

Master Thesis

Application of neural networks for
short-term prediction of flowing
wellhead pressures in horizontal gas
storage wells

Written by:

Markus Sachsenhofer, BSc
1035241

Advisors:

Univ.-Prof. Dipl.-Ing. Dr.mont. Herbert Hofstätter
Dipl.-Ing. Dr.mont. Rudolf Fruhwirth
Dipl.-Ing. Alexander Gubik

Leoben, 21.11.2016

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht habe.

AFFIDAVIT

I hereby declare that the content of this work is my own composition and has not been submitted previously for any higher degree. All extracts have been distinguished using quoted references and all information sources have been acknowledged.

Acknowledgement

First of all I would like to express my gratitude to Dipl.-Ing. Bernhard Griess and everyone else at Rohoel-Aufsuchungs AG (RAG) for this exceptional opportunity. I want to especially thank Dipl.-Ing. Alexander Gubik, who was my supervisor at RAG. Without his experience, encouragement and support this thesis would not have been possible.

I would like to thank the Chair of Petroleum and Geothermal Energy Recovery and its head Univ.-Prof. Dipl.-Ing. Dr. mont. Herbert Hofstätter for their support and efforts throughout my studies.

In particular I would like to thank my advisor at the Chair Dipl.-Ing. Dr. mont. Rudolf Fruhwirth, who has been an outstanding thesis advisor. His experience and dedication were the main inspiration for this thesis.

Finally I want to thank my parents for their effort and encouragement throughout the whole time. I would also like to express my gratitude to my aunt and uncle for their support, which made living as a student much easier. Additionally I would like to mention my brother Klaus, who was a great help when writing this thesis.

Kurzfassung

Die Speicherung von Erdgas und die Versorgungssicherheit damit sind wichtig für ein Land wie Österreich, das vom Import von fossilen Energiequellen abhängig ist. Typischerweise ist der Betrieb eines Untertagespeichers für Erdgas durch den saisonal bedingten Verbrauch von Gas und der nahezu konstanten Lieferkapazität von Pipelines gekennzeichnet. Eine kurzfristigere Fahrweise des Speichers unterstützt durch eine numerische Simulation verschiedenerer Szenarien könnte die Energieeffizienz verbessern und durch verbesserte Schaltzeiten Equipment schonen.

Diese Masterarbeit behandelt die Themenstellung ob künstliche neuronale Netze zur Modellierung des Betriebs eines Erdgasspeichers verwendet werden können. Neuronale Netze sind heuristische Tools die nahezu jeden Trend in den Daten wiedergeben können, ohne Kenntnisse der zugrundeliegenden physikalischen Gesetzmäßigkeiten. Es wird untersucht ob es eine Architektur von neuronalen Netzen gibt, die eine Alternative zur klassischen Lagerstättensimulation im Bereich der Simulation kurzfristiger Fahrweisen eines Erdgasspeichers darstellen können. Eine Genauigkeit der Vorhersage der Sondenkopffließdrücken von ± 1 Bar ist die Voraussetzung einer weiteren Optimierung des Betriebsplans. Daten eines echten Untertageerdgasspeichers, betrieben von der Rohölaufsuchungs AG (RAG) sind das Fundament aller Simulationen dieser Arbeit mit Hilfe neuronaler Netze.

Mehrere Typen von neuronalen Netzen werden geprüft hinsichtlich ihrer Fähigkeit korrekte Sondenkopfdrucke während der Injektion und Entnahme von Gas vorherzusagen. Die Anwendung neuronaler Netze setzt eine geeignete Vorgangsweise der Datenverarbeitung voraus. Diese inkludiert die Bearbeitung von fehlenden Datenpunkten und Ausreißern in den gemessenen Daten sowie die Umwandlung der irregulär gesampelten Signale in regulär gesampelte. Zwei verschiedene Methoden zur Abstratenkonvertierung werden verwendet um sicherzustellen, dass ein Signal frei von Artefakten zum Training der neuronalen Netze angewendet wird. Alle Netze werden mit den gleichen „History Matching“ und Testintervallen aus einer Datenquelle trainiert.

Die Resultate aus den „History Matching“ und Testdatensätzen haben gezeigt, dass manche Netzarchitekturen die Sondenkopfdrucke, die während des Betriebs des Speichers gemessen wurden, erfolgreich simulieren konnten. Bei der Verwendung von „recurrent“ neuronalen Netzen, wie den „fully recurrent“, Elman und NARX konnten gute Ergebnisse erzielt werden. Die besten Ergebnisse im „History Matching“ und auch im unabhängigen Testintervall konnten mit den „fully recurrent“ Netzen erreicht werden. Obwohl keines der trainierten Modelle die gewünschte Genauigkeit von ± 1 Bar erzielte, konnte dennoch gezeigt werden, dass die „recurrent“ neuronalen Netze generell im Stande waren die Sondenkopfdrucke zu simulieren. Abschließend kann man sagen, dass neuronale Netze eine brauchbare Alternative sein können, wenn noch zusätzlich Verbesserungen hinsichtlich ihrer Genauigkeit erzielt werden können.

Abstract

The underground storage of natural gas and the security of supply thereof is an important issue for a country like Austria, which depends on imports of fossil fuels. Typically the operation mode of an underground gas storage facility is determined by the seasonal swing in the demand of gas opposed by the nearly constant supply through pipelines. A more short-term oriented operation schedule supported by a numerical simulation of various scenarios could improve the energy efficiency and reduce wear through optimized switching cycles.

This master's thesis evaluates the application of neural networks to the simulation of gas storage behaviour. Neural networks are heuristic tools that can be used to model nearly any trend in the data fed to them, without prior knowledge of underlying physical laws. It is investigated whether a neural network architecture exists that can act as an alternative to the classical reservoir simulation to model the short-term operation of an underground gas storage facility. An accuracy of the wellhead pressure prediction in the order of ± 1 Bar is the prerequisite to the optimization of operation schedules. Data from an actual underground gas storage plant, operated by the Rohöl-Aufsuchungs AG (RAG) is the basis for all neural network simulations.

Several neural network types are tested with regards to their suitability of producing adequate wellhead pressure predictions during injection and production of natural gas. A prerequisite towards the application of these networks is an appropriate data management workflow. This includes the treatment of missing samples and outliers in the measured data as well as the transformation of the irregularly sampled signal to a regular one. Two different resampling methods are built to ensure a clean signal is used to train the neural networks. All networks are trained using equal history matching and testing intervals coming from one data set.

The results on the history matching and testing data set showed that there were some network architectures that could successfully model the wellhead pressures during gas storage operation. Good results were achieved when using recurrent neural network architectures, like the fully recurrent one, the Elman network and NARX model. The best model outputs on the history matching as well as the independent test interval were produced by the fully recurrent network. Although none of the models trained and tested could reach the desired accuracy of ± 1 Bar, it could be shown that these recurrent neural networks were capable of modelling the wellhead pressures in general. Finally it can be said that neural networks could be a viable alternative if additional improvements towards an enhanced accuracy are made.

List of Tables

4.1	Automatic deletion of double time stamps	49
4.2	Resampling intervals and respective filter configurations	64
5.1	Overview of neural network software packages	78
5.2	Overview of MLP network generations	79
5.3	Overview of fully recurrent network generations	81
5.4	Overview of Elman network generations	82
5.5	Overview of NARX network generations	82
5.6	Overview of ESN generations	84

List of Figures

2.1	Gas Demand Profile	4
3.1	Joshi Horizontal Well Model	13
3.2	Babu & Odeh Horizontal Well Model	14
3.3	Skin Effect & Darcy Coefficient	17
3.4	Houptert Coefficients from Flow-after-Flow Test	19
3.5	Influence of Tubing Size on the Gas VLP Curve	25
3.6	Effect of Damage and Turbulence	26
3.7	Concept of Upscaling	28
3.8	Multi-Layer Perceptron	31
3.9	Neuron Model and Transfer Function	32
3.10	Schematic of ANN Input Data Treatment	34
3.11	Flow of Error Signal through an ANN	35
3.12	Effect of Overfitting	36
3.13	Fully recurrent ANN	39
3.14	Elman - simple recurrent ANN	41
3.15	NARX recurrent ANN	42
3.16	BPTT Training Method	44
3.17	ESN with sparsely connected Reservoir	46
4.1	Treatment of missing Data	49
4.2	Distinction between Outliers and valuable Samples	50
4.3	Histogram of sampled Time Intervals	51
4.4	Time Series Plot of Well 1	52
4.5	Scatter plots of flow rate vs pressure for all six operating wells	54
4.6	Box plot of pressures of all wells	56
4.7	Boxplot of flow rates of all wells	56
4.8	Histograms of the Pressure for all six operating Wells	58
4.9	Histograms of the Flow Rate for all six operating Wells	59
4.10	Relation between Nyquist Frequency and Aliasing	61
4.11	Comparison of a Gaussian & Median Filter	63
4.12	Spectral analysis of a Gaussian filter	64
4.13	Comparison of Original & Resampled Signal	68
4.14	Critical Sections in the Original & Resampled Signal	69
5.1	Input & Output configuration of ANN models	72
5.2	Input & Output configuration Obs-Well 1	73
5.3	Time Series Plot of Obs-Well 1 Prediction	74

5.4	Output - Target correlation plot & error histogram - Obs-Well 1	75
5.5	Data Separation for ANN Models	76
5.6	Computation Time of ANN	77
6.1	MLP ANN Validation & Test Set Errors	86
6.2	Time Series Plot of MLP ANN Prediction	87
6.3	Output - Target correlation plot & error histogram - MLP	89
6.4	Fully recurrent ANN Validation & Test Set Errors	90
6.5	Time Series Plot of Fully Recurrent ANN Prediction	92
6.6	Time Series Plot of Fully Recurrent ANN Prediction	93
6.7	Output - Target correlation plot & error histogram - fully recurrent ANN	95
6.8	Elman ANN Validation & Test Set Errors	96
6.9	Time Series Plot of Elman ANN Prediction	98
6.10	Output - Target correlation plot & error histogram - Elman ANN	99
6.11	NARX ANN Validation & Test Set Errors	100
6.12	Time Series Plot of NARX ANN Prediction	102
6.13	Output - Target correlation plot & error histogram - NARX ANN	103
6.14	ESN Validation & Test Set Errors	104
6.15	Time Series Plot of ESN Prediction	106
6.16	Output - Target correlation plot & error histogram - ESN	107
6.17	Ranking of networks according to their validation errors	108
A1	Moody Diagram	119
A2	Nodal Analysis	120
A3	Overview of common ANN Transfer Functions	122
C1	Time Series Plot of Fully Recurrent ANN Prediction - Well 1	154
C2	Time Series Plot of Fully Recurrent ANN Prediction - Well 2	155
C3	Time Series Plot of Fully Recurrent ANN Prediction - Well 3	156
C4	Time Series Plot of Fully Recurrent ANN Prediction - Well 4	157
C5	Time Series Plot of Fully Recurrent ANN Prediction - Well 5	158
C6	Time Series Plot of Fully Recurrent ANN Prediction - Well 6	159

Abbreviations

ANN	Artificial Neural Network
AOF	Absolute Open Flow
BP	Backpropagation
BPTT	Backpropagation Through Time
EFM	Electronic Flow Measurement
ESN	Echo State Network
FVM	Finite Volume Method
GLR	Gas Liquid Ratio
GOR	Gas Oil Ratio
GRU	Gated Recurrent Unit Network
IMPES	Implicit Pressure Explicit Saturation
IMPIMS	Implicit Pressure Implicit Saturation
IPR	Inflow Performance Relationship
IQR	Inter-Quartile Range
LNG	Liquefied Natural Gas
MLP	Multilayer Perceptron
NARX	Nonlinear AutoRegressive Model with eXogenous Inputs
PVT	Pressure-Volume-Temperature
QC	Quality Control
RAG	Rohölaufsuchungs AG
RMSE	Root Mean Square Error
SCADA	Supervisory Control and Data Acquisition
UGS	Underground Gas Storage
VLP	Vertical Lift Performance

Contents

1	Introduction	1
1.1	Problem Statement	1
2	Fundamentals	3
2.1	Underground gas storage concept	3
2.2	Types of UGS & Field data	6
2.3	Data measurement & Management	7
3	Literature Review	8
3.1	Classical approach to reservoir management	8
3.1.1	IPR for gas wells	8
3.1.2	Skin & Non-Darcy Effect	15
3.1.3	VLP for gas wells	19
3.1.4	Nodal Analysis	24
3.1.5	General reservoir simulation	26
3.1.6	Wellbore storage & Transient effects	29
3.2	Mathematical approach with artificial neural networks	31
3.2.1	Multilayer Perceptron	31
3.2.1.1	Neuron Model & Number of Weights	31
3.2.1.2	Input normalization & Weight initialization	33
3.2.1.3	Training of ANN	34
3.2.1.4	Generalization	35
3.2.2	Recurrent ANN	37
3.2.2.1	Fully connected recurrent ANN	38
3.2.2.2	Simple recurrent ANN	40
3.2.2.3	NARX recurrent network	41
3.2.2.4	Training of recurrent ANN	43
3.2.2.5	Vanishing Gradients	43
3.2.2.6	Echo State Network	44
4	Data Management	47
4.1	Description of available data	47
4.1.1	Confidentiality agreement	47
4.2	Gathering and preparation of data	48
4.2.1	Missing values	48
4.2.2	Double values	48
4.2.3	Erroneous measurement	49
4.2.4	Irregular sampling intervals	50

4.3	Data statistics	51
4.3.1	Time series plot	51
4.3.2	Scatter plot	53
4.3.3	Box plot	55
4.3.4	Histogram	57
4.4	Resampling	60
4.4.1	Linear resampling and filtering	60
4.4.1.1	Filtering in the time domain - Gaussian filter	61
4.4.1.2	Filtering in the frequency domain	62
4.4.1.3	Median filter	62
4.4.1.4	Applied filtering approach	63
4.4.2	Resampling with local polynomial kernel regression	65
4.5	Quality control	66
5	Methodology	70
5.1	Agenda	70
5.2	Data preprocessing	70
5.2.1	Selection of input & output for ANN	71
5.2.1.1	Obs-Well 1 as input	72
5.2.2	Data separation	76
5.2.3	Sampling frequency & Computation time	77
5.3	Model Setup	78
5.3.1	Overview of neural network programs	78
5.3.2	MLP with RapidMiner	78
5.3.3	Fully Recurrent ANN with MATLAB	79
5.3.3.1	Smaller resampling interval	81
5.3.4	Elman ANN with MATLAB	81
5.3.5	NARX ANN with MATLAB	82
5.3.6	ESN with MATLAB	83
6	Results	85
6.1	MLP ANN	85
6.2	Fully recurrent ANN	90
6.2.1	Smaller resampling interval	91
6.3	Elman ANN	96
6.4	NARX ANN	100
6.5	Echo State Network	104
6.6	Ranking of ANN types	108
7	Conclusion	109
8	Discussion and Outlook	112

9 References	113
Appendices	119
Appendix A	119
Appendix B	123
Appendix C	129

1 Introduction

Natural gas is one of the most important energy sources not only in Austria but also in the whole European Union and covers around 17 % of Austria's total energy consumption. The domestic yearly production of natural gas in Austria remained in a range of 1.3 to 2 MMm³ since 1990. The net import volumes of gas rose from 5 in 1990 up to 9 MMm³ in 2011. In the same time period a remarkable development with regard to underground gas storage capacity could be observed in Austria. As of April 2014 the working gas volume of all storage facilities in Austria had reached 8.1 MMm³, which means that Austria can store nearly the whole yearly domestic gas consumption [1, p. 43-50].

This impressive amount of storage capacity is mainly used in a seasonal cycle by storing gas in summer and producing it again in winter. By utilizing spare storage capacities different short-term oriented operation modi could be analysed to generate additional income or to optimise facility usage. This thesis investigates if and how neural networks can be used to model the gas storage's behaviour for short-term periods.

1.1 Problem Statement

The possibility of modelling the short-term behaviour of an underground gas storage facility, especially the wellhead pressure during injection and production opens many ways of optimization. These improvements would additionally increase the overall energy efficiency and reduce wear at valves and other equipment through enhanced switching cycles.

In general the classical reservoir simulation approach is used to generate a geological model and do finite volume numerical simulation of the hydrocarbon reservoir. This approach reaches its limit of unrestrained applicability when numerous prediction scenarios are required to optimize short-term operation of a gas storage facility. This limitation results from the requirements involved with the set up and the computation of different injection/production scenarios in classical reservoir simulation software, therefore the method is hardly applicable when operational decisions have to be taken in short time, requiring the numerical simulation of various scenarios in a restricted time period.

The aim of this thesis is to investigate if neural networks can act as an alternative to the classical reservoir simulation approach for short-term applications. This investigation is based on actual data from underground gas storage facilities of Rohölaufsuchungs AG (RAG). The data provided by RAG includes static reservoir parameters and dynamic data coming from wellhead measurements. To check the suitability of neural networks a workflow including data gathering and preparation as well as the setup of different network architectures has to be developed.

A simulation of wellhead pressures with an accuracy of ± 1 Bar is required to investigate the optimization potential of different operation modi.

Data Management

The challenge in data management is to extract suitable data from a multitude of different sources with varying quality. Data is provided by different databases within RAG, which store static information about the gas storage reservoir and dynamic information coming from wellhead measurements. The raw data has to be refined in order to be applicable for neural networks and this refinement process includes the selection of appropriate parameters in the model building process. A workflow dedicated to data handling and identification of valuable information is developed. This process discusses all necessary steps from identifying valuable information, statistical analysis of gathered data and their transformation into a suitable format, in particular the issue of suitable types of data for a neural network is dealt with. Typically data used in classical reservoir simulation is in time series format, which requires that data from different sources is scaled to identical time stamps. A similar approach is developed to treat dynamic measurements coming from different wells.

Neural Networks

Neural networks are heuristic machine learning tools which allow modelling any observable relationship in data without prior knowledge of underlying physical processes. There are numerous types and architectures to be found in the literature and several of them are evaluated. A review of literature on existing types of neural networks is done which gives an indication of architectures suitable for modelling underground gas storage operation. A workflow is proposed that discusses the steps involved in model building and testing of those. These steps include the selection of suitable variables and the data pre-processing required for neural network inputs. Furthermore an investigation concerning the use of static and dynamic data is done to find out which kind of data is required to model the gas storage's behaviour by neural networks. Several network architectures are evaluated regarding their predictive performance of wellhead pressures during injection and production cycles.

All networks models are based on an equal data set as far as possible to ensure comparability and validity of results. In a final step all networks are trained and evaluated on a history matching period, as it would also be done in classical reservoir simulation. The result in the history matching interval gives an indication if certain neural network architectures are suitable at all. The network's predictive capabilities are tested on an independent time interval that is the same for all networks.

2 Fundamentals

2.1 Underground gas storage concept

"The gas supply chain is characterized by large hourly, daily and annual imbalances between supply and demand. Although on the one hand, producers and transporters prefer to deliver the gas at the same constant rate at all times, users only need gas at certain times [...]" [2, p. 25].

Underground gas storage acts as a balancing measure, without which producers and transporters would have to at least double the production and transport capacity to satisfy the demand. The imbalances in demand do not only happen on seasonal, but also on a daily and hourly basis. Different types of UGS, namely seasonal, daily and hourly storage facilities allow to satisfy the gas supply demand at every instance. 'Daily storage' is characterized by an injection during night time and an increased production in the morning and evening hours. 'Seasonal storage' aims at the increased gas demand in the year's colder months and an injection phase in the summer months, where generally the gas supply is larger than the demand. Figure 2.1 shows the variation in the yearly demand of gas and the balancing done by the operation of gas storage. Another important application of UGS is 'strategic reserve'. Especially for Europe, which is no longer able to produce its entire gas demand within Europe this form of storage is of utmost significance. The gas stored in those facilities is primarily aimed to mitigate supply shortages and interruptions, caused by technical nature and failures in the gas distribution network or political events in producing countries or countries along the main transport routes. Additionally UGS are sometimes found in the vicinity of gas producing areas. The purpose of this so-called 'production storage' is to ensure a steady feed into the pipeline in case of production fluctuation, common in associated gas production and areas of severe weather events, e.g. hurricanes in the Gulf of Mexico. A more recent development sees UGS in combination with LNG terminals to act as buffer between continuous pipeline and non-continuous transport via ship. This requires a special type of UGS with unordinary high production and injection rates [2, p. 25-27].

The history of UGS dates back to 1915 with the conversion of a depleted gas field in Welland County, Ontario, Canada followed a year later by the Zoar field in the western New York state, USA. Thereafter more and more gas fields in the USA were converted into gas storages. The first UGS in Europe was commissioned in 1953 with the Engelborstel aquifer storage. A more recent type of gas storage was the development of salt cavern storage, the first salt cavern was commissioned in 1961 in Michigan, USA and in 1970 in Tersanne, France, Europe [2, p. 31].

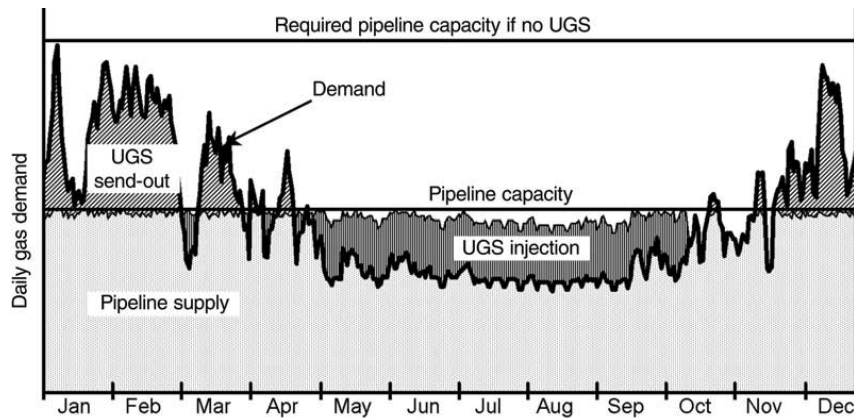


Figure 2.1: Yearly gas demand profile (black) and pipeline capacity [2, p. 26]

The first UGS in Austria was commissioned in 1977 by OMV AG and 1982 by Rohölaufsuchungs AG. In 2015, 715 UGS were in operation worldwide with a working gas volume of 396.000 MMNm³, out of these 9 were located in Austria with a working gas volume of 8.200 MMNm³ [3].

The gas market liberalization and the associated ‘unbundling’ in the 1990s in the USA followed by Europe in the last decade resulted in a change within the whole gas storage business. The gas supply chain was transformed from integrated companies, which were able to optimize a whole system, into numerous independent operators, where each one is responsible to balance its own demand and supply requirements [2, p. 26].

The standing of natural gas as a reliable, versatile and environmentally friendly source of energy is steadily increasing and so does its demand. In the past the gas storage has been seasonal with a clear gap in demand between winter and summer. A new phenomenon could be seen in the USA since 2008, that there is no longer a correlation between the working gas in storage and the wellhead price of natural gas. A main reason for this evidently undesirable situation for storage operators is the abundance of gas, caused by the success of hydraulic fracturing.

Since the gas price is strongly tied to the oil price it saw a rapid decline in conjunction with the oil prices from mid-2014 onward. Resulting in a similar situation in Europe where gas storage operators are facing lower profit margins in their business. A remedy for this situation is to take gas prices into account. In the ideal case the operator would inject gas when the price is low and produce at a high price; thereby the working gas plot should mirror the gas price in the opposite direction. The authors have proposed several simple modifications to generate profitable injection and withdrawal schedules [4]

The endeavor of optimizing gas storage development and operation is no new topic at all; already starting at the end of the 1960s people came up with numerous solutions addressing this issue. In the following paragraph a few examples until now are introduced.

One of the first publications concerning this topic dates back to the year 1968 dealing with a gas storage optimization problem composed of well placement considerations and operational strategy. Henderson et al. applied a two-dimensional single-phase reservoir simulator to the task of finding the optimal well placement scenario which could satisfy the constraints in a beforehand schemed operation plan. It demonstrated the practicality of multidimensional models for planning in petroleum reservoirs. Especially for storage operations it was said to be of utmost importance to evaluate alternate methods of operation as it was to determine the proper placement of wells. Particularly for UGS not only high initial but also an appropriate end-season production is required, to deliver performance based on operational constraints set down in contracts [5]

A paper published in 1970 by Wattenbarger continued with the approach set by Henderson et al. The described work proposed a method for putting the gas withdrawal scheduling problem into a linear programming format. In order to apply linear programming well pressures had to be expressed in terms of well rates. This was done via superposition in time and space so that the pressures of several interacting wells could be calculated via the transient drawdown curve. The author showed that with this approach an optimal solution to the gas withdrawal problem could be found [6].

McVay and Spivey presented a specific procedure for efficient optimization of gas storage reservoir performance by using a minimum number of simulation runs, through rigorous computer reservoir simulation. This procedure was applied to two different tasks, each of major importance. The first is maximizing the working gas volume and peak rates for a predetermined arrangement of reservoir, well and surface facilities. Secondly the minimization of cost to fulfill a specific production and injection schedule resulting from the working gas volume and peak rate requirements. The core of this method is to use the least number of simulation runs necessary to generate plots of average reservoir pressure versus gas in place, production and injection rate. The combination of the beforehand listed plots can later be utilized to find the maximum working gas volume and peak rates. Using the scheme in the opposite direction, by setting performance requirements first, the optimum number and location of wells and surface facilities can be determined [7].

Zangl et al. investigated an approach to reduce computation time needed to execute a numerical full field in an optimization loop. Numerical models are highly capable tools in reservoir management, running these models thousands of times searching for the most likely or profitable solution is a very time consuming task. An ANN can be used as a surrogate to mimic the behavior of the numerical model. Several runs of the numerical model are done, to generate the training input for the ANN, the created inputs should span over the whole range of input parameter variation. The trained neural network is then able to interpolate between the individual simulation runs. In a last step the ANN is used as a proxy function in a genetic algorithm optimizer to find the best possible solution in a minimum time. The proposed workflow can be used to investigate uncertainties of designated input variables, to improve the history matching

procedure and to optimize operational settings [8].

2.2 Types of UGS & Field data

A short introduction of the terms ‘hourly, daily and seasonal storage’ has already been given in the previous subchapter. As for now UGS can be classified roughly into three types according to the underground storage environment:

- **Storage in gas fields:**

This the most common type of UGS, due to its proven ability of containing the gas over a long period of time. The reservoir behavior is usually well understood, due to an abundance of wells and long production history. Considerations with regard to the suitability of a field for conversion to an UGS include the sealing capacity of the cap rock, the condition of wells, reservoir properties, well productivity/injectivity and the influence of a potential aquifer. This type is also often referred to as pore storage.

- **Storage in aquifers:**

“An aquifer is a porous reservoir filled with water (generally saline at depth). By injecting gas into this type of reservoir, it can be converted to a gas storage reservoir. Once in operation the behavior and the operation of the facility is similar to storage in gas reservoirs.” [2, p. 29]

Another noteworthy difference between storage in gas fields and aquifers is that the second one is commonly operated at higher pressures than the initial pressure in the aquifer.

There are a lot of unknowns involved in the development of aquifer storage. A behavior inherent with aquifer UGS is that at the end of an operation usually a large portion of the cushion gas cannot be produced and remains in the reservoir, due to the high capillary pressure of water.

- **Storage in salt caverns:**

A salt cavern is typically formed in a suitable underground formation by dissolving the salt (halite) and thereby creating a cavity which can later on be used to store the gas. This type of storage is primarily used for low volume – high deliverability UGS, which is common in ‘hourly and daily storage’ also sometimes referred to as ‘peak shaver’ facilities. The main difference between a cavern and a pore storage regarding the deliverability, is that in the first one the productivity is only limited by the well configuration, while in the case of the later one also the reservoir itself has a much larger effect [2, p. 29-31].

As previously stated, storage in gas fields is the most common and practical approach to underground storage of gas. The suitability of a gas field for conversion to storage depends on several qualities. First of all, the containment of gas in the underground formation, which includes the cap rock’s pressure threshold, as well as the integrity along all wellbores. Obviously, also the size of the former gas field is of importance, since it has to be large enough to hold the planned working and cushion gas, but it should not be too large, in order to prevent excessive demand of cushion gas. The sheer size only does not decide whether a field is ideal for UGS, it is

also determined by reservoir properties, like porosity, permeability and degree of homogeneity which allow for efficient move of gas through the formation. The well performance is governed by reservoir properties, depth and tubing size. Generally UGS tend to have larger diameter ($7^{5/8}$ " & $9^{5/8}$ ") and horizontal sections in the reservoir's more productive areas. Other properties to consider are the strength and influx from a potential aquifer, the distance to existing infrastructure and market, as connection to an existing pipeline grid can be a major economic consideration [2, p. 27-28].

2.3 Data measurement & Management

Data measurement and management is of essential significance for an efficient and reliable operation of an UGS. Brown and Meikle have shown that a wellhead electronic flow measurement (EFM) system can be used for several applications, ranging from immediate identification of operational problems, pressure transient testing using data measured at the wellhead and automated daily assessment of individual well performance. The key components of the EFM system are located at three locations: the wellsite, the field office and the corporate headquarters. The system at the wellsite comprises several sensors recording the pressure, temperature, flow rate and cumulative flow at 10-second intervals. This dataset is sent via telecommunication to the field and corporate office, where it is stored in relational databases. The authors have come to several points in their conclusion of which the most important are summarized [9].

1. "Wellhead EFM systems can provide the data necessary for effective surveillance of gas storage fields."
2. "Surveillance using EFM can be highly automated."
3. "Well deliverability can be effectively monitored, and declines in well deliverability identified ... "
4. "Wellhead EFM data is of sufficient quality for use in pressure transient analysis."

From Tiani et al. it is evident that the advances in the computer and telecommunication industry allow improving the beforehand mentioned possibilities even further. The usage of real time data from a supervisory control and data acquisition (SCADA) system allowed the authors to create workflows tailored to reservoir management in the gas storage environment. The outcome was a workflow dedicated to data quality control which had to be applied before the dataset could be used in later applications. Secondly a workflow dealing with matching well models with real time data and live testing of this workflow. It is stated that the proposed workflows are just the tip of the iceberg of what is possible and valid real time data bears much more applications [10].

3 Literature Review

3.1 Classical approach to reservoir management

This chapter gives an overview of the methods used in the more classic approach to reservoir management. It discusses the standard "Inflow Performance Relationship" models used with horizontal wells. Furthermore the implementation of the skin effect, due to wellbore damage and turbulence effects, into these models is presented. This is then followed by a comprehensive discussion of "Vertical Lift Performance" and "Nodal Analysis" for system analysis in gas wells. The final chapter presents the approach done in classic reservoir simulation and how the previously described models contribute to it.

3.1.1 IPR for gas wells

The Inflow performance relationship is a mathematical tool used mainly in oil and gas production engineering which represents the well rate as a function of the pressure drop from the reservoir into the wellbore. The IPR depends on several reservoir properties and on the flow regime of the producing well. Usually the data necessary to create the IPR is acquired by measuring the production rates under various drawdown pressures. The reservoir fluid composition and behaviour of the fluid phase under flowing condition determines the shape of the curve.

The most basic form of an IPR for an oil reservoir above the bubble point pressure, also referred to as an under-saturated reservoir, is a linear relationship between the pressure drawdown, which is the pressure drop between reservoir and wellbore, and the flow rate. The relationship between pressure and flow rate can be expressed as given in **eq 3.1** below [11, p. 19-22].

$$PI = \frac{q_o}{p_{res} - p_{wf}} \quad (3.1)$$

PI	Productivity Index [$\text{Sm}^3/\text{day}/\text{bar}$]
q_o	Flow Rate Oil [$\text{Sm}^3/\text{day}/\text{bar}$]
p_{res}	Reservoir Pressure [bar]
p_{wf}	Well Flowing Pressure [bar]

In order to account for gas coming out of solution when dropping below the bubble point pressure, it was vital to introduce non-linear IPR's. These models are especially important for solution-gas drive reservoirs, where dissolution of gas out of the oil phase is the main driving factor.

One of these models was proposed by Vogel [12, p. 83-92] which shows a quadratic relation between pressure and flow rate. This model is valid for vertical oil wells with low to moderate GOR below the bubble point. A combination of the linear and the Vogel model can be applied to reservoirs, which are initially under-saturated and then drop below the bubble point pressure during production.

$$\frac{q_o}{q_{o,max}} = 1 - 0.2 \cdot \frac{p_{wf}}{p_{res}} - 0.8 \cdot \left(\frac{p_{wf}}{p_{res}} \right)^2 \quad (3.2)$$

$q_{o,max}$ Maximum Oil Flow Rate at an atmospheric Bottom hole flowing Pressure [Sm^3/day]

A second common model for an oil reservoir below the bubble point was proposed by [13, p. 1-24]. It was derived from the empirical backpressure gas well deliverability equation proposed by Rawlins and Schellhardt in 1935 [14, p. 1-11].

$$q_{q,o} = C_{g,o} \cdot (p_{res}^2 - p_{wf}^2)^n \quad (3.3)$$

$C_{g,o}$ Flow Coefficient [$\text{Sm}^3/\text{day}/\text{bar}$]

n Deliverability Exponent [-]

In the early days, estimates of gas well performance were conducted by opening the well to the atmosphere and then measuring the flow rate. This so-called "open flow" posed danger to personnel and equipment, wasted otherwise sellable gas and could possibly damage the reservoir. Another downside was that it delivered only limited information of the producing capabilities under varying flow conditions. This practice did however spark the concept of AOF, which is a common indicator of well productivity and denotes the maximum rate at which the reservoir could flow against a theoretical atmospheric backpressure at the reservoir [15, p. 5-7].

The common way to determine the productivity of a gas well is via deliverability testing. These tests provide information used to develop reservoir rate-pressure relations for the well and generate an inflow performance or gas-backpressure curve. One method is the empirical backpressure relationship developed by Rawlins and Schellhardt in 1935. The method was based on the analysis of more than 500 well tests, with the observation that when the differences between squared average and flowing bottomhole pressures were plotted on a log-log scale they appeared to be a straight line.

The Flow Coefficient C is a factor dependent on reservoir parameters like reservoir permeability, thickness, extension and temperature as well as wellbore radius and fluid properties like viscosity, density and compressibility. The deliverability Exponent n is in a way a measure of the flow regime and ranges between 0.5 and 1, where 1 denotes laminar and 0.5 fully turbulent flow. From n being an exponent in the Fetkovich or Rawlins-Schellhardt equation it is apparent that the

flow regime has a major influence on the pressure drop. [15, p. 7-12]

Another analytical approach is the derivation of the single-phase radial inflow equation from the basic material balance equation for flow through a volume element of thickness Δr , height h and porosity Φ at a distance r from the well.

$$q\rho|_{r+\Delta r} - q\rho|_r = 2\pi r h \phi \Delta r \frac{\partial \rho}{\partial t} \quad (3.4)$$

Assuming a homogeneous reservoir with constant thickness, substituting q by Darcy's law and rewriting the compressibility term yields the pressure diffusion equation for radial coordinates:

$$\frac{\partial p}{\partial t} = \frac{k}{\phi \mu c_t} \cdot \frac{1}{r} \cdot \frac{\partial}{\partial r} \left(r \frac{\partial p}{\partial r} \right) \quad (3.5)$$

The solution of the equation above requires a linearisation, because of the dependence of the physical properties ρ , c_t and μ on pressure. For gas it can be linearised using two approximations.

The pressure squared approach substitutes $\rho = \frac{2.7pY_g}{zT}$ and uses the assumption that $\frac{p}{\mu z}$ is a linear function of pressure. By replacing then p by p^2 the relationship is linearised. This assumption is valid up to around 200 bar [16, p. 178-180].

The second method uses pseudo-real pressure $m(p)$, which was defined by [17, p. 624-627]. This is a more rigorous approach, as no assumptions regarding the variations in physical properties as functions of pressure are necessary. This yields a much more precise solution at the cost of increased numerical effort. In the further work only the pseudo-real pressure approach is considered due to its superior precision and the possibility of utilizing a computer for its calculation [16, p. 180-181].

$$m(p) = 2 \cdot \int_{p_{base}}^p \frac{p}{\mu z} dp \quad (3.6)$$

$m(p)$	Pseudo-real Pressure [$\text{Pa}^2/(\text{Pa} \cdot \text{s})$]
r	Distance from well [m]
Δr	Thickness of Volume Element [m]
h	Height of Volume Element [m]
t	Time[s]
q	Flow Rate [m^3/s]
ρ	Density [kg/m^3]
p	Pressure [Pa]
T	Temperature [K]
c	Compressibility [Pa^{-1}]
μ	Viscosity [$\text{Pa} \cdot \text{s}$]
k	Permeability [m^2]
ϕ	Porosity [-]

There are two flow regimes that usually take place in nature; these are transient and pseudo-steady state flow. Transient state flow normally occurs when production from a new reservoir begins. The pressure near the wellbore drops significantly as fluid is withdrawn, however further away from the well no pressure drop can be observed at early times. During transient condition the pressure drop moves, similar to a wave, through the reservoir until it reaches its boundaries. Pseudo-steady state is reached when the pressure drop can be measured at all reservoir locations and the whole reservoir is contributing to the well's production. The mathematical formulation of transient and pseudo-steady state flow regimes are as follows [16, p. 181-189]:

- Transient state: $\frac{dp}{dt} = f(t)$
- Pseudo-steady state: $\frac{dp}{dt} = \text{constant}$

The most convenient way is to utilize pseudo-steady state inflow equations, because these are rather easy to handle and applicable to most reservoirs after a transient period at the early phase of production. The transient solution of the pressure diffusion equation is much more complex and also requires more reservoir parameters to be available for application.

