be presented and discussed separately according to the different three cases.

4.1.1 Result of case one

In case one, GPR is applied to predict oil production rate as doing well test 7 hours per day, by using training dataset which is measured by doing well test per month. As the result shown in Figure 4.1, the oil prediction (blue solid line) from GPR is quite similar with the fitting result from features X and target y or actual oil production rate (red dashed line). It means that by using pure machine learning algorithm to train loosely tested training data from oil well within a specific period, the reliable and logic prediction is available to be obtained by inserting relative tightly tested test dataset. However, since the prediction is done by using pure machine learning algorithm, so the model itself can not explain the physical behavior of what the electric submersible pump really did in the oil production procedure.

In order to test the estimated regression makes sense or not, the coefficient of determination, also known as R^2 (R squared) is used. This is used as a measure of how well the regression equation actually describes the relationship between the dependent variable or target (y) and the independent variable or feature (X). The closer the coefficient of determination or R^2 is to 1, the more closely the regression line fits the sample data. Calculation result showing that R^2 = 0.9993513414860301, which is quite close to 1. So, it means that the kernel based GPR model, which applied in case one is quite proper for the training dataset. The calculation details of R^2 can be found from appendix 5.0.5.

The standard deviation of case one at the ending area is quite large, it may caused by the sharp decreasing oil production rate or sharp slope. Maybe can use more proper kernel

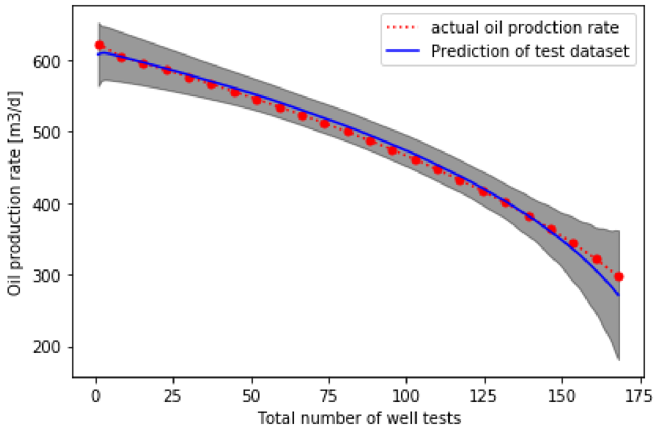


Figure 4.1: Oil prediction from pure machine learning

functions to eliminate this phenomena.

4.1.2 Result of case two

In case two, the hybrid modeling of oil well is done by using first principle model of ESP to get the head performance curve and further using this curve to get oil production rate. Then it is combined with data read from MATLAB simulation to generate the training dataset for Gaussian process regression. The head performance curve (H-Q curve) of ESP, REDA H22500N pump, for each stage is shown in Figure 4.2.

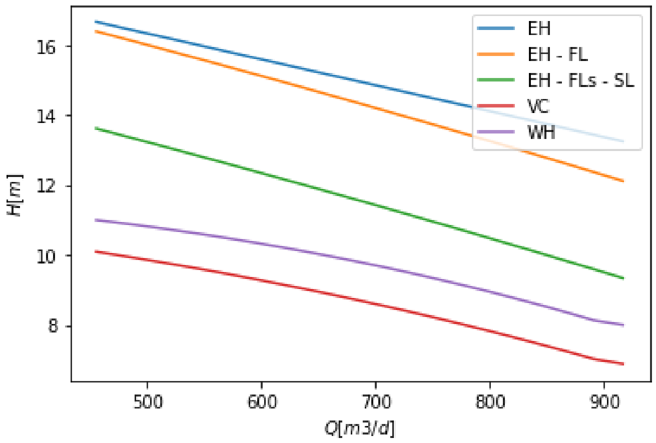


Figure 4.2: Head performance curve of ESP

For H-Q curve, stating each curve from top to down is Euler head, Euler head corrected after subtracting friction loss, Euler head corrected after subtracting friction loss and shock loss, water head for REDA H22500N pump from manufacture for Q ranges from 450 to 920 m^3/d and final actual head after doing viscosity correction.

The oil production rate which is calculated from the first principle model of ESP by using up.interpolation to fit actual head curve to the corresponding oil production rates is shown in Figure 4.3.

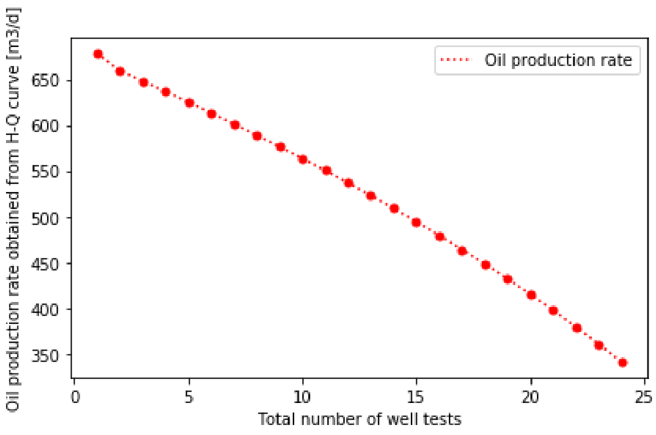


Figure 4.3: Oil production rate obtained from H-Q curve

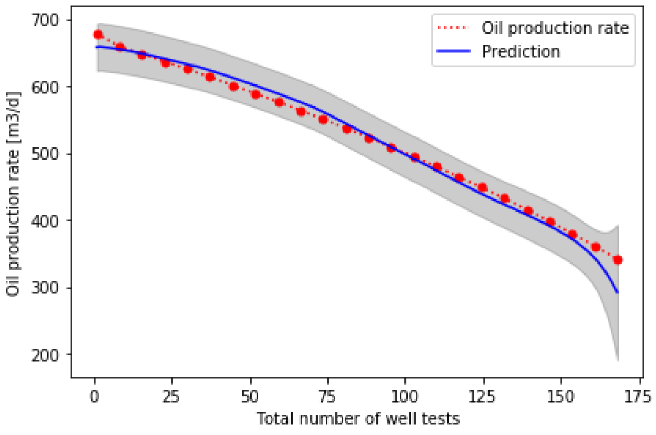


Figure 4.4: Oil prediction from case two

By inserting test dataset into the GPR to get prediction of oil production rate which is illustrated in Figure 4.4. By analysing two curves, we can see that, the prediction curve (blue solid line) is reasonable and logical, because the curve has the same descending trend with oil production rate from first principle model or real oil production tendency. However, the predictions of case one and case two seem like no large difference. But, in fact, it can imply that the advantages of using hybrid model which contains physical behaviours of certain chemical process, like what ESP really did in the oil production procedure, and those physical behaviours would provide more reliable insight of the process rather than just using pure machine learning model. Moreover, the predictions of these two cases are based on the same kernel function with different length-scales. For case two, due to the FPM's complexity which stands behind physical process, requires larger length-scale to enforce the prediction to be smooth. So, I think, hybrid modeling requires more proper kernel functions which can properly smooth and fit the physical behaviors of the process into machine learning algorithm.

4.1.3 Result of case three

In case three, the oil prediction process is done by learning and predicting from Gaussian process regression, which is trained and tested by only one dataset. Furthermore, the dataset includes the oil rates flow through ESP. And another important detail of this dataset is that it contains the pressure difference of the tube, which will provide more general pressure trend of a well under production. The prediction of oil production rate of case three is illustrated in Figure 4.5 below.