Integrating the pressure diffusion equation with the boundary condition imposed by the pseudo-steady state flow regime yields the following IPR for a vertical gas well, which does, at this state, not include skin effect and non-Darcy effects [16, p. 186-189].

$$q_{sc} = \frac{7.6326 \cdot 10^{-7} kh [m(\bar{p}) - m(p_{wf})]}{T \left[\ln \frac{r_e}{r_w} - \frac{3}{4} \right]} \quad (3.7)$$

q_{sc}	Gas Flow Rate [Sm^3/day]
k	Permeability [md]
h	Reservoir Thickness [m]
T	Reservoir Temperature [K]
\bar{p}	Average Reservoir Pressure [kPa]
p_{wf}	Well Flowing Pressure [kPa]
r_e	Reservoir Radius [m]
r_w	Wellbore Radius [m]

Horizontal wells were more widely introduced in the 1980s and 1990s and since then they have been a vital part of the oil and gas production worldwide. The key benefits of horizontal wells according to [18] are:

- Increased productivity compared to vertical wells
- Reduction of water or gas coning
- Possibility to intersect fractures, i.e. efficient reservoir drainage
- Increased drainage area compared to vertical wells and reduction of vertical wells in low permeable reservoirs
- Increased injectivity

Horizontal wells have proven to be tremendously significant in UGS, as high deliverability is of key importance to storage operators in order to meet the needs of its customers especially during peak demand periods. Another benefit of horizontal wells compared to vertical ones is that their total performance is not impacted with the same severity by formation damage through drilling. This is mainly because of their much larger overall reservoir contact compared to a vertical well. Additionally because of lower delta pressures required for a given flow rate, horizontal wells have the potential of reducing cushion gas and compression requirements [19].

Due to the completely different drainage areas and inflow mechanisms of horizontal wells compared to vertical ones, the IPR of vertical wells cannot adequately describe the inflow into horizontal wells. This resulted in the development of equations for horizontal wells in the last decades which were based on assumptions of flow patterns for those wells. Most of them are combinations of more than one standard pattern e.g. radial, linear, elliptical and spherical flow. Originally horizontal IPR models were developed for oil wells and then modified for compressible fluids like natural gas. Two pseudo-steady state IPR models for horizontal wells are presented in the following.

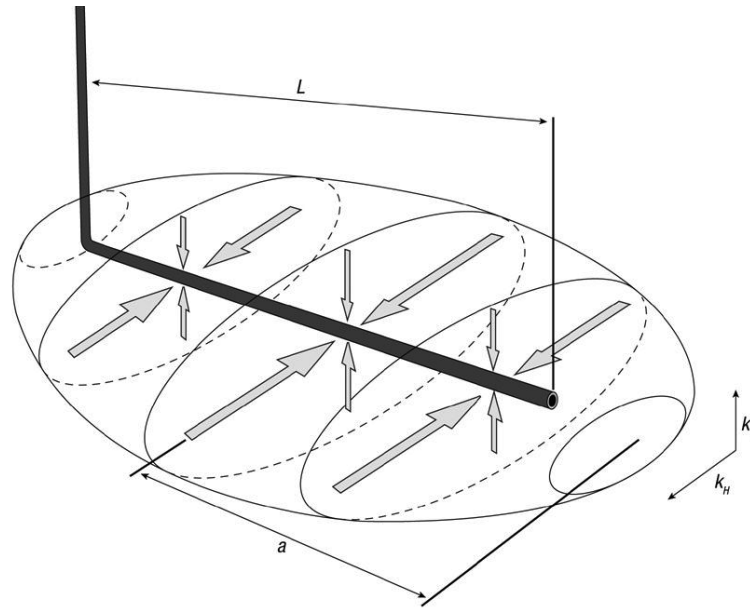


Figure 3.1: Ellipsoidal drainage area of the Joshi horizontal well IPR model [11, p. 98]

The Joshi model from 1989, which was modified by Economides and published in 1994, assumes an ellipsoid as drainage area around the wellbore. Joshi treated the horizontal well inflow as a combination of flow in the $x - y$ plane and the flow in the $y - z$ plane separately. This leads to the hypothesis of a drainage ellipse, which is depicted in Figure 3.1. Economides et al. have extended this model to a general one, for one or more horizontal wells in no-flow boundary “boxes”, in order to create any possible arbitrary well trajectory. The following formula is the Joshi model for using an elliptic drainage area [20, p. 77-79]:

$$q_{sc} = \frac{k_H h (m(\bar{p}) - m(p_{wf}))}{1424T \left(\ln \left\{ \frac{a + \sqrt{a^2 - (0.5L)^2}}{0.5L} \right\} + \frac{I_{ani} h}{L} \left\{ \ln \frac{I_{ani} h}{r_w (I_{ani} + 1)} - 0.75 + Dq_{sc} \right\} \right)} \quad (3.8)$$

$$I_{ani} = \sqrt{\frac{k_H}{k_V}} \quad (3.9)$$

$$a = \frac{L}{2} \left\{ 0.5 + \left[0.25 + \left(\frac{r_{eH}}{0.5L} \right)^4 \right]^{0.5} \right\}^{0.5} \quad \text{for } \frac{L}{2} < 0.9 \cdot r_{eH} \quad (3.10)$$

q_{sc}	Gas Flow Rate [MMscf/d]
k_H, k_V	Horizontal & Vertical Permeability [md]
h	Reservoir Thickness [ft]
L	Horizontal Well Length [ft]
T	Reservoir Temperature [°R]
\bar{p}	Average Reservoir Pressure [psi]
p_{wf}	Well Flowing Pressure [psi]
I_{ani}	Measurement of Vertical to Horizontal Permeability Anisotropy [-]
a	The Large Half-Axis of the Drainage Ellipsoid [ft]
r_{eH}	Drainage Radius in the Horizontal Well [ft]

The Babu and Odeh (1988, 1989) model has a box-shaped drainage area with a horizontal well placed parallel to the x -direction. Figure 3.2 shows a schematic of the box-shaped drainage area and all parameters describing its extension. The well can be in any location within the reservoir, with the constraints that the well must lie in the x -direction and cannot be too close to any boundary. This model is based on radial flow in the $y - z$ plane. The deviation of the drainage area from circular shape is accounted for by a geometry factor and flow from beyond the wellbore in the x -direction accounted for with a partial penetration skin factor.

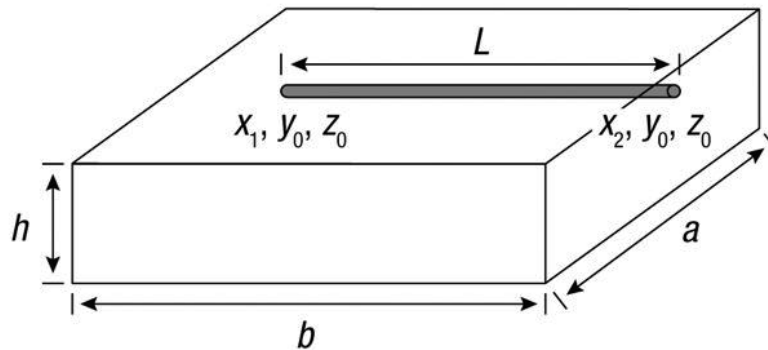


Figure 3.2: Box-shaped drainage area of the Babu & Odeh horizontal well IPR model [11, p. 105]

$$q_{sc} = \frac{b\sqrt{k_y k_z}(m(\bar{p}) - m(p_{wf}))}{1424T \left[\ln \left(\frac{A^{0.5}}{r_w} \right) + \ln C_H - 0.75 + s_R \right]} \quad (3.11)$$

b	Reservoir Length [m]
k_y, k_z	Permeability in y and z Direction [md]
r_w	Wellbore Radius [m]
A	Reservoir Cross-section Area [m ²]
C_H	Shape Factor [-]
s_R	Partial Penetration Skin [-]

In essence the Babu and Odeh model is the procedure for calculating the shape factor and partial penetration skin factor. These factors are dependent on the reservoir geometry, the relation of reservoir length to width and the position of the well in the box-shaped reservoir. Equations describing C_H and s_R can be found in [21, p. 417-421].

3.1.2 Skin & Non-Darcy Effect

As already indicated the inflow equations presented in the previous chapter do not take into account any formation damage, effects from stimulation or hydraulic fracturing, as well as turbulence effects.

A well is rarely drilled under ideal conditions, therefore in most cases the reservoir properties around the wellbore are altered during drilling and completion, which results in a change of permeability in the near wellbore region. Consequently cleanup, stimulation treatment or hydraulic fracturing can increase the permeability in the wellbore's vicinity to a level above the original state. To quantify the changes, which have a tremendous impact on the well performance the concept of the skin factor was developed. Damage in the near wellbore zone and consequently a reduction in permeability causes an additional pressure drop, which can be defined in the following way:

$$S = \frac{7.6326 \cdot 10^{-7} kh}{q_{sc} \cdot T} \Delta m(p)_s \quad (3.12)$$

$$\Delta m(p)_s = m(\dot{p}_{wf}) - m(p_{wf}) \quad (3.13)$$

S	Skin Factor [-]
\dot{p}_{wf}	Ideal Well Flowing Pressure [kPa]
p_{wf}	Actual Well Flowing Pressure [kPa]

A positive skin factor specifies a damaged near wellbore zone, with a reduction of permeability, whereas a negative value indicates a higher permeability around the wellbore than the original one [16, p. 189-191].

The assumption that Darcy's Law represents the relationship between pressure gradients and flow velocity in the reservoir in the derivation of IPR from the pressure diffusion equation is only valid for low velocity or laminar flow. Therefore to account for the turbulence effect around the wellbore which causes an additional pressure drop to the skin effect due to formation damage and completion effects; the apparent skin \acute{s} is given by:

$$\acute{s} = s + Dq_{sc} \quad (3.14)$$

\acute{s}	Apparent Skin Factor [-]
s	Total Skin Factor [-]
D	Non-Darcy Coefficient [Day/Sm ³]
q_{sc}	Flow Rate [Sm ³ /Day]

The total skin factor is the sum of various contributing skin effects, of which some only apply at certain reservoir and completion types. The most prominent are mechanical skin due to formation damage and perforations, completion geometry skin, anisotropy skin and additional mechanical skin from gravel packs, prepacked screens and filter cakes. A more thorough discussion of total skin contributors can be found in the referenced sources [15, p. 241-267], [22, p. 25-38].

At high flow velocities the pressure drop in the wellbore vicinity increases more rapidly than the linear relationship would suggest. This is called non-Darcy flow. Non-Darcy flow typically occurs when the reservoir's Reynolds number which is based on particle diameter in the porous medium, exceeds 10. Kelkar states two common ways in which the reservoir fluid's velocity increases. The first being the reduction of cross-sectional area available for fluid flow when the fluid approaches the wellbore in the radial flow region. Since velocity can be seen as the ratio between flow rate and cross-sectional area, a decrease in area available for fluid results in an increase of velocity. The second way in the expansion of the reservoir fluid as it approaches the wellbore. The reason is the lower well flowing pressure compared to the average reservoir pressure, this phenomenon is especially important for compressible fluids like natural gas.

The Forchheimer equation describes the relation between the velocity and the pressure gradient in a porous medium [16, p. 197-199].

$$-\frac{dp}{dx} = \frac{\mu_g}{k_g}v_g + \rho_g\beta_gv_g^2 \quad (3.15)$$

μ_g	Gas Viscosity [Pa · s]
v_g	Superficial Velocity [m/s]
k_g	Permeability to Gas [m ²]
β_g	Turbulence Factor [m ⁻¹]
ρ_g	Gas Density [kg/m ³]

The turbulence factor β can be calculated by using published theoretical and empirical correlations, which are valid for single-phase gas flow only, therefore the subscript g is dropped from now on. The correlation presented in this thesis is from [23, p. 799-806].

$$\beta = \frac{5.5 \cdot 10^9}{k^{1.25}\phi^{0.75}} \quad (3.16)$$

One way to derive the non-Darcy coefficient D , which was introduced in **eq 3.14** is via the integration of the Forchheimer equation. Substituting the expression for velocity by flow rate and with the assumption that non-Darcy flow is only significant in the near wellbore region allows the integration of the Forchheimer equation, which yields the following expression for D [16, p. 200-204]:

$$D = 2.226 \cdot 10^{-21} \frac{\beta \gamma_g k h}{h_p^2 \mu r_w} \quad (3.17)$$

β_g	Turbulence Factor [ft ⁻¹]
k	Permeability [md]
ϕ	Porosity [-]
D	Non-Darcy Coefficient [Day/Sm ³]
γ_g	Specific Gravity of Gas [-]
h	Reservoir Thickness [m]
h_p	Length of Perforated Interval [m]
μ	Gas Viscosity [Pa · s]
r_w	Wellbore Radius [m]

The literature provides several other empirical correlations for D . However correlations should only be used when no well test data is available. This is because usually D is determined by multi-rate pressure test analysis. Figure 3.3 displays how D and s can be determined from well test analysis.

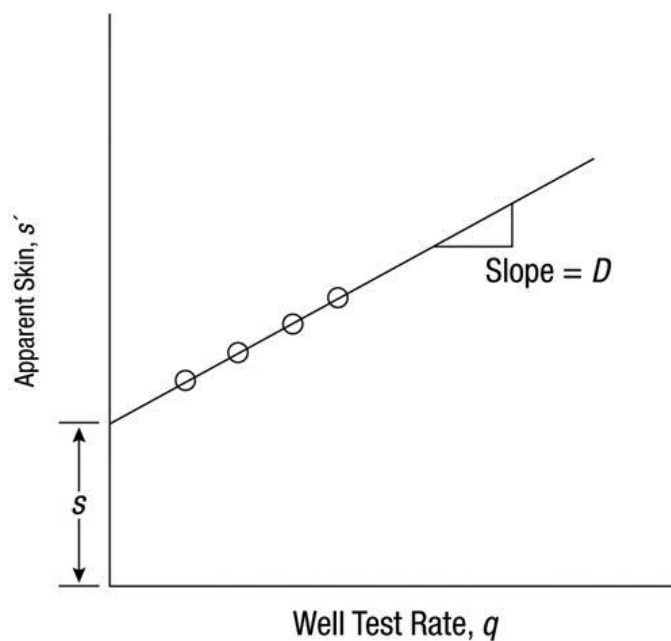


Figure 3.3: Determination of skin effect and Darcy coefficient from multi-rate well tests [11, p. 131]

To assess the non-Darcy flow in a horizontal well a few adjustments have to be made to the correlation for D by Tek et al. [20, p. 78-79]

$$k = \sqrt[3]{k_x k_y k_z} = \sqrt[3]{k_H^2 k_V} \quad (3.18)$$

$$\beta_H = \frac{5.5 \cdot 10^9}{(k_x k_y k_z)^{\frac{5}{12}} \phi^{\frac{3}{4}}} \quad (3.19)$$

$$D_H = \frac{2.22 \cdot 10^{-15} (k_x k_y k_z)^{\frac{1}{3}} \gamma_g \beta_H}{\mu h r_{wH}} \quad (3.20)$$

$$r_{wH} = \frac{r_w (1 + I_{ani})}{2 \cdot I_{ani}} \quad (3.21)$$

k_x, k_y, k_z	Permeability in x, y and z Direction [md]
k_H, k_V	Permeability in Horizontal and Vertical Direction [md]
β_H	Turbulence Factor for Horizontal Well [ft ⁻¹]
r_{wH}	Effective Wellbore Radius for Horizontal Wells [ft]
I_{ani}	Measurement of Vertical to Horizontal Permeability Anisotropy [-]
r_w	Wellbore Radius [ft]

A convenient way to integrate the non-Darcy flow effects into a deliverability relationship derived from the radial pressure diffusion equation was presented by [24, p. 1468-1684]. The relationship can be expressed in terms of pseudo pressure in a quadratic form:

$$m(\bar{p}) - m(p_{wf}) = a_g q_{sc} + b_g q_{sc}^2 \quad (3.22)$$

a_g	Houpeurt Coefficient [Pa ² /Sm ³]
b_g	Houpeurt Coefficient [Pa ² /(Sm ³) ²]

The coefficient a_g is also referred to as the laminar flow coefficient while b_g is the turbulent one. Coefficients a_g and b_g can be determined either by rearranging the deliverability equation 3.22 or preferably by using data from a four point flow after flow well test. Plotting the delta pseudo pressure - $\Delta m(p)$ from the well test data on the ordinate and corresponding gas flow rate q_{sc} on the abscissa should yield a straight line behaviour. From this plot, shown in Figure 3.4, a_g can be read as the intercept and b_g is obtained from the slope of the straight line [15, p. 5-12].

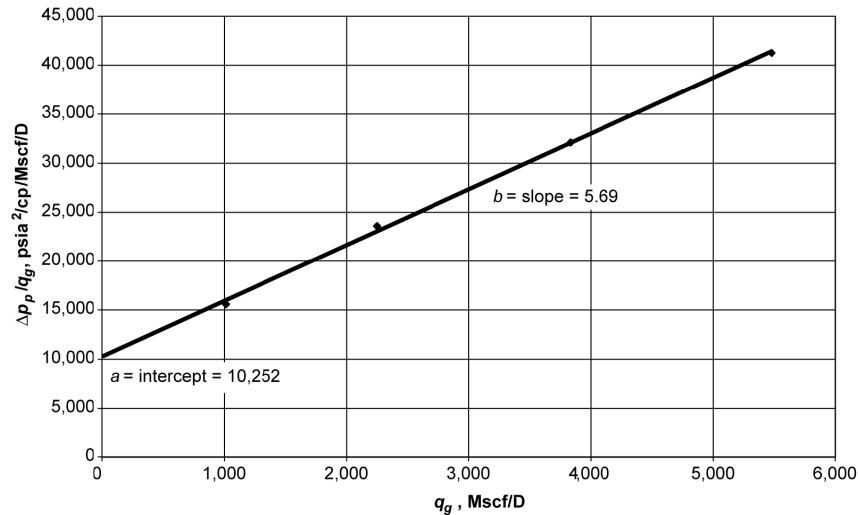


Figure 3.4: Determination of Houptert coefficients from flow-after-flow test data using the Pseudopressure method [25, p. 12]

It is also possible to apply the quadratic formulation in pseudo pressure form for horizontal gas wells. Billiter et al. have applied this concept to the Babu and Odeh pseudo-steady state IPR **eq 3.11** for horizontal gas wells. It is stated that this set of equations is suitable to generate IPR curves from buildup or drawdown well tests, accompanied by the knowledge of key reservoir parameters. Using the idea of apparent skin, various contributions from different skin factors can be included into the IPR as a total skin in addition to the non-Darcy flow effect. The set of equations is as follows [26]:

$$q_{sc} = \frac{-a_{pss} + \sqrt{a_{pss}^2 + 4b_{pss}(m(\bar{p}) - m(p_{wf}))}}{2 \cdot b_{pss}} \quad (3.23)$$

$$q_{pss} = \frac{1422T \left[\left(\ln \left(\frac{\sqrt{A}}{r_w} \right) + \ln C_H - 0.75 + s_R \right) + \frac{b}{L} s_d \right]}{b \sqrt{k_x k_z}} \quad (3.24)$$

$$b_{pss} = \frac{1422T \left[\frac{b}{L} D \right]}{b \sqrt{k_x k_z}} \quad (3.25)$$

3.1.3 VLP for gas wells

The vertical lift performance equation is a mean to relate the downhole pressure in a well to the pressure at the wellhead, with consideration of the flowing fluid, the flow rate and the conduit of flow. With regard to well flow performance the objective is usually to predict the pressure as a function of position between the bottomhole location and the surface. Generally the well flow may be divided into several categories, depending on the flow geometry, the flow rate and fluid properties. Additionally the properties of the fluid, especially their PVT behaviour and rheological features have to be considered. Flow in the well can either be laminar or turbulent depending on the flow rate and fluid properties and this will strongly influence the flow behaviour.

A differentiation of single-phase flow as being laminar or turbulent is typically done via the value of a dimensionless group, the Reynolds number N_{Re} . This number is the ratio of inertial forces to viscous forces in a flowing fluid. The flow regime in a pipe strongly influences the velocity profile in the pipe and the frictional pressure drop among other factors. Generally the transition from laminar to turbulent flow in circular pipes happens around a Reynolds number of 2100 [11, p. 173-174].

$$N_{Re} = \frac{D\rho u}{\mu} \quad (3.26)$$

N_{Re}	Reynolds Number [-]
D	Pipe Inside Diameter [m]
u	Flow Velocity [m/s]
ρ	Density of Fluid [kg/m ³]
μ	Dynamic Viscosity of Fluid [Pa · s]

By solving the mechanical energy balance equation, the pressure drop over a distance L , of a single-phase flow in a pipe can be gained; its differential form is given below.

$$\frac{dp}{\rho} + u du + g dz + \frac{2f_f u^2 dL}{D} = 0 \quad (3.27)$$

dp	Pressure Drop [Pa]
g	Constant of Gravitation circa 9.81 [m/s ²]
dz	Difference in Elevation [m]
dL	Difference in Distance [m]
f_f	Fanning Friction Factor [-]

For an incompressible fluid ($\rho = \text{constant}$) **eq.3.27** can be integrated without additional assumptions and the pressure drop split up into three contributors.

$$\Delta p = \Delta p_{PE} + \Delta p_{KE} + \Delta p_F \quad (3.28)$$

Δp_{PE}	Pressure Drop due to Potential Energy [Pa]
Δp_{KE}	Pressure Drop due to Kinetic Energy [Pa]
Δp_F	Frictional Pressure Drop [Pa]

The pressure drop due to the potential energy Δp_{PE} accounts for the pressure change due to the weight of the fluid column. It is given by the following formula:

$$\Delta p_{PE} = \rho g L \cdot \sin(\theta) \quad (3.29)$$

- θ Angle between Horizontal and Direction of Flow [°]
 L Length of Pipe Segment [m]

The pressure drop due to kinetic energy change Δp_{KE} is the pressure drop resulting from a change in the velocity of the fluid between two positions.

$$\Delta p_{KE} = \frac{\rho}{2}(\Delta u^2) \quad (3.30)$$

The frictional pressure drop Δp_F is calculated with the Fanning equation. The Fanning friction factor differs from the Darcy-Weisbach friction factor f_D , which is four times f_f . In laminar flow the friction factor is a function only depending on the Reynolds number, whereas in the turbulent flow regime, the friction factor does not only depend on the Reynolds number but also on the relative pipe roughness ϵ . The relative pipe roughness is the ratio of the size of surface structures on the pipe wall protruding into the flowing fluid to the pipe diameter [11, p. 178-183].

$$\Delta p_F = \frac{2 \cdot f_f u^2 L}{D} \quad (3.31)$$

The Fanning friction factor for laminar flow can be read from the laminar region of the Moody diagram or calculated via the Hagen-Poiseuille law for fully laminar flow at low flow rate, or for high flow rates in a hydraulically smooth pipe via the formula by Prandtl [27, p. 529-534].

$$f_f = \frac{16}{N_{Re}} \quad (3.32)$$

$$\frac{1}{\sqrt{f_f}} = 4 \cdot \log \left(\sqrt{f_f} \cdot N_{Re} \right) - 4 \quad (3.33)$$

Commonly the Fanning friction factor in the turbulent flow regime is obtained from the Moody friction factor chart (see Figure A1 in Appendix A), which was created from the Colebrook-White equation.

$$\frac{1}{\sqrt{f_f}} = -4 \cdot \log \left(\frac{\epsilon}{3.7065} + \frac{1.2613}{N_{Re} \sqrt{f_f}} \right) \quad (3.34)$$

$$\epsilon = \frac{k}{D} \quad (3.35)$$

- ϵ Relative Pipe Roughness [-]
 k Pipe Surface Roughness [m]

It can be seen that the Colebrook-white equation is implicit in f_f , requiring an iterative solution process. The Chen equation is an explicit equation for the friction factor with similar accuracy as the Colebrook-White equation.

$$\frac{1}{\sqrt{f_f}} = -4 \cdot \log \left\{ \frac{\epsilon}{3.7065} - \frac{5.0452}{N_{Re}} \log \left[\frac{\epsilon^{1.1098}}{2.8257} + \left(\frac{7.149}{N_{Re}} \right)^{0.8981} \right] \right\} \quad (3.36)$$

The Colebrook-White and the Chen equation are said to describe the turbulent flow regime the most accurate at Reynolds numbers higher than 3250. There are numerous other friction factor equations in the literature, some of those are only applicable at certain ranges of N_{Re} and ϵ , where they show comparable performance to the Colebrook-White equation, which is suitable for $N_{Re} > 3250$ and all ranges of ϵ [11, p. 182-185].

The calculation of pressure drops in a gas well requires that the compressibility of the fluid is taken into account. In a compressible medium the fluid density and fluid velocity vary along the pipe. These deviations must be considered when integrating the mechanical energy balance equation. The pressure drop resulting from the change in kinetic energy may be ignored when the depth is greater than 1250m or the flowing wellhead pressure is higher than 35 Bar [28, p. 547-550]. These conditions are fulfilled for the gas wells featured in this thesis.

The mechanical energy balance equation can then be simplified as stated below:

$$\frac{dp}{\rho} + \left(g \cdot \sin(\theta) + \frac{2 \cdot f_f u^2}{D} \right) dL = 0 \quad (3.37)$$

From the real gas law follows the following expression for gas density in terms of specific gas gravity:

$$\rho = \frac{28.97 \gamma_g p}{ZRT} \quad (3.38)$$

γ_g	Gas Specific Gravity [-]
Z	Real Gas Factor [-]
R	Gas Constant circa 8.314 [JK ⁻¹ mol ⁻¹]
T	Temperature [K]

After inserting the expression for density into **eq 3.27**, this equation still contains three variables, Z, T and pressure, which are functions of the position in the pipe. A fast but less accurate way to solve this equation is to use single, average values of temperature and compressibility factor along the pipe segment of interest. This can be done in a semi-iterative approach, whereby the Z factor calculated with the first assumption of starting values is compared to the Z factor computed with the pressure that has been calculated in a previous step [11, p. 185-188].

$$p_2^2 = e^s p_1^2 + \frac{32 f_f}{\pi^2 D^5 \sin(\theta)} \left(\frac{\bar{Z} \bar{T} q p_{sc}}{T_{sc}} \right)^2 (e^s - 1) \quad (3.39)$$

$$s = \frac{-2 \cdot 28.97 \gamma_g g \sin(\theta) L}{ZRT} \quad (3.40)$$

p_{sc}	Pressure – Standard Condition [Pa]
T_{sc}	Temperature – Standard Condition [K]
q	Flow Rate [m^3/s]
Z	Average Real Gas Factor [-]
T	Average Temperature [K]

A second similar method for calculating the pressure in a gas well was proposed by Cullender and Smith and later on modified which lead to reduction in error compared to the original formulation by Cullender and Smith [29, p. 1-10]. A comparison of the average temperature and pressure method stated in **eq 3.39** and the Cullender and Smith method showed nearly the same error, 3.4% to 3.5%, referenced to measurements from 144 gas wells. The specific gas gravity in both approaches can be replaced by a gravity adjusted term γ_m which takes the density variation due to produced liquid into account. It is also stated that both methods produce considerably larger errors at GLR's lower than 10,000 SCF/STB [30, p. 1-10].

It is obvious that in the above described methods a longer flow distance will yield higher errors due to the used approximations. One way to reduce the error is to use more segments along the wellbore and calculate the pressure drop for each segment. Another way is a rigorous solution of **eq 3.37** which will require temperature profiles along the pipe and the compressibility factor being substituted by a function of temperature and pressure using an equation of state. The result would be a more accurate but numerically more intensive solution [11, p. 185-188].

The calculation of bottomhole pressures and pressure gradients in gas wells is a crucial issue concerning the performance evaluation of these wells. It is highly recommended to pay attention to the prevalent flow regimes and fluid types. In an UGS there are generally high flow rate wells, where pressure losses due to friction are very prominent. Therefore it is obligatory to carefully choose suitable friction factor correlations and adjustments for concomitant fluids. Although pressure loss due to kinetic energy change can typically be neglected for many wells, it is beneficial to be aware of its presence. Nowadays softwares used for well performance analysis have an automatic selection of proper correlations already integrated. Additionally software packages use equation of state solutions for PVT parameters accompanied by a temperature profile, to reduce error compared to the average value method.

3.1.4 Nodal Analysis

Nodal system analysis is defined as a system approach to the optimization of oil and gas wells. The objectives of nodal analysis are, to determine the optimal flow rate, while considering wellbore and reservoir limitations, to optimize the system towards producing a target flowrate most economically and to check the whole system for unnecessary restrictions.

The nodal analysis approach aims at optimizing a total well system from the reservoir's outer boundary to the sand face, across perforations and through the completion up to the separator, including any restrictions in tubing, surface chokes at wellhead as well as flowlines. To solve the optimization problem for the total producing system, nodes are used to split the whole into portions represented by different equations or correlations. The nodes themselves can be divided into functional and non-functional nodes. The first ones having a pressure discontinuity across the node, where the pressure – flowrate relationship can be described by a mathematical function and the second type having no pressure discontinuity. A graphical representation of a typical producing system including characteristic node positions is presented in Appendix A, Figure A2 [31, p. 1-19].

The classical way to evaluate well deliverability is via the combination of IPR and VLP curves. The standardized representation of inflow performance relationship curve commonly has the bottomhole flowing pressure plotted on the ordinate and the corresponding production rate on the abscissa. The vertical lift performance curve is depicted again in a standardized manner, with bottomhole flowing pressure on the ordinate and production rate on the abscissa.

The traditional way of solving the optimization of flow rate – pressure relation can be done graphically. Commonly the starting point is the IPR depicted in a p_{wf} vs q_{sc} plot. Then for a given wellhead pressure the bottomhole flowing pressure is calculated by utilizing the mechanical energy balance equation, this yields the so-called VLP curve. The intersection of these curves provides the expected production rate and the corresponding flowing bottomhole pressure. The VLP of gas wells consists of the hydrostatic and frictional pressure drops, though for gas the frictional pressure dominates. This results in a nearly linear VLP curve, with tubing diameter and its selection being the most important factor. Figure 3.5 displays an IPR curve and the influence of the tubing diameter on the well flowing pressure and production rate [11, p. 267-279].

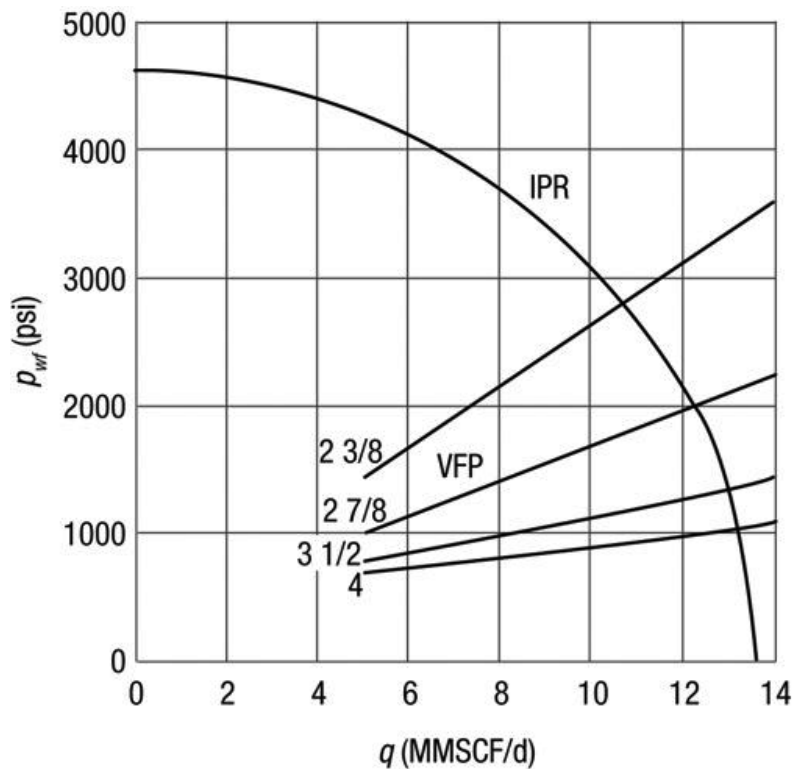


Figure 3.5: Influence of tubing size on the gas VLP curve [11, p. 279]

As already stated the VLP and IPR play an essential role in finding restrictions in the completion and possible damage in the reservoir which prevents the system's operation at an optimal condition. The VLP covers the section from bottomhole to the wellhead, and evaluates the ability of the completion to deliver the reservoir potential to surface. As shown in Fig 3.5 the careful selection of the appropriate tubing and consideration of restriction along its course are vital factors concerning the pressure requirements from the reservoir and possible production rates.

The IPR curve is a measure of reservoir performance and can in the context of nodal analysis be used to detect damage in the reservoir and insufficient connection from it into the wellbore. To recognize reservoir damage or improvement it is necessary to have a base case as reference. This is best done via modelling the IPR from well test data; however it is hardly possible to exactly quantify an already existing initial damage. There are numerous models to mathematically describe the IPR, several of those, which are especially applicable for horizontal gas wells have been presented in a previous chapter. Figure 3.6 shows that generally an additional damage compared to the base case can be recognized as stronger curvature downwards in the IPR, which consequently results in a lower AOF. In contrast an improved reservoir performance, from stimulation or hydraulic fracturing, can be perceived as a less strong downwards curvature and higher AOF. Without continuous performance assessment, it would not be likely to detect a change in well deliverability [32, p. 1751-1763].

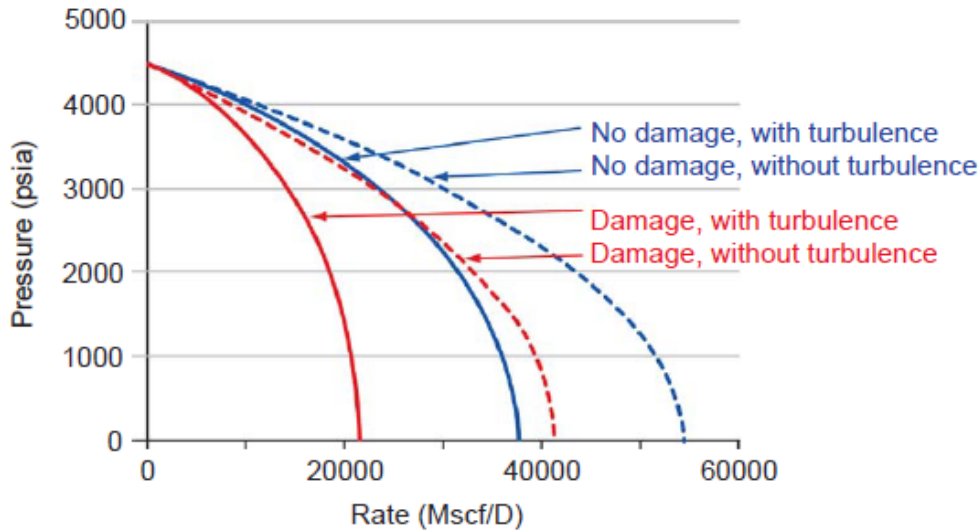


Figure 3.6: Effect of damage and turbulence on the gas inflow performance [33, p. 27]

3.1.5 General reservoir simulation

Reservoir simulation is the application of a proxy model to estimate the performance of a hydrocarbon reservoir under various development scenarios. The main reason is that the actual field can only be produced once, however a simulation model can be run an infinite number of times, trying to find an optimal production scheme, targeted at maximizing net profits and ultimate recovery. The common stages in reservoir simulation are the creation of a reservoir model from available data (static model), the history match of this model with historic production data (dynamic model) and a subsequent simulation of desired operation scenarios. Reservoir simulation is not a new topic, but rather a long standing approach that started with rather simple models and as computational capabilities increased throughout the development of microelectronics, more complicated and complex models became feasible [25, p. 1399-1400].

The foundation of reservoir simulation is a mathematical model describing the material balance, pressure evolution and fluid flow in all computation cells of the reservoir model. This is the so-called mass conservation equation for a porous medium, the general concept behind is that since mass must be conserved; the inflow and outflow in a control volume C_V plus a source or sink term equal the rate of accumulation. The mass conservation equation for three-dimensional single-phase flow is as follows:

$$\nabla(\rho q) \pm \rho q = -\frac{\partial(\rho\phi)}{\partial t} \quad (3.41)$$

Substituting the flow rate q by Darcy's law using the interstitial velocity v , which is flow rate divided by flow area and porosity, yields the general equation for single-phase flow in a porous

medium.

$$\nabla \left[\frac{\rho k}{\mu} (\nabla p - \rho g \nabla D) \right] + q = \frac{\partial(\phi \rho)}{\partial t} \quad (3.42)$$

ρ	Fluid Density [kg/m ³]
q	Flow Rate [m ³ /s]
ϕ	Porosity [-]
k	Permeability [m ²]
μ	Viscosity [Pa · s]
p	Pressure [Pa]
g	Gravitational Constant circa 9.81 [m/s ²]
D	Depth [m]

These partial differential equations are commonly solved numerically as their analytical solution is not trivial and only partly suitable for computer algorithms. The most common numerical approaches are the fully implicit formulation and implicit-explicit formulation, which are also referred to as IMPIMS and IMPES. The implicit formulation solves the **eq 3.42** directly, using an iterative method. The starting point is an assumption of saturations in the control volume, and then the pressures of the respective phases are computed. From these pressures, capillary pressures can be calculated, which can then be converted to saturations again. The deviation of assumed and calculated saturations, determines if additional iterations are required. The IMPES approach is based on the fact that the saturations of all phases add up to one; this allows the modification of conservation equations in such a way as to yield a pressure equation. In this solution method the pressure equation is solved first, resulting in a pressure distribution in the cell, from the pressure distribution the saturations can be computed utilizing the conservation equations. In general, it can be said that the implicit formulation yields higher accuracy at the cost of increased computation time compared to the IMPES method [34, p. 1383-1388]. [35] has improved the stability of IMPES by combining the explicit pressure equation with the solution of implicit saturation for the water phase in a two-phase system. The Adaptive Implicit Method by [36] is a combination of the implicit and explicit formulation which lays both in accuracy and computation time between the former both.

In order to convert the geological model of the reservoir into a discrete one, on which flow equations can be solved, it is divided into a grid of computation cells. There are several types of grids ranging from structured ones, like the regular Cartesian and hexahedral grids to unstructured grids with uneven polygonal cells. The size and number of cells in the fluid flow model depend on the complexity and amount of features in the geological model and on the computation capability available. One way to raise the accuracy in the vicinity of important features like wells or geologic faults is local grid refining, by increasing the number of cells in a limited extent. A higher number of cells allow a more detailed investigation of fluid flow in the reservoir at cost

of increased calculation time. The conversion of the commonly finer meshed geological to the coarser fluid flow model involves upscaling of static and dynamic reservoir parameters. Figure 3.7 depicts a schematic overview of this upscaling process which includes a reduction of geological detail and an averaging of properties. Though there are numerous methods for upscaling the individual reservoir parameters, the selection of appropriate ones is different for every reservoir [25, p. 1415-1432].

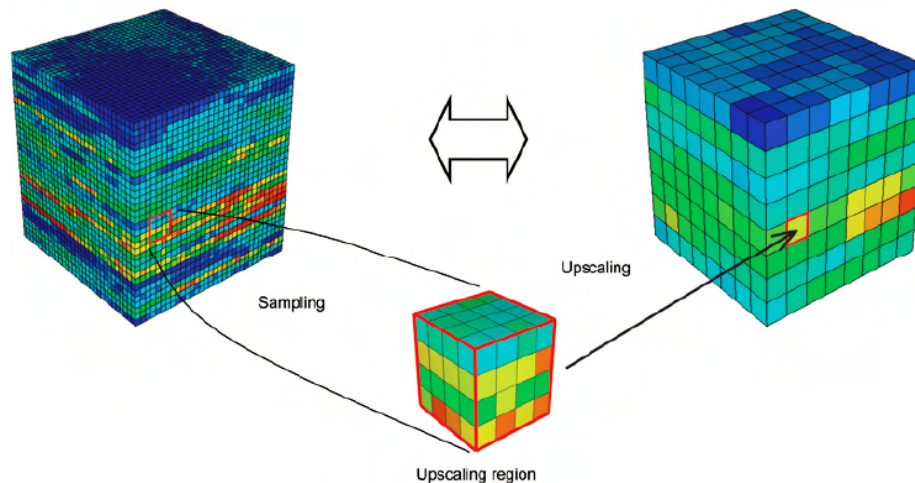


Figure 3.7: Conceptual illustration of the Upscaling process from the fine-scaled geological model to the coarse-scaled simulation model [15, p. 1422]

The solution of the fluid flow equations in the gridded reservoir model requires a kind of discretization to apply those equations on the computation cells; a common type is the finite volume method (FVM). FVM is mainly employed for the solution of numerical fluid mechanics problems, which is also the case for a hydrocarbon reservoir. The basic steps in FVM encompass gridding the domain of application into computational volumes. The next step is formulating the conservation equations including possible boundary conditions in integral form for each cell in dependence of the neighbouring cells. The integrals are then approximated using rules of numerical integration, like the midpoint or Simpson rule. Additionally it is necessary to approximate the fluxes at the faces of the control volume by using neighbouring nodal values. The last step is to assemble the approximations from all control volumes into an algebraic system and consequently its solution. The whole approach can be done in an iterative fashion as described in the section concerning the solution methods [37, p. 77-105].

The generation of a reservoir model involves expertise from petro physics, geology, reservoir and production engineering. The typical constituents are a geological model which acts as the basis, as well as a PVT model describing the fluid properties and their changes with temperature, pressure and interaction between phases. The integration of wells requires models describing governing processes concerning the inflow into the well from the reservoir and the flow in the wells itself.

3.1.6 Wellbore storage & Transient effects

The start of injection or production from a wellbore and in succession from the reservoir is commonly affiliated to transient effects from the reservoir and the wellbore itself. Already in 1953 [38] topics of including wellbore storage and skin effects in the analysis of fluid flow into a wellbore were dealt with. Two years later Gladfelter et al. [39] proposed a method to include wellbore storage effect in the investigation of oilwell pressure buildup data and presented a way to correct for the beforehand mentioned effect. Since then the wellbore storage effect moved into the focus of research on transient well test analysis. Ramey's investigation concerned the effect of Non-Darcy flow and wellbore storage on the pressure in buildup and drawdown tests of gas wells. It was stated that the examination of these effects is vital to retrieve true results from well tests and correction methods for them in gas wells have been proposed [40, p. 223-233].

Wellbore storage describes two phenomena happening immediately after a change of rate. In the case of drawdown part of the production comes from the reservoir but the other part is due to expansion of the fluid already in the wellbore. In the case of a shut-in at the wellhead, the pressure in the wellbore is lower than in the reservoir, resulting in an inflow into the reservoir, the so called afterflow, and compression of the wellbore's fluid. Generally the wellbore storage effect can have two physical explanations, first the expansion and compression of the fluid in the wellbore and second a change of liquid level in the tubing and tubing-casing annulus. An accurate evaluation of wellbore storage is essential for a precise analysis of transient well test data and rate adjustment in the early times of flow. For a packed off gas well only fluid expansion or compression is of significance, which can be expressed as follows [41, p. 147-151]:

$$C = \frac{\Delta V}{\Delta p} = cV_w \quad (3.43)$$

C	Wellbore Storage Coefficient [m^3/Pa]
ΔV	Change in Volume of Fluid in the Wellbore [m^3]
Δp	Change in Bottomhole Pressure [Pa]
V_w	Volume of Wellbore [m^3]
c	Fluid Compressibility [Pa^{-1}]

The wellbore storage effect is a transient phenomenon and therefore it is important to have an idea on how to investigate its duration. A convenient way to investigate the transient flow of real gas through a porous medium is via dimensionless and in some cases additionally dimensionless-pseudo parameters. Ramey [40] reported in 1965 that there is a critical time after which wellbore storage can be neglected. Agarwal et al. [42] in 1970 and in 1988 Oren et al. [43] have extended this approach to include the effect of a positive skin and the later study furthermore the effect of turbulence on the duration of wellbore storage. The following expression utilizing dimensionless

pseudo variables can be used to determine its duration:

$$t_{pD} \geq (60 + 3.5 \cdot s_D) C_D \quad (3.44)$$

The authors also proposed a revised method to calculate the actual flowrate at the sand face from the measured flowrate and the dimensionless pressure. It is stated that van Everdingen and Hurst were the first to emphasize the difference in measured and sand face rate and that the latter should be used to calculate the pressure at the wellbore. While still being influenced by the wellbore storage effect and turbulence through Non-Darcy flow, the following approximation can be used to calculate the flow rate at the sand face [43, p. 547-554].

$$q_{sf} = q_{sc}(1 - e^{-C_2 \cdot t_{pD}}) \quad (3.45)$$

$$C_2 = f(C_D, t_{pD}, s_D, N_{TD}, q_{scD}) \quad (3.46)$$

An overview of the necessary pseudo variables can be found in Appendix A.

Another noteworthy point addressed by Oren et al. is the assumption that the Non-Darcy coefficient D , presented in a previous chapter, is only lightly dependent on the gas viscosity and can therefore be kept constant. The basis for a further investigation on this issue was done in a study by Lee et al. [44] where it was shown that the gas properties indeed have a significant impact on the pseudo pressure drop which can be related to an actual pressure drop. They have defined a dimensionless turbulence number, also called Forchheimer number, N_{Fo} , to specify different flow regions from laminar, transitional to turbulent. It was shown that the assumption of constant D becomes unreliable at high N_{Fo} and at small pseudo times [44, p. 108-120].

This issue is also of importance in modelling of an UGS, as wells in this field of application are usually large in diameter and high in flow rate. The first fact clearly indicates that wellbore storage might play an important role in the performance of an UGS well, as it is directly linked to wellbore dimensions. This becomes even more important when a change in the operation mode from seasonal to daily storage is planned, because it is vital to have experience about the duration of wellbore storage effects during injection and production operations. Secondly the high flow rate clearly promotes turbulence in the wellbore and in its connection to the reservoir. Turbulence and wellbore storage combined can lead to an increased duration of transient effects. Modelling of these transient effects becomes increasingly important in case of a more flexible UGS operation mode. Currently not all reservoir simulation software fulfills those requirements. This is mainly due to their time range being more oriented at long term simulations of reservoirs in pseudo-steady and steady state regimes. This may lead to an incorrect prediction of pressures or rates at the start of production and injection phases.

3.2 Mathematical approach with artificial neural networks

This chapter gives an overview of existing types and architectures of ANN, which are considered as an alternative to the tools of the classical reservoir simulation. In general there are numerous different types of ANN, with various fields of application. Due to the sheer extent of research done on ANN in the past decades only architectures dedicated to function approximation and time series modelling are presented.

3.2.1 Multilayer Perceptron

The Multilayer Perceptron (MLP) can be seen as an evolution of Rosenblatt's perceptron, which is a single layer ANN. The characteristics typical for a MLP can be summed up in three points. First it contains one or more hidden layers which are concealed from the input and output layer, second each neuron in the MLP includes a nonlinear activation function, which is differentiable. Finally there is a high degree of connectivity in the network, which is determined by the synaptic weights of the individual neurons in the ANN.

Generally MLP are fully connected, whereby a neuron in a hidden layer is connected to all neurons in the previous and the subsequent layer. Figure 3.8 below shows an architectural graph of a fully connected MLP, with an input, two hidden and an output layer. The signal from the input layer on the left passes through the network layer by layer to generate an output on the right [45, p. 123-125].

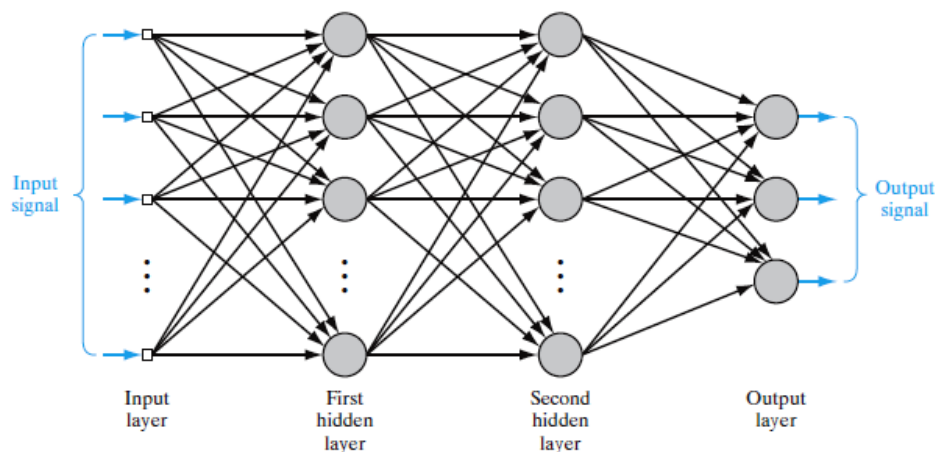


Figure 3.8: Architectural overview of a MLP with two hidden layers and an output layer [45, p. 124]

3.2.1.1 Neuron Model & Number of Weights

The neuron model is inspired by the human brain consisting of a large number of neurons, which are highly connected. In a very broad sense a biological neuron can be divided into three parts. First the cell body which thresholds and processes the incoming signals. The dendrites are receptive networks of fibres which carry the signal coming from another neuron

to the cell body. The axon acts as a transmitter for signals emitted from a neuron to other neurons. The synapse is the point of contact of the axon of a neuron with the dendrite of another one. Although ANN pale compared to the brain's complexity, there are still noteworthy similarities. The first is the high connectivity of neurons in an ANN and secondly that the weights associated with neuron-neuron connections determine the network's functionality [46, p. 1-8 - 1-9].

Figure 3.9(a) depicts the model of neuron commonly used in ANN. The inputs to the neuron are scaled by the synaptic weights associated with each individual connection. The body of the neuron applies a summation of the inputs, which are multiplied with their respective synaptic weight. The activation function processes the input and generates an output. The addition of a bias allows shifting the zero-centered activation function to the left or right. Common activation functions are sigmoid; one of these is shown in Figure 3.9(b). The following set of equations can be used to describe the mathematical operations applied at each neuron [45, p. 10-12]. A broader overview of available activation functions for MLP is given by Figure A3 in Appendix A.

$$v_k = \sum_{j=1}^n w_{kj} x_j \quad (3.47)$$

$$y_k = \phi(v_k) \quad (3.48)$$

v_k	Output of Linear Combiner [-]
w_k	Synaptic Weights of the Neuron [-]
x	Input Signals to Neuron [-]
y_k	Output from Neuron [-]
ϕ	Activation Function [-]

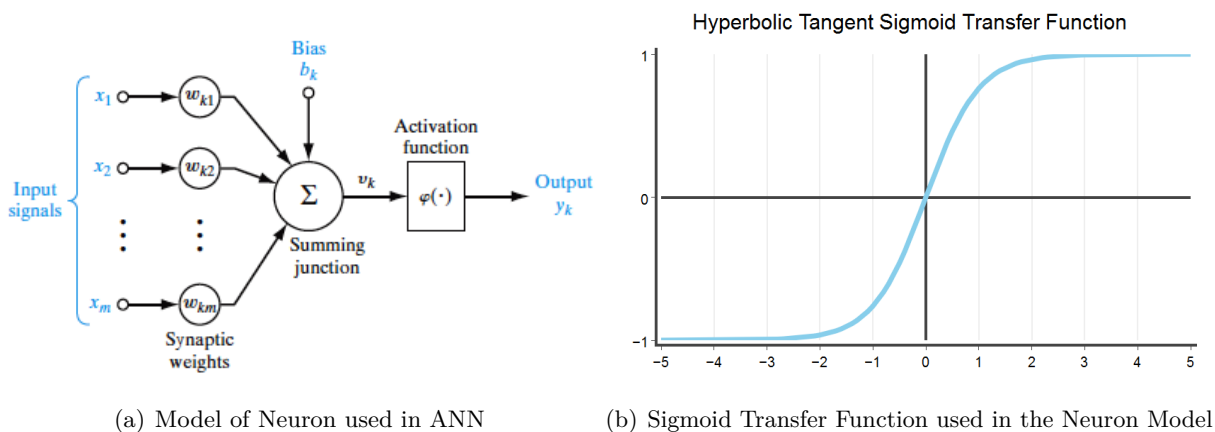


Figure 3.9: Neuron model and sigmoid transfer function [45, p. 11]

The number of weights in an ANN depends on the number of inputs, outputs as well as the number and size of hidden layers. The network's complexity is governed by the number of weights.

The adjustable parameters of the MLP shown in Figure 3.8 can be computed as follows:

$$N_w = (I + 1) \cdot H1 + (H1 + 1) \cdot H2 + (H2 + 1) \cdot O \quad (3.49)$$

N_w	Number of Weights [-]
I	Number of Inputs [-]
$H1$	Number of Neurons in the first Hidden Layer [-]
O	Number of Outputs [-]

3.2.1.2 Input normalization & Weight initialization

For any model development process it is vital to be familiar with the available data. The performance of an ANN is significantly influenced by data preprocessing. An essential part is to investigate the value range of input and target data that are going to be used in the ANN. If not much is known about the physical phenomenon that is to be modelled, it is advised to use a standardization procedure. The purpose is to weigh all input and output variables equally, independent of their actual range.

A common way of scaling the input data is by mapping each input signal to a mean of zero with a standard deviation of one. This type of scaling is typically used in conjunction with sigmoid activation functions, which are symmetrical around zero (see Figure 3.9b). LeCun et al. state that the training process converges faster if additionally to the treatment described above, inputs are scaled in way that they have nearly the same covariance. An exception from scaling to the same covariance should be made if it is known that some inputs are less significant.

$$C_i = \frac{1}{N} \sum_{j=1}^N (z_i^j)^2 \quad (3.50)$$

C	Covariance [-]
N	Number of Samples [-]
z	Sample [-]
i	Index of Input Variable [-]
p	Index of Individual Sample of i^{th} Input Variable [-]

A final step is to check if input variables are correlated. If this is the case redundant information is fed to the ANN, which may slow down the learning process. Correlated input should be decorrelated if possible or removed otherwise. Figure 3.10 depicts the schematic that is applied to the input data [47, p. 17-19].

The training process and ultimately the performance of an ANN are significantly influenced by its initial weights. A simple way to initialize the weights is to choose them randomly. LeCun et al. state that if the above described standardization procedure is applied, whereby the training

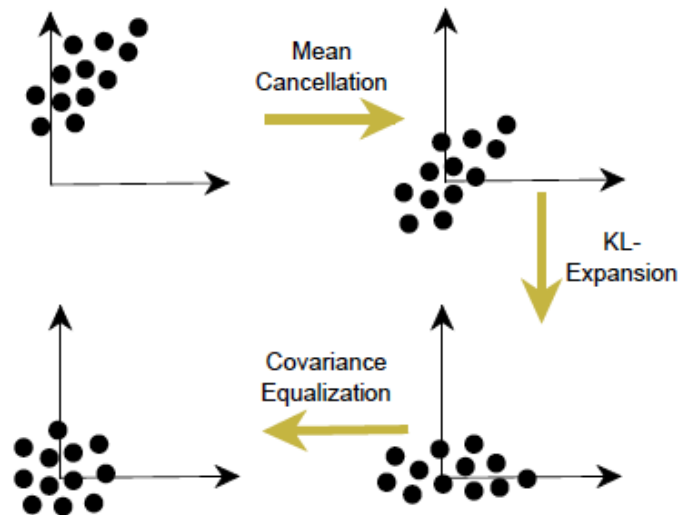


Figure 3.10: Schematic of proposed ANN input data treatment [47, p.19]

set has been normalized to zero mean and a standard deviation of one. By using a symmetric sigmoid activation function, a neuron's initial weights can be randomly drawn from a distribution with zero mean and the following standard deviation [47, p. 22]:

$$\sigma_w = m^{-\frac{1}{2}} \quad (3.51)$$

σ_w Standard Deviation [-]

m Number of Connections Feeding into the Neuron [-]

3.2.1.3 Training of ANN

In order to successfully apply an ANN to unseen data it is obligatory to train the network with known data. This data normally consists of a certain number of samples, with corresponding input and output values. Numerous training algorithms have been developed in the last few decades. The algorithm, which was a breakthrough allowing an efficient training of an ANN, was formulated by Werbos in 1974 and further popularized by Rumelhart and McClelland in 1986.

This so called “Backpropagation” approach is a supervised learning algorithm suitable for MLP networks. Figure 3.11 describes the general flow of information in a MLP network. The function signal graphed in black is the signal generated by the network's input. It passes through the ANN layer by layer and leaves at the output layer. At every pass through a neuron the signal is calculated as a function of the inputs and synaptic weights tied to each individual neuron. Then the output of the ANN is being compared to the desired output and the difference is then considered as the error. Finally the error, shown in blue, is back propagated from the output to the input layer. Thereby the synaptic weights of all neurons are changed according to their leverage on the error. This procedure should result in a better output at the next iteration.

Generally speaking the learning algorithm can be seen as an optimization procedure whereby an error function is minimized. In most cases only a local minimum of this error function can be found, which is highly dependent on initial starting conditions [45, p. 124-125, 129-137].

Generally the complete dataset is split into three parts, which are the training, validation and testing dataset. The network is trained by using the training dataset, the validation dataset is utilized to monitor, how well the network adapts to untrained samples. The testing dataset is totally independent of the training process and is used to check the ANN's response to new data. Another application of the testing dataset is to compare the performance of different networks on the same independent dataset [46, p. 13-5 - 13-6].

Typically the training process is terminated if one or more of the following conditions apply:

- The maximum number of training iterations has been reached.
- The desired error goal has been reached.
- Early Stopping.
- Training does not converge and is therefore aborted.

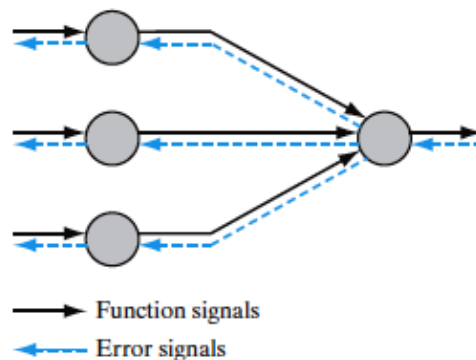


Figure 3.11: Flow of the error signal through an ANN [45, p. 125]

3.2.1.4 Generalization

The generalization of an ANN is its ability to compute a correct input output mapping. To evaluate a network's degree of generalization typically the independent test dataset is utilized. A network that generalizes well is able to produce a most correct output even from an input that slightly differs from the training data set. This means in a best case scenario it will perform as well on unseen data as it does on the training dataset.

Overfitting is a condition of a network that can arise when the training process has not been done properly or the network is much too complex for the problem at hand. A model that suffers from overfitting has a poor predictive performance on unseen data and overreacts to slight fluctuations

in the input signal. The ANN has not appropriately learned to distinguish random noise from the underlying trend in the training data. The problem of overfitting is most prevalent in cases where the number training samples is limited and in the same order of magnitude as the amount of parameters in the network. This allows the network to “remember” the input-output mapping rather than detecting the signal’s underlying trend. The occurrence of overfitting is unlikely if the amount of training samples is much larger than the number of the ANN’s parameters. Figure 3.12a shows the response of a network, graphed in black, which generalizes the underlying function, graphed in blue, well. In Figure 3.12b an overfitting model, which is disturbed by the noise in the input data, can be seen [46, p. 13-1 - 13-6].

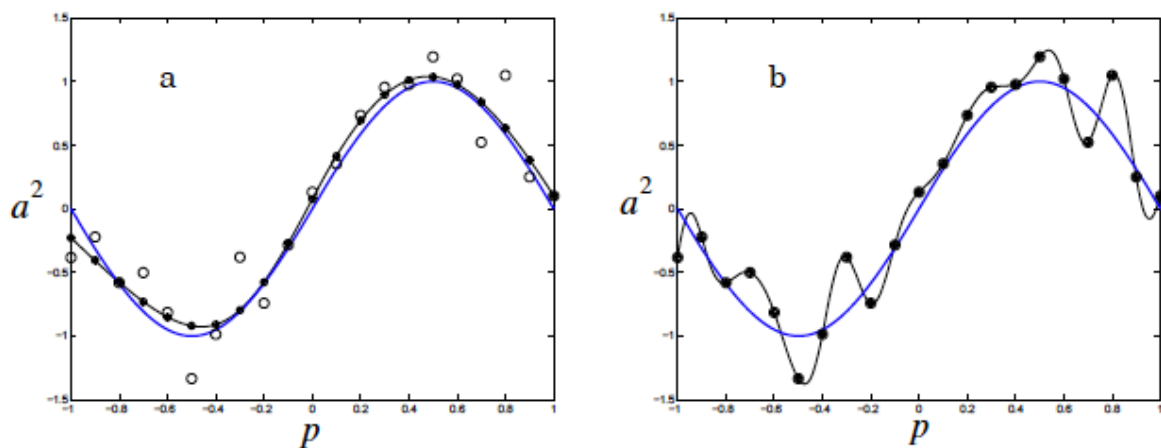


Figure 3.12: The left graph shows a well adapted network vs. an overfitting one on the right hand side [46, p. 13-17]

Improving the generalization capability of an ANN can basically be reduced to the task of finding the simplest network that fits the data. Generally two concepts can be applied which are restricting the number of weights in the network and restricting their magnitude. The first method is done by training a certain number of networks with specified complexity and increase the complexity incrementally. In order to preserve comparability it is vital to use identical datasets and initial conditions when increasing network size. Tracking the error of the validation and testing datasets allows finding the optimal complexity for the problem.

There are two very common means to restrict the magnitude of weights in the network. The first one is the so called “Early Stopping” approach which is used to improve generalization and prevent overfitting. The ANN is trained using only the training dataset and weight changes are based solely on it. The validation set is used to evaluate the ANN’s error on untrained samples. Training is aborted when the validation error stays the same or increases for a specified number of iterations. The weights that produced the best error in the validation set are used as the final ones.

The second approach is called “Regularization” and was introduced by Tikhonov in 1963. This method modifies the error function, generally the sum squared error, in a way as to penalize

network complexity by introducing a new term, which comprises of the weights in an ANN. Thereby reducing the sum squared weights, which will result in a more smooth network output. Equation 3.52 gives the regularized error function used in ANN [46, p. 13-6 - 13-10].

$$E = \gamma \cdot \frac{1}{N} \sum_{i=1}^N (t_i - a_i)^2 + (1 - \gamma) \cdot \frac{1}{N} \sum_{i=1}^N w_i^2 \quad (3.52)$$

E	Error Function [-]
γ	Performance Ratio [-]
N	Number of Samples [-]
t	Target Output [-]
a	Network Output [-]
w	Weights in the Network [-]

The challenge in using regularization is to find the optimal value of γ , which produces the best model. The larger γ is, the smoother the network response.

Another method to improve the generalization is the application of "Classifier Ensembles". The most simple approach thereby is to train several neural networks or any other machine learning tool on the same data set. Then the ensemble output is a combination of the outputs of each of these networks, whereby the simplest way of combining is by averaging. Generally this type of ensemble is only useful if there is sufficient disagreement between the individual outputs. The greatest improvement over the output of a single ANN can be achieved by averaging highly correct networks that disagree as much as possible. This means that not only networks of the same topology and complexity should be combined, but also several ANN of varying complexity [48].

3.2.2 Recurrent ANN

The distinguishing factor between the beforehand discussed MLP architecture and a recurrent ANN is that the later one has at least one feedback loop. The term feedback loop describes a type of neuron-neuron connection whereby the output signal of one neuron is fed back to its own input and/or those of all other neurons. Feedback commonly exists in dynamic systems where the input to a certain part of the system is influenced and dependent on the output of another one and vice versa. This creates one or several closed paths of information transport in the system [45, p. 18-21].

Due to containing feedback loops recurrent ANN are suitable to operate on sequential input, where the ordering of these is of the same importance as their numerical value. The feedback loops enable recurrent ANN to generate a response that depends on historic and current values of the input sequence, thereby allowing the network to learn time dependent patterns. This behaviour can be used to approximate dynamic systems. Typical fields of application of recurrent ANN are listed below [46, p. 14-1 - 14-2]:

- Control of dynamic systems
- Time series prediction
- Speech recognition
- Learning of grammar
- Robotics

3.2.2.1 Fully connected recurrent ANN

There are numerous references to the term “Fully connected recurrent ANN” in the literature concerning recurrent networks; unfortunately its definition varies depending on the individual source. To clarify the terminology in this thesis, the following definition by Williams and Zipser is used. They define a fully recurrent ANN as a network in which any neuron in the hidden layer is connected to:

- All other neurons in the hidden layer
- All inputs to the ANN and outputs from all other neurons
- Itself via a feedback loop

In this configuration any subset of the hidden neurons may act as the designated output and the remaining ones are treated as hidden nodes. Figure 3.13 depicts this configuration in a slightly modified way. The lines in blue show the feedback connections from an individual neuron to itself and all others. The feed forward connections are graphed in black. The recurrent ANN can also be described in the form of a state space model [49, p. 437-440].

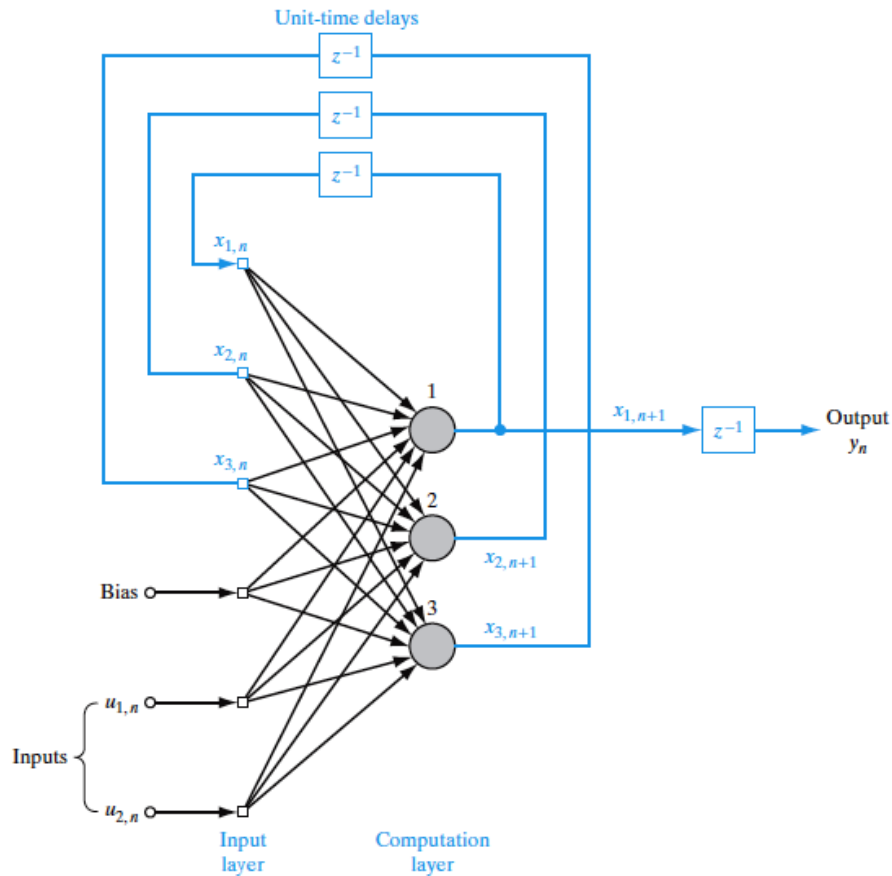


Figure 3.13: Configuration of a recurrent ANN with feedback connections shown in blue [45, p. 799]

The term state of a system is defined as follows:

"The state of a stochastic dynamic system is defined as the minimal amount of information about the effects of past inputs applied to the system that is sufficient to completely describe the future behavior of the system." [45, p. 732].

Mathematically the behaviour of the recurrent network can be described as a dynamical system by the pair of equations:

$$x_k(t) = \begin{cases} x_k^{net}(t), & k \in I \\ y_k(t), & k \in I \end{cases} \quad (3.53)$$

$$s_k(t+1) = \sum_{l \in U} w_{kl} y_l(t) + \sum_{l \in I} w_{kl} x_l^{net}(t) \quad (3.54)$$

$$y_k(t+1) = \phi_k(s_k(t+1)) \quad (3.55)$$

x_k, y_k	Output of a Neuron [-]
s_k	Input to Neuron [-]
w_k	Synaptic Weight [-]
ϕ	Activation Function [-]
t	Time Step [-]

The index k determines whether x_k is the output of a neuron in the network or an external input. The first case applies if $k \in U$ the second one if $k \in I$. A similar rule also holds for the weights in the network. Again U denotes weights associated with neuron–neuron connections and I input to neuron connections [49, p. 437-440].

Williams and Zipser state that this type of network can be applied to any problem, even if both the input and output may be time-varying. Additionally recurrent ANN offer a rich set of possibilities for representing the internal state. The adaptive nature of the state in a recurrent network, allows the creation of delay line structures and other forms of internal memory. This type of memory allows the adjustment to nearly any time variation in input and output variables [49, p. 433-434].

Lo et al. state that a recurrent ANN described by the state space equations listed above is a universal approximator of all non-linear dynamic systems. More specifically a recurrent ANN can describe any dynamic system to any desired degree of accuracy, if it is equipped with an adequate number of hidden neurons [50, p. 217-218].

3.2.2.2 Simple recurrent ANN

One of the most prominent species of simple recurrent ANN is the “Elman Neural Network”. The neural network architectures proposed by Elman were designed to learn time-varying pattern or dynamic systems. The foremost importance was language processing, but in general these architectures could be applied to any problem involving sequences.

This type of network is a simplified version of the fully recurrent ANN, where several neuron–neuron connections are not considered. This reduces the Elman neural network to basically the combination of a MLP with recurrent connections from the hidden layer to a context unit. The purpose of the context unit is to remember the previous internal state, which is defined by the synaptic weights in the hidden layer. Thus the context units provide the network with sort of a memory, which allows it to associate current events with historic sequences. There are feed forward connections from both the input layer and the context unit to the neurons in the hidden layer. The task of the hidden neurons is to find a correct input-output mapping by utilizing both an external input and the previous internal state provided by the context unit. Forcing the neurons to adapt to current and previous inputs, their weights have to represent a time dependent model. Therefore the time effect is inherent in these internal states. Figure 3.14 shows

the block diagram of an Elman network with a single hidden layer. Generally there are as many neurons in the context unit as in the hidden layer [51, p. 182-185].

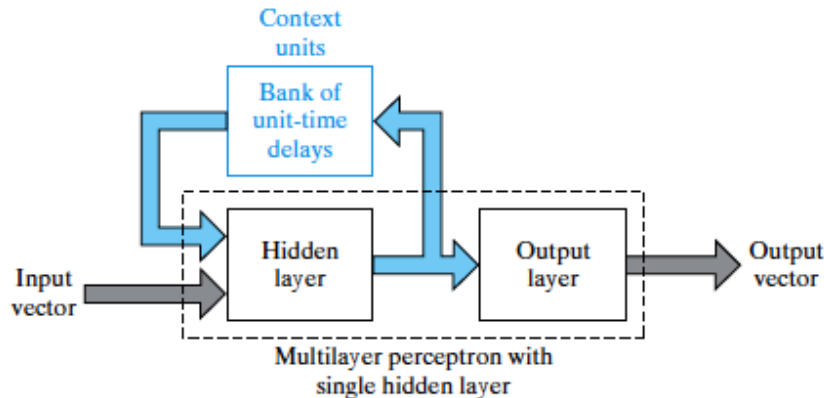


Figure 3.14: Block diagram of the Elman simple recurrent ANN [45, p. 794]

The training process in an Elman network modifies the connection weights from the input layer and the context unit to the hidden layer. The recurrent connections are fixed and not subjected to any adjustment.

3.2.2.3 NARX recurrent network

Another type of recurrent ANN dedicated to tasks involving long term dependencies between the input and output is the so called NARX network. It is based upon the “Nonlinear AutoRegressive model with eXogenous Inputs”. The neural network representation of the NARX model consists of a MLP with two tapped delay lines feeding into the input layer of the MLP. The concept behind these tapped delay lines is to provide the network with a memory. This memory comprises of the current value of an input variable and a certain amount of previous values thereof. Due to the fact that the NARX has two tapped delay lines it is able to draw on memory from previous inputs as well as outputs when computing the output for the current time step. Figure 3.15 depicts a single input – single output NARX network and the related delay lines. The time steps stored in these memory units are treated as additional inputs to the MLP architecture [52, p. 208-210].

Mathematically the operation of a NARX network can be defined as follows:

$$y(t) = \Psi(u(t - n_u), \dots, u(t - 1), u(t), y(t - n_y), \dots, y(t - 1)) \quad (3.56)$$

- u Present and Past Values of the Input, which represent the Exogenous Input [-]
- y Output and Delayed Values thereof [-]
- $n_{u,y}$ Number of Input and Output Delays [-]
- Ψ Mapping performed by the MLP [-]

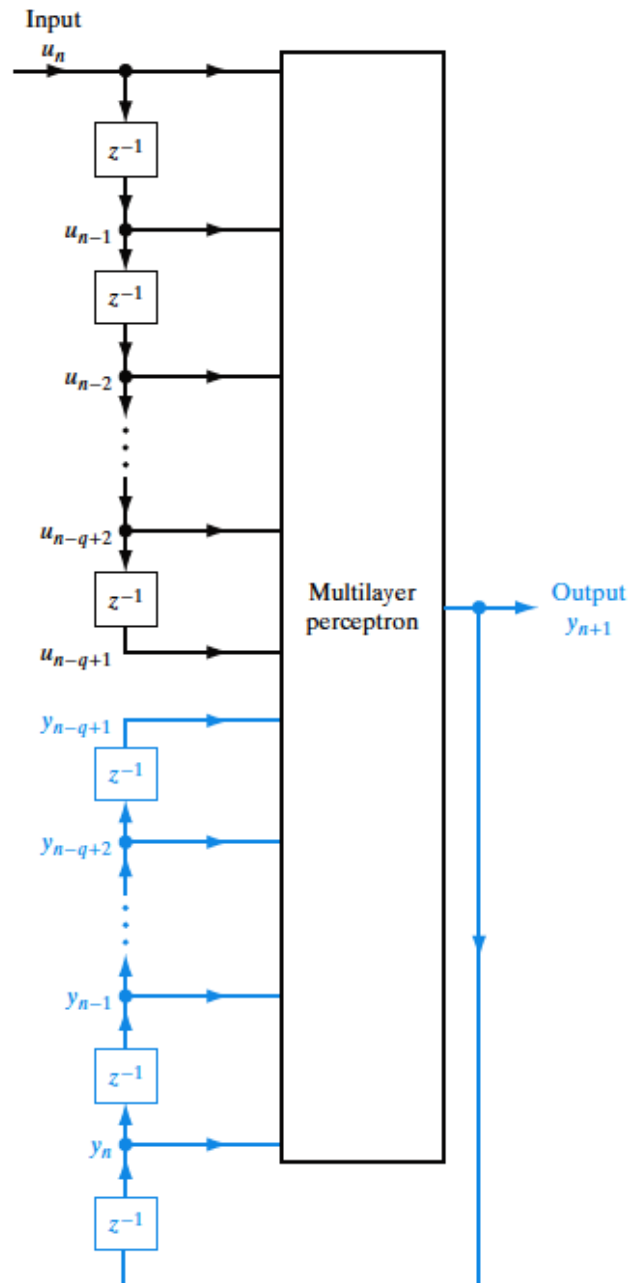


Figure 3.15: Configuration of a NARX model with feedback connections shown in blue [45, p. 792]

The structure of a NARX network with a MLP in its core generally has fewer connections to train compared to fully recurrent ANN. The training of a NARX ANN only adjusts the weights inside the MLP and the feedback loops are replaced by the tapped delay line coming from the network output. Siegelmann et al. could prove that although a NARX network is a lot less complex than the fully recurrent ANN, it is capable to simulate a fully recurrent one. Therefore the NARX can also be considered as a universal approximator. This led to the postulation that NARX networks can replace fully recurrent neural networks without losing any computational

power [52, p. 213-214].