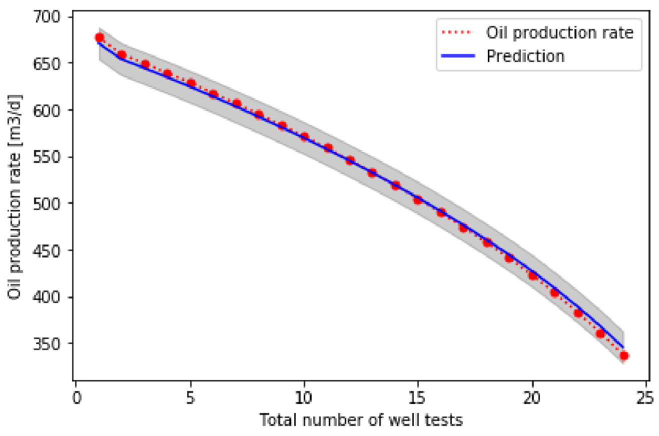


Figure 4.5: Oil prediction from case three

As shown in Figure 4.5, the prediction of oil production rate in case three is reliable and logical. Compared with the Figure 4.1, the standard deviation area of predictive distribution is more average than that in case one. In addition, learning and predicting from trained dataset can avoid the fitting noise caused by test dataset. Prediction seems good, however, the prediction results are higher than practical oil production rate at the wellhead.

4.2 Discussion

By comparing case one and case three, we can know that, the prediction results are depends on the correlations of dataset with the predicted output. For example, in case one, oil production rate is used as one of the feature to train the GPR, which provides the strong correlations with other training features to get a more practical distribution. However, in case three, the oil rate used as training feature is oil rate through the pump. Obviously, it is has not such a strong correlation with oil production rate, but it still can be used as the rough prediction of oil production rate, since they both have the same changing tendency. Machine learning algorithm is a kind of black-box model, but when solving problems like chemical process or oil gas production system, I think it would be much better to fit data which have stronger correlations with predicted output.

For case two, since it is not possible to get the proper geometric parameters of ESP, which is REDA H22500N in this project, the head performance result would more or less have deviations with actual head performance. In addition, some simplifications and assumptions which used in ESP first principle modeling, may also cause some deviations. But as a combination of black-box model and first principle model, hybrid model is more reliable to be implemented to predict oil production.

So, by comparing three results from three different cases, the result of case two is the best one, which is more realistic and reliable. As a result, hybrid modeling is the good method to more accurately predict oil production rate. However, I found from the programming experience, the prediction of case two needs larger length-scale to make prediction smoother. So, I think it may be caused by the complex physical behaviours of FPM and hybrid modeling should need more accurate and fine kernel functions to biuld up GPR.

Of course, the oil well manipulator would meet a situation to have to replace the pump with new one, due to some unknown issues would occur in the future production. When this situation occurs, instead of doing the whole hybrid modeling process like done in case two, manipulator can easily predict the new friction corrected pump head performance by implementing as follows. (1) Setting up a second order polynomial with friction head as x and actual head of current pump as y by using np.polyfit. (2)After structuring the polynomial, insert the head performance of a certain new pump from pump manufacture into one-dimensional polynomial class, numpy.poly1d to get the actual head after subtracting friction loss. As following above mentioned two steps, the actual pump head performance of a certain new pump after doing viscosity correction is illustrated in Figure 4.6. In this figure, the blue solid line is head performance of current pump, and orange solid line is head performance of new pump which will be used in the future.

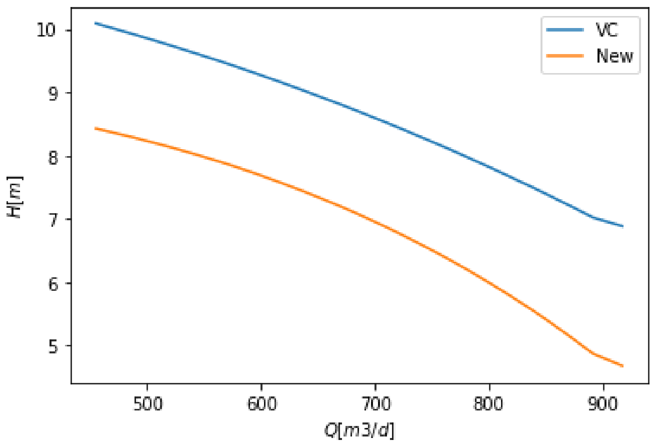


Figure 4.6: Actual pump performance of certian new pump

Chapter 5

Conclusion

First principle model is hard to derive, but it actually contains the physical behaviour of the ESP in this project. So, hybrid modeling not only can provide what ESP is really doing in oil production process and also can exert the ability of machine learning algorithm when fitting input and output variables. In this project, hybrid modeling is the best way to estimate oil production in oil well. Furthermore, hybrid modeling is the robust way to implement. Since, it is my first time to focus on near real application in oil and gas production system, I spent lots of time on how to set up first principle model and how to apply machine learning algorithm in oil production application. Maybe this project is not absolutely perfect, but I actually dedicated hundred percent of myself into this project. Now, I actually learnt lots of things from this project, not only the specific knowledge and programming and software skills, but also learnt how to think and how to learn. In the upcoming semester, I hope I can become more familiar with machine learning application and implement more powerful algorithm in real problems. Moreover, the further modification and study of this project is the most important thing should I do. If possible, I will implement choke model and separator and let this oil well model be as practical as possible.
Anything which can not destroy you will make you.

Bibliography

Datong Sun, M.P., 2006. Single-phase model for electric submersible pump (esp) head performance. Society of Petroleum Engineers 11.

American national standards institute, H.i., 2010. American national standard (guidence) for effects of liquid viscosity on rotodynamic (centrifugal and vertical pump performance) .

Jianjun Zhu, Haiwen Zhu, G.C.J.Z.J.P.H.B.H.Q.Z., 2019. A new mechanistic model to predict boosting pressure of electrical submersible pumps esps under high-viscosity fluid flow with validations by experimental data. Society of Petroleum Engineers .

Low Soon Tiong, A.A., 1997. Hybrid model for chemical process modeling. Artificial Intelligence in Real Time Control .

Qiang Xiong, A.J., 2001. Grey-box modelling and control of chemical processes. Chemical Engineering Science 57 (2002), 1027 – 1039.

Rasmussen, C.E., Williams, C.K.I., 2006. in: Gaussian Processes for Machine Learning.

Takacs, G., 2018. in: Electrical Submersible Pumps Manual Design, Operations, and Maintenance Book • 2nd Edition, Oxford OX2 8DP, UK.

Tatiane Silva Vieira a, n, J.R.S.b.A.D.B.b.R.E.M.c.V.E.a., March 2015. Analytical study of pressure losses and fluid viscosity effects on pump performance during monophase flow inside an esp stage. Journal of Petroleum Science and Engineering 127, 245–258.

Timur Bikmukhametov, J.J., January 2020, 106487. First principles and machine learning virtual flow metering: A literature review. Journal of Petroleum Science and Engineering 184.

Appendix

5.0.1 OLGA oil field model with ESP

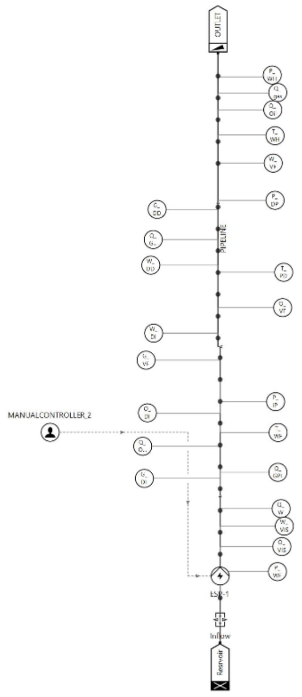
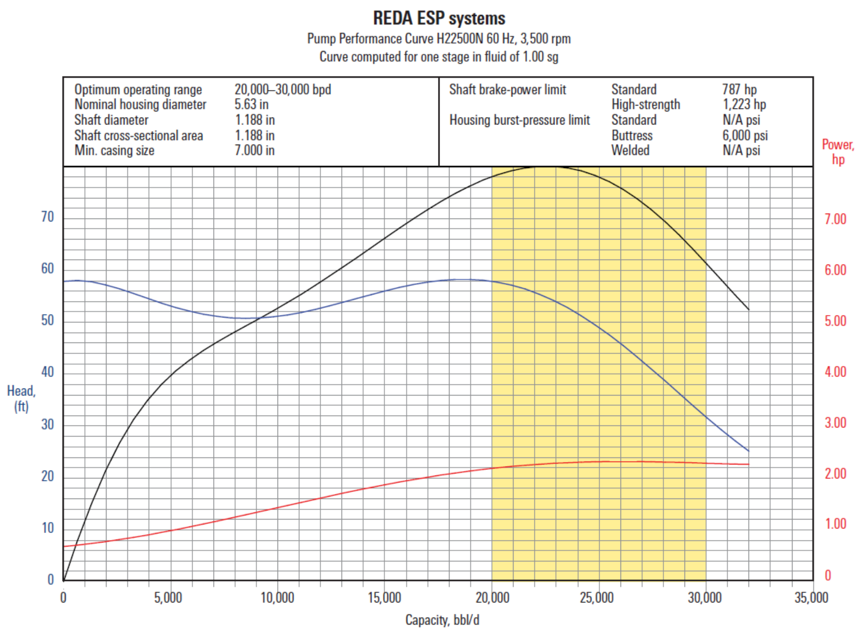


Figure 5.1: Oil well model with ESP in OLGA

5.0.2 Hydraulic diameter (Datong Sun, 2006)

The hydraulic diameter of impeller channel can be calculated from

$$D = \frac{2ab}{a+b} \tag{5.1}$$



Optimum operating range	20,000–30,000 bpd	Shaft brake-power limit	Standard	787 hp
Nominal housing diameter	5.63 in		High-strength	1,223 hp
Shaft diameter	1.188 in	Housing burst-pressure limit	Standard	N/A psi
Shaft cross-sectional area	1.188 in		Buttress	6,000 psi
Min. casing size	7.000 in		Welded	N/A psi

Figure 5.2: REDA H22500N, 60Hz, 3500rpm in head[ft] and flow rate [bbl/d], performance curve

where,

$$a = \frac{2\pi r}{n} \sin \beta_m; b = h \tag{5.2}$$

Radius of near-rectangular impeller in hydraulic diameter calculation can be expressed as:

$$r = r_2 - r_1 \tag{5.3}$$

Average impeller balde angle in hydraulic diameter calculation can be expressed as:

$$\beta_m = \frac{\beta_1 + \beta_2}{2} \tag{5.4}$$

5.0.3 Friction factor calculation (Jianjun Zhu, 2019)

Churchill correlation

$$f_c = 2[(\frac{8}{R_e})^{12} + \frac{1}{(A+B)^{1.5}}]^{1/12} \tag{5.5}$$

where A and B can be obtained from:

$$A = [2.475 \ln(\frac{1}{(\frac{7}{R_e})^{0.9} + 0.27\frac{\epsilon}{D}})]^{16} \tag{5.6}$$

$$B = (\frac{37530}{R_e})^{16} \tag{5.7}$$

where R_e is Reynold number, which determines the flow regime in the impeller channel. Reynold number can be expressed as:

$$Re = \frac{DW\rho_l}{\mu_l} \tag{5.8}$$

where ρ_l is liquid density, μ_l is liquid viscosity, W is relative velocity, which can be computed as:

$$W = \frac{Q_l}{2\pi rb\sin\beta_m} \tag{5.9}$$

Pipe shape effect F_γ

The calculation of pipe shape effect F_γ is determined by Reynold number.
If $Re \leq 2300$, the calculation of F_γ is:

$$F_\gamma = [\frac{2}{3} + \frac{11}{24}L(2-L)]^{-1} \tag{5.10}$$

If $Re > 2300$, the calculation of F_γ is:

$$F_\gamma = [\frac{2}{3} + \frac{11}{24}L(2-L)]^{-0.25} \tag{5.11}$$

where, L is given by:

$$L = \frac{\min(a,b)}{\max(a,b)} \tag{5.12}$$

Rotational speed effect F_ω

In order to obtain F_ω, the critical Reynolds number is redefined. Because Ito and Nanbu (1971), they suggested that the flow regime and friction factor for rotational pipes were influenced by rotational Reynolds number. Where rotational Reynolds number ($Re\omega$) is given as:

$$Re_\omega = \frac{D^2\omega\rho_l}{\mu_l} \tag{5.13}$$

So, the critical Reynolds number is
If $Re_\omega \geq 28$:

$$N_{Re} = 1070Re_\omega^{0.23} \tag{5.14}$$

If $Re_\omega < 28$:

$$N_{Re} = 2300 \tag{5.15}$$

Thus, when $Re < N_{Re}$, the calculation of F_ω have three possibilities. If $ReRe_\omega \leq 220$ and $Re_\omega/Re < 0.5$:

$$F_\omega = 1 \tag{5.16}$$

If $220 < ReRe_\omega < 10^7$ and $Re_\omega/Re < 0.5$:

$$F_\omega = 0.0883(ReRe_\omega)^{0.25}(1 + 11.2(ReRe_\omega)^{-0.325}) \tag{5.17}$$

If $Re_\omega/Re \geq 0.5$:

$$F_\omega = \frac{0.0672Re_\omega^{0.5}}{1 - 2.11(Re_\omega)^{-0.5}} \tag{5.18}$$

For $Re > N_{Re}$, F_ω also have three possibilities. If $Re_\omega^2/Re < 1$:

$$F_\omega = 1 \tag{5.19}$$

If $1 < Re_\omega^2/Re < 15$:

$$F_\omega = 0.942 + 0.058[(\frac{Re_\omega^2}{Re})^{0.282}] \tag{5.20}$$

If $Re_\omega^2/Re > 15$:

$$F_\omega = 0.942[(\frac{Re_\omega^2}{Re})^{0.05}] \tag{5.21}$$

The final fiction factor in this project is the multiplication of all the correction factors with Churchill correlation, which has been shown by Eq.2.9

5.0.4 Viscosity correction (American national standards institute, 2010)

Determining pump performance of pumping vicous fluid when water performance is known

Step 1

Calculate parameter B based on the water performance best efficiency flow (Q_{BEP-W})

$$B = 16.5\frac{(V_{vis})^{0.5} \times (H_{BEP-W})^{0.0625}}{(Q_{BEP-W})^{0.375} \times N^{0.25}} \tag{5.22}$$

where N is pump speed. If $1.0 < B < 4.0$, go to Step 2.