Lin et al. could show that NARX networks are superior in tasks that involve long-term dependences. The reason is said to be that the problem of vanishing gradients is not as prominent in NARX networks as compared to fully recurrent ones. This allows a NARX ANN to better latch on long term dependences prevalent in the input and output data. It was discovered that in those kinds of problems the memory provided by external delay lines made it easier to learn these long term relationships compared to the hidden states in fully recurrent networks [53, p. 1332-1335].

3.2.2.4 Training of recurrent ANN

The training of a recurrent ANN is a much more complex task than for a feed forward network like the standard MLP. The increased complexity in training recurrent ones results from the feedback connections, which are their characteristic feature. The standard “Backpropagation” algorithm which proved to be an efficient way to train feed forward networks cannot be used with recurrent ANN. This is due to the fact that the standard method requires that the network has only feed forward connections and it cannot deal with recurrent ones. Werbos developed an extension of the BP method that made it possible to resolve the issue of recurrent connections. This extension is the so called “Backpropagation Through Time” (BPTT) [54, 1550-1560].

The idea behind the BPTT approach is to unfold the recurrent ANN in time. Thereby identical copies of the network are stacked and the recurrent connections are redirected. This results in a MLP in which all former recurrent loops are represented by only feed forward connections. Now the standard BP algorithm can be applied to this network. Figure 3.16 depicts recurrent network with two neurons, and its unfolded representation. The layered MLP in the lower part grows by a layer at each step of the temporal operation. The feed forward connections are then trained by utilizing the standard BP algorithm [45, p. 808-812].

The BPTT algorithm is generally used with a batch training scheme. This type of training uses the whole time sequence, where at each network copy at the respective time step n the input $u(n)$ is read in and the internal state $x(n)$ and current output $y(n)$ is calculated from $u(n)$ and $x(n - 1)$. Then the error is back propagated through the whole network and the weights are adjusted accordingly. The whole procedure is repeated numerous times until the error function has been minimized to the desired level. The same rules concerning the termination of the training process are applied as in the standard BP [55, p. 12-15].

3.2.2.5 Vanishing Gradients

The problem of vanishing gradients may arise while training recurrent ANN. It is of importance when training is done on sequences in which the desired output is dependent on the input data and output data in the distant past. The root of this issue is the way the error is propagated through the network layer by layer. The BP algorithm calculates the partial derivative of

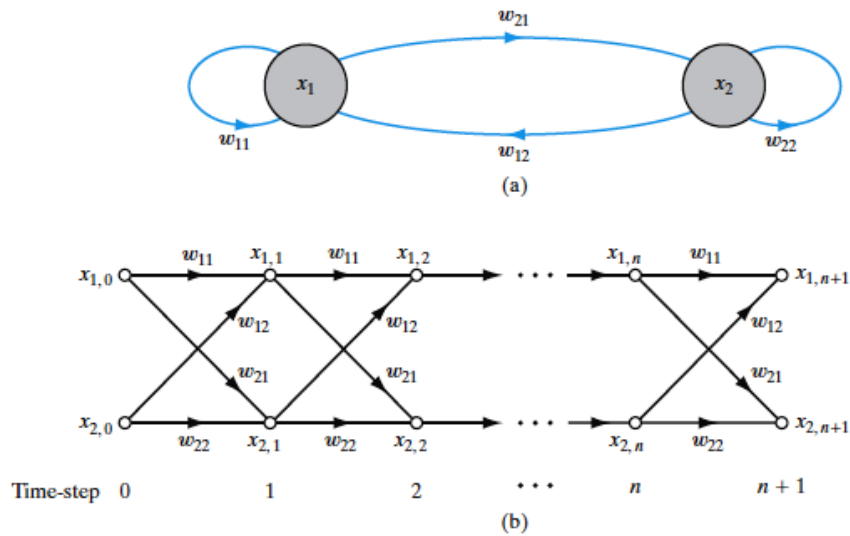


Figure 3.16: Unfolding procedure for a recurrent ANN done by the BPTT method [45, p. 808]

the error with respect to a certain weight by using the chain rule. Generally speaking the more layers a network has the longer the terms have to be considered by the partial derivative. This makes it difficult to honour the influence of temporally distant input, even if a large change in it has an influence on the desired output. The reason behind is that this influence is not measurable by the gradient. The issue of vanishing gradients makes it hard to learn long term dependencies by gradient based algorithms [45, p. 818-821]. Jaeger states the memory span that can be captured by gradient based training is between 10 to 20 time steps [55, p. 14].

Utilizing second order optimization algorithms, such as Quasi Newton, instead of gradient based ones, to adjust the weights might be a remedy to the vanishing gradients. A drawback of these methods is that they are more likely to converge into poor local minima [45, p. 821].

3.2.2.6 Echo State Network

The Echo State Network (ESN) and similar network architectures known as “Liquid State Machines” are summarized by the term “Reservoir Computing”. There are several resemblances between the ESN and previously discussed architectures of recurrent ANN. The concept of the ESN was first published in 2001 by Jaeger. The idea behind this type of network is on the one hand to benefit from the computational power of recurrent networks and on the other hand to avoid some of their weaknesses. They are mainly dedicated to modeling of time series and dynamic system as a whole [56].

The ESN consists of a random recurrent ANN with an arbitrary number of neurons in it. The neurons in this so called reservoir typically are the same as used in other ANN, which utilize sigmoid activation functions. The neurons in the reservoir can be sparsely connected, meaning

that the degree of connectivity in the reservoir can range from a small fraction to one. The reservoir's actual size depends on the task at hand. The input and output units are typically connected to all neurons in the ESN, additionally feedback connections from the output to the reservoir can be used if required. After the reservoir has been created the network is driven by the input sequence of training data. The result is a sequence of reservoir states; these are signals created by the non-linear transformation of the input sequence. The reservoir's state and output signal can be described by the following equations:

$$x(n+1) = f(Wx(n) + W^{in}u(n+1) + W^{fb}y(n)) \quad (3.57)$$

$$y(n) = g(W^{out}[x(n); u(n)]) \quad (3.58)$$

x	Reservoir State [-]
W	Reservoir Weight Matrix [-]
W^{in}	Input Weight Matrix [-]
W^{fb}	Feedback Weight Matrix [-]
W^{out}	Output Weight Matrix [-]
u, y	Input & Output Signal [-]
g, f	Activation Function [-]

The training of an ESN only adjusts the output weights from the reservoir. The internal weights are initialized when the reservoir is created and kept untouched by the training process. The output weights are calculated as the linear regression of the desired outputs on the reservoir states. After this step the ESN is fully trained and ready for use [55, p. 20-32].

A decisive advantage of ESN over other recurrent networks is the incredibly simple training scheme, due to the fact that only the output weights are trained by a computationally least expensive linear regression. Although there are these simplifications in the ESN, they are still able to capture dynamics over time and can successfully model dynamic systems [57, p- 36-38]. Figure 3.17 depicts the model of an ESN, where the five little graphs are the reservoir states generated from the input signal. The solid arrows indicate fixed connections, while the dotted lines represent trainable ones.

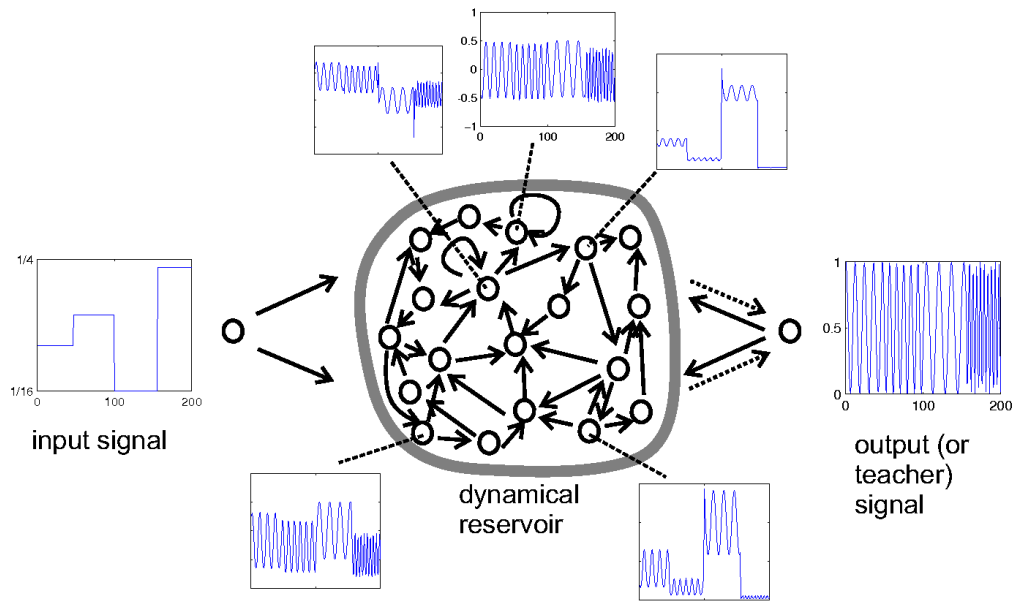


Figure 3.17: ESN with a sparsely connected reservoir. The little graphs show the input & output signal and a representation of the reservoir states [56]

4 Data Management

4.1 Description of available data

The data used in this thesis was provided by several database sources within RAG. Static and dynamic data from the UGS under investigation were gathered from database systems within the RAG company. The static variables gathered consist of the number of wells and their locations within the gas storage field, as well as the size and volume of the UGS. Additionally the length, diameter and extension of each well and the wellhead configurations were readily available. In addition to the static data several dynamic parameters describing the UGS operation were gathered. The most important dynamic variables were the pressure, the flow rate and the temperature measured at the wellhead for all operating and observation wells in the UGS field.

There are currently six operative wells used for injection and production of gas and two observation wells which are mainly used to observe the development of pressure within the reservoir. The six operative wells are denoted as Well 1 – 6 and the two observation wells as Obs-Well 1 and 2. Obs-Well 2 was not used in this thesis, as its wellhead pressure was highly influenced by operating wells in its vicinity. Therefore it could not be used as an input to predict the dynamic pressure at the operating wells. Furthermore long intervals were missing from the measured data.

4.1.1 Confidentiality agreement

Due to a confidentiality agreement with RAG it was not possible to disclose technical details, which would allow conclusions about the UGS being worked on. In order to fulfill this arrangement with RAG, it was necessary to alter the gathered data. This included the names, locations and exact configuration of the wells, but the dynamic data itself was left untouched. This greatly improved the meaningfulness of all statistical plots as well as the time series plots of measured variables. Furthermore the validity of the results from the various ANN and their respective errors is enriched. Normalizing the dynamic data and therefore also the resulting errors of the network prediction would disguise the actually achieved performance. Any assessment of network results without proper knowledge about the numeric range of measured data would lead to invalid and indiscriminate performance estimates.

4.2 Gathering and preparation of data

The dynamic data, namely pressure, temperature and flow rates for all wells were gathered for a continuous 27 months' time period. The datasets retrieved from the database were raw data provided by the sensors at the wellhead, which therefore required extensive data preparation and quality control before it could be used for further data analysis and simulation with an ANN. Another undesired effect of the raw data gathered from databases was the irregular sampling frequency of measurements ranging from a time interval of one millisecond to one hour.

The following list gives the most prominent issues that were encountered during inspection of dynamic data gathered from the database:

- Missing values
- Double values for a specific stamp
- Erroneous measurement
- Irregular sampling intervals and high variation in sampling frequency

4.2.1 Missing values

The issue of missing values was encountered in all data sets of each well, occurring at totally different time intervals for each well. The term missing values refers to intervals in the data set where no data was measured or available in the database for a time interval larger than the maximum sampling interval of one hour. Further investigation showed that the main reason for this were workover operations at the wellsite, malfunctions of sensor equipment, faulty data transmission and failures in the database. Remediation of data sets with regard to missing values had to be done manually, due to the fact that there was no reliable and automatic way of replacing missing data in a meaningful fashion. Missing values at time intervals shorter than one day were replaced by linear interpolation where possible while thereby honouring the overall trend in data. At higher time intervals than one day, data had to be replaced using mathematical expressions tailored to the field of application e.g. approximation of the pressure behaviour at the wellhead after shut-in with an exponential function. Figure 4.1 shows an example of Well 2, where missing pressure data due to a workover operation has been replaced by linear interpolation. The pressure drop in this eight days shut-in period had only been roughly 1.7%, therefore this method was applicable.

4.2.2 Double values

Another noteworthy issue encountered in the raw data was the occurrence of double time stamps. It was possible to distinguish between a systematic and a random occurrence of double values in the datasets under investigation. The systematic event could be traced back to a failure in the database, where adding new sensor data led to the creation of double values at the exact same time each day. An Excel VBA script was issued to check for the occurrence of double time stamps in the gathered data sets. Double time stamps were deleted from the data set, in a way as to preserve the one time stamp with valid readings for pressure, temperature and flow rate.

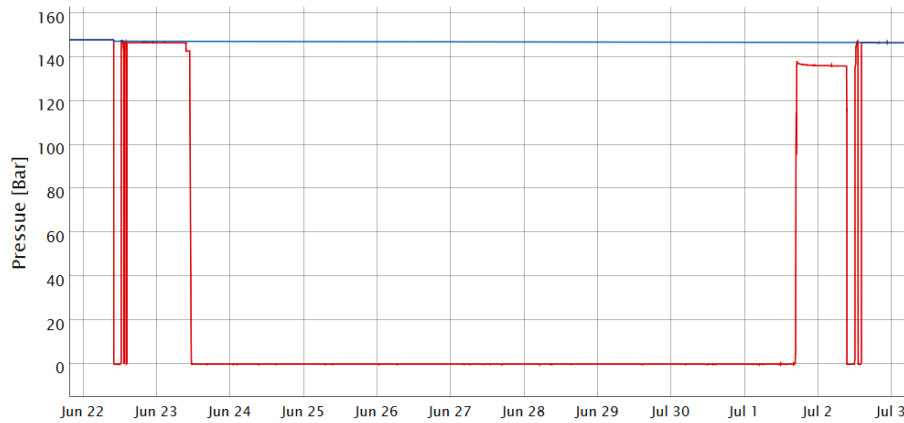


Figure 4.1: Treatment of missing data samples due to workover operations. The measured signal is plotted in red and the replacement is shown as the blue line. The replacement of the missing samples is done by a linear interpolation.

Table 4.1 below shows the occurrence of a systematic double time stamp in the data set. The second time stamp with erroneous data was deleted from the data set.

Table 4.1: Automatic deletion of double time stamps

Wellname	Time Stamp	Pressure	Temperature	Rate
Well 1	01.05.20XX 01:00:00	0.8	0.2	0
Well 1	01.05.20XX 01:00:00	NA	NA	0

4.2.3 Erroneous measurement

This term describes a situation where data could be gathered from the database but it was not usable for further analysis. The main characteristics of such data samples were outliers, which were far off the overall trend of data and constant readings for a parameter, where a certain trend in data over time was expected to happen. The key reasons for this were thought to be sensor malfunctions that led to wrong measurements and workover operation at the well. Detection of faulty data samples was a tricky task which could not be automated, due to the fact that no generally applicable rules for outlier detection could be found. Additionally no reliable mechanism for an automatic classification and remediation of intervals with constant readings was available. Remediation of this kind of flaws in the dataset had to be done manually in an analogous manner as with the missing values. Figure 4.2 shows a situation where automatic outlier detection would have falsely classified valid samples as outliers. The red line indicates that samples lower than e.g. 70 Bar would have been cut if a certain deviation from a mean value was used as the threshold. In reality the sharp decline in pressure was not caused by any sensor malfunction but an unexpected inflow of water into the well.

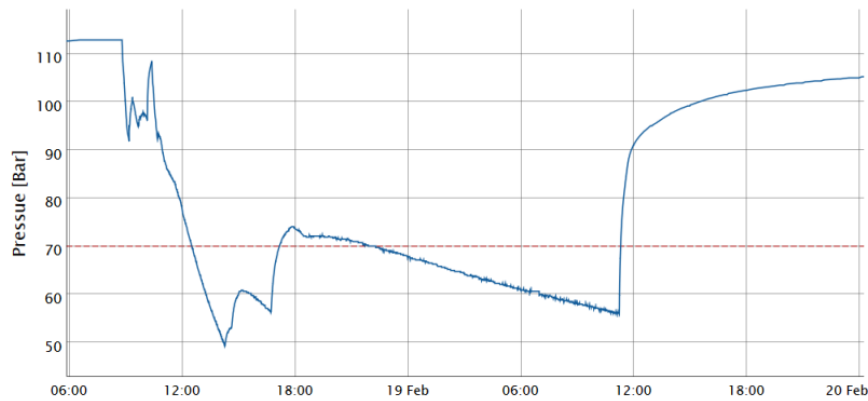


Figure 4.2: Manual distinction between outliers and valuable samples. The blue line is the wellhead pressure and the red dotted line resembles the cutoff of an automatic outlier detection.

4.2.4 Irregular sampling intervals

Gathering of dynamic data from the database made the issue of non-continuously sampled datasets apparent. The sensors measuring pressure, temperature and flow rate were not configured to deliver measurements at a constant frequency but rather at time intervals between one millisecond up to a maximum of one hour. The actual sampling interval was a function of the minimum sampling interval the sensor equipment was able to deliver, which was in this case the beforehand mentioned one millisecond. Another determining factor happened to be the resolution of the sensors for the parameters measured at the wellhead. The resolution of pressure was ± 0.2 bar, for temperature ± 0.15 °C and around ± 100 m³/h for gas flow rate. Therefore during production or injection operations a change of any of these measured variables larger than the threshold triggered the creation of a new time stamp, at which a value for all parameters was logged. After a certain time period of shut-in of a well, when pressure had stabilized at a nearly constant value, only an hourly measurement was delivered. An above threshold change of temperature during shut-in intervals did not trigger the generation of a new time stamp, between the regular hourly ones.

The advantage of the above outlined measuring scheme is a reduced demand of database storage, while the data transmission capacity still has to be designed for full load scenario. A major disadvantage is the high range of time intervals between measurements in the datasets and the wide variation in the distribution of these. Figure 4.3 shows a histogram of time intervals in the dataset from Well 1. It is clearly visible that there is a high concentration of sampling intervals less than 100 seconds with a peak at 80 seconds. There are only few values between the 100 second mark and the final peak at one hour sampling interval.

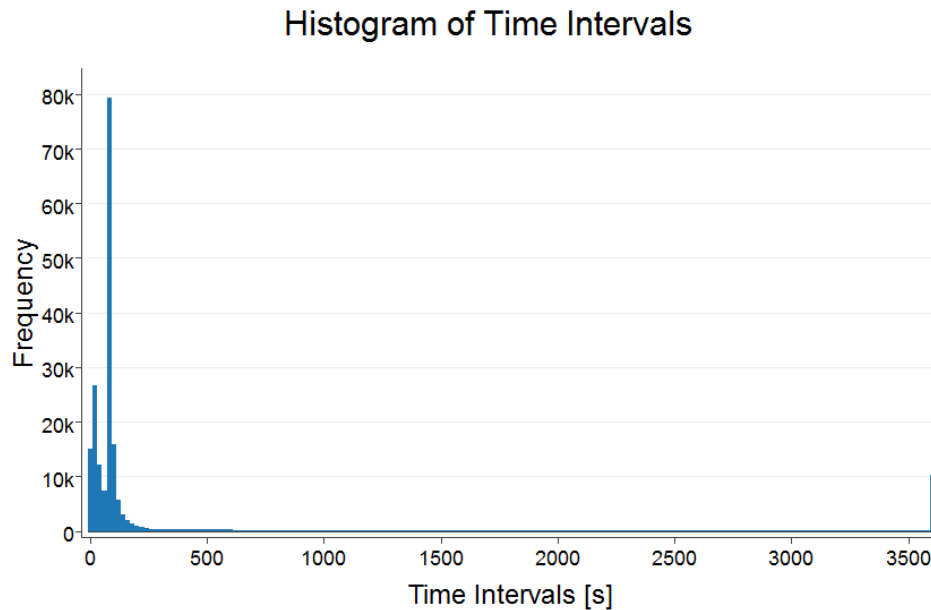


Figure 4.3: Histogram of time intervals in the measured signal coming from Well 1

4.3 Data statistics

To get a better understanding of data and its distribution, a graphic representation of it was advised. Four different types of graphs were utilized namely time series plots, box plots, scatter plots and histograms. The purpose of these graphs was to allow an investigation of the data's features at a glance.

4.3.1 Time series plot

Figure 4.4 displays a time series plot over the whole 27 months of measured data for Well 1. The plot on top displays the pressure measured at the wellhead, the graph in the middle the flow rate and on the bottom the temperature which was also measured at the wellhead is depicted. Negative flow rates indicate injection into the UGS, whilst positive ones resemble gas production. All three graphs are already remedied of measurement errors, outliers and double time stamps. These time series graphs were the starting point for all further data analysis steps using various other plots.

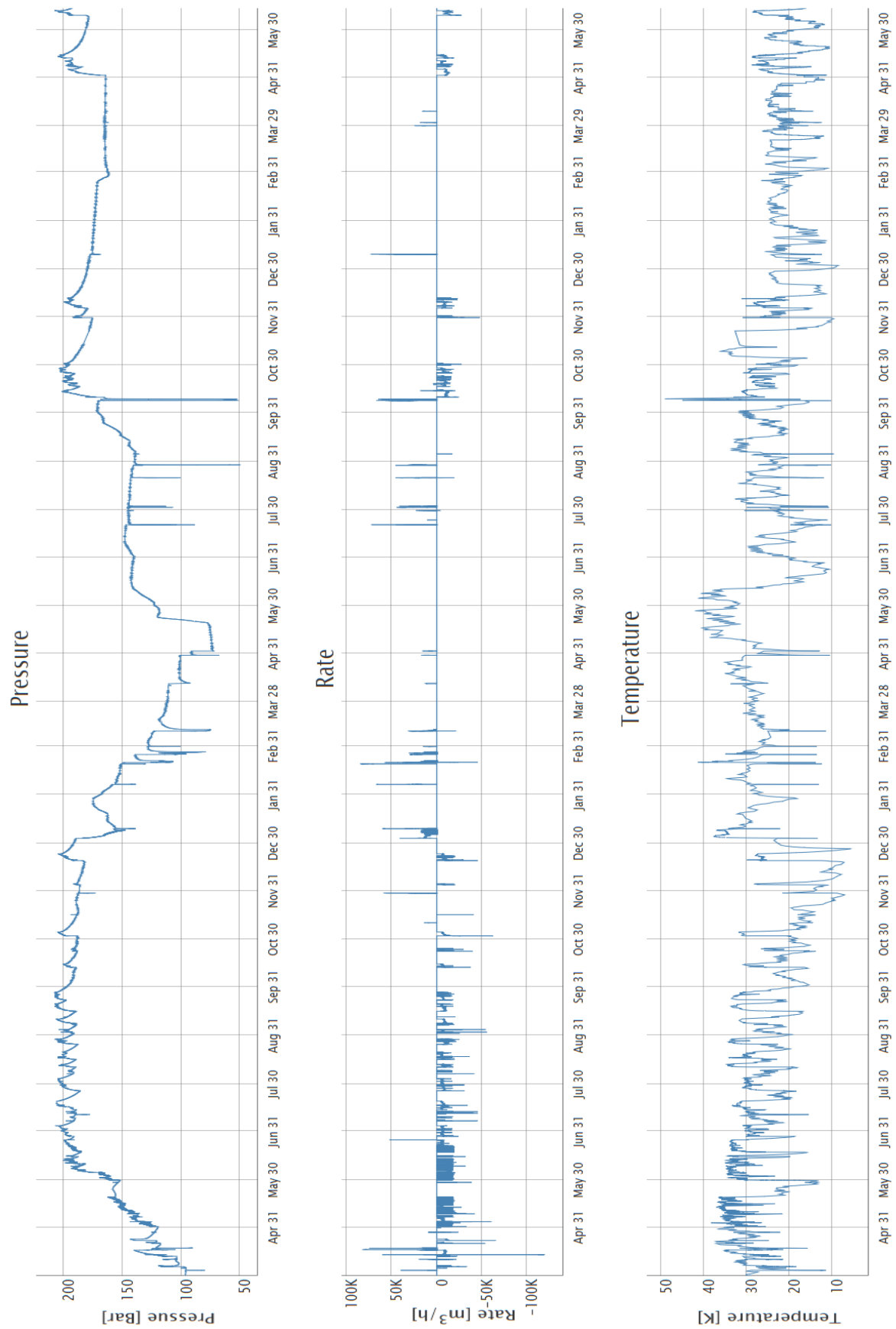


Figure 4.4: Time series plot of pressure, flow rate and temperature measured at the wellhead for Well 1. The unit of pressure is Bar, m³/h for the flow rate and K for the temperature measurement. A positive flow rate indicates production from the UGS, whilst a negative one resembles an injection of gas.

4.3.2 Scatter plot

The purpose of a scatter plot is to allow a first look to check if there is a notable correlation between two variables. This plot can suggest variable kinds of correlations between variables with a certain confidence interval. Another application is to see whether two comparable datasets agree with each other. One of its biggest advantages is the ability to not only show linear but also non-linear relationships between variables. Adjusting the marker's opacity used in the scatter plot allows a graphic estimation of accumulation points and underlying trends [58, p. 125-156].

From the parameters measured at the wellhead three different scatter plots were made. The most relevant graph was used to evaluate if and to which degree a relation between flow rate and pressure at wellhead existed. Therefore a plot of flow rate on the horizontal axis against the pressure on the vertical axis was made. Figure 4.5 (a) - (f) shows scatter plots of flow rate against pressure of all six operating wells. It is visible that there is not just one single identifiable relationship in the data, but rather regions where a certain trend is visible. Another noteworthy fact was that there has not been just a single pressure value for a zero flow rate, but rather numerous values. This circumstance contributed to the selection of a recurrent ANN. Although the actual ranges in between these graphs vary, there are several similarities to be observed. It is clearly visible that all wells show numerous different pressure readings for zero flow rate and all show similar non linear relationships between flow rate and pressure.

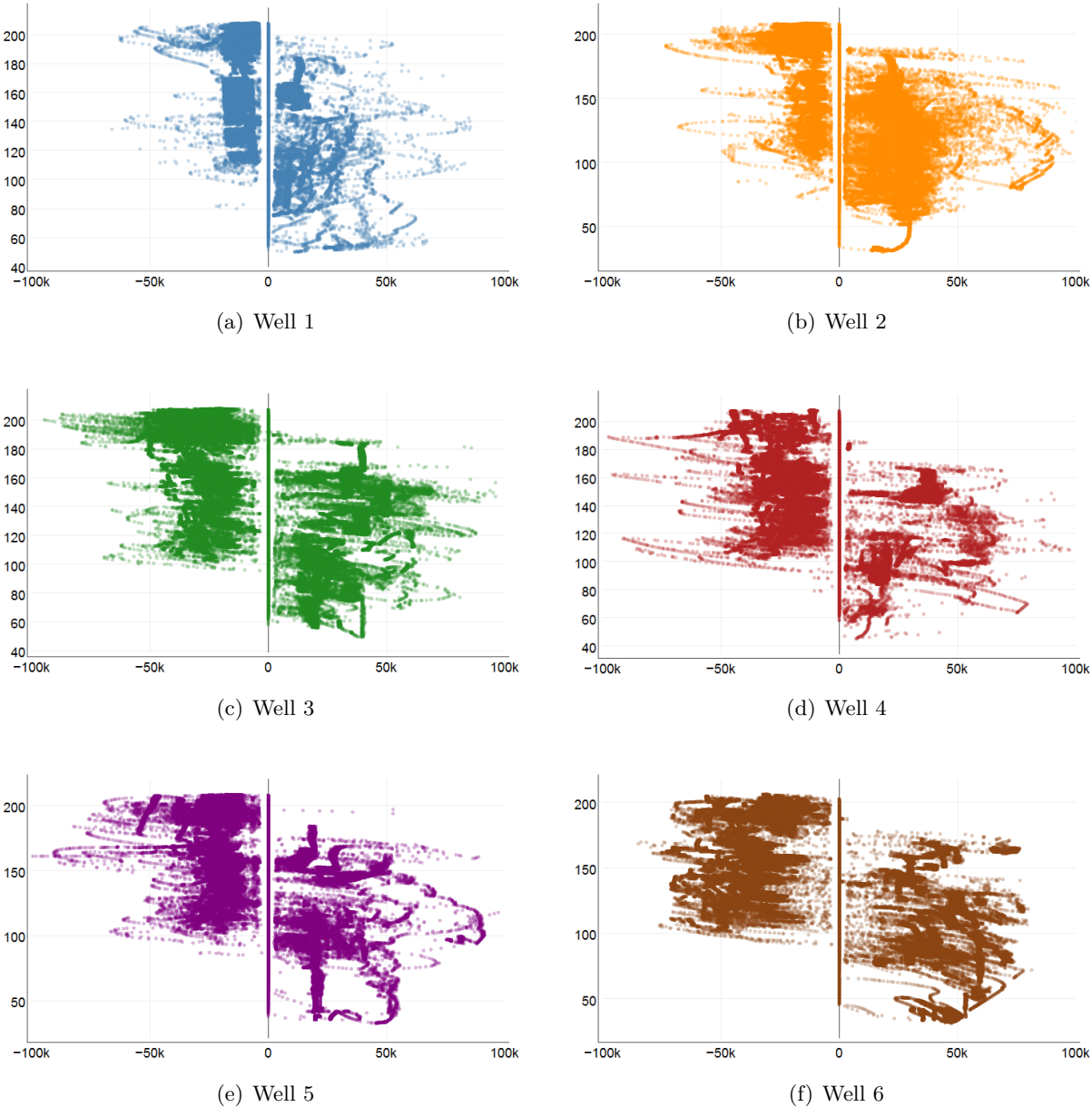


Figure 4.5: Scatter plots of flow rate vs pressure for all six operating wells. The unit of flow rate is m^3/h and Bar for pressure.

4.3.3 Box plot

The box plot, common in descriptive statistics, was the second type of plots used to analyse the gathered data. This is a convenient way of plotting numerical data through their quartiles. In general the top and the bottom of the box are the first and third quartile of data, while the band inside the box is the second quartile also called median. In order to represent data extending beyond the first and third quartile, lines called whiskers are added vertically to the box. The most common way to define their extension is to add 1.5 times the interquartile range (IQR) above and below the box. Any samples lying outside of the whiskers are usually plotted as individual points and considered outliers. Although one would expect equal whiskers lengths from this configuration, it is also a general convention to set the whiskers' end where the last data point inside of $1.5 \cdot \text{IQR}$ lies. A box plot is an easy and suitable way of displaying and comparing groups of similar data. It depicts the variation in data without making an assumption of the underlying statistical distribution [58, p. 39-49].

Grouped box plots were utilized for the investigation and comparison of pressure and flow rate data sets. The box plot of pressure data contained the data sets of Wells 1 - 6 and Obs-Well 1, while the plot of flow rate was limited to the six operative wells. Figure 4.6 shows the box plot of pressure data. It is clearly visible that all six operative wells lie in a quite similar range, while the observation well shows a noticeable lower pressure range. The explanation for this is that the measurements at the observation well were purely static pressures while the pressure at the operative wells was heavily influenced by the flow rate. Noteworthy information was the lack of outliers above the upper whisker, resulting from an imposed regulatory maximum pressure for the operating wells. Further investigation showed that the causes of the outliers at low pressures were mainly an inflow of water and/or a high production rate.

The box plot of flow rates seen in Figure 4.7 shows rather different IQRs for the various wells. This resembled in a way the performance of a well compared to the others. The position of the boxes and thus also the medians' with respect to the zero line presented the allocation of injection and production operation in the available time range. The reason for the high number of outliers, especially in wells 1 - 5, was thought to stem from transients like the wellbore storage effect. However no satisfactory explanation for the lack of outliers in Well 6 could be found. Further investigation concerning the outliers in the box plots was necessary to distinguish between faulty samples and data points which contain valuable information.

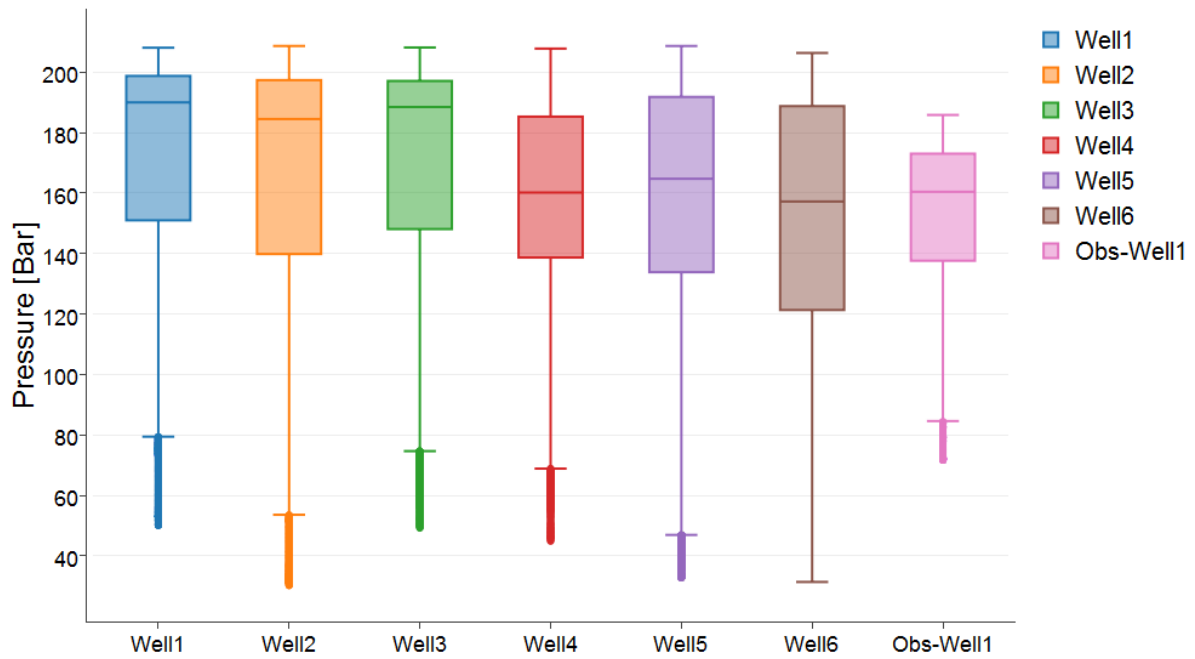


Figure 4.6: Box plot of pressures of all wells

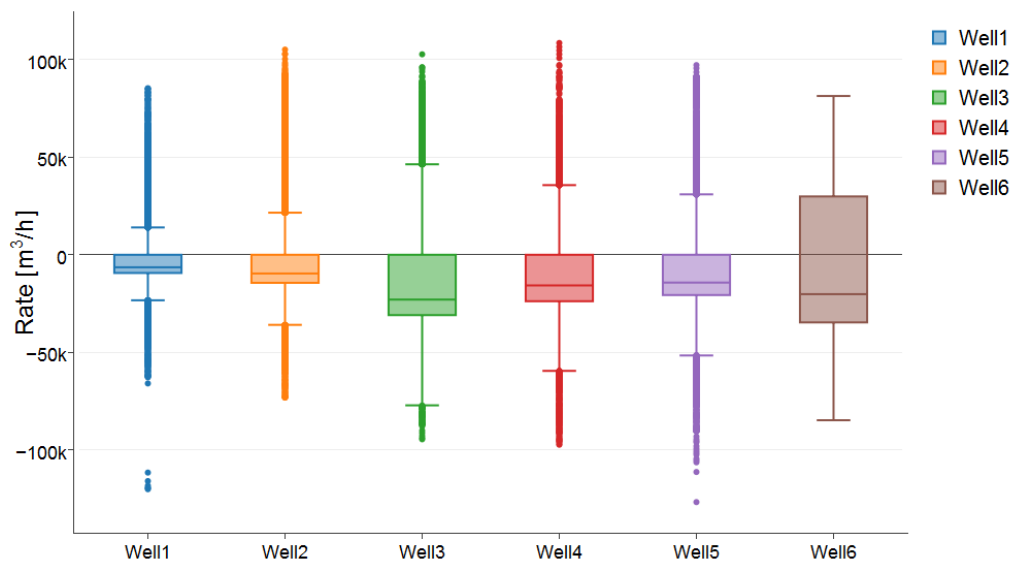


Figure 4.7: Boxplot of flow rates of all wells

4.3.4 Histogram

The purpose of a histogram is to provide a graphical representation of the distribution of numerical data. It is commonly used in descriptive statistics to estimate the probability density distribution of a quantitative variable. In a histogram the data is divided into a number of bins, which contain all samples falling in a certain value range. Then bars are erected above the bins with their height being proportional to the count of samples, also called frequency, in a bin. In general, a lower number of bins reduces noise originating from sampling randomness. A higher count increases the precision of the distribution estimation, while being more vulnerable to the adverse influence of random noise [59, p. 77-83].