Step 2

Calculate correction factor C_Q

$$C_Q = (2.71)^{-0.165\times(\log_{10} B)^{3.15}} \tag{5.23}$$

Correct water performance flow to viscous flow

$$Q_{vis} = C_Q \times Q_W \tag{5.24}$$

Correct the water performance head (H_{BEP-W}) that corresponds to water performance best efficiency flow (Q_{BEP-W})

$$C_{BEP-H} = C_Q \tag{5.25}$$

Step 3

Calculate head correction factor (C_H), and corresponding values of viscous head (H_{vis})

$$C_H = 1 - ((1 - C_{BEP-H}) \times (\frac{Q_W}{Q_{BEP-W}})^{0.75}) \tag{5.26}$$

$$H_{vis} = C_H \times H_W \tag{5.27}$$

5.0.5 The coefficient of determination R^2

Total sum of squares, TSS

$$TSS = \sum (y_i - \bar{y})^2 \tag{5.28}$$

Residual sum of squares, RSS

$$RSS = \sum (y_i - \hat{y})^2 \tag{5.29}$$

R^2

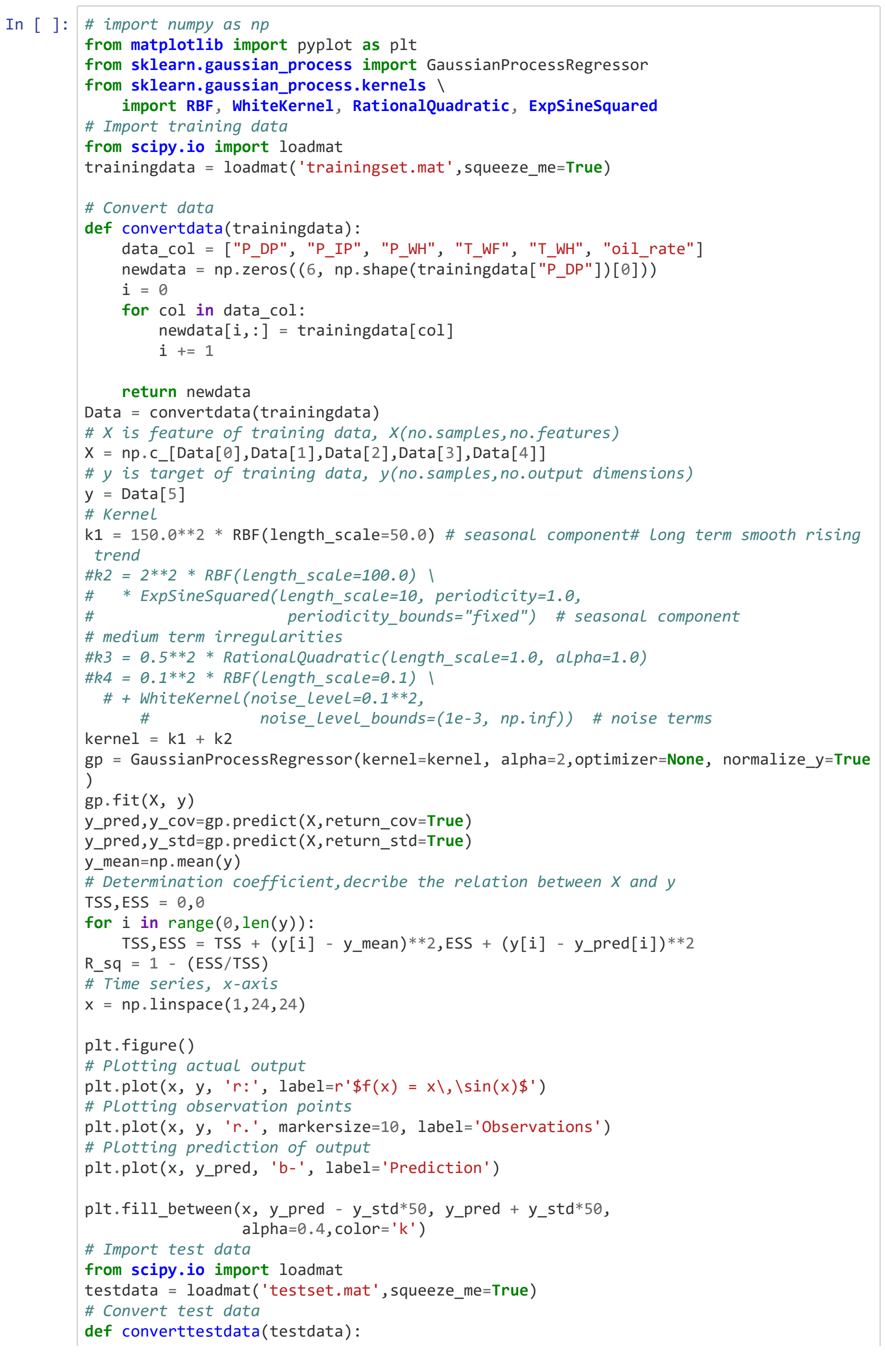
$$R^2 = 1 - \frac{RSS}{TSS} \tag{5.30}$$

where, $\bar{y}$ is mean of y and $\hat{y}$ is the estimation from regression equation.