In order to identify any present probability density function in the measured data sets, histograms for pressure and flow rate were made. Figure 4.8(a) - (f) depicts histograms of pressure for all six operating wells and Figure 4.9(a) - (f) shows the corresponding histograms of flow rates. The patterns shown in Figure 4.8(a), (b), (c) and (e) are quite similar and could be described as skewed left, showing a tendency to higher pressures in the range of 180 to 210 Bar. The remaining two wells displayed in Figure 4.8(d) & (f) show a less clear distribution of pressure readings. The histograms of flow rate for the six operating wells are depicted in Figure 4.9(a) - (f). All samples with zero flow rate have been removed in those histograms, as these samples would have resulted in a huge peak in the middle obscuring any recognizable distribution. The histograms' patterns could be described as bimodal, with a strong emphasis on negative flow rates. These histograms reveal that during the 27 months long sequence more gas has been injected than produced. It is also clearly visible that the examined wells vary significantly in their achievable injection and production rates. It can also be observed that the histograms of wells 1, 2 and 3 depicting the measured pressure and those of the corresponding flow rate are quite similar, indicating that these wells were operated at comparable conditions.

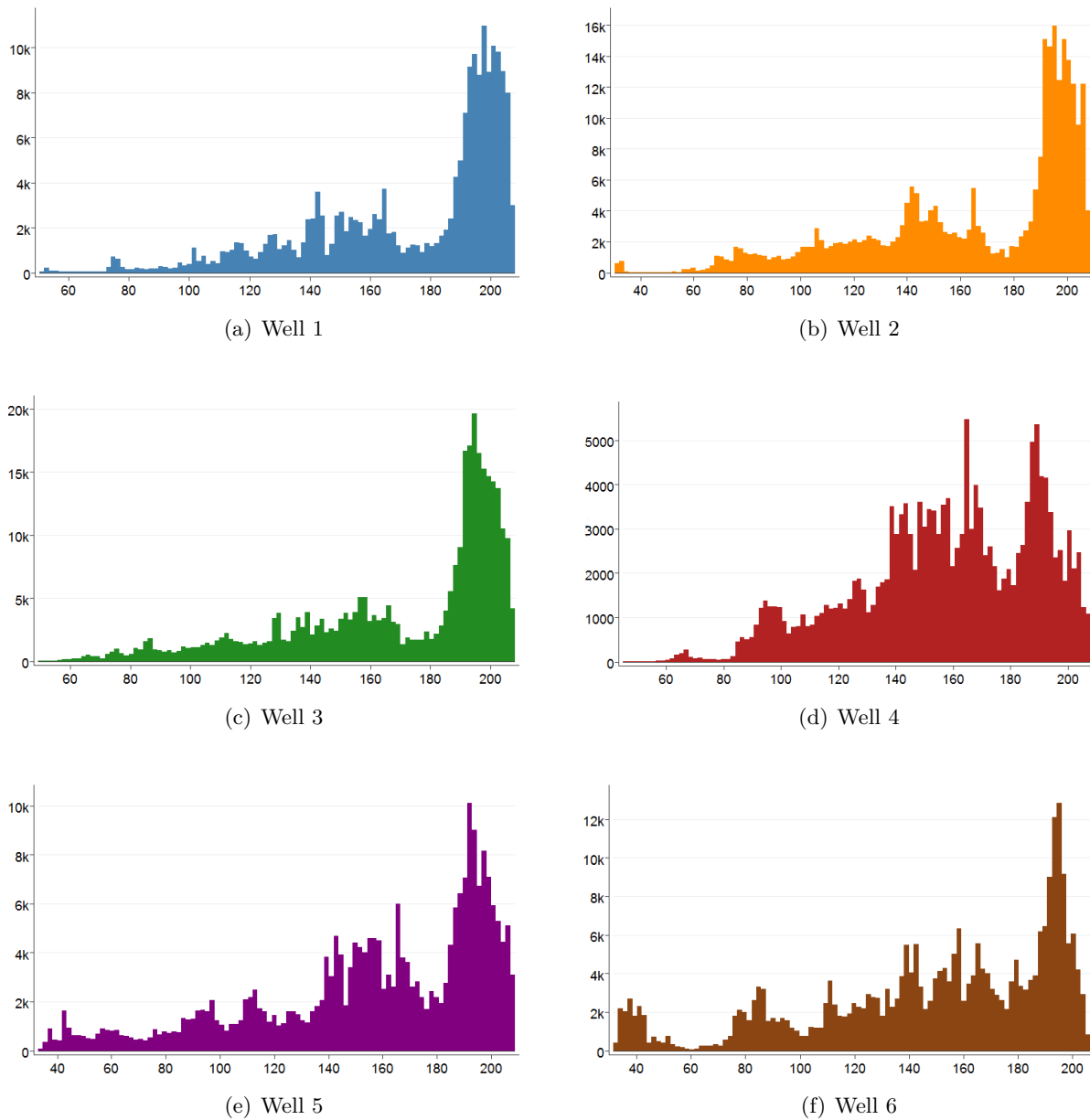


Figure 4.8: Histograms (100 bins) of the pressure for all six operating wells. The x-axis displays the pressure in Bar and the y-axis the frequency.

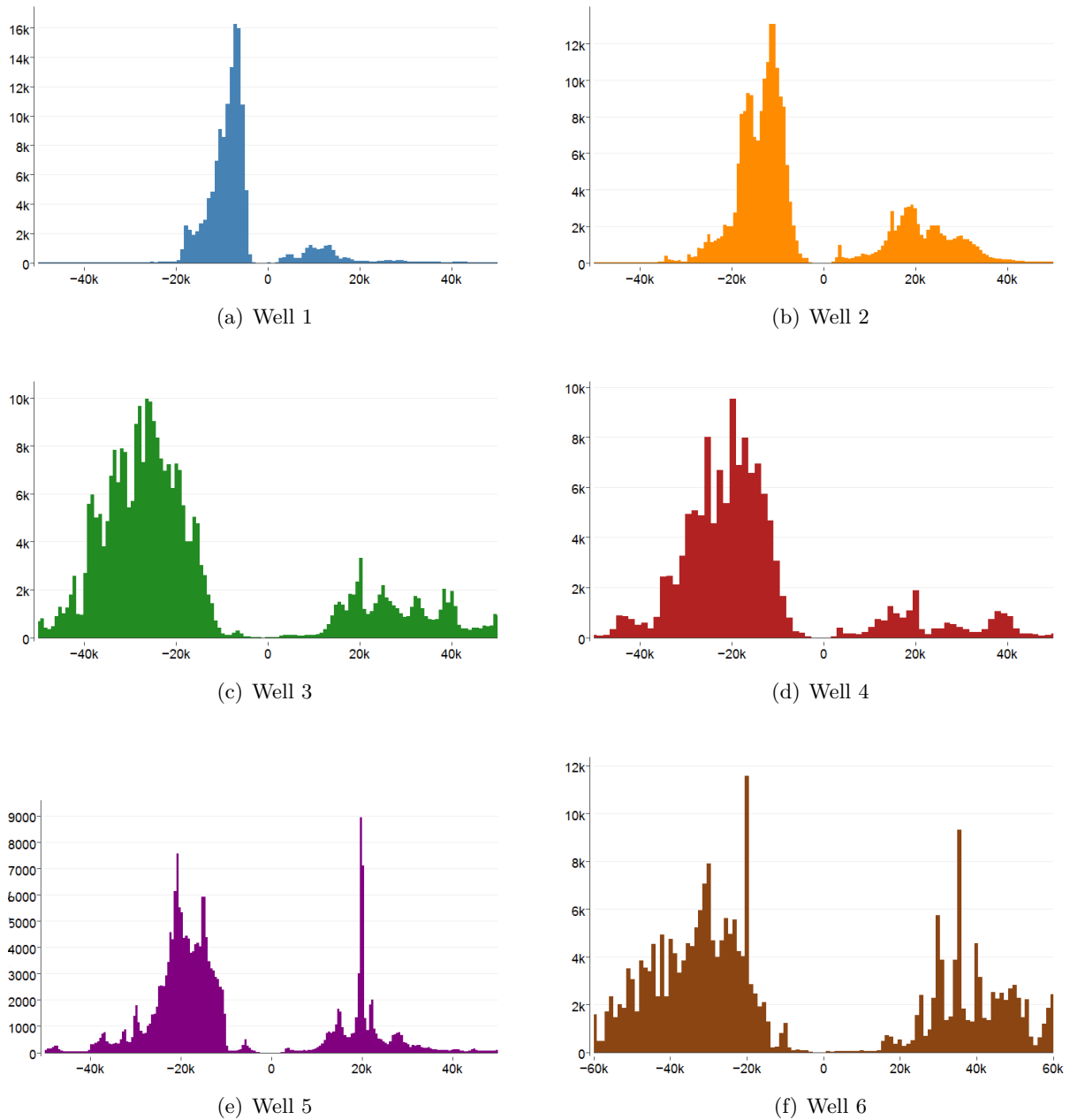


Figure 4.9: Histograms (250 bins) of the flow rate for all six operating wells. The x-axis displays the flow rate in m^3/h and the y-axis the frequency.

4.4 Resampling

This chapter describes the procedure applied to compile the data sets from all wells into a single set with an uniform sampling interval. The wide distribution of sampling intervals in the individual data sets and the variation in between different wells made this compilation to a uniform sampling frequency a challenging task. There were several reasons why it was an absolute requirement to have a single regularly sampled dataset.

First of all to enable reasonable training, simulation and prediction with an ANN using data coupled to a time stamp, also referred to as a time series, it was essential to have a uniform sampling frequency. In addition the data from the sensors was delivered with a substantial amount of undesired noise in it. The various filters available for noise reduction require a constant sampling frequency to work properly. Two different approaches to assemble time series data from all wells into one coherent dataset were utilized. The MATLAB scripts written for resampling the irregularly sampled signal to a regular one are to be found in Appendix B. This includes the linear resampling with filtering and the kernel regression method.

4.4.1 Linear resampling and filtering

The first method utilized linear resampling of time series data to a new a uniformly sampled time vector and applying a median and low pass filter afterwards to reduce noise and mitigate aliasing effects.

The term aliasing describes a phenomenon in signal processing where undesired effects occur at digitalizing of analogue signals or during resampling. During resampling the original signal is sampled at regular time intervals and can later be restored with a low pass filter. In order to ensure a proper restoration of the signal, the original signal has to be sampled with at least twice the highest frequency occurring in the original signal, this relation is the so-called Nyquist-Shannon sampling theorem. If a too low sampling frequency is being used, frequency content higher than half of the sampling frequency is under sampled. Figure 4.10 shows the outcome of under sampling. The upper graph shows the original signal, whose frequency increases with time and is sampled at regular time intervals. The lower graph shows the reconstructed signal, which is correct until the Nyquist frequency, marked as red line, is reached. The Nyquist frequency is half of the sampling rate, and the following **eq.4.1** & **eq.4.2** describe the relation between sampling, signal and Nyquist frequency required to prevent the occurrence of aliasing [60]:

$$f_{Ny} = \frac{1}{2} \cdot f_{samp} \quad (4.1)$$

$$f_{samp} > 2 \cdot f_{orig} \quad (4.2)$$

f_{Ny}	Nyquist Frequency [Hz]
f_{samp}	Sampling Frequency [Hz]
f_{orig}	Highest Frequency occurring in the original Signal [Hz]

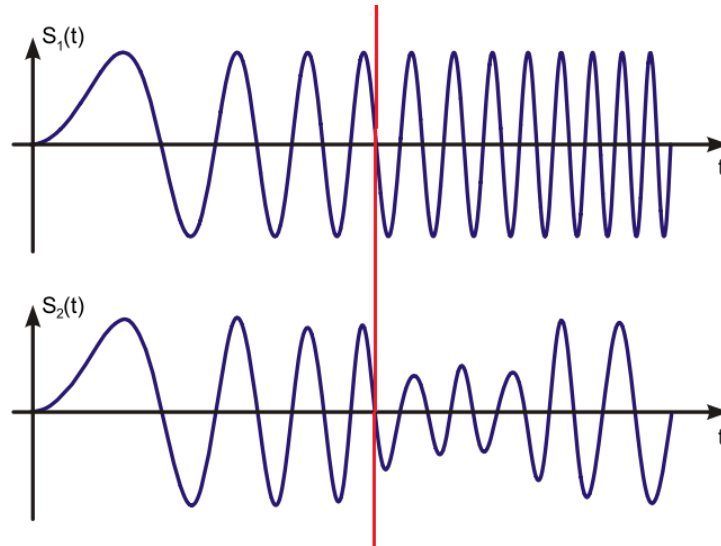


Figure 4.10: The lower graph shows an erroneous reproduction of the signal from the upper graph due to aliasing. The red line represents the time where the Nyquist frequency is greater than half the sampling frequency. Modified after [60]

4.4.1.1 Filtering in the time domain - Gaussian filter

The process of filtering in the time-domain can be viewed as a moving averaging method applied to the noisy original signal. By using this procedure the noisy raw signal is smoothed and the medium to long wavelength components are revealed more clearly by diminishing the influence of short wavelength features. At the start of the procedure the weighting window is positioned at one end of the signal and a weighted average is calculated for the first sample. Then the window is moved by one data point and the average is computed once more, this is done until the window has reached the signal's last point. The length of the weighting window is governed by the so-called cutoff, whereby a higher value increases the window length and therefore leads to a higher degree of smoothing. Another major influencing factor is the characteristic of the window itself and how the samples inside of it are weighted. The Gaussian filter puts the highest weight to the central element and the weights on both sides of the center decrease gradually. The process of applying the moving average is also called convolution of the signal with a window function, thereby creating a third function, which is the smoothed response. The specialized MATLAB function performs a convolution from both sides of the signal, thereby adjusting for the shift of data inherent to the moving average process. The Gaussian window is given as [61, p. 23-38]:

$$w(n) = e^{-\frac{1}{2} \cdot \left(\frac{\alpha \cdot \frac{n}{(N-1)}}{2} \right)^2} = e^{-\left(\frac{n}{\sqrt{2} \cdot \sigma} \right)^2} \quad (4.3)$$

w	Window Function [-]
N	Window Length [-]
σ	Standard Deviation [-]
α	Constant which is inversely proportional to σ [-]

4.4.1.2 Filtering in the frequency domain

Generally the digitized signal of any observable physical quantity exists in the time domain, because it is measured as a function of time. The Fourier theory states that any signal measured in the time domain can be understood as being composed of sinusoids of varying amplitude, frequency and phase. Applying a Fourier transform to the time domain signal decomposes it into the frequencies which make it up. This method allows identifying sinusoidal components of specific wavelengths and their respective amplitude and phase. Additionally this transformation enables identifying the contributions of valuable information and unwanted noise in the measured signal. The noise can be suppressed by selecting a cutoff frequency and an appropriate filter, which cuts all signal content lying outside of the beforehand defined frequency. The basic concept of filtering in the frequency domain can be summed up into three phases. The first step is transforming the signal from the time to the frequency domain, secondly applying a filter to suppress certain frequencies and lastly the transformation of the filtered signal back into the time domain [61, p. 13-21].

4.4.1.3 Median filter

The procedure used in the application of a median filter is in a way similar to any other filtering in the time domain, as a window function is convoluted with the original signal, thereby resulting in a filtered response. The window function of this filter replaces the data point in its center with the median value of all points covered by the window. The median is the middle point of a sequence ordered in ascending value. Similar to the moving average approach an increase in window length results in a smoother response signal, due to the fact that more data points are taken into account. A clear advantage of a median compared to a Gaussian filter is its ability to automatically reject outliers. Another distinguishing feature is the capability to pass step functions. A moving average filter would inevitably smear the edge, due to averaging of data points across the jump in the step function. It is also stated that median filtering is useful for reducing random noise and periodic patterns [62, p. 54-63]. The built-in MATLAB function for median filtering is capable of automatic edge detection and correction of shift induced by the moving window. Figure 4.11 shows the application of a Gaussian and a median filter to a step function. The median filter is able to apply its advantageous features compared to a filter

using any kind of moving average window, thereby rejecting the outlier and displaying the step function without smearing at the edges.

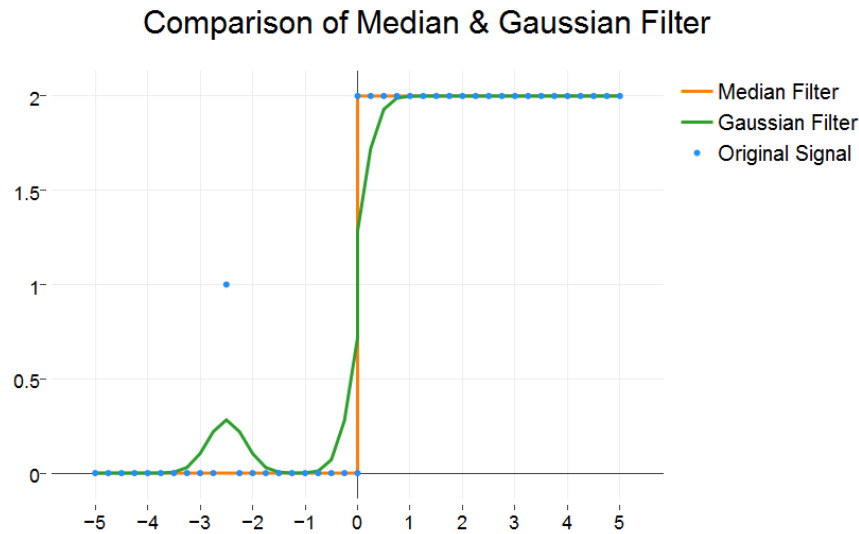


Figure 4.11: Comparison of a Gaussian & Median filter on the behaviour at a step function

4.4.1.4 Applied filtering approach

The first step was to resample the non-uniformly sampled signal of pressure, rate and temperature from each well to a new uniform auxiliary time vector, which was identical for all wells. The purpose of this auxiliary time vector with a rather high sampling frequency of 0.1 HZ was to mitigate aliasing effects and loss of information through the use of oversampling. Followed by down sampling to the chosen sampling interval, which would later be used for further data analysis and application in the ANN. Lastly the application of an appropriate filter to reduce noise still present in the signal and aliasing effects, which could not be fully prevented.

A Gaussian filter was utilized to filter the pressure and temperature signal. Those two signals were characterized by gradual changes over time compared to the fluctuations in the flow rate measurements. The configuration of the Gaussian filter had to be adapted to the desired resampling interval. A modification of the resampling interval led to a change of the number of individual data points in a certain time interval, e.g. a sampling interval of five compared to ten minutes reduced the data points per hour from twelve to six.

Using a resampling interval of 10 minutes, which was equal to a sampling frequency of 1.67 mHz, required all higher frequency content to be cut from the signal. This step was necessary to not introduce errors and artefacts in the down sampled signal, originating from high frequency content which could not be reproduced correctly in the lower frequency signal. Furthermore ignoring this fact would have led to the introduction of false information and noise to the signal.

These disturbances are caused by the so called aliasing effect. To mitigate the occurrence of aliasing the cutoff frequency was set to $2/3$ of Nyquist frequency of the desired resampling time interval, ensuring that higher frequency content was successfully suppressed. Therefore larger resampling intervals led to wider filter windows in the domain time and as a consequence thereof to narrow ones in the frequency domain.

Using the auxiliary step of oversampling resulted in a signal with 0.1 Hz, on which the following filter configurations listed in Table 4.2 were applied.

Table 4.2: Resampling intervals and respective filter configurations

Resampling Interval [Min]	Frequency [mHz]			Window Width [-]
	Resampling	Nyquist	Cutoff	
0.5	33.34	16.67	11.11	9
5	3.34	1.67	1.11	91
10	1.67	0.83	0.56	181
30	0.56	0.28	0.18	541
60	0.28	0.14	0.09	1081

Figure 4.12a shows the time domain representation of the windows specified in Table 4.2 and Figure 4.12b their frequency domain depiction. The dotted vertical line resembles the Nyquist frequency for each resampling interval, this is exactly half of the Nyquist frequency's value tabulated in Table 4.2, because a single sided representation of frequency and magnitude is used. Both figures display the Gaussian filter window for three different sampling frequencies. It can be seen that an increase of the window width results in a decrease in the frequency domain and vice versa. Both figures combined show the advantageous feature of a Gaussian window which remains a Gaussian curve after applying a Fourier transform to it.

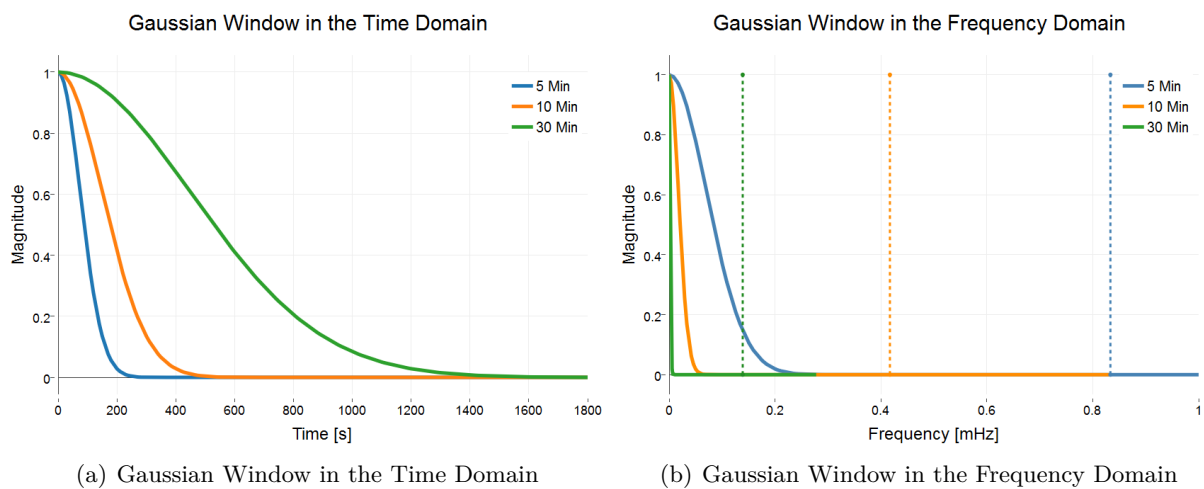


Figure 4.12: Spectral analysis of a Gaussian filter

In contrast to the pressure and temperature, the signal generated by the flow rate measurement contained a substantial amount of random noise at times of operation. Another distinguishing feature was the nearly instantaneous change of flow rate from zero at shut-in to a certain value while operating the well. This bore an undeniable resemblance to a step function. Both the random noise, in form of fluctuating rate measurements, and the appearance of jumps in the signal, made the flow rate signal a promising candidate for median filtering. The application of the median filter to the high frequency auxiliary signal before down sampling to the desired sampling interval, proved to be the best solution for the flow rate data. Analogous to the Gaussian filter a change in the resampling interval demanded an adaptation of window size. In order to ensure nearly equal outcome from filtering to pressure, temperature and rate signal, identical window sizes (see Table 4.2) have been used.

4.4.2 Resampling with local polynomial kernel regression

The second method deployed for resampling was a local polynomial kernel regression. This type of non-parametric regression was found suitable for problems, where no other general model could capture and reproduce the trend in data on the small as well as on the large scale.

The non-parametric regression estimates an unknown regression function from the given observations. A simple approach is to construct a local mean estimator; this approach was first proposed by Nadaraya and Watson in 1964 and therefore referred to as Nadaraya-Watson kernel estimator.

$$m(x) = \frac{\sum_{i=1}^n w(x_i - x_j h) y_i}{\sum_{i=1}^n w(x_i - x_j h)} \quad (4.4)$$

m	Estimated Value [-]
n	Number of Observations [-]
w	Window Function [-]
h	Bandwidth [-]
y	Observation [-]
x	Corresponding x-Variable [-]

The window or also called kernel is generally a smooth positive function which has a peak at zero and decreases monotonically at the sides. The Gaussian density function is an appropriate choice, because it ensures that most weight is given to data points close to the point of interest. Similar to the application of the Gaussian window in filtering, the bandwidth controls the window width and thereby the degree of smoothing. An alternative to the local mean estimator is to fit data with a local linear regression. Using the later type of estimator requires solving the following least squares problem:

$$\min_{\alpha, \beta} \sum_{i=1}^n [y_i - \alpha - \beta(x_i - x)]^2 w(x_i - x_j h) \quad (4.5)$$

α, β	Regression Coefficient [-]
-----------------	----------------------------

As with the local mean estimator, there is also an explicit formula for the local linear regression estimator:

$$m(x) = \frac{1}{n} \sum_{i=1}^n \frac{[s_2 - s_1(x_i - x)]w(x_i - x_jh)y_i}{s_2s_0 - s_1^2} \quad (4.6)$$

$$s_\tau(x, h) = \frac{1}{n} \sum (x_i - x)^\tau w(x_i - x_jh) \quad (4.7)$$

The operation mode of this non-parametric regression is similar to the moving average filtering approach, as again a window function moves along the data points. At each step coefficients of a linear regression and consequently an estimated value are calculated from the data points within the window, until all observations have been taken into account. A key difference compared to the beforehand mentioned filtering is that in kernel regression, the number of data points created by the regression is independent from the count of observations present in the dataset. In contrast to filtering, where only the existing data points are manipulated. Integrating the standard model of linear regression into non-parametric regression made this a very reliable tool. The local linear estimator proves to be superior to the local mean one, at edges and at start and end segments of the data sequence [63, p. 48-52].

The nature of the non-parametric regression allowed resampling to a new uniformly sampled time vector from the edited original signal without any intermediate steps. This approach merely required choosing a desired sampling interval for the output and specifying the kernel bandwidth. As with the alternative approach the default resampling interval was set to ten minutes. Testing several bandwidths for various resampling intervals, found that a kernel width of 25 to 35 samples yielded satisfactory results. Due to the kernel regression's characteristics it was not necessary to adapt the bandwidth when modifying the resampling interval.

The kernel regression acted as an alternative to the approach using linear resampling and filtering. Therefore it was then possible to check the soundness of the resampled signal not only by comparing to the original signal but also to the alternative resampling algorithm's result.

4.5 Quality control

After the resampling had been done using one of the two previously described methods it was necessary to check the integrity of the generated signal. Although precautions against aliasing effects had been taken a quality control (QC) that lays its focus on critical sections was inevitable. This QC comprised of checking if the signal's trend was reflected in the resampled one and how the filtering methods behaved in critical segments of the original signal.

The QC had to be done manually as there was no simple way to check all segments of an original

signal, which was characterized by highly various frequency content, noise coming from the measurement process and abrupt changes in the quality of individual segments. Figure 4.13 displays an overview of the original and resampled signals of pressure and flow rate for Well 1. The following Figures 4.14(a),(b) & (c) marked in green, orange and purple in the overview plot show critical sequences in the measured signal that required special attention. In all plots the originally measured signal is shown in red, while the blue line depicts the resampled signal of 10 Min sampling interval. Linear resampling was used as a standard method for generating the resampled signal.

The first set (Figure 4.14(a)) displays a sequence that is plagued by high frequency noise, which was undesired and had to be removed. The chosen filter configuration mitigated the high frequency components and produced a smoother signal without inducing any aliasing effects. The blue line shows that the filter configuration worked properly and returned the desired result.

Figure 4.14(b) shows the smoothing of the signal at time intervals below the specified sampling interval of 10 Min. While at time intervals greater than the specified resampling frequency the signal is not influenced and returned nearly unaltered. It can be seen that accurate results are achieved at both pressure and rates, although two different filters were applied.

The last plot (Figure 4.14(c)) depicts a situation where mostly time intervals above 10 Min are present in the original signal. It can be observed that the specified filter configurations smooth the signal's high frequency content, while again leaving the low frequency parts nearly unchanged.

As seen in the beforehand mentioned figures the applied filtering approach was proved to be fitting for each of the sections. Furthermore this method provided a smooth and nearly noise free signal without introducing aliasing effects. The resampled and filtered signal could then be used as an appropriate input for the training of neural networks.

In general it can be said that both resampling methods provided good results, although the linear resampling approach proved to be superior. The high degree of adaptability in the linear resampling was its deciding advantage. The specification of an appropriate filter window with a proper cutoff frequency allowed producing nearly noise free signals without introducing unwanted aliasing effects. The biggest advantage of this filter could also be its greatest drawback, as it was a challenging task to find suitable filter configurations. The benefit of the kernel smoother was the ease of application and the generation of an acceptable and usable signal, without evaluating numerous configurations for different resampling intervals.

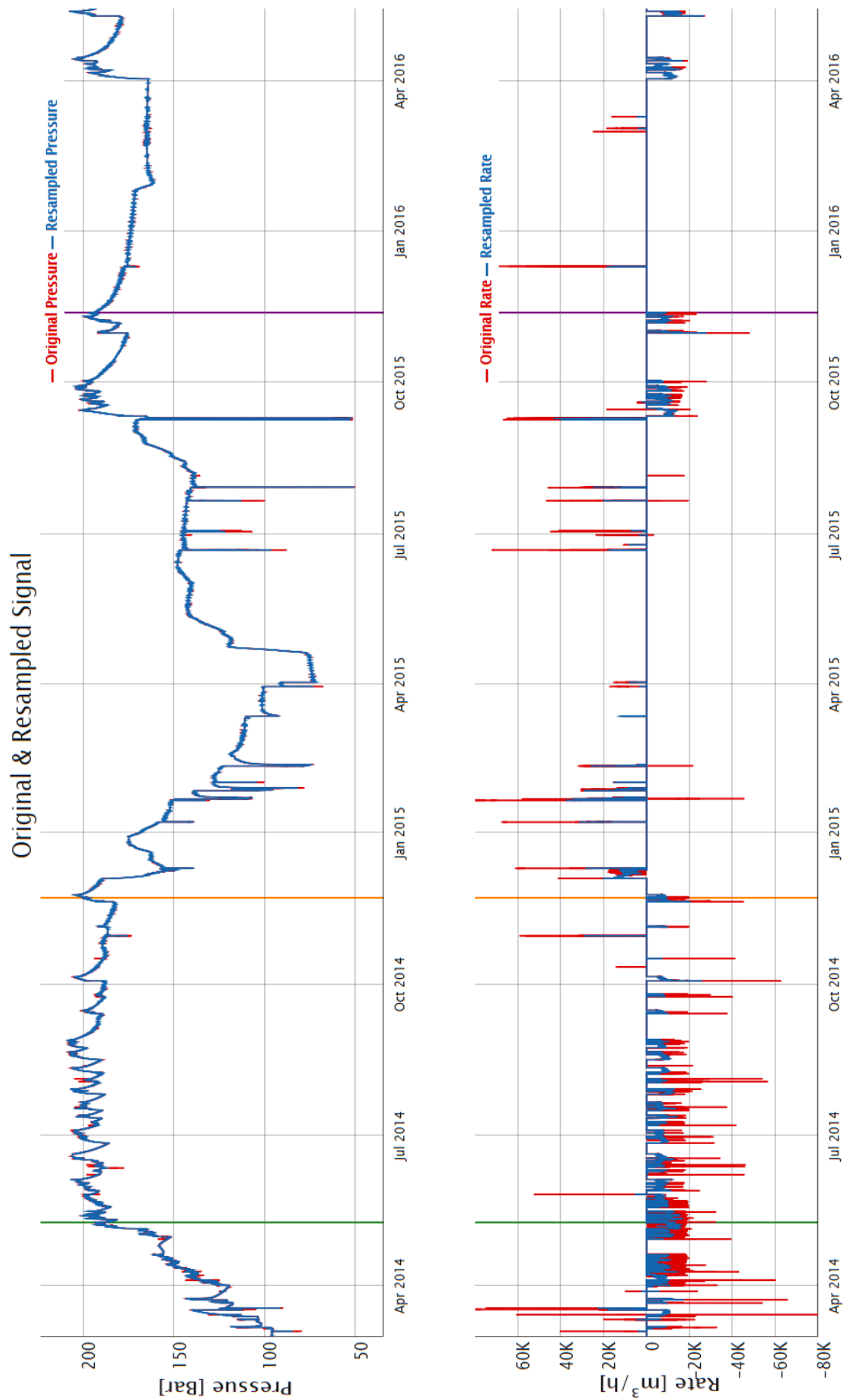


Figure 4.13: Comparison of Original signal in red & Resampled signal in blue coming from Well 1 for the whole 27 months sequence. The green, orange and purple lines indicate critical sections in the Original signal which required special attention.

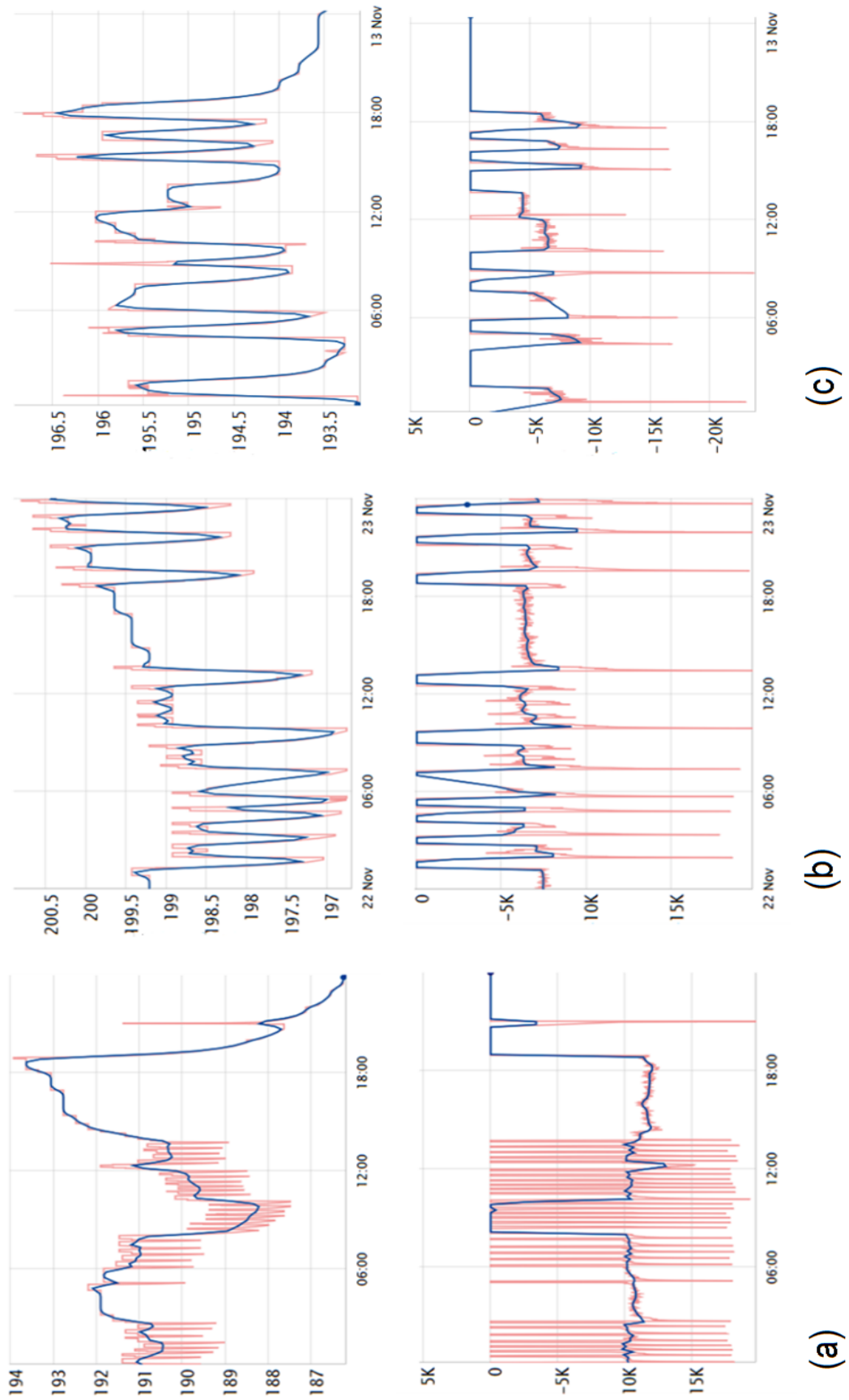


Figure 4.14: Critical sections in the Original & Resampled signal. The plots show a comparison of the Original signal in red & Resampled signal in blue at three critical intervals.

5 Methodology

This chapter discusses the workflow developed for the generation and training of various ANN models. The key steps in model generation are described one after the other. The starting point is a short description of the data preprocessing scheme tailored to neural networks. This is then followed by the steps involved in the generation of ANN models with the software packages RapidMiner and MATLAB. Finally the details of training and application of these prototypes are discussed.

5.1 Agenda

The driving goal of this thesis was to test whether artificial neural networks can be used to simulate the behaviour of an UGS. Therefore it was necessary to develop a suitable methodology to apply these networks to the data gathered from databases within RAG. This data was the foundation on which all networks were built upon and tested against. A good part of the workflow was dedicated to the investigation if the provided data was sufficient to build sound models. The selection of appropriate network types and architectures was another essential task in the whole workflow. Finally the results should allow judgement whether there is an ANN architecture that can adequately capture the underlying dynamics present in underground gas storage operation. This judgement process included a history match of wellhead pressures for more than two years' time of operation and about one and a half months of data that was used to test the network's capabilities to predict future pressures. The results will determine if neural networks can compete with classical reservoir simulation and be considered as a viable alternative. The following sections will discuss the steps that were taken from gathering the data, building models and their application and testing.

5.2 Data preprocessing

This chapter discusses the practical aspects of data preprocessing that were necessary to shape data in a suitable form for the application of an ANN. The description of available data, statistical aspects and the resampling procedure applied were already outlined in the previous chapter. After the gathered data had been remedied of deficiencies and resampled to a regular sampling frequency it was required to compile it into a usable form. Arranging data streams from all wells into a single table was found to be the most practical approach. The resampling script written in MATLAB automatically compiled the resampled signals from all wells into a single table that could be used in various other programs. This method guaranteed that all programs and models were based on the same data basis.

5.2.1 Selection of input & output for ANN

The available dynamic data consisted of pressure, temperature and flowrate measurements at the wellhead. Additionally static data about the wells locations, their design and completion were available. It was found that the static data could not be integrated into the neural network models in a meaningful way. There were several reasons why the static data was more or less neglected. The first and most crucial reason was that the static data as the name implies was purely static. The ANN model on the other side was solely driven by dynamic time series information coming from the wellhead measurements. From a modeling point of view the ANN could be seen as a black box, which should learn the dynamics veiled in the measured signals. The static properties of the wells were not changed throughout the gathered time period. This meant that including an input, which remained constant during the whole history matching period would not have had any beneficial influence on the ANN's modelling behavior. Secondly it was not intended to create a static reservoir model that would be coupled with neural networks. In theory the ANN could replace the numerical computations in a classic reservoir simulator. Zangl et al. showed a conceptual way how a neural network could replace the numerical computation model of a total reservoir model. They state that the computation speed is the decisive advantage of neural networks over the numerical models in the classic reservoir simulation [8].

The beforehand mentioned circumstance reduced the data actually used to the pressure and flow rate measured at the wellhead. The pressure data was coming from the operating wells 1 to 6 and the sensor readings of Obs-Well 1. The UGS' filling level at each time step was calculated by cumulating the flow rate measurements. This cumulative volume was then divided by the working gas volume of the reservoir. Figure 5.1 displays the configuration of network input and output signals used in this work. Although also wellhead temperature measurements were available, but they were not utilized in the designed models. The main reasons behind this decision were firstly that the temperature was influenced by ambient surface conditions and it was not possible to find out at which time intervals the wellhead heating systems was active. Secondly the models were designed to be trained on defined input variables and the desired outputs. In order to apply them for any prediction tasks it was required to feed future inputs to the ANN to generate future desired outputs. Using the temperature as an input would have required an additional model to predict the future temperature at the wellhead. It was concluded that the possible error in forecasting the temperature would outweigh the benefits of having it as an input.

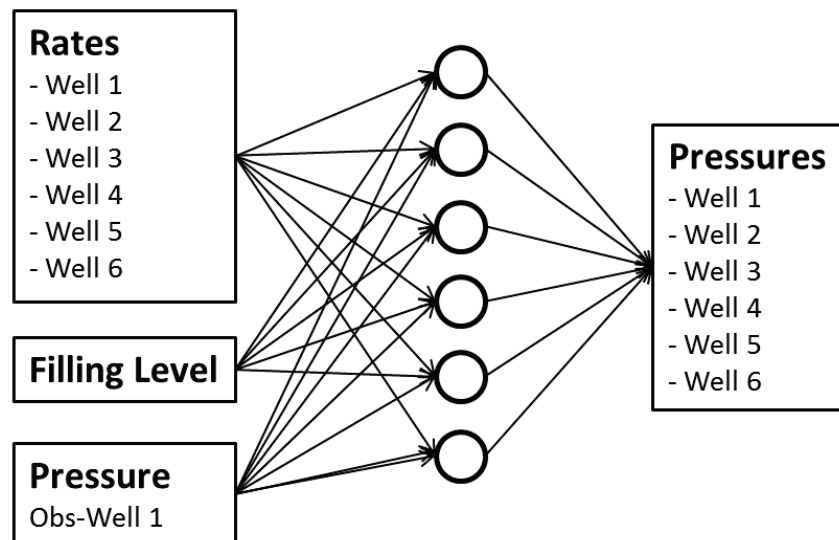


Figure 5.1: Input & Output configuration of ANN models.

5.2.1.1 Obs-Well 1 as input

Using the pressure measured at Obs-Well 1 as an input to predict the pressures at Wells 1 to 6, as depicted in Figure 5.1, required knowing it in advance. This pressure was utilized as an input to provide a sort of pressure baseline besides the gas storage filling level for the models forecasting the dynamically influenced wellhead pressures. Due to the fact that the pressure at Obs-Well 1 is only marginally influenced by immediate changes of flow rate at the operational wells it could act as a pseudo reservoir pressure.

To forecast the pressure at Obs-Well 1 a fully recurrent ANN with ten hidden neurons was used. The utilized networks, which are discussed in more detail in section 5.3.3, were identical to network generation 6 shown in Table 5.3. The only difference were the altered inputs and output. Figure 5.2 shows the inputs used to model and predict the wellhead pressure at Obs-Well 1. It was obvious that this pressure was governed by the filling level and the flow rates at the six operating wells. Therefore the filling level, rates of all individual wells, the sum of all rates at each time step and the cumulative injection/production were used as inputs.

The same data separation method as used in the models proposed to predict the wellhead pressures for the operating wells was applied. Additionally the identical hourly resampled data set was used to forecast the pressure at the observation well. Therefore the whole data sequence was split into the same three time intervals for network training and validation as well as the independent test sequence as were used when forecasting the pressures at the operating wells. The data separation approach and selection criteria regarding the chosen resampling interval are explained in the following two sections.

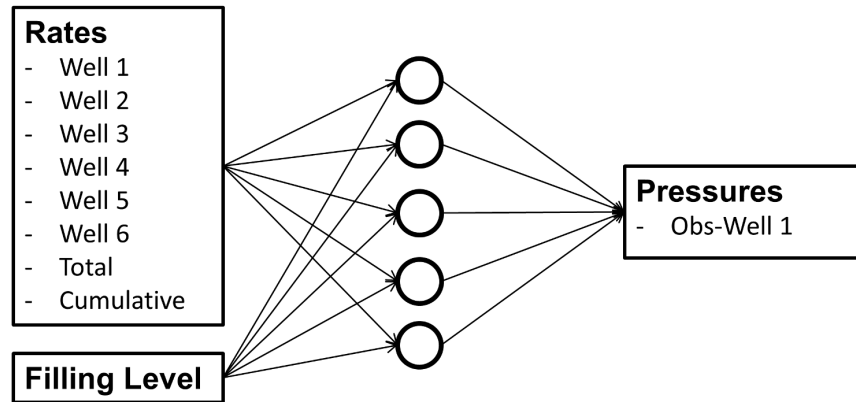


Figure 5.2: Input & Output configuration of the ANN model used to forecast the wellhead pressure at Obs-Well 1.

Figure 5.3 displays a time series representation of the measured pressure and the output of the fully recurrent network for Obs-Well 1 along with the total flow rate, which is the summation of the rates from the six operating wells. It is clearly visible that the proposed model is able to forecast the required pressure with a sufficient accuracy of less than 2 Bar deviation from the desired value.

Figures 5.4(a), (c) and (e) show scatter plots of the fully recurrent network's output against the desired one and a superimposed coefficient of determination R^2 for the training, validation and test data sets. Figures 5.4(b), (d) and (f) on the right hand side depict an error histogram of each data set.

Figures 5.4(a), (c) and (e) clearly show that the fully recurrent ANN achieved a very good generalization. The coefficient of determination R^2 is greater than 0.999 for both the training and validation data sets, which indicates a nearly perfect match. The R^2 of 0.97 in the independent test sequence still resembles an adequate network model. The three error histograms in Figures 5.4(b), (d) and (f) underpin the conclusion that the proposed model is suitable to predict the pressure for Obs-Well 1. The majority of errors lie in the range of -1 to $+1$ in the training and validation sets. The slightly worse performance in the test interval is also resembled in the somewhat skewed error histogram with a increased range of errors from -2 to $+2$.

A possible explanation for worse performance and the skewed distribution of errors in the test interval was the nearly a month long period of inactivity where there were no changes in the rates and filling level. Furthermore it can be seen that in the test interval the deviations between the network output and the desired target are generally less in the active parts. This observation allows the conclusion that the proposed model can forecast the pressure at Obs-Well 1 with an accuracy of roughly 1 Bar in the active periods.

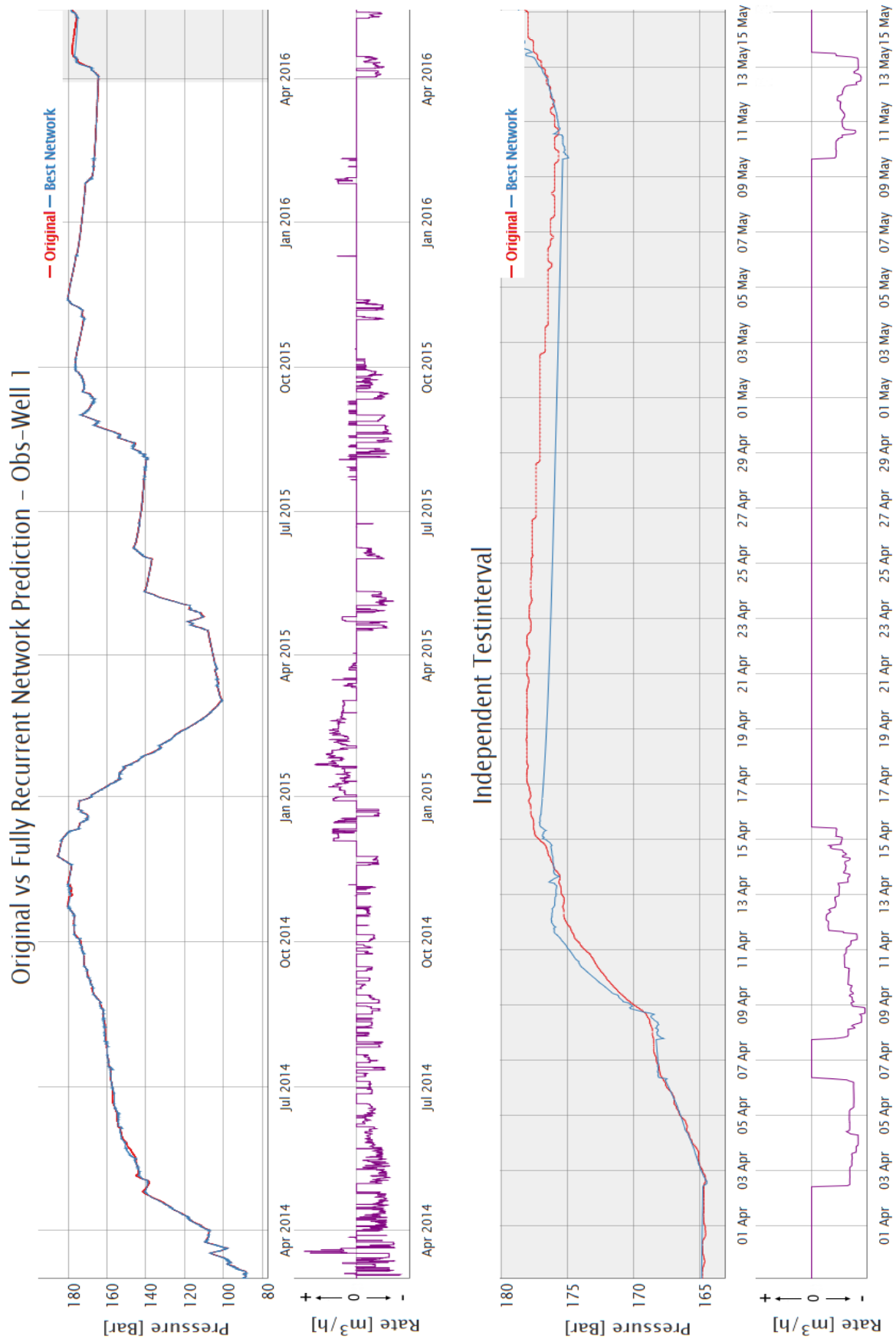
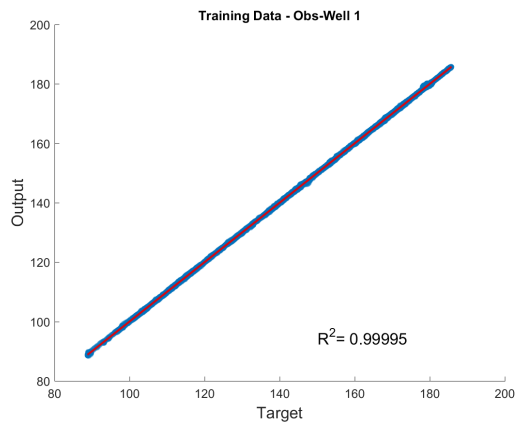
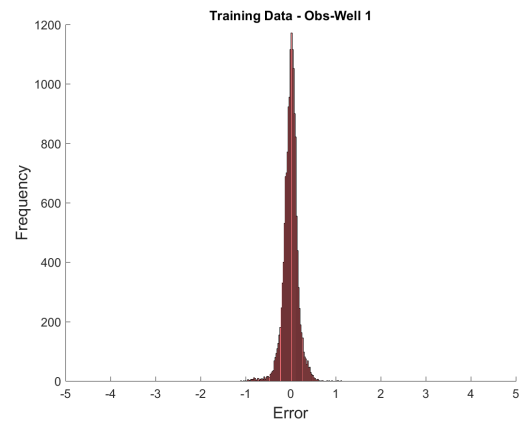


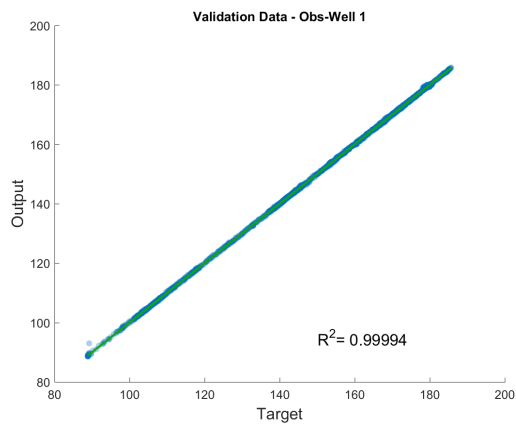
Figure 5.3: Comparison of Original & fully recurrent network (using 10 hidden neurons) Output for Obs-Well 1 in a time series plot. The upper graph depicts the whole 27 months period. The independent test set is shaded in grey.



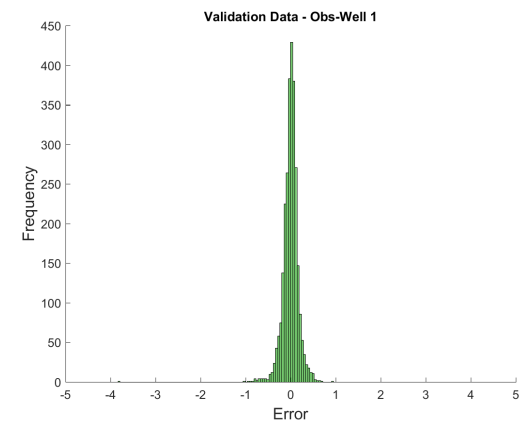
(a) Training Set - Correlation



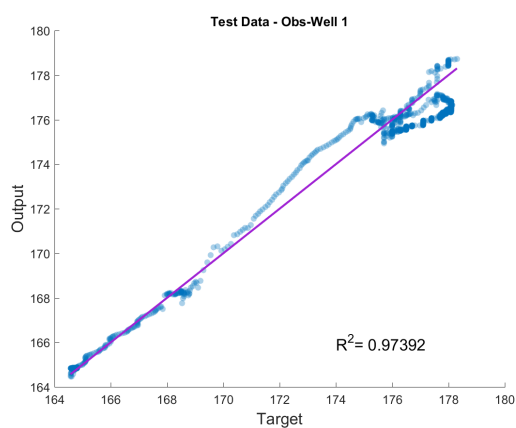
(b) Training Set - Histogram of Errors



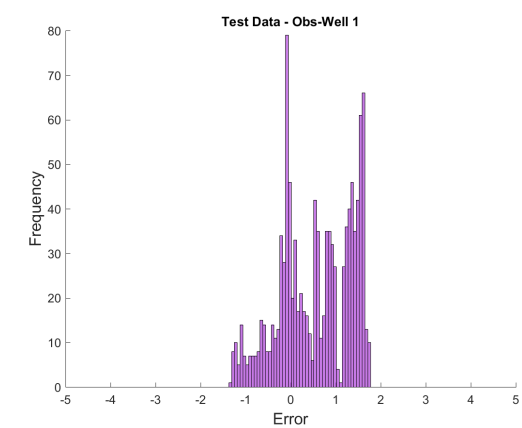
(c) Validation Set - Correlation



(d) Validation Set - Histogram of Errors



(e) Test Set - Correlation



(f) Test Set - Histogram of Errors

Figure 5.4: Output - Target correlation plot & error histogram for Obs-Well 1 using a fully recurrent ANN. The training set is shown on top, the validation set in the middle and the test interval on bottom.

5.2.2 Data separation

A similar data separation scheme as described in the chapter concerning the MLP network was used to split the data into three parts. For feedforward networks the standard is to allocate samples randomly into the three subsets. The time series data used in modelling the gas storage operation required a different approach of data separation. This data was characterized by short and long term dependencies in each individual signal and between input and output variables. Randomly assigning the data into the three subsets would have destroyed the underlying dependencies. The majority of the roughly 27 months of measured data was dedicated to training and validation of the models. The remaining part acted as an independent sequence used to test the model's response to unseen data. The training and validation dataset with a length of 25 months was split again into separate training and validation samples. The first one consisted of 80% of the data and the second of the remaining 20%. The MATLAB "Neural Network Toolbox" offered several data separation methods. The "interleaved" approach divided the data as in dealing a deck of cards. The whole 25 months of data were in the first step divided into a certain number of subsets, and these were thereafter split into two blocks, namely training and validation set. The percentages of training and validation samples were the beforehand specified 80% and 20%. Figure 5.5 depicts a schematic overview of the applied data separation approach. The segments shaded in red are the intervals dedicated to training of the ANNs, while the subsequent green segments are used as validation sets. The shorter purple segment at the end of the whole sequence is the independent test set, which is identical for all networks.

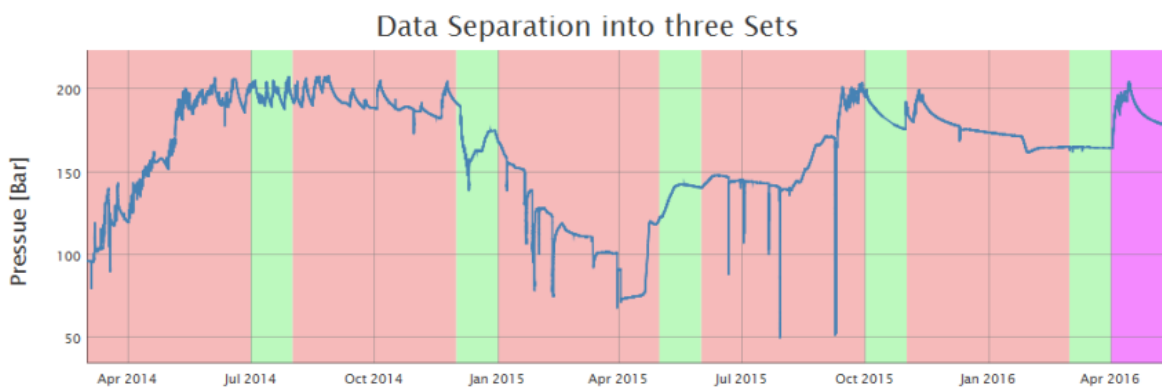


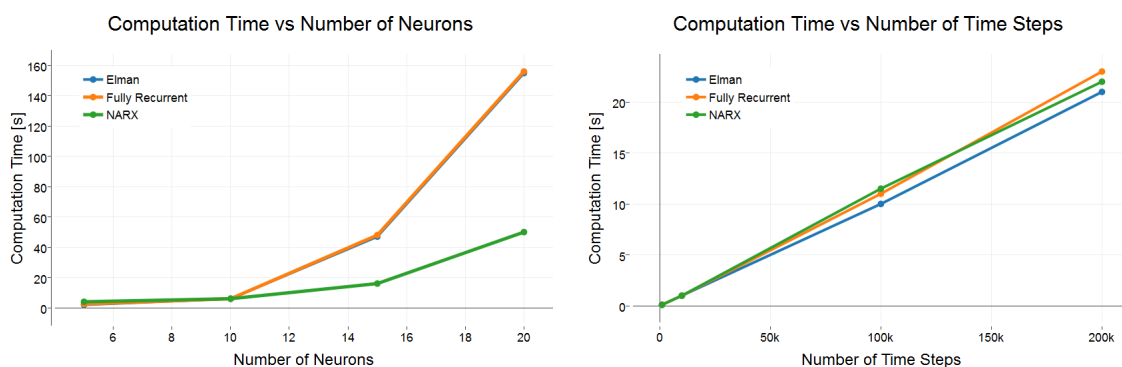
Figure 5.5: Data separation for ANN models. The red segments are the training intervals and green parts are used for validation. The independent test sequence is shown in purple.

5.2.3 Sampling frequency & Computation time

Apart from data separation it was also necessary to select an appropriate sampling frequency for the dataset to be utilized in the modelling scheme. This frequency had to be a compromise between ANN computation times and the capability of preserving the dependences in the sequential data. Jaeger stated that the computational complexity of the BPTT algorithm is $O(TN^2)$, where N is the number of hidden neurons and T the time steps. Additionally the memory span coverable by BPTT training was said to be 10 to 20 time steps [55, p. 14].

In order to have an idea about the actual computation times to be expected, several trials were run. Generic models of the Elman, fully recurrent and NARX network were used to investigate the influence of the number of hidden neurons and the number of samples on the computation time. Figure 5.6(a) shows the computation time per iteration for the different models with a fixed number of 100,000 time steps. This clearly indicates a highly non-linear trend between number of neurons and computational complexity. The second Figure 5.6(b) depicts the computation time per iteration as a function of the number of time steps. The quantity of neurons in the hidden layer was fixed at 10 for all network architectures. The straight line behaviour reveals the linear trend between the computation time and the number of time steps.

The resulting compromise suggested an hourly sampling frequency for the data fed to the ANN. The hourly data still maintained the signal's trend and allowed to train several networks in a convenient time frame. The chosen sampling frequency reduced the whole dataset to roughly 20,000 time steps.



(a) Computation time of ANN architectures against amount of hidden neurons (b) Computation time of ANN architectures against number of time steps

Figure 5.6: Computation time per iteration in seconds of various network types

5.3 Model Setup

This chapter discusses the steps taken in setting up neural network models to fit the gathered time series data. Several different architectures were tested to find out if any of them proved to be suitable. Additionally the performance of pure feed forward ANN was tried against several recurrent architectures.

5.3.1 Overview of neural network programs

The two software packages RapidMiner and MATLAB were used to setup and train the various neural network architectures. The following Table 5.1 gives an overview which software was used for a specific type of ANN. The fully recurrent, Elman and NARX ANN were implemented using the "Neural Network Toolbox" which is integrated into MATLAB. Each individual architecture and how it was set up is discussed in more detail in the following sections.

Table 5.1: Overview of neural network software packages

Network Architecture	Software Package	Software Module
Multilayer Perceptron	RapidMiner	Neural Net
Fully Recurrent ANN	MATLAB	layreinet (modified)
Elman ANN	MATLAB	elmannet
NARX ANN	MATLAB	narxnet
Echo State Network	MATLAB	ESN Toolbox [64]

5.3.2 MLP with RapidMiner

The representative of the feed forward MLP network was implemented with the open source software "RapidMiner". This software was aimed at general data mining purposes whereby neural networks were just a part of the tools available and not the centerpiece. The package offered predefined tools dedicated to data loading and processing. In order to retain comparability with neural network models done using other programs, the data was preprocessed in MATLAB using the previously outlined scheme. The MLP model in RapidMiner could be configured with respect to the number of neurons in the hidden layer and the input normalization. The standard BP algorithm was the only option available for training of the ANN. The input was normalized using the min-max approach which scaled data into the range of -1 to $+1$. The networks were trained using early stopping and a maximum of 2000 iterations.

Several generations of networks were trained starting from five up to 20 neurons in the hidden layer. Each generation included five individual networks, initialized with different random seed values. Using several networks in one generation is typically done to mitigate the influence of poor starting conditions. Analysing the results of several networks allowed formulating a more general statement about a network generation's generalization capabilities. When increasing the network's complexity the random seed values remained unchanged, thereby ensuring equivalent

starting conditions. The MLP implementation of RapidMiner had one noteworthy drawback which allowed it to have only one numerical output. This meant that for each well an individual model had to be set up. Additionally it was not possible to consider the interaction of pressures and flow rates from different wells. An overview of trained networks is given in Table 5.2 below.

Table 5.2: Overview of MLP network generations

Network Generation	# Networks	# Hidden Neurons	Random Seed	Training Algorithm	Training Conditions
1	5	5	1 - 10	BP	ES - 2000
2	5	10	1 - 10	BP	ES - 2000
3	5	15	1 - 10	BP	ES - 2000
4	5	20	1 - 10	BP	ES - 2000

5.3.3 Fully Recurrent ANN with MATLAB

The models utilizing recurrent ANNs were done in MATLAB, which offers an extensive toolbox with various network architectures. Analogous to the setup of the MLP networks, the data for the recurrent ones was prepared using the scheme outlined in the previous sections of this chapter.

The MATLAB toolbox offered various options in configuring the individual ANN types. The possibilities ranged from adjusting the number of layers and neurons in the network, adapting the transfer function for each neuron to choosing from a wide variety of training algorithms. Mostly the MATLAB default setup for the neuron-neuron connections and transfer function was adopted. The hyperbolic tangent sigmoid transfer function was set as the standard for all neurons in the hidden layer while the output layer utilized linear ones. The advantage of a linear transfer function in the output layer was the, although limited, capability of extrapolation beyond the range covered by the training samples. Input was normalized by mapping them all to zero mean and a standard deviation of one. The MATLAB toolbox used the “Levenberg-Marquardt” optimization method in conjunction with BPTT as the standard training algorithm for recurrent ANN dealing with time series modelling.

The Levenberg-Marquardt (LM) algorithm, also known as dampened least-squares method, is a second order optimization technique used to solve non-linear least squares problems. It was first published in 1944 by Levenberg and rediscovered by Marquardt in 1963. The LM method optimizes the parameters of a model so that the sum of the deviations **eq 5.1** becomes minimal. It can be seen as a middle way between gradient descent and the Gauss-Newton method. The LM algorithm **eq 5.2** extends the standard Gauss-Newton method by a damping coefficient λ , which is adjusted at every iteration step. For large values of λ the LM algorithm behaves like a gradient descent method and for small λ more like the Gauss-Newton approach. The introduction of the dampening coefficient λ makes the LM algorithm very robust, which means

that it will find a local minimum even if it starts very far off [65, p. 164-168], [66, p. 431-441].

$$E(\beta) = \sum_{i=1}^n [y_i - f(x_i, \beta)]^2 \quad (5.1)$$

$$\beta_{k+1} = \beta_k - [J^T J + \lambda I]^{-1} J^T \cdot [y - f(\beta)] \quad (5.2)$$

x_i, y_i	Independent and dependent Variable [-]
β	Model Parameter(s) [-]
J	Jacobian Matrix [-]
I	Identity Matrix [-]
λ	Dampening Coefficient [-]

The computation of the Jacobian matrix is the crucial part of the LM algorithm. To apply the Levenberg-Marquardt technique to neural networks an adaption of the BP algorithm had to be done. The main idea behind this modification is to calculate the elements of the Jacobian matrix as the derivatives of the errors, instead of the derivatives of the sum squared errors, like it is done in the standard BP algorithm. The LM method appears to be the fastest ANN training algorithm for a moderate number of network parameters. It is stated that this method becomes impractical for more than a few thousand parameters [67, p. 989-993].

Analogous to the training of MLP networks, several generations of fully recurrent ANN were trained and tested. These network generations spanned from five up to 20 neurons in the hidden layer. The maximum number of neurons was limited to a maximum of 20 neurons to ensure realizable training times. Identical random seed values were used when increasing the network complexity to ensure equal starting conditions. All ANNs were trained using the early stopping approach and a maximum of 350 iterations per individual network. Table 5.3 below offers an overview of the trained networks, their complexity and the respective random seed value. This way of setting up models ensured reproducibility and allowed comparing results from different network generations.

Several MATLAB scripts were written which automated the training process. The core of this package was a script that allowed specifying all networks options by using a customized input dialogue. These scripts made it possible to increase the network complexity automatically while leaving all other network parameters unchanged. In addition a script dedicated to testing the network's response to the independent test set and storing the results was written. Combining the described scripts enabled a nearly full automation of the training process thereby ensuring utmost reproducibility. Finally another script compiled datasets and results from all network generations into a file that could be opened and processed with various other programs. This was mainly done to plot results using software packages that offered a broader variety of graphing options. All of the beforehand described scripts were also adapted to be applicable with Elman, NARX and the echo state networks. Using MATLAB as the software basis allowed adapting

Table 5.3: Overview of fully recurrent network generations

Network Generation	# Networks	# Hidden Neurons	Random Seed	Training Algorithm	Training Conditions
1	10	5	1 - 10	LM	ES - 350
2	10	6	1 - 10	LM	ES - 350
3	10	7	1 - 10	LM	ES - 350
4	10	8	1 - 10	LM	ES - 350
5	10	9	1 - 10	LM	ES - 350
6	10	10	1 - 10	LM	ES - 350
7	10	11	1 - 10	LM	ES - 350
8	10	12	1 - 10	LM	ES - 350
9	10	13	1 - 10	LM	ES - 350
10	10	15	1 - 10	LM	ES - 350
11	5	20	1 - 10	LM	ES - 350

these scripts without requiring major changes for the Elman and NARX network. Reaching compatibility with the ESN extension required more changes, because the ESN was not integrated into the MATLAB Neural Network Toolbox. In Appendix C the three major scripts for training and testing of fully recurrent, Elman and NARX networks are shown one after the other.

5.3.3.1 Smaller resampling interval

In addition to the fully recurrent ANNs listed in Table 5.3 a network generation utilizing a resampling interval of 10 Min was trained. This was motivated by the investigation how the networks would cope with a change in the dependences in between time intervals, brought upon by altering the resampling interval. Network generation 9 with 13 hidden neurons was selected as the test candidate, because it showed the best validation error on the hourly data. To enable reasonable computation times a five month long training interval, which equalled approximately 20,000 time steps was selected. The independent test set was also reduced to one and a half weeks.

5.3.4 Elman ANN with MATLAB

The MATLAB Neural Network Toolbox also offered an out of the box usable implementation of an Elman ANN. Due to the fact that the Elman network is a simplification of a fully recurrent architecture, it was feasible to adopt nearly all specifications from the former network type. Apart from the neuron-neuron connections all other options did also apply for the Elman ANN. The similarity of implementations in the MATLAB Neural Network toolbox allowed an easy adaption of all beforehand mentioned scripts.

Analogous to the fully recurrent type several network generations, using equivalent initial conditions and normalization, were set up and trained. The span of hidden neurons again ranged from five up to 20 and identical random seed values as in the fully recurrent architecture were used. Training was again done utilizing the Levenberg-Marquardt algorithm combined with

early stopping and a maximum of 350 iterations. Table 5.4 gives an overview of the individual generations of Elman networks and their properties.

Table 5.4: Overview of Elman network generations

Network Generation	# Networks	# Hidden Neurons	Random Seed	Training Algorithm	Training Conditions
1	10	5	1 - 10	LM	ES - 350
2	10	10	1 - 10	LM	ES - 350
3	10	15	1 - 10	LM	ES - 350
4	5	20	1 - 10	LM	ES - 350

5.3.5 NARX ANN with MATLAB

The NARX models were once more realized using the in MATLAB implemented NARX network architecture. The main difference between the NARX model and the two other recurrent types was the kind of feedback connection. In the NARX model the feedback loop was realized via two tapped delay lines, one from the input and the other one delaying the network output. A similar set of scripts which introduced an automatic training procedure to NARX models was done, by adding the option to specify delay lines.

Table 5.5: Overview of NARX network generations

Network Generation	# Networks	# Hidden Neurons	Random Seed	Training Algorithm	Training Conditions
1	10	5	1 - 10	LM	ES - 350
2	10	6	1 - 10	LM	ES - 350
3	10	7	1 - 10	LM	ES - 350
4	10	8	1 - 10	LM	ES - 350
5	10	9	1 - 10	LM	ES - 350
6	10	10	1 - 10	LM	ES - 350
7	10	11	1 - 10	LM	ES - 350
8	10	12	1 - 10	LM	ES - 350
9	10	13	1 - 10	LM	ES - 350
10	10	14	1 - 10	LM	ES - 350
11	10	15	1 - 10	LM	ES - 350
12	10	20	1 - 10	LM	ES - 350

Several generations of NARX models were trained in a very similar fashion compared to all other recurrent ANN types. The amount of neurons in the hidden layer ranged between five and 20. Although the feedback was realized in a different way in the NARX network, the BPTT method in conjunction with the LM algorithm has proved to be a viable choice for training. Again the early stopping scheme with a maximum of 350 iterations was applied as the training process stopping criterion. An investigation concerning the appropriate number of input and output delays for the hourly resampled data was done. It was concluded that a delay of four time steps

in the input and output loop proved to be suitable choice. Table 5.5 depicts all generations of NARX networks and their parameters.

5.3.6 ESN with MATLAB

The ESN networks were set up using a MATLAB toolbox done by Jaeger [64] which was free for non-commercial purposes. This toolbox comprised of several individual scripts which enabled the user to set up different types of ESN and adjust any parameter desired.

A comprehensive guideline which parameters could be adjusted in an ESN and how any modification would influence the network's functionality was done by Lukoševičius [68, p. 659-686]. It is stated that the three most significant global parameters of a standard ESN are:

- Reservoir Size
- Spectral Radius
- Input Scaling

Generally, for a challenging task a reservoir as big as computationally possible can be used. Its size is bounded by the number of independent real values from the input it has to recall to generate the desired output and on the other side by the amount of time steps in the whole input sequence. If very large reservoirs are used whose sizes are in the order of the amount of total input time steps measures against overfitting have to be taken. Typically, other network parameters are defined first and then followed by a stepwise increase of reservoir size. Growing the reservoir is useful as long as the error coming from the training and testing data set decreases.

The spectral radius of the reservoir weight matrix, equivalent to the maximal absolute eigenvalue, is one of the most global parameters of an ESN. Generally, it can be said that the spectral radius governs the ESN's treatment of long term dependences in the data sequence. The spectral radius acts as sort of a memory in the reservoir; the smaller it is the faster an input's influence dies out. Typically, larger spectral should be utilized when more historic input values are required to model the desired output. Commonly the spectral radius ranges between 0 – 1, but also values greater than 1 can be used if required [68, p. 663-667].

A common way of computing ESN's output weights is by using the "Pseudoinverse" approach. This method solves the over-determined linear system of equations for output weights (**eq. 5.3**) by applying a pseudoinverse matrix of X .

$$Y_t = W^{out} X \quad (5.3)$$

$$W^{out} = Y_t X^+ \quad (5.4)$$

W^{out}	Output Weights [-]
Y_t	Targets [-]
X	States produced by the Reservoir [-]
X^+	Pseudoinverse of X [-]

Typically, the pseudoinverse method is numerically stable, but its high memory demands are a limiting factor to reservoir size and amount of time steps. The pseudoinverse approach enables network training with high precision but it does not have any regularization capability. The lack thereof is a drawback that demands that the reservoir size is much smaller than the number of training time steps in order to mitigate overfitting [68, p. 671]. Utilizing the pseudoinverse method resulted in a change of data separation scheme used in training the ESN compared to all other network types. The former training and validation sets were combined into the new training set, while the independent test sequence was left untouched.

Several ESN generations were trained to investigate optimal values for the above described parameters. For each network generation several trial run iterations were done, whereby the input weight scaling and the spectral radius were increased incrementally. The same procedure was repeated when increasing the reservoir size. The connectivity in the reservoir itself was fixed at 10% for every network generation. Table 5.6 gives an overview of the network generations trained and the utilized ranges of the two beforehand mentioned parameters.

Table 5.6: Overview of ESN generations

Network Generation	# Networks	Reservoir Size	Random Seed	Input Scaling	Spectral Radius
1	100	50	1 - 10	0.1 - 1	0.1 - 1.5
2	100	100	1 - 10	0.1 - 1	0.1 - 1.5
3	100	250	1 - 10	0.1 - 1	0.1 - 1.5
4	100	500	1 - 10	0.1 - 1	0.1 - 1.5
5	100	1000	1 - 10	0.1 - 1	0.1 - 1.5
6	100	2000	1 - 10	0.1 - 1	0.1 - 1.5
7	50	5000	1 - 10	0.1 - 1	0.1 - 1.5

In order to train and test numerous networks in an efficient way several scripts were written as an extension to the toolbox provided by Jaeger [64] which automated certain training steps. Analogous to the training of other recurrent architectures these scripts added an input dialogue to define training parameters. All network parameters were increased incrementally while ensuring that identical random seed values were used. Furthermore, a script dedicated to testing the network on the same independent data set, as used with all other recurrent ANNs, was written. Finally, another script compiled data sets and results from all networks in a generation into a file that could be stored.