5.0.6 Python code of case one

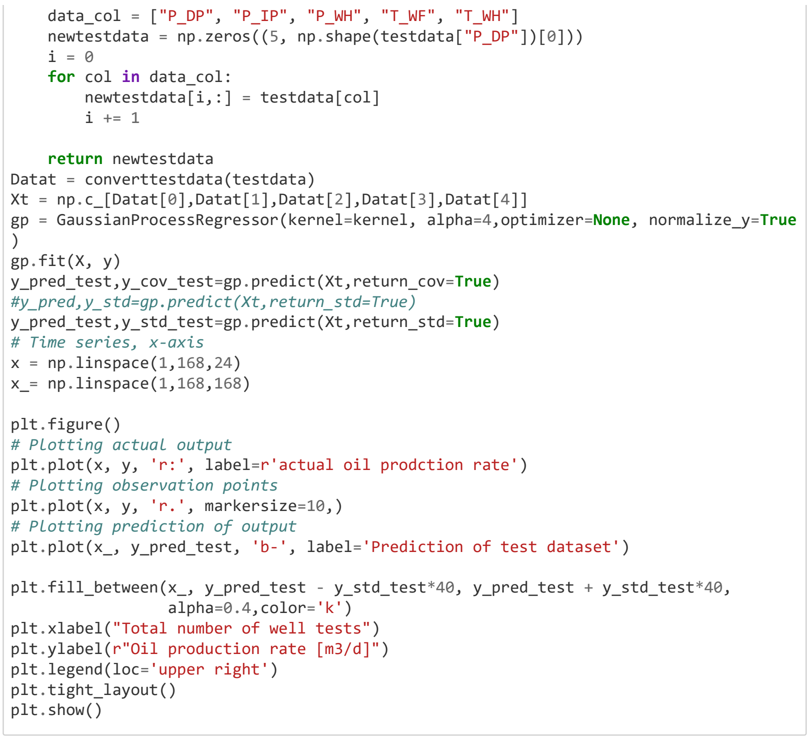
5.0.7 Python code of case two

5.0.8 Python code of case three



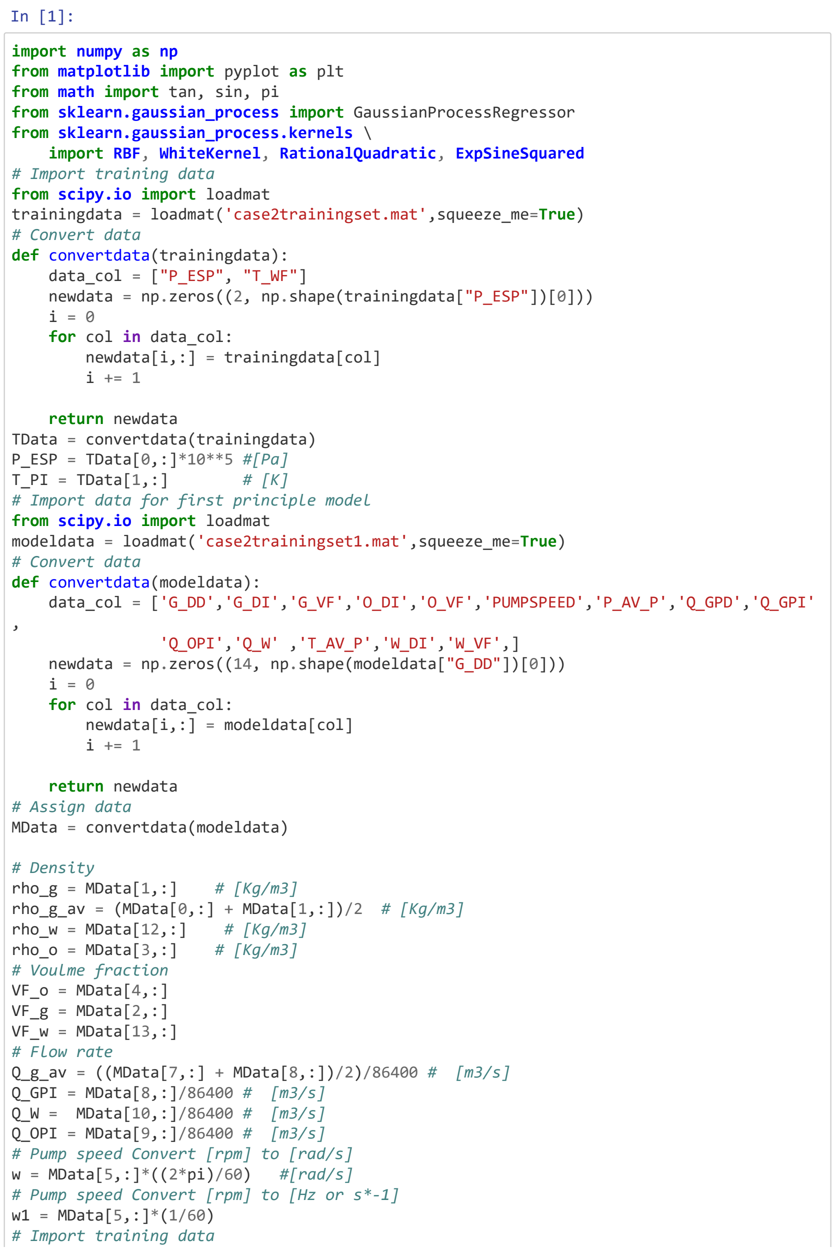
```
In [ ]: # import numpy as np
        from matplotlib import pyplot as plt
        from sklearn.gaussian_process import GaussianProcessRegressor
        from sklearn.gaussian_process.kernels \
            import RBF, WhiteKernel, RationalQuadratic, ExpSineSquared
        # Import training data
        from scipy.io import loadmat
        trainingdata = loadmat('trainingset.mat',squeeze_me=True)

        # Convert data
        def convertdata(trainingdata):
            data_col = ["P_DP", "P_IP", "P_WH", "T_WF", "T_WH", "oil_rate"]
            newdata = np.zeros((6, np.shape(trainingdata["P_DP"])[0]))
            i = 0
            for col in data_col:
                newdata[i,:] = trainingdata[col]
                i += 1

            return newdata
        Data = convertdata(trainingdata)
        # X is feature of training data, X(no.samples,no.features)
        X = np.c_[Data[0],Data[1],Data[2],Data[3],Data[4]]
        # y is target of training data, y(no.samples,no.output dimensions)
        y = Data[5]
        # Kernel
        k1 = 150.0**2 * RBF(length_scale=50.0) # seasonal component# long term smooth rising
         trend
        #k2 = 2**2 * RBF(length_scale=100.0) \
        #    * ExpSineSquared(length_scale=10, periodicity=1.0,
        #                     periodicity_bounds="fixed")  # seasonal component
        # medium term irregularities
        #k3 = 0.5**2 * RationalQuadratic(length_scale=1.0, alpha=1.0)
        #k4 = 0.1**2 * RBF(length_scale=0.1) \
          # + WhiteKernel(noise_level=0.1**2,
              #            noise_level_bounds=(1e-3, np.inf))  # noise terms
        kernel = k1 + k2
        gp = GaussianProcessRegressor(kernel=kernel, alpha=2,optimizer=None, normalize_y=True
        )
        gp.fit(X, y)
        y_pred,y_cov=gp.predict(X,return_cov=True)
        y_pred,y_std=gp.predict(X,return_std=True)
        y_mean=np.mean(y)
        # Determination coefficient,decribe the relation between X and y
        TSS,ESS = 0,0
        for i in range(0,len(y)):
            TSS,ESS = TSS + (y[i] - y_mean)**2,ESS + (y[i] - y_pred[i])**2
        R_sq = 1 - (ESS/TSS)
        # Time series, x-axis
        x = np.linspace(1,24,24)

        plt.figure()
        # Plotting actual output
        plt.plot(x, y, 'r:', label=r'$f(x) = x\,\sin(x)$')
        # Plotting observation points
        plt.plot(x, y, 'r.', markersize=10, label='Observations')
        # Plotting prediction of output
        plt.plot(x, y_pred, 'b-', label='Prediction')

        plt.fill_between(x, y_pred - y_std*50, y_pred + y_std*50,
                         alpha=0.4,color='k')
        # Import test data
        from scipy.io import loadmat
        testdata = loadmat('testset.mat',squeeze_me=True)
        # Convert test data
        def converttestdata(testdata):
```

Figure 5.3: Case one



```python
    data_col = ["P_DP", "P_IP", "P_WH", "T_WF", "T_WH"]
    newtestdata = np.zeros((5, np.shape(testdata["P_DP"])[0]))
    i = 0
    for col in data_col:
        newtestdata[i,:] = testdata[col]
        i += 1

    return newtestdata
Datat = converttestdata(testdata)
Xt = np.c_[Datat[0],Datat[1],Datat[2],Datat[3],Datat[4]]
gp = GaussianProcessRegressor(kernel=kernel, alpha=4,optimizer=None, normalize_y=True
)
gp.fit(X, y)
y_pred_test,y_cov_test=gp.predict(Xt,return_cov=True)
#y_pred,y_std=gp.predict(Xt,return_std=True)
y_pred_test,y_std_test=gp.predict(Xt,return_std=True)
# Time series, x-axis
x = np.linspace(1,168,24)
x_= np.linspace(1,168,168)

plt.figure()
# Plotting actual output
plt.plot(x, y, 'r:', label=r'actual oil prodction rate')
# Plotting observation points
plt.plot(x, y, 'r.', markersize=10,)
# Plotting prediction of output
plt.plot(x_, y_pred_test, 'b-', label='Prediction of test dataset')

plt.fill_between(x_, y_pred_test - y_std_test*40, y_pred_test + y_std_test*40,
                 alpha=0.4,color='k')
plt.xlabel("Total number of well tests")
plt.ylabel(r"Oil production rate [m3/d]")
plt.legend(loc='upper right')
plt.tight_layout()
plt.show()
```