6 Results

This chapter compiles the results of all network types and generations. The performance of a network is determined by its error in the history matching period and the independent test set. In a first step the results of each network type are discussed individually and then the best performing networks of all architectures are compared against each other. Due to the sheer number of trained networks it is not possible to discuss the result of every network explicitly. Therefore it was decided to show them in a progressive way. An overview graph depicting the performance of every network generation was plotted and the results from the best performing one are discussed in more detail. The output of the best individual network of each ANN type was plotted in a time series representation. In addition to the best network's output an averaged output of all networks in the specific ANN generation was depicted along of it.

The root mean square error (RMSE), **eq 6.1**, was the best choice to evaluate a network's performance and to compare them with each other. The RMSE was a good measure of accuracy when comparing desired values with predicted ones. A drawback of the RMSE was its scale dependency, meaning that only forecasts of a certain variable could be compared [69]. This disadvantage did not influence the comparability of errors as only pressures of a similar range were tested. The fact that the standard error function used in the training of ANN was a mean squared one, which made it easy to compute the RMSE in one further step. Another advantage of the RMSE was that it has the same unit as the forecasted variable.

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{n}} \quad (6.1)$$

y	Desired Value [-]
\hat{y}	Forecasted Value [-]
n	Number of Values [-]

6.1 MLP ANN

Several generations of feed forward MLP networks (see Table 5.2) were run and tested. The purpose behind using static MLP was to investigate how well this type of ANN can handle dynamic time series data. A major advantage of pure feed forward networks over recurrent ones was the drastically reduced complexity and therefore a reduction of computation time per iteration by at least an order of magnitude. Figure 6.1 depicts the validation error in blue and the testing error in orange against the number of neurons in the hidden nodes.

It is clearly visible from Figure 6.1 that the validation error decreases with increasing network complexity. Another noteworthy fact is that due to the reduced complexity of MLP networks

compared to recurrent ANNs, the spread in the validation errors of an individual network generation is remarkably low. The testing error does not resemble the definite trend of better performance at higher numbers of hidden neurons. It rather seems that there is no benefit from having more than ten neurons in the hidden layer. Additionally there is a wider spread of achievable testing performances throughout an individual network generation.

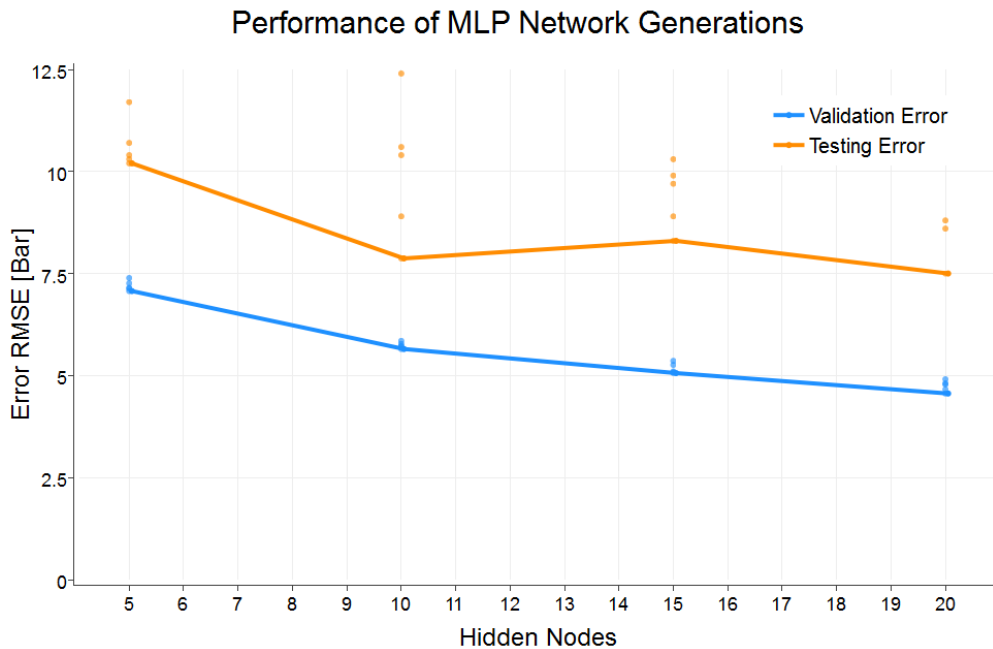


Figure 6.1: Development of validation & testing error with increasing network complexity. Single points display the errors coming from all networks in a MLP generation.

Figure 6.2 compares the desired output and network output for Well 1 in a time series plot. Although the best network's validation error of about 6 did not look that bad on a first glance, figure 6.2 reveals the MLP's weaknesses. It is clearly visible that the static network manages to reproduce the overall trend in the training dataset but it is not able to capture more specific dynamic sequences, like pressure buildup and drawdown. Even though the testing error of the best network is just slightly worse at 7.5, the time series plot clearly shows that the MLP did not generalize well. The network forecast of the test set is far off the actual trend and does not resemble the dynamic pressure behaviour.

This led to the conclusion that static feed forward networks are not suitable to model the dynamic wellhead pressure system. The lack of feedback connections did restrict the MLP's generalization capability of a system where future outputs obviously did not only depend on current inputs but also on historic inputs and outputs.

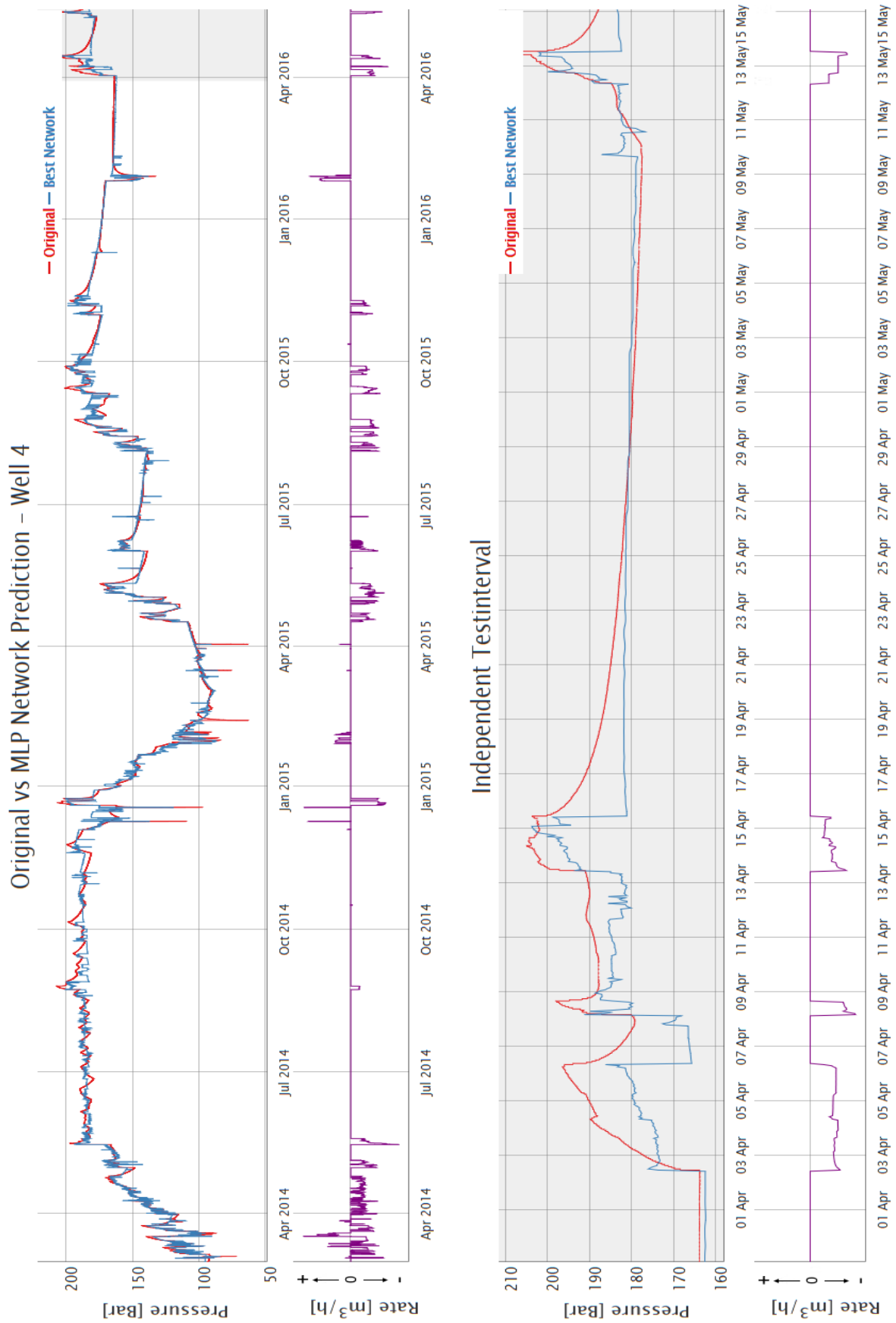
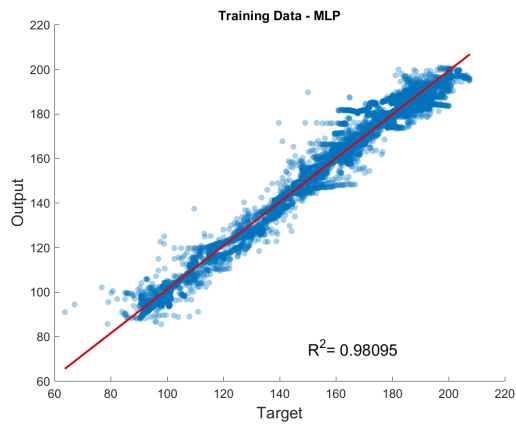


Figure 6.2: Comparison of Original & Output of a MLP network using 20 hidden neurons of Well 4 in a time series plot. The upper graph depicts the whole 27 months period. The independent test set is shaded in grey.

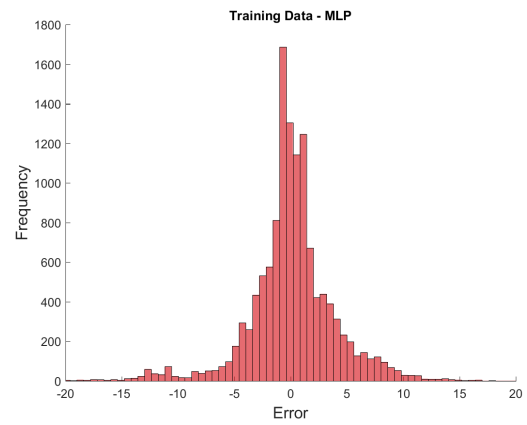
Figures 6.3(a), (c) and (e) show scatter plots of the network output against the desired one and a superimposed coefficient of determination R^2 for the training, validation and test data sets. Figures 6.3(b), (d) and (f) on the right hand side display an error histogram of each data set.

It can be seen that the static MLP reached an acceptable generalization fitness for the history matching period, which comprised of the training and validation test set. The R^2 greater than 0.97 for both the training and validation sequence are only slightly worse compared to the recurrent ANN types. Figure 6.3(e) reveals the weakness of the static feed forward MLP, an R^2 of 0.64 on the test interval demonstrates that no generalization of the overall dynamics could be achieved and therefore the network's response on the unseen data is rather poor.

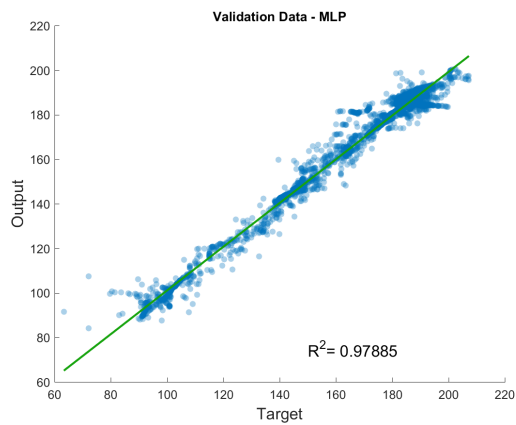
The three error histograms draw a similar picture, where the majority of errors lie in the range of -10 to $+10$ in the training and validation interval. The test set error histogram reveals once more the insufficient modelling done by the static MLP network. It can be seen that the errors lie in the range of -5 to $+20$, with a skewed distribution, while the errors in the training and validation set are nearly normally distributed with the mean around zero.



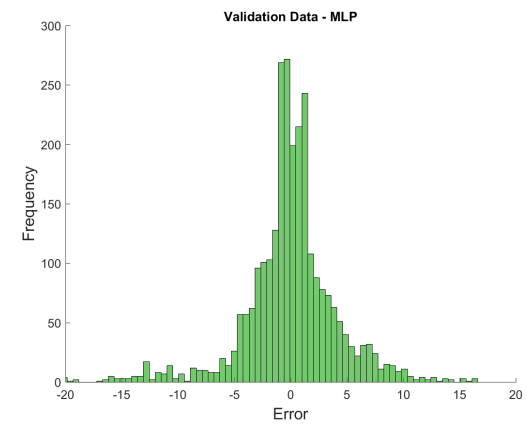
(a) Training Set - Correlation



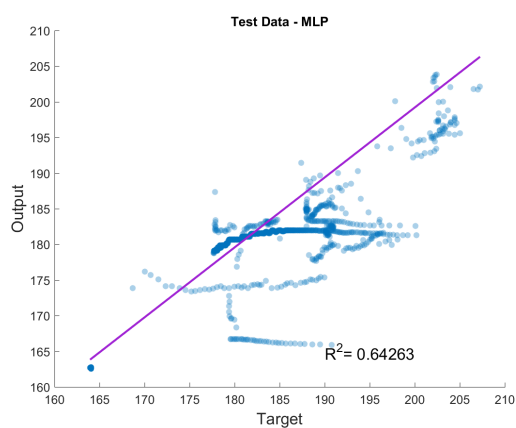
(b) Training Set - Histogram of Errors



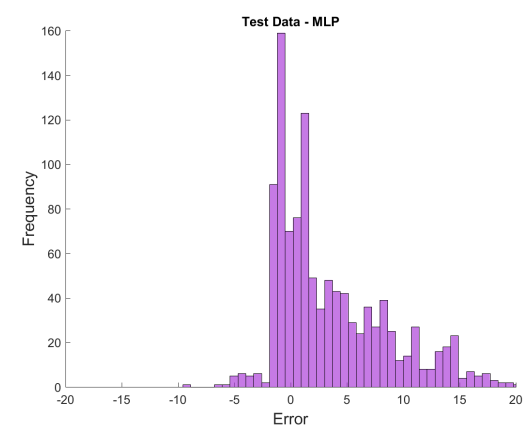
(c) Validation Set - Correlation



(d) Validation Set - Histogram of Errors



(e) Test Set - Correlation



(f) Test Set - Histogram of Errors

Figure 6.3: Output - Target correlation plot & error histogram for the MLP network. The training set is shown on top, the validation set in the middle and the test interval on bottom.

6.2 Fully recurrent ANN

The investigation of recurrent neural network architectures became inevitable after noticing the bad performance and generalization capability of feed forward networks on the task of modelling the dynamic system at hand. Analogous to the feed forward ANN several generations of fully recurrent networks were trained. Figure 6.4 shows the validation and testing error of all trained network generations. The lines connect the best validation set errors and test set errors of the networks achieving the best validation error. It is evident that the majority of both the validation and testing errors are lower than those of MLP networks. Another remarkable circumstance is the almost equal range of errors coming from the two data sets, leading to the conclusion that the recurrent networks' generalization capability is far superior. The substantially more complex training procedure of a recurrent ANN compared to a feed forward MLP explains the much wider spread in validation errors throughout individual network generations. The reason for this is thought to be the much higher tendency of getting stuck in local minima during training of recurrent networks with the BPTT method.

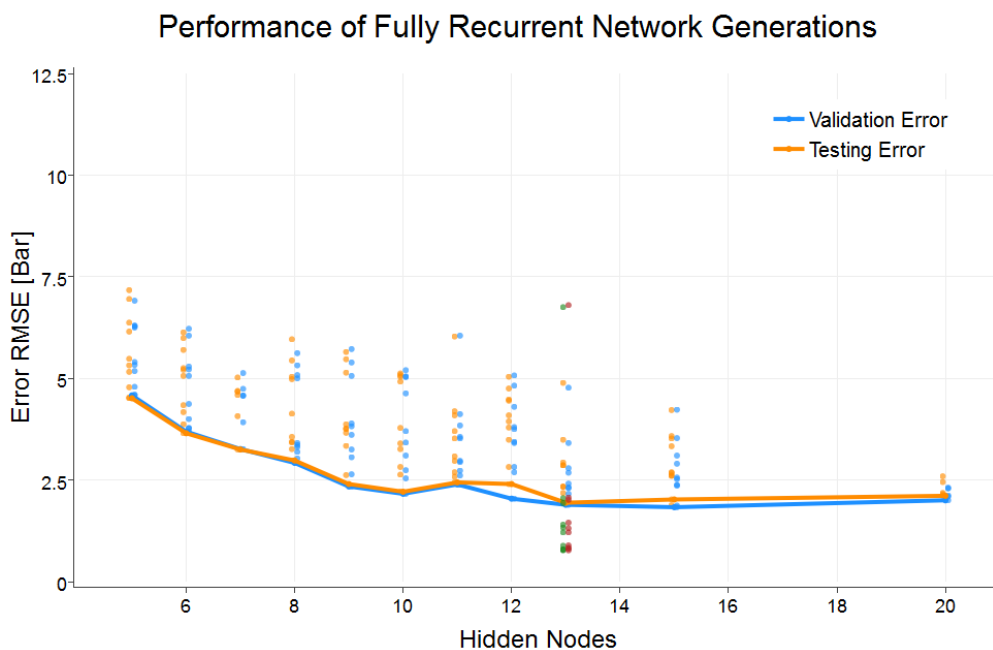


Figure 6.4: Development of validation & testing error with increasing network complexity. Single points display the errors coming from all networks in a fully recurrent generation. The red and green points are the validation and testing errors of the 10 Min sampling interval.

Figure 6.4 allows drawing the conclusion that very good performances can be achieved in the range of 13 to 20 neurons in the hidden layer, although more neurons reduce the required number of individual networks in a generation to find a "good" one. This claim can be substantiated by the reduction in the spread of errors from 13 to 20 neurons. Contrary to this especially the network generation with seven hidden nodes performed rather poor and was plagued by local minima.

A time series plot of the best fully recurrent network's forecast on Well 4 is shown in Figure 6.5. It can already be seen that the history matching period and the forecast on the independent data set is superior to the output from the MLP network. Especially in the lower figure, which depicts a zoom into the forecasting interval, an improvement of the generalization fitness is evident. The recurrent network is able to resolve the underlying dynamics and can therefore model the pressure behaviour successfully. The deviations between forecasts and desired outputs generally stay below 4 Bar, which is also the case for Wells 1, 2, 3 and 6. Time series plots for these wells are presented in Appendix C. All tested models struggled when forecasting pressure for Well 5, because this well had not been operated during the test interval and the wellhead pressure was just a function of the reservoir pressure and interaction with the other wells.

Generally, it can be said that the fully recurrent ANN exhibited a substantial enhancement compared to the feed forward architecture not only in the validation and test set error but also in the actual generalization. The contrast between the predictions done by the MLP (Figure 6.2) and the fully recurrent ANN (Figure 6.5) visualized this drastic improvement. These developments could be seen as the proof that recurrent connections could tip the scales towards being able to model the dynamic system governing the wellhead pressure.

6.2.1 Smaller resampling interval

A direct comparison of the same network generation on the 10 Min and hourly resampled data is shown in Figure 6.4. It can be observed that nearly all networks utilizing the lower resampling interval exhibit an improved performance compared to the hourly resampling interval. Figure 6.6 depicts a time series representation of the five month long training data set and independent test interval of Well 4. It is clearly visible that the increase of the resampling frequency does not decrease the network's generalization capability at all. It can even be said that the deviations between forecasts and desired outputs are generally lower than those of the network generation utilizing an hourly resampling interval. The errors in the test set stay below 2 Bar, which is also the case for Wells 1, 2, 3, 5, and 6.

Analysing the results depicted in Figures 6.5 & 6.6 allows drawing a few conclusions concerning the generalization capability of fully recurrent networks with respect to the selected resampling interval. It can be stated that successful training of ANNs using the BPTT method in conjunction with the LM algorithm is possible for a wide range of resampling intervals. Therefore the short and long-term dependences in the measured data can be resolved by the recurrent connections regardless of the selected resampling frequency. It can be assumed that an arbitrarily long training interval, which covers the whole range of possible input and target values that might occur in the dynamic system, is sufficient to generate adequate forecasts on unseen test data. The previous statement requires that the ranges of inputs encountered in the test and the training data set are the same.

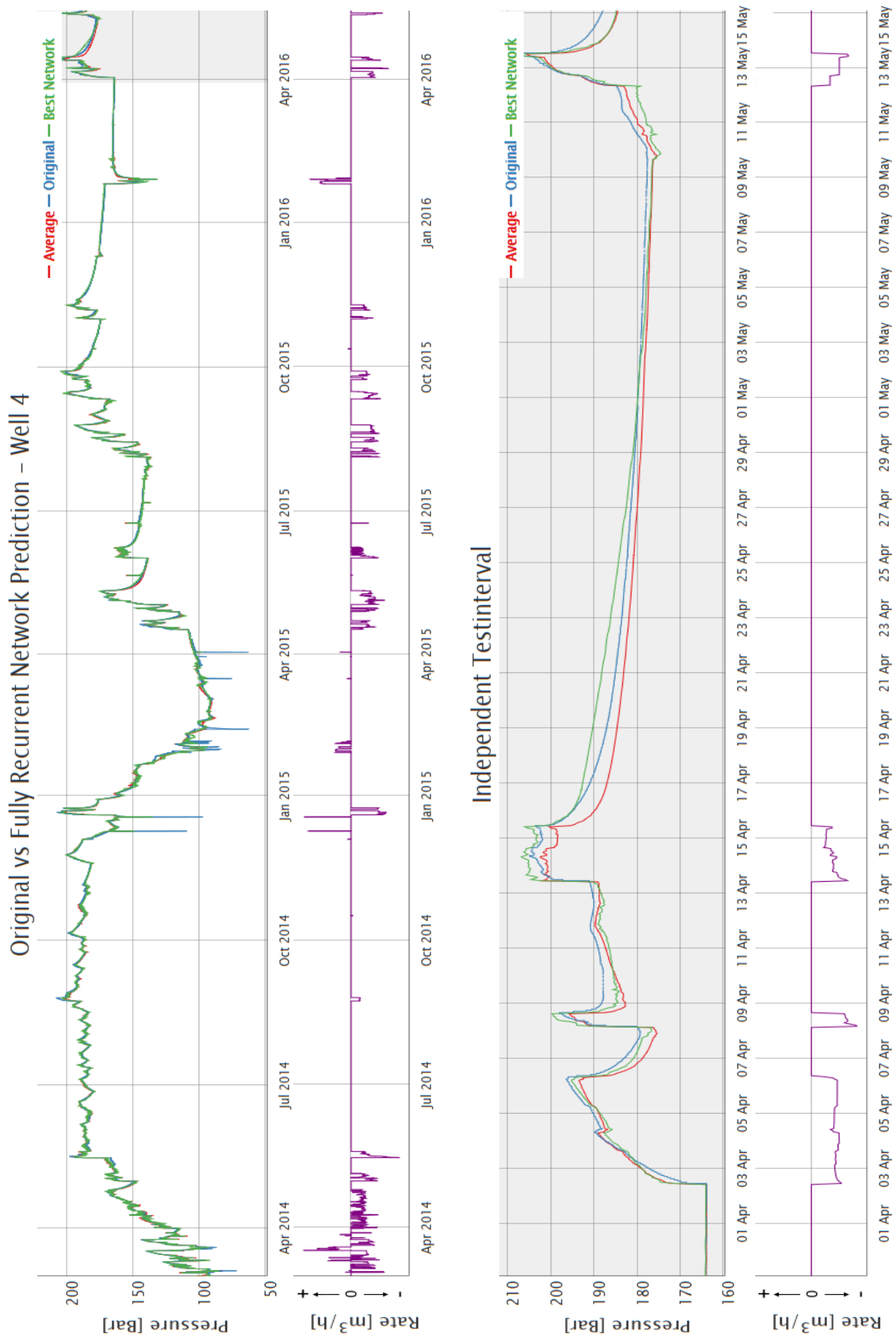


Figure 6.5: Comparison of Original & fully recurrent Network (Generation 9 with 13 hidden neurons) Output for Well 4 in a time series plot. The upper graph depicts the whole 27 months period. The independent test set is shaded in grey.

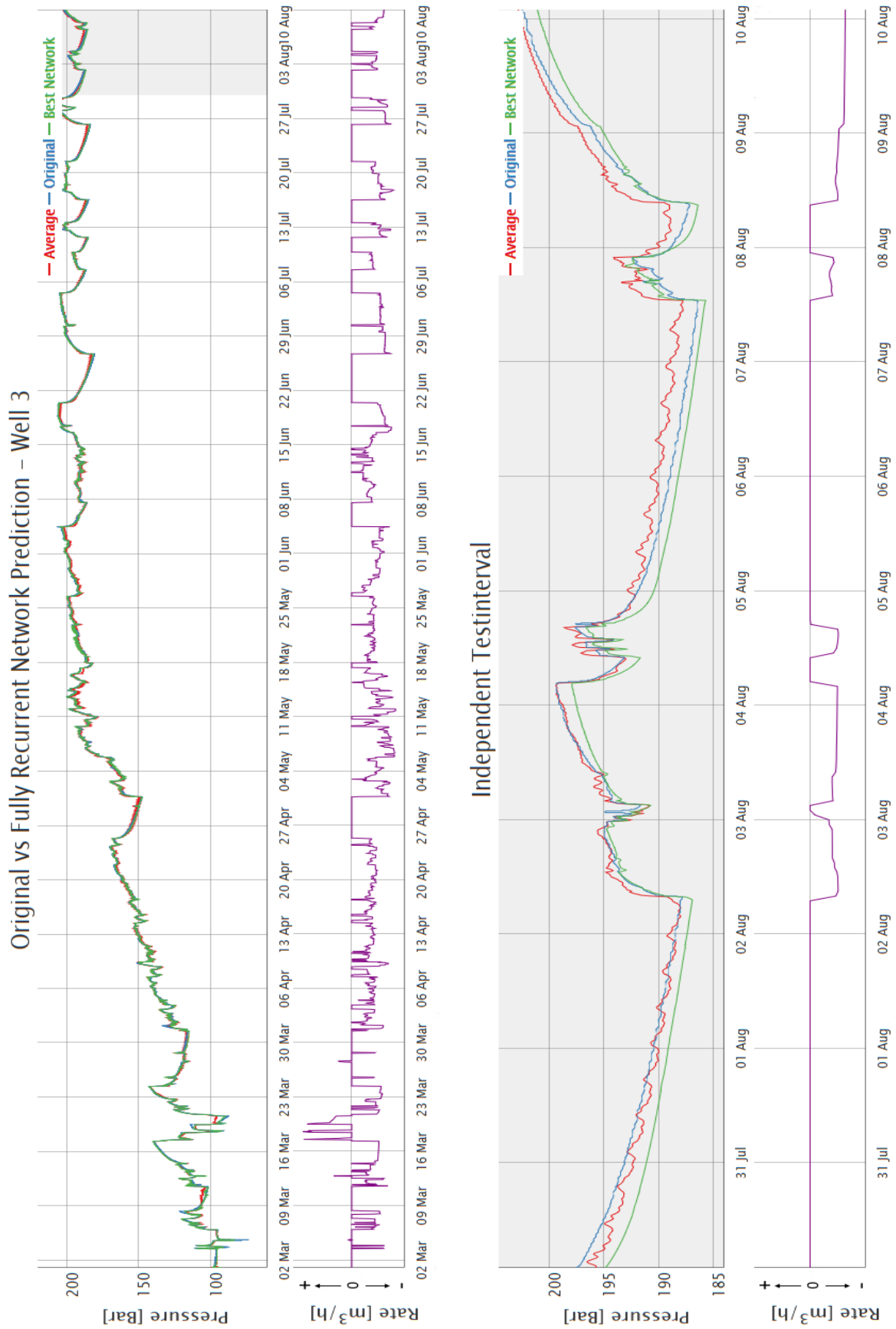
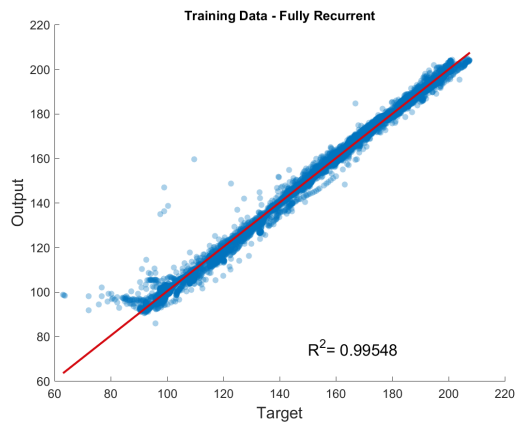


Figure 6.6: Comparison of Original & fully recurrent Network (Generation 9 with 13 hidden neurons) Output for Well 3 in a time series plot. The upper graph depicts the five months training period. The one and a half weeks long independent test set is shaded in grey.

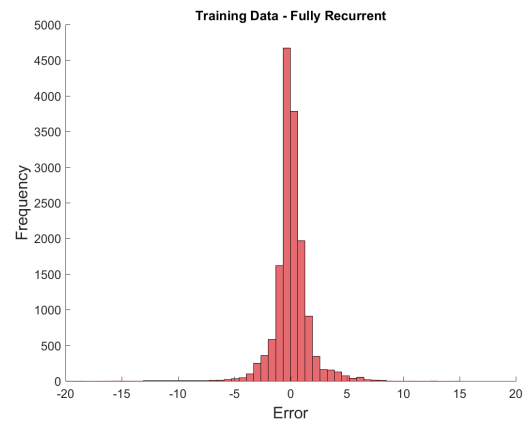
Figures 6.7(a), (c) and (e) show scatter plots of the fully recurrent network's output against the desired one and a superimposed coefficient of determination R^2 for the training, validation and test data sets. Figures 6.7(b), (d) and (f) on the right hand side display an error histogram of each data set. These figures are valid for the ANN using the hourly resampled data set, which is depicted in Figure 6.5.

From Figures 6.7(a), (c) and (e) it can be observed that the fully recurrent ANN achieved a nearly excellent generalization. Both the training and validation set have a coefficient of determination R^2 greater than 0.995 and even on the test interval the R^2 is higher than 0.985. The three scatter plots show that the fully recurrent ANN's is clearly superior to the static MLP network. The scatter plot of the test interval shows an impressive improvement of the fully recurrent network over the static one. It can again be said that the feedback connections in the fully recurrent ANN enabled this improvement.

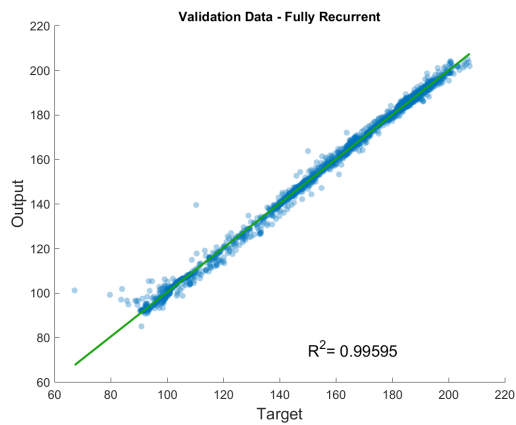
All three error histograms also resemble the enhanced performance of the fully recurrent ANN. The majority of errors lie in the range of -5 to $+5$ for all three data sets. In contrast to the static MLP network all three histograms show a normal distribution of errors in the training and validation as well as in the test interval. This normal distribution with a mean at zero indicates the enhanced modelling capability enabled through the recurrent nature of this network type.



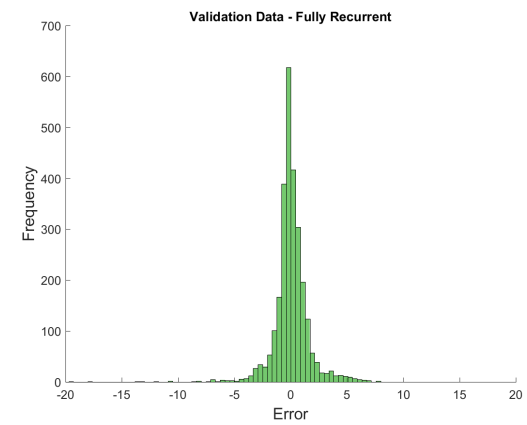
(a) Training Set - Correlation



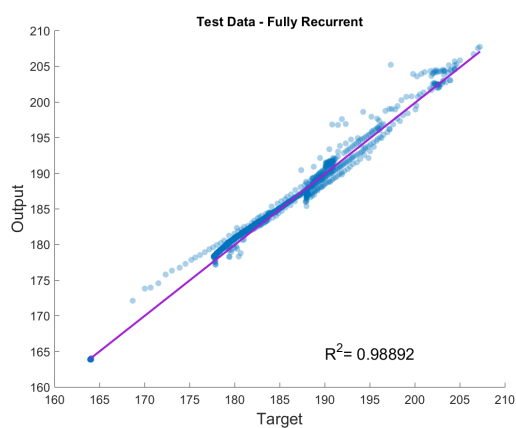
(b) Training Set - Histogram of Errors



(c) Validation Set - Correlation



(d) Validation Set - Histogram of Errors



(e) Test Set - Correlation



(f) Test Set - Histogram of Errors

Figure 6.7: Output - Target correlation plot & error histogram for the fully recurrent network. The training set is shown on top, the validation set in the middle and the test interval on bottom.

6.3 Elman ANN

Several generations of Elman networks were trained to study which influence the simplifications done to the fully recurrent ANN would have on the performance and generalization fitness. Figure 6.8 once more depicts the validation and testing errors of all network generations. The lines connect the best validation set errors and test set errors of the networks achieving the best validation error. It is apparent that also the Elman ANN shows enhanced performance over the feed forward MLP network. The Elman network's errors on the validation and testing set are comparable to those of the fully recurrent ANN. Analogous to the beforehand discussed network type, the Elman ANNs show similar deviations between validation and testing errors. This fact clearly indicates that the generalization capability of the Elman architecture is more or less equal to the fully recurrent ANN. The Elman network also suffers from the higher tendency of getting trapped in a local minimum, which explains the spread of validation and testing errors throughout a network generation. It is once more fair to say that an increase of neurons in the hidden layer reduces this phenomenon's occurrence. The results shown in Figure 6.8 allow claiming that at least 10 hidden neurons are required to ensure a decent network generalization, tantamount to small errors in both data sets. Due to the fact that just four Elman generations were trained no arbitrarily poor performing network complexity could be found.

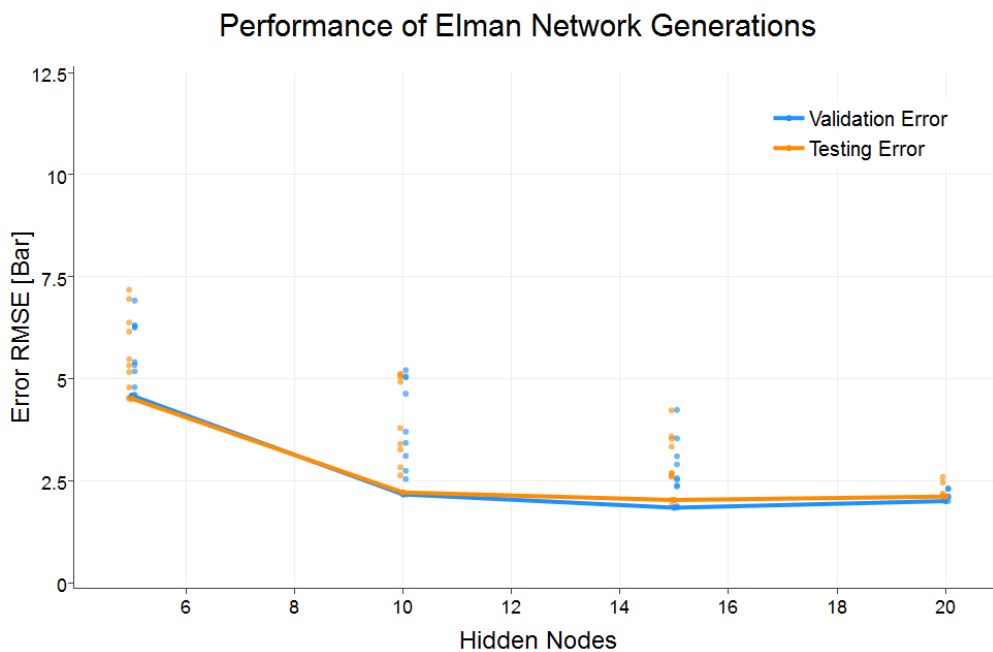


Figure 6.8: Development of validation & testing error with increasing network complexity. Single points display the errors coming from all Networks in an Elman generation.

Figure 6.9 graphs the best Elman network's prediction of wellhead pressures for Well 4. Again the performance in the history matching period and the forecast on the independent test set is far better than what the best feed forward network was able to achieve. It is fair to say that the Elman network is also capable of reproducing the underlying tendency. Although the errors of the best Elman and fully recurrent ANN are close together, the zoom into the forecasting period reveals later one's slightly better performance. While the fully recurrent type managed to keep the deviations between forecast and desired output below 4 Bar, the Elman ANN's prediction accuracy deteriorates with an increasing prediction horizon. This leads to errors up to 7 Bar, at prediction horizons greater than one week. The Elman network showed equal outcomes for Wells 1, 2, 3, 4 and 6 while also struggling when dealing with Well 5, for the same reasons as discussed in the previously.