Figure 5.4: Case one



```
In [1]:

import numpy as np
from matplotlib import pyplot as plt
from math import tan, sin, pi
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels \
    import RBF, WhiteKernel, RationalQuadratic, ExpSineSquared
# Import training data
from scipy.io import loadmat
trainingdata = loadmat('case2trainingset.mat',squeeze_me=True)
# Convert data
def convertdata(trainingdata):
    data_col = ["P_ESP", "T_WF"]
    newdata = np.zeros((2, np.shape(trainingdata["P_ESP"])[0]))
    i = 0
    for col in data_col:
        newdata[i,:] = trainingdata[col]
        i += 1

    return newdata
TData = convertdata(trainingdata)
P_ESP = TData[0,:]*10**5 #[Pa]
T_PI = TData[1,:]        # [K]
# Import data for first principle model
from scipy.io import loadmat
modeldata = loadmat('case2trainingset1.mat',squeeze_me=True)
# Convert data
def convertdata(modeldata):
    data_col = ['G_DD','G_DI','G_VF','O_DI','O_VF','PUMPSPEED','P_AV_P','Q_GPD','Q_GPI'
,
                'Q_OPI','Q_W' ,'T_AV_P','W_DI','W_VF',]
    newdata = np.zeros((14, np.shape(modeldata["G_DD"])[0]))
    i = 0
    for col in data_col:
        newdata[i,:] = modeldata[col]
        i += 1

    return newdata
# Assign data
MData = convertdata(modeldata)

# Density
rho_g = MData[1,:]    # [Kg/m3]
rho_g_av = (MData[0,:] + MData[1,:])/2  # [Kg/m3]
rho_w = MData[12,:]    # [Kg/m3]
rho_o = MData[3,:]    # [Kg/m3]
# Voulme fraction
VF_o = MData[4,:]
VF_g = MData[2,:]
VF_w = MData[13,:]
# Flow rate
Q_g_av = ((MData[7,:] + MData[8,:])/2)/86400 #  [m3/s]
Q_GPI = MData[8,:]/86400 #  [m3/s]
Q_W =  MData[10,:]/86400 #  [m3/s]
Q_OPI = MData[9,:]/86400 #  [m3/s]
# Pump speed Convert [rpm] to [rad/s]
w = MData[5,:]*((2*pi)/60)   #[rad/s]
# Pump speed Convert [rpm] to [Hz or s*-1]
w1 = MData[5,:]*(1/60)
# Import training data
```

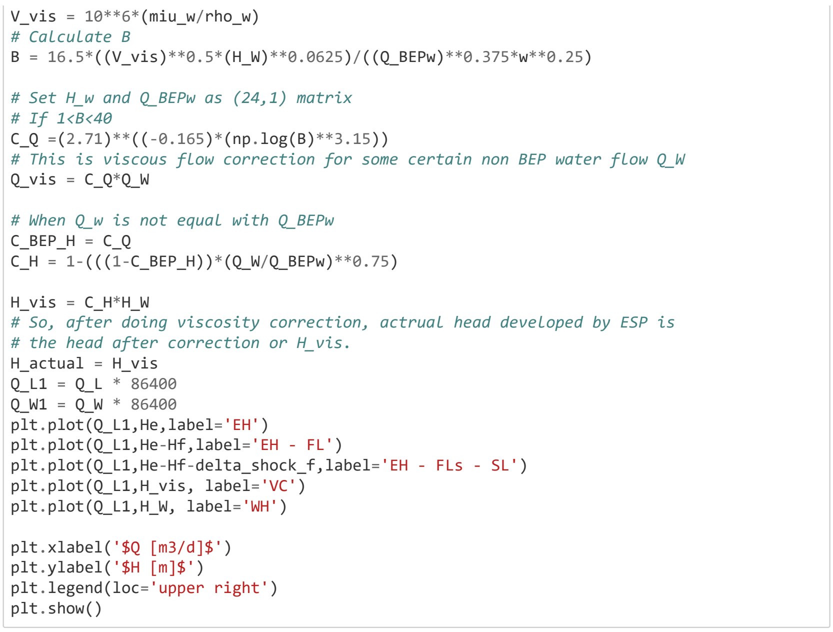
Figure 5.5: Case two

```python
from scipy.io import loadmat
modeldata1 = loadmat('case2trainingset2.mat',squeeze_me=True)
# Convert data
def convertdata(modeldata1):
    data_col = ["O_VIS", "W_VIS"]
    newdata = np.zeros((2, np.shape(modeldata1["O_VIS"])[0]))
    i = 0
    for col in data_col:
        newdata[i,:] = modeldata1[col]
        i += 1

    return newdata
# Assign data
MData1 = convertdata(modeldata1)*10**3
# Viscosity; check how to calculate mixture viscosity
miu_o = MData1[0,:]* 0.001 #[Pa.s]
miu_w = MData1[1,:]* 0.001 #[Pa.s]
# Import training data
from scipy.io import loadmat
modeldata2 = loadmat('case2trainingset3.mat',squeeze_me=True)

# Convert data
def convertdata(modeldata2):
    data_col = ['P_AV_P', 'P_IP','T_AV_P']
    newdata = np.zeros((3, np.shape(modeldata2['P_AV_P'])[0]))
    i = 0
    for col in data_col:
        newdata[i,:] = modeldata2[col]
        i += 1

    return newdata
# Assign data
MData2 = convertdata(modeldata2)
P_i = MData2[1,:]   #[bar]
# Pump inner diameter from literatures
# b1: impeller intake blade angle; b2: impeller diacharge blade angle; n: number of cha
nnels; h: channel height
# w: angular velovity[rad/s]; r1: Impeller entrance radius; r2: Impeller discharge radi
us
# epsilon: Channel wall roughness; Q_L: liquid flow rate;
b1 = 38*(pi/180) # [radian]
b2 = 23*(pi/180) # [radian]
n = 7
h = 0.01 # [m]
r1 = 0.029  # [m]
r2 = 0.048 #[m]
epsilon = 10**(-4) #[m]
g = 9.80665   # [m/s-2]
Q_L = (Q_W + Q_OPI) #[m3/s]
# Euler head calculation; He: [m] ,[m]*3.28084 = [ft]
# But I can not get pump inside geometries, so neglect Euler head.
He = (w**2/g)*(r2**2-r1**2)-((Q_L*w)/(2*pi*g*h))*(1/tan(b2)-1/tan(b1))
# If the fluids enter the impeller without pre-rotation
He_ideal = (w**2*r2**2/g)-(Q_L*w)/(2*pi*g*h*tan(b2))
# bm: average blade angle,[degree]; Di: channel hydraulic diameter, [m];
# Q: liquid flow rate, [m3/s] (Have to convert it into volumetic flow rate)
# average blade angle
bm = (b1+b2)/2
# Impeller channel hydraulic diameter,To calculate the friction factor, the hydraulic d
iameter is
# needed and is related with the cross-sectional geometry
```

Figure 5.6: Case two

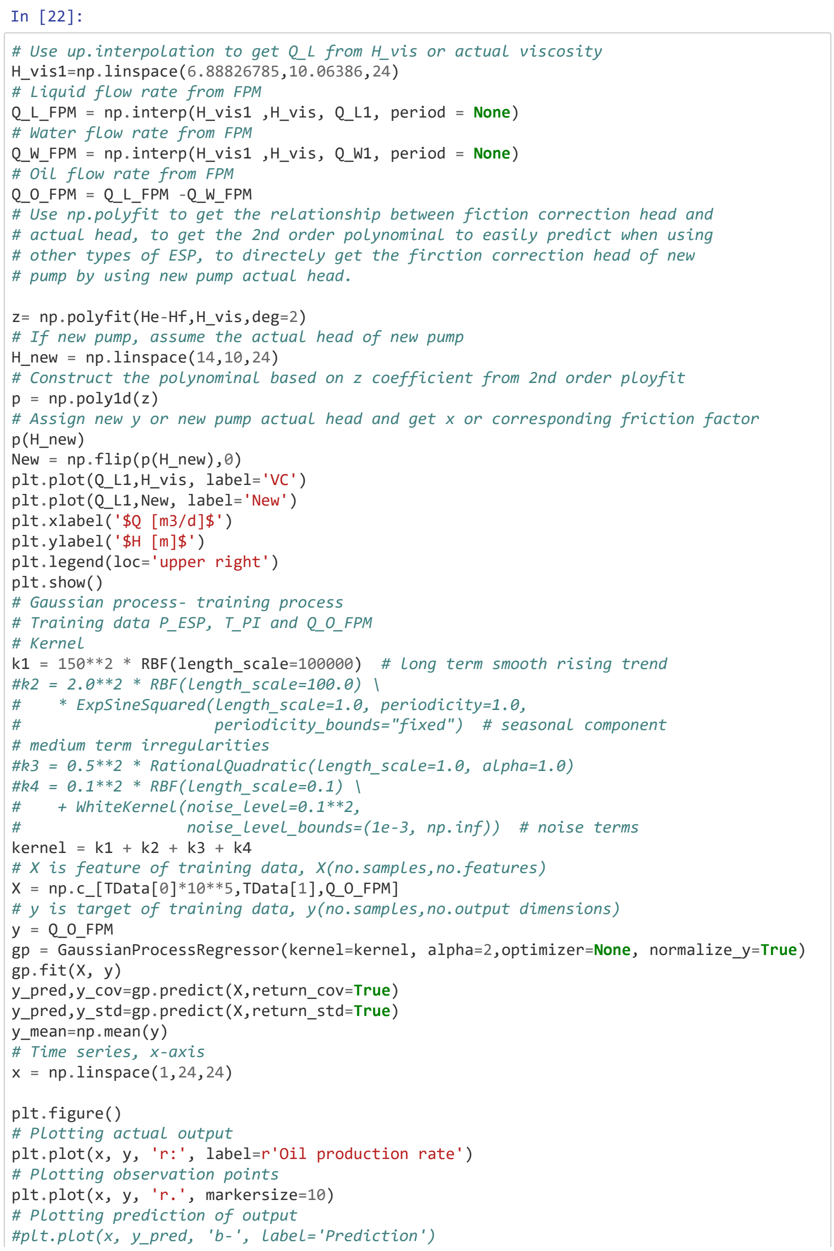
```
a = ((2*pi*(r2-r1))/n)*sin(bm)
Di = (2*a*h)/(a+h)
# Calculate liquid mixture density ; VF: volume fraction
rho_m=VF_o*rho_o + VF_w*rho_w
# Calculate liquid mixture density
miu_m=VF_o*miu_o + VF_w*miu_w
# Fre, pipe shape effect
# Relative velocity
W = Q_L/(2*pi*(r2-r1)*h*sin(bm))
# The friction factor depends on whether the flow regime occurring in the channel is la
minar or turbulent.
# determination of the flow regime depends on the Reynolds number, which is related to
 the relative velocity W
# along ESP channels as
Re = Di*W*rho_m/ miu_m
l = a/h
# If laminar
Fre = (2/3+(11/24)*l*(2-l))**(-1)
# If turbulent
#Fre = (2/3+(11/24)*l*(2-l))**(-0.25)
Re_ro =(Di**2)*w1*rho_m/miu_m
# Churchill friction factor,f
A = (2.457*np.log(1/((7/Re)**0.9)+0.27*epsilon/Di))**16
B = (3750/Re)**16
f = 2*((8/Re)**12+1/((A+B)**1.5))**(1/12)
# Fro, rotational speed effect
# Re_ro: rotational Reynolds number; If the Re_ro < 28, the pipe can be considered stat
ionary. If the Re_ro >= 28,
# rotational speed effects must be considered.

# If Re_ro < 28
N_Re = 2300
# If Re_ro >= 28
# N_Re = 1070 * Re_ro**0.23
# If Re < N_Re
# If 220<Re_ro*Re<10*7 and Re_ro/Re<0.5
Fro = 0.0883*(Re*Re_ro)**0.25*(1+11.2*(Re*Re_ro)**(-0.325))
# Firction factor
fi = Fre*f*Fro
# Head loss by impeller friction
Hf = (fi*(Q_L)**2)/(8*g*Di*pi**2*h**2*(sin(bm))**3)*((r2-r1)/(r1*r2))
# Compute shock loss;  Approximate with a polynominal
A = 3.3*10**(-6)
B = -5.122*10**(-3)
C = 2.042
w_base = 50*2*pi        #[rad/s]
K = w/w_base
delta_shock_f = K**2*(A*(Q_L*(1/K))**2+B*(Q_L*(1/K))+C)
#delta_shock_f1 = K**2*(A*(Q*(1/K))**2)+B*(Q*(1/K)+C
# Avtual head
H_actual = He_ideal - Hf - delta_shock_f
# Water head at BEP per stage[m]
H_W = np.linspace(8, 11, 24)
# H_w = 10
# H_W = H_W
# REDA H22500N has 26 stages in my case[m**3/d]
H_wt = H_W*26
# Water flow rate at BEP
Q_BEPw = 3000/86400  #[m**3/s]
# Fluid viscosity
# V_vis = 10**6*max((miu_w/rho_w),(VF_o*miu_o)/rho_o)
```

Figure 5.7: Case two



```
V_vis = 10**6*(miu_w/rho_w)
# Calculate B
B = 16.5*((V_vis)**0.5*(H_W)**0.0625)/((Q_BEPw)**0.375*w**0.25)

# Set H_w and Q_BEPw as (24,1) matrix
# If 1<B<40
C_Q =(2.71)**((-0.165)*(np.log(B)**3.15))
# This is viscous flow correction for some certain non BEP water flow Q_W
Q_vis = C_Q*Q_W

# When Q_w is not equal with Q_BEPw
C_BEP_H = C_Q
C_H = 1-(((1-C_BEP_H))*(Q_W/Q_BEPw)**0.75)

H_vis = C_H*H_W
# So, after doing viscosity correction, actrual head developed by ESP is
# the head after correction or H_vis.
H_actual = H_vis
Q_L1 = Q_L * 86400
Q_W1 = Q_W * 86400
plt.plot(Q_L1,He,label='EH')
plt.plot(Q_L1,He-Hf,label='EH - FL')
plt.plot(Q_L1,He-Hf-delta_shock_f,label='EH - FLs - SL')
plt.plot(Q_L1,H_vis, label='VC')
plt.plot(Q_L1,H_W, label='WH')

plt.xlabel('$Q [m3/d]$')
plt.ylabel('$H [m]$')
plt.legend(loc='upper right')
plt.show()
```

Figure 5.8: Case two


In [22]:

```python
# Use up.interpolation to get Q_L from H_vis or actual viscosity
H_vis1=np.linspace(6.88826785,10.06386,24)
# Liquid flow rate from FPM
Q_L_FPM = np.interp(H_vis1 ,H_vis, Q_L1, period = None)
# Water flow rate from FPM
Q_W_FPM = np.interp(H_vis1 ,H_vis, Q_W1, period = None)
# Oil flow rate from FPM
Q_O_FPM = Q_L_FPM -Q_W_FPM
# Use np.polyfit to get the relationship between fiction correction head and
# actual head, to get the 2nd order polynominal to easily predict when using
# other types of ESP, to directely get the firction correction head of new
# pump by using new pump actual head.

z= np.polyfit(He-Hf,H_vis,deg=2)
# If new pump, assume the actual head of new pump
H_new = np.linspace(14,10,24)
# Construct the polynominal based on z coefficient from 2nd order ployfit
p = np.poly1d(z)
# Assign new y or new pump actual head and get x or corresponding friction factor
p(H_new)
New = np.flip(p(H_new),0)
plt.plot(Q_L1,H_vis, label='VC')
plt.plot(Q_L1,New, label='New')
plt.xlabel('$Q [m3/d]$')
plt.ylabel('$H [m]$')
plt.legend(loc='upper right')
plt.show()
# Gaussian process- training process
# Training data P_ESP, T_PI and Q_O_FPM
# Kernel
k1 = 150**2 * RBF(length_scale=100000)  # long term smooth rising trend
#k2 = 2.0**2 * RBF(length_scale=100.0) \
#    * ExpSineSquared(length_scale=1.0, periodicity=1.0,
#                     periodicity_bounds="fixed")  # seasonal component
# medium term irregularities
#k3 = 0.5**2 * RationalQuadratic(length_scale=1.0, alpha=1.0)
#k4 = 0.1**2 * RBF(length_scale=0.1) \
#    + WhiteKernel(noise_level=0.1**2,
#                  noise_level_bounds=(1e-3, np.inf))  # noise terms
kernel = k1 + k2 + k3 + k4
# X is feature of training data, X(no.samples,no.features)
X = np.c_[TData[0]*10**5,TData[1],Q_O_FPM]
# y is target of training data, y(no.samples,no.output dimensions)
y = Q_O_FPM
gp = GaussianProcessRegressor(kernel=kernel, alpha=2,optimizer=None, normalize_y=True)
gp.fit(X, y)
y_pred,y_cov=gp.predict(X,return_cov=True)
y_pred,y_std=gp.predict(X,return_std=True)
y_mean=np.mean(y)
# Time series, x-axis
x = np.linspace(1,24,24)

plt.figure()
# Plotting actual output
plt.plot(x, y, 'r:', label=r'Oil production rate')
# Plotting observation points
plt.plot(x, y, 'r.', markersize=10)
# Plotting prediction of output
#plt.plot(x, y_pred, 'b-', label='Prediction')
```

Figure 5.9: Case two

```python
#plt.fill_between(x, y_pred - y_std*20, y_pred + y_std*20,
#                  alpha=0.2,color='k')
plt.xlabel("Total number of well tests")
plt.ylabel(r"Oil production rate obtained from H-Q curve [m3/d]")
plt.legend(loc='upper right')
plt.tight_layout()
plt.show()
# Determination coefficient,decribe the relation between X and y
TSS,ESS = 0,0
for i in range(0,len(y)):
    TSS,ESS = TSS + (y[i] - y_mean)**2,ESS + (y[i] - y_pred[i])**2
R_sq = 1 - (ESS/TSS)
print(R_sq)
# Import test data

from scipy.io import loadmat
testdata = loadmat('case2testset.mat',squeeze_me=True)
# Convert data
def convertdata(testdata):
    data_col = ['P_ESP', 'Q_OPI','T_WF']
    newdata = np.zeros((3, np.shape(testdata['P_ESP'])[0]))
    i = 0
    for col in data_col:
        newdata[i,:] = testdata[col]
        i += 1

    return newdata
# Assign data
TData1 = convertdata(testdata)
Xt = np.c_[TData1[0]*10**5,TData1[2],TData1[1]]
# Kernel
k1 = 150**2 * RBF(length_scale=100000)  # Long term smooth rising trend
k2 = 2**2 * RBF(length_scale=1)
# medium term irregularities
#k3 = 0.1**2 * RationalQuadratic(length_scale=1, alpha=0.0001)
#k4 = 0.1**2 * RBF(length_scale=0.1) \
#    + WhiteKernel(noise_level=0.1**2,
#                  noise_level_bounds=(1e-3, np.inf))  # noise terms
kernel = k1 + k2 + k3 + k4
#kernel = k1
gp = GaussianProcessRegressor(kernel=kernel, alpha=2,optimizer=None, normalize_y=True)
gp.fit(X, y)
#y_pred_test,y_cov_test=gp.predict(Xt,return_cov=True)
y_pred_test,y_std_test=gp.predict(Xt,return_std=True)
# Time series, x-axis
x = np.linspace(1,168,24)
x_= np.linspace(1,168,168)

plt.figure()
# Plotting actual output
plt.plot(x, y, 'r:', label=r'Oil production rate')
# Plotting observation points
plt.plot(x, y, 'r.', markersize=10)
# Plotting prediction of output
plt.plot(x_, y_pred_test, 'b-', label='Prediction')

plt.fill_between(x_, y_pred_test - y_std_test*15, y_pred_test + y_std_test*15,
                 alpha=0.2,color='k')
plt.xlabel("Total number of well tests")
plt.ylabel(r"Oil production rate [m3/d]")
```

Figure 5.10: Case two

```
plt.legend(loc='upper right')
plt.tight_layout()
plt.show()
```

Figure 5.11: Case two

```
In [2]:
```

```
# Training data
# Pressure difference of tube: welldown P - wellhead P
# Pressure difference of pump
# Pump intake temperature; Wellhead temperature;
# Pump intake oil rate or oil rate through pump
import numpy as np
from matplotlib import pyplot as plt
from math import tan, sin, pi
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels \
    import RBF, WhiteKernel, RationalQuadratic, ExpSineSquared
# Import training data
from scipy.io import loadmat
trainingdata = loadmat('case3trainingset.mat',squeeze_me=True)
# Convert data
def convertdata(trainingdata):
    data_col = ['P_ESP', 'P_T', 'Q_OPI', 'T_WF', 'T_WH']
    newdata = np.zeros((5, np.shape(trainingdata['P_ESP'])[0]))
    i = 0
    for col in data_col:
        newdata[i,:] = trainingdata[col]
        i += 1

    return newdata
TData = convertdata(trainingdata)
# X feature of training data
X = np.c_[TData[0]*10**5,TData[1],TData[3],TData[4]]

# y feature of training data
y = TData[2]
# Kernel
k1 = 0.01**2 * RBF(length_scale=50.0)  # long term smooth rising trend
k2 = 1.5**2 * RBF(length_scale=100.0) \
    * ExpSineSquared(length_scale=1.0, periodicity=1.0,
                     periodicity_bounds="fixed")  # seasonal component
# medium term irregularities
k3 = 0.5**2 * RationalQuadratic(length_scale=1.0, alpha=1.0)
k4 = 0.1**2 * RBF(length_scale=0.1) \
    + WhiteKernel(noise_level=0.1**2,
                  noise_level_bounds=(1e-3, np.inf))  # noise terms
kernel = k1 + k2 + k3 + k4
gp = GaussianProcessRegressor(kernel=kernel, alpha=0.1,optimizer=None, normalize_y=True
)
gp.fit(X, y)
y_pred,y_cov=gp.predict(X,return_cov=True)
y_pred,y_std=gp.predict(X,return_std=True)
y_mean=np.mean(y)
# Determination coefficient,decribe the relation between X and y
TSS,ESS = 0,0
for i in range(0,len(y)):
    TSS,ESS = TSS + (y[i] - y_mean)**2,ESS + (y[i] - y_pred[i])**2
R_sq = 1 - (ESS/TSS)
print(R_sq)
# Time series, x-axis
x = np.linspace(1,24,24)

plt.figure()
# Plotting actual output
plt.plot(x, y, 'r:', label=r'Oil production rate')
```

Figure 5.12: Case three

```
# Plotting observation points
plt.plot(x, y, 'r.', markersize=10)
# Plotting prediction of output
plt.plot(x, y_pred, 'b-', label='Prediction')

plt.fill_between(x, y_pred - y_std*40, y_pred + y_std*40,
                 alpha=0.2,color='k')
plt.xlabel("Total number of well tests")
plt.ylabel(r"Oil production rate [m3/d]")
plt.legend(loc='upper right')
plt.tight_layout()
plt.show()
```

Figure 5.13: Case three