Figures 6.10(a), (c) and (e) show scatter plots of the Elman network's output against the desired one and a superimposed coefficient of determination R^2 for the training, validation and test data sets. Figures 6.10(b), (d) and (f) on the right hand side display an error histogram of each data set.

The scatter plots of the training and validation data sets are comparable to those produced by the fully recurrent ANN, with an R^2 greater than 0.995. The plot coming from the test interval with an R^2 of 0.92 shows at a glance that the Elman network's generalization capability is worse than the fully recurrent one's but way better than what the static MLP network could achieve.

The three error histograms draw a similar picture, where the majority of errors lie in the range of -5 to $+5$ in the training and validation interval. The worse performance in the test interval is also resembled in the error histogram, as the range in which the majority of the error lies has increased to -10 to $+10$. In comparison to the static MLP network all errors in the Elman network are nearly normally distributed with a mean at zero.

Analysing the results shown in Figures 6.8, 6.9 & 6.10 led to the conclusion that the generalization capability of the Elman ANN is on the one hand superior to the feed forward ANN but on the other hand surpassed by the fully recurrent one's. It can still be said that even the simplified feedback in the Elman network made it possible to model a dynamic system as it is prevalent in the data gathered from an UGS.

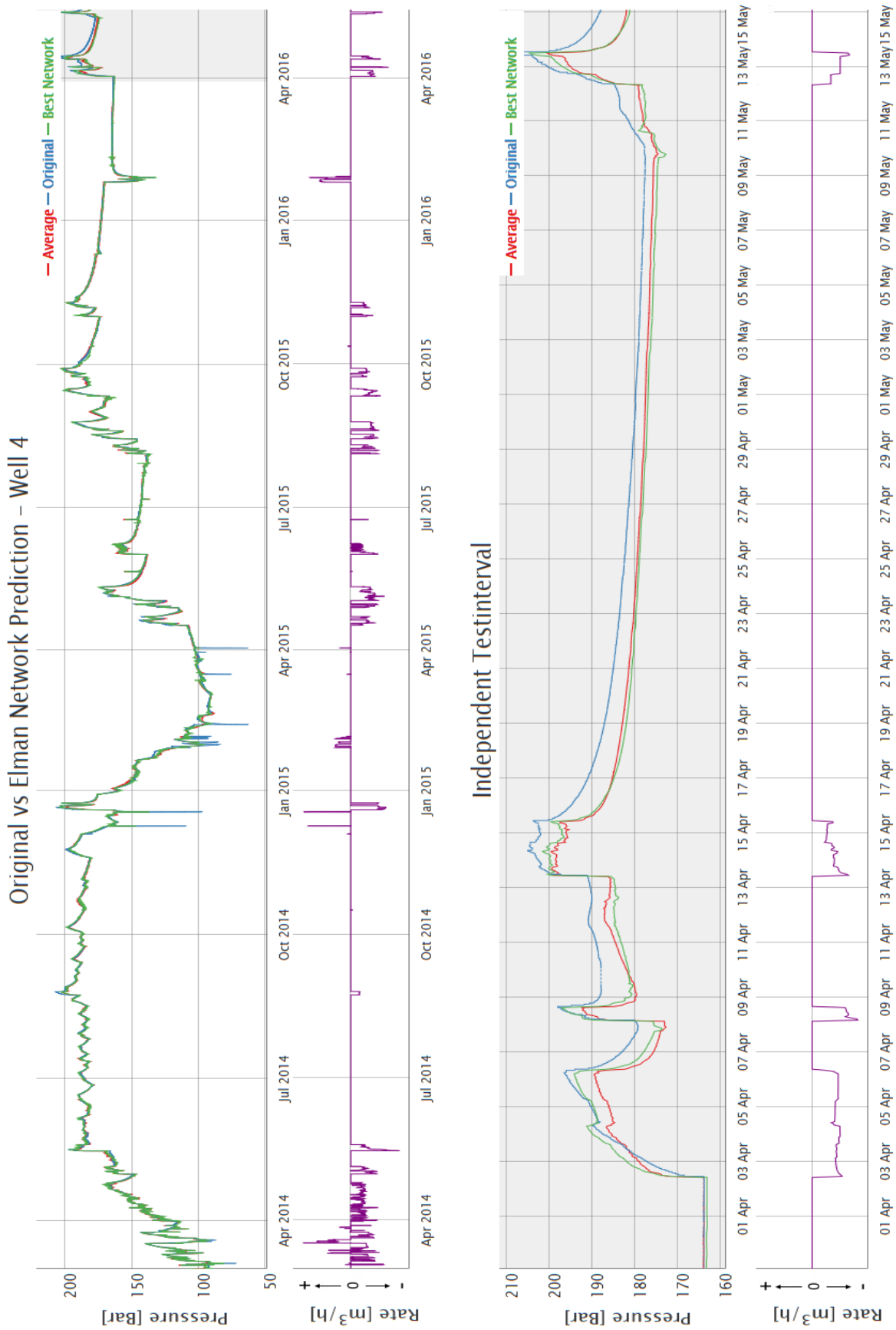
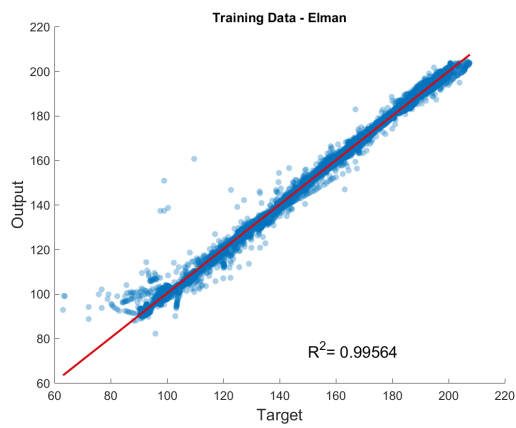
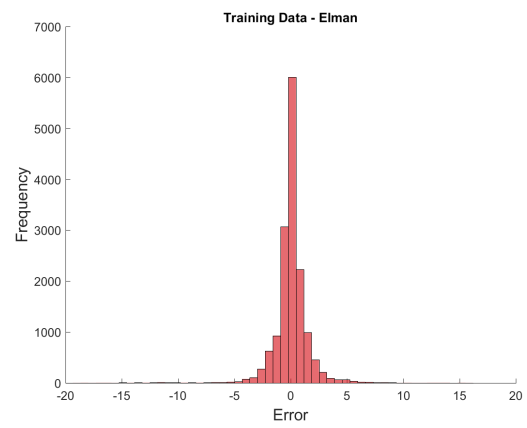


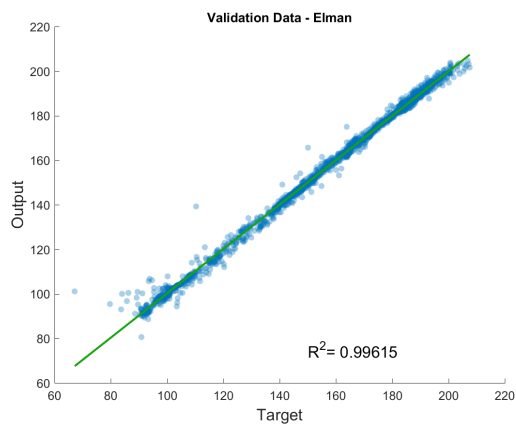
Figure 6.9: Comparison of Original & Elman network (Generation 3 with 15 hidden neurons) Output for Well 4 in a time series plot. The upper graph depicts the whole 27 months period. The independent test set is shaded in grey.



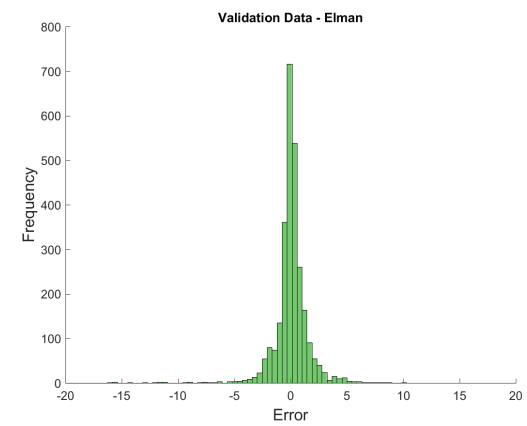
(a) Training Set - Correlation



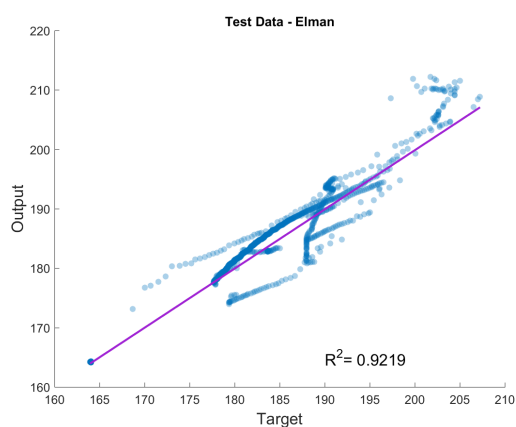
(b) Training Set - Histogram of Errors



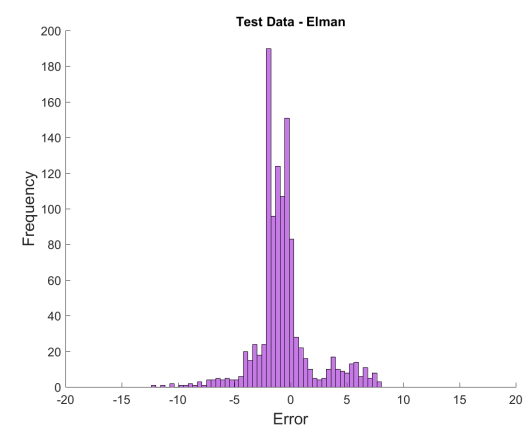
(c) Validation Set - Correlation



(d) Validation Set - Histogram of Errors



(e) Test Set - Correlation



(f) Test Set - Histogram of Errors

Figure 6.10: Output - Target correlation plot & error histogram for the Elman ANN. The training set is shown on top, the validation set in the middle and the test interval on bottom.

6.4 NARX ANN

The NARX model was another examined variant of recurrent neural networks. As done with all other architectures before, numerous NARX ANN generations (see Table 5.5) were trained and tested. The resulting validation and testing errors are displayed in Figure 6.11. The lines connect the best validation set errors and test set errors of the networks achieving the best validation error. Similar to all previously discussed recurrent networks the errors reached are lower than those of the feed forward MLP ANN. Another parallel shared between all recurrent network architectures is the almost equal range of validation and test set errors, leading once more to the conclusion that the NARX ANN is able to generalize adequately. In contrast to all similarities described, the NARX generations do show a rather constant spread of deviations from 5 up to 20 neurons. Additionally there is no such clear trend of decreasing errors with increasing number of neurons in the hidden layer, as exhibited by the other two recurrent ANN types. The performance more or less stabilizes for network complexities above 10 hidden neurons.

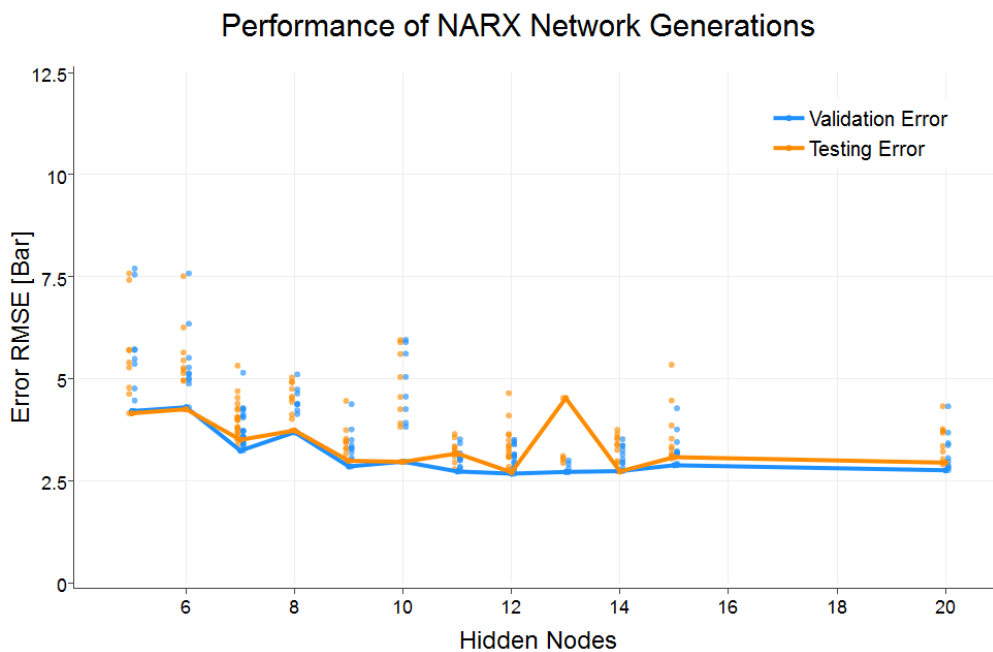


Figure 6.11: Development of validation & testing error with increasing network complexity. Single points display the errors coming from all networks in a NARX generation.

A time series plot of the best NARX ANN's prediction on wellhead pressures for Well 4 is displayed in Figure 6.12. The NARX model again displays the strength of feedback connections, thereby producing far better outcomes both in the history matching and testing interval. The zoom into the forecast period reveals the NARX network's generalization fitness and its ability of modelling the inherent dynamic system. There are errors up to 7 Bar in the network predictions for the independent test set thereby achieving a similar performance as the Elman ANN. Similar to the Elman model the prediction accuracy starts to deteriorate at prediction horizons greater than a week. It can also be concluded that NARX model is not able to catch up with the fully re-

current ANN's superior forecasting capabilities. A drawback encountered in the NARX networks were the larger fluctuations in the prediction performance on the different wells. This circumstance made it difficult to identify an individual NARX model that showed equal generalization for all wells. Additionally the issue regarding the pressure forecast for Well 5 was also encountered.

Figures 6.13(a), (c) and (e) show scatter plots of the NARX network's output against the desired one and a superimposed coefficient of determination R^2 for the training, validation and test data sets. Figures 6.13(b), (d) and (f) on the right hand side display an error histogram of each data set.

From Figures 6.13(a), (c) and (e) it can be observed that the results achieved by the NARX network are comparable to those of the Elman network and worse than the fully recurrent ANN. Both the training and validation set have a coefficient of determination R^2 around 0.995. The scatter plot of the results from the test interval with an R^2 of 0.93 depict the worse generalization of the NARX network compared to fully recurrent one. It can be observed that in general any feedback connections can enhance the network's predictive performance and produce better results than the pure feed forward MLP.

The error histograms of the NARX network are quite similar to those of the Elman network. The majority of errors coming from the training and validation data sets lie in the range of -5 to $+5$ with a mean around zero. The histogram of errors from the independent test interval lie in a more skewed range of -7 to $+5$ with a mean around -2 . The error histograms also confirm the observation that the NARX and Elman networks show a comparable performance, which is slightly worse than the fully recurrent one's but way better than what the static MLP network can achieve.

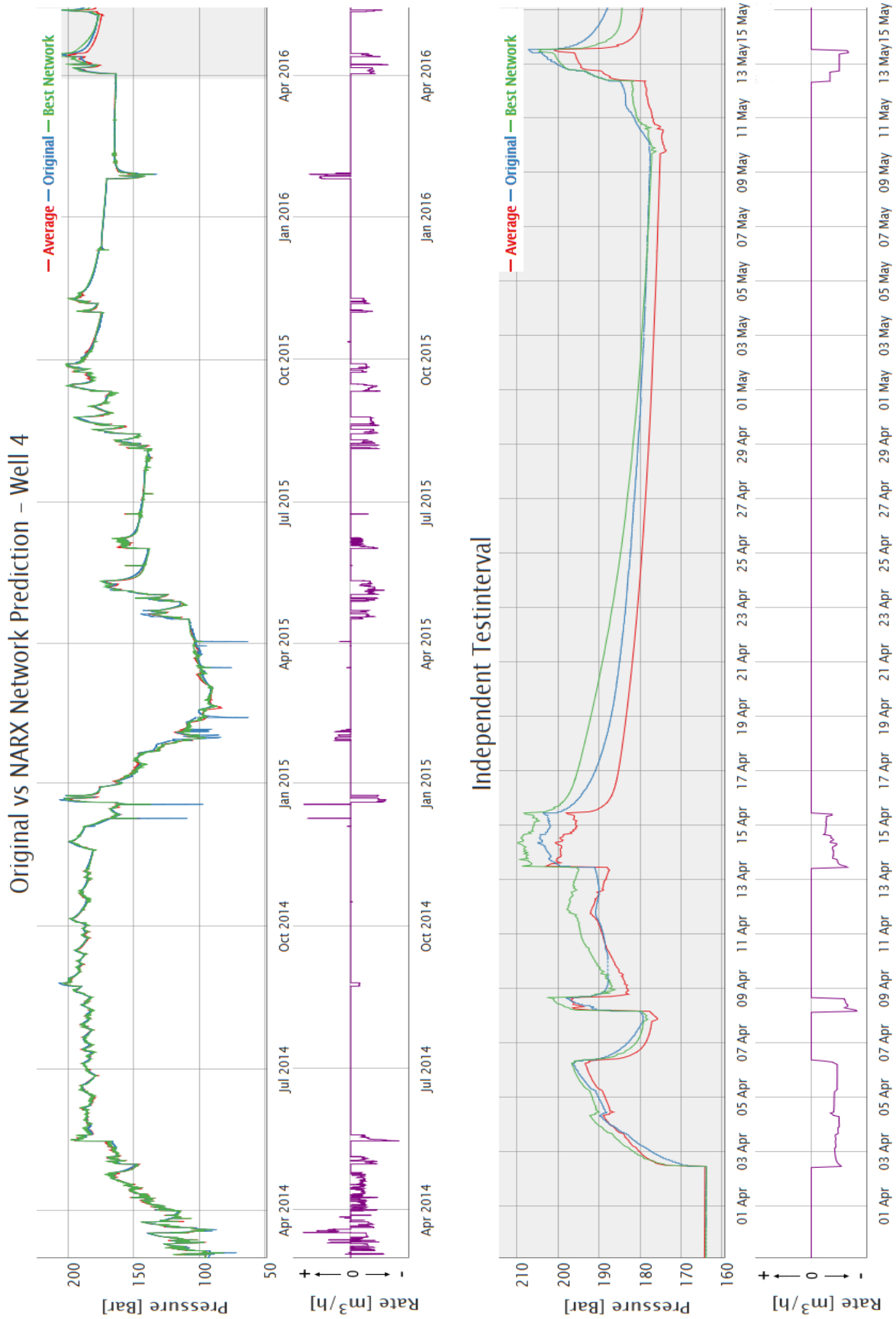
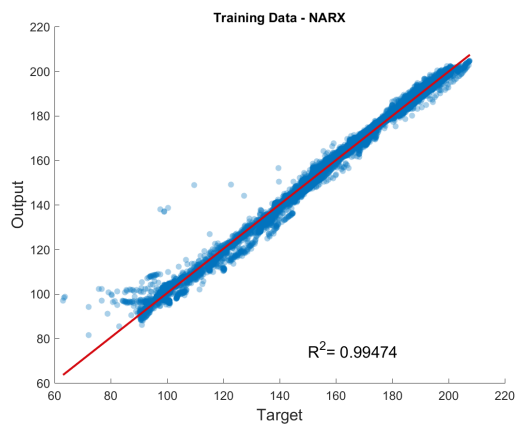
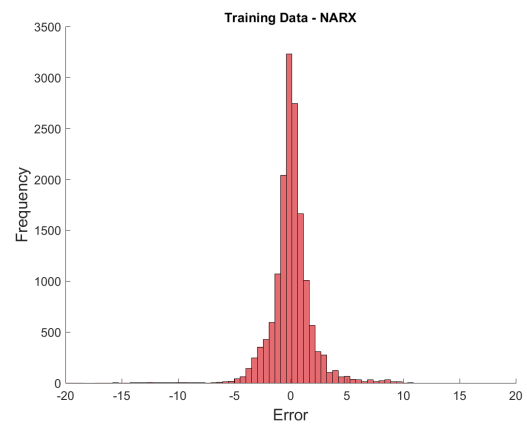


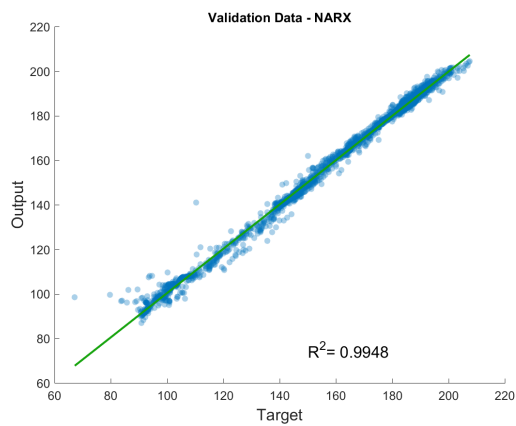
Figure 6.12: Comparison of Original & NARX network (Generation 8 with 12 hidden neurons) Output for Well 4 in a time series plot. The upper graph depicts the whole 27 months period. The independent test set is shaded in grey.



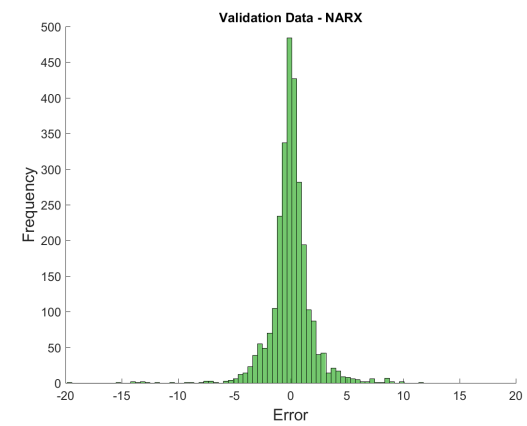
(a) Training Set - Correlation



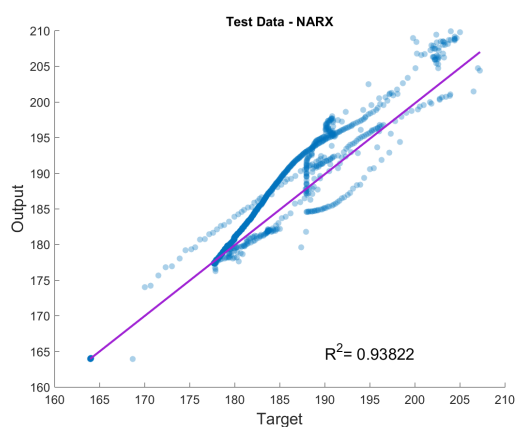
(b) Training Set - Histogram of Errors



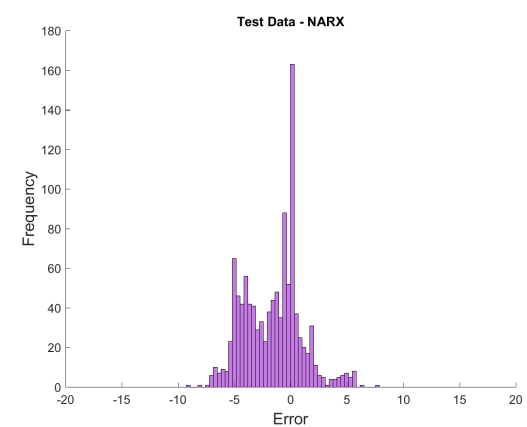
(c) Validation Set - Correlation



(d) Validation Set - Histogram of Errors



(e) Test Set - Correlation



(f) Test Set - Histogram of Errors

Figure 6.13: Output - Target correlation plot & error histogram for the NARX ANN. The training set is shown on top, the validation set in the middle and the test interval on bottom.

6.5 Echo State Network

Several ESN generations (see Table 5.6) were trained and tested to investigate how the ESN architecture would compete with feed forward networks and other types of recurrent ANN. As already discussed the major difference between an ESN and the other recurrent architectures tested was the network training itself. In an ESN only output weights had to be trained whereas in all other tested networks all connections were modified. Figure 6.14 displays the training and testing error of the best set of trained networks for each generation. The lines connect the best validation set errors and test set errors of the networks achieving the best validation error. It is clearly visible that the error in the training set decreases monotonically with increasing reservoir size, whereas the testing error shows a more irregular behaviour. A remarkable observation is the reverse trend of errors in the test set with increasing complexity. This allows claiming that the general rule of using a reservoir, as big as affordable does not apply for the problem at hand. The best performing network generation is found to have 50 hidden neurons while the performance drastically decreases at sizes greater than 500. The reason for this is thought to be overfitting of the ESN which leads to a higher performance in the training interval at the cost of overall generalization capability. It even results in clearly non reasonable outcomes as can be seen for more than 1,000 neurons, where the trainable weights and the amount of time steps ($\approx 18,000$) are just one order of magnitude off.

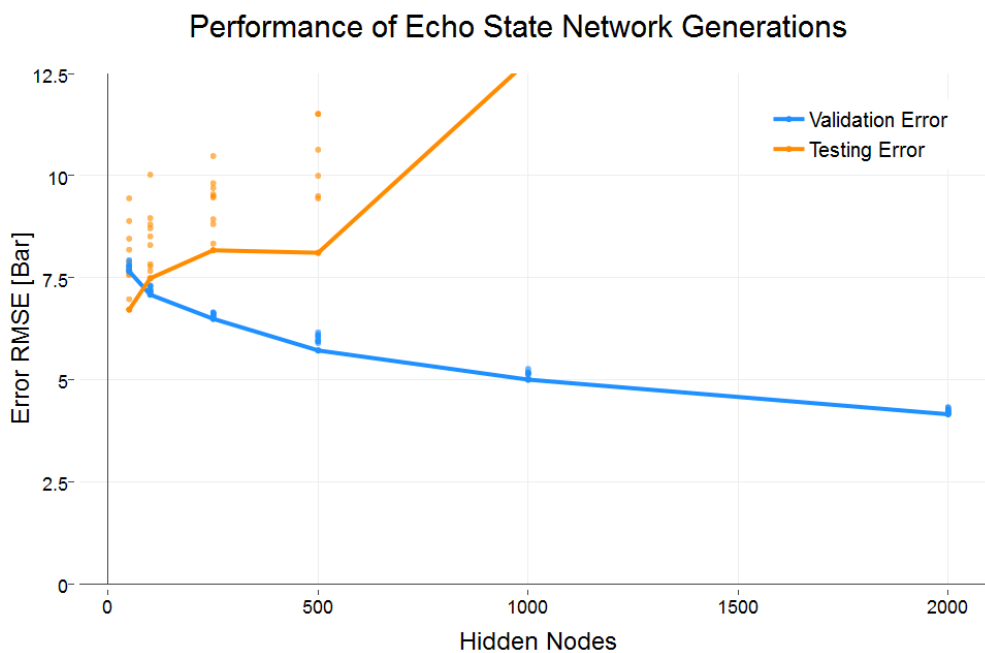


Figure 6.14: Development of validation & testing error with increasing network complexity. Single points display the errors coming from all networks in an ESN generation.

The investigation regarding the input scaling did reveal one already beforehand assumed circumstance. It could be observed that increasing the relative importance of the UGS filling level and the pressure measurement coming from the one observation well did yield better outcomes.

Doubling the influence of those two inputs compared to the other six produced the lowest error.

No clear trend, concerning the impact of different spectral radii on the ESN's performance could be perceived. Testing additional ESN generation with a sampling interval of 10 Min instead of the standard hourly interval revealed a few hints. The ESN generations trained with the hourly sampled data set showed a tendency towards smaller spectral radii, whereas the ESN trained with the smaller sampling interval indicated an opposing trend. The reason for this was thought to be the memory span required to model the long term dependences in the data sequence properly. Obviously when a smaller sampling interval had been used the reservoir had to remember more individual time steps to cover a certain memory span and larger intervals had an opposite effect.

Figure 6.15 shows a time series plot of the best ESN generation's forecast of wellhead pressure on Well 4. A noteworthy observation is the spiky behaviour of the network's response and although the network is able to capture the overall trend in the data, it is not fit to resolve details like a pressure buildup with a satisfactory accuracy. Especially in the independent test sequence this lack of generalization is depicted by the unsatisfactory pressure modelling. The forecast for the five other wells show the same rather underwhelming performances.

Figures 6.16(a) and (c) show scatter plots of the ESN's output against the desired one and a superimposed coefficient of determination R^2 for the training and test data sets. Figures 6.16(b) and (d) on the right hand side display an error histogram of each data set.

Figures 6.16(a) and (c) it can be observed that the results produced by the ESN are comparable to those by a static MLP network. Especially the plot of network outputs against desired targets of the ESN training period, which is a combination of the training and validation sets of the other network types, with an R^2 of 0.90 shows its poor generalization. Although this marks the worst performance of all ANN types on the training set, the results on the test set with an R^2 of 0.71 are slightly better than those achieved by the feed forward MLP.

The error histograms of the ESN are similar to the ones from the MLP network. From Figure 6.16(b) it can be observed that the majority of training errors lie in the range of -20 to $+20$ with a mean around zero. The error histogram coming from the independent test interval displays a skewed distribution with a range from -5 to $+20$, which is comparable to the MLP type.

It can be concluded that the ESN networks did not exhibit much of an improvement over the static feed forward MLP. Comparing the time series plots from the other tested network types clearly indicated that at the current state a fully recurrent, Elman or NARX model should be prioritized over the ESN ANN. Due to the exceptionally fast training speed of ESN it was possible to test various configurations in little time.

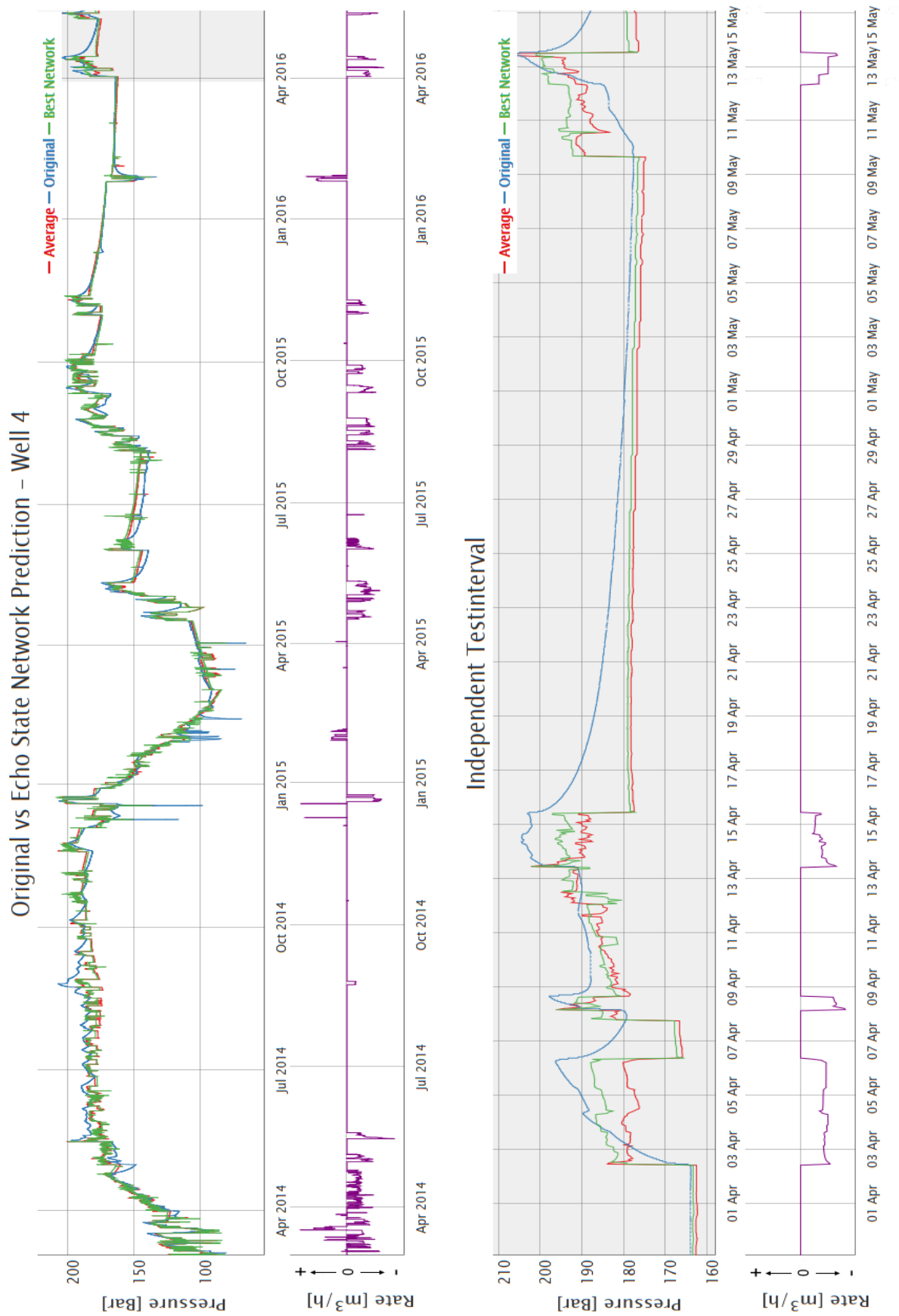
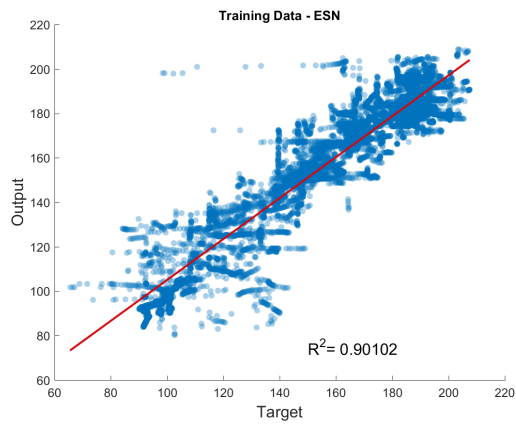
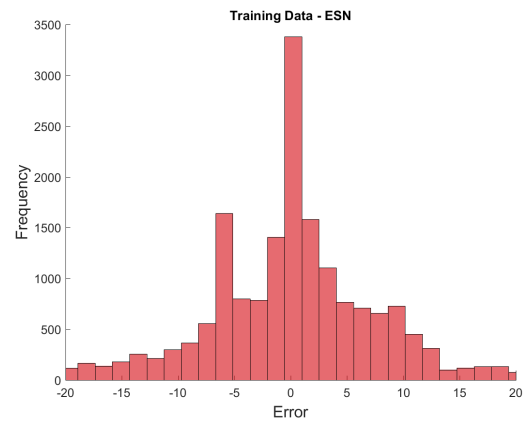


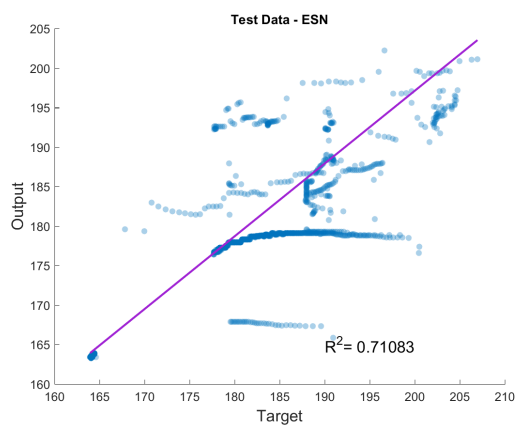
Figure 6.15: Comparison of Original & ESN (Generation 1 with a reservoir size of 50 neurons) Output for Well 4 in a time series plot. The upper graph depicts the whole 27 months period. The independent test set is shaded in grey.



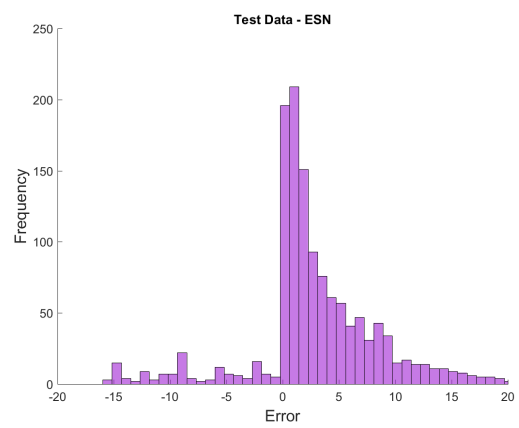
(a) Training Set - Correlation



(b) Training Set - Histogram of Errors



(c) Validation Set - Correlation



(d) Validation Set - Histogram of Errors

Figure 6.16: Output - Target correlation plot & error histogram for the ESN. The training set is shown on top and the test interval on bottom.

6.6 Ranking of ANN types

A comparison of the results from all ANN types clearly indicated that the static feed forward MLP network could not compete with the recurrent network architectures. The fully recurrent, Elman and NARX network have exhibited an undoubtedly superior performance not only in the training and validation data set but also in the independent test interval. Especially the time series representations of network prediction and original values showed the enhanced generalization capability of recurrent ANN types. In addition to those graphs the correlation plots and error histograms further underline this conclusion. It could be concluded that the integration of historic input and output variables into the generation of current network outputs enabled the recurrent networks' enhanced performance. The ESN was the only recurrent network architecture which could not outperform the static MLP network and just achieved similar performance compared to the former type.

Figure 6.17 ranks all tested ANN types according to their error in the validation data set with the best architecture at the bottom. It can be seen that the fully recurrent ANN achieved the least error in the validation as well as in the independent test set. It is closely followed by the Elman and NARX network, whereas the ESN and MLP network are outperformed by a substantial margin.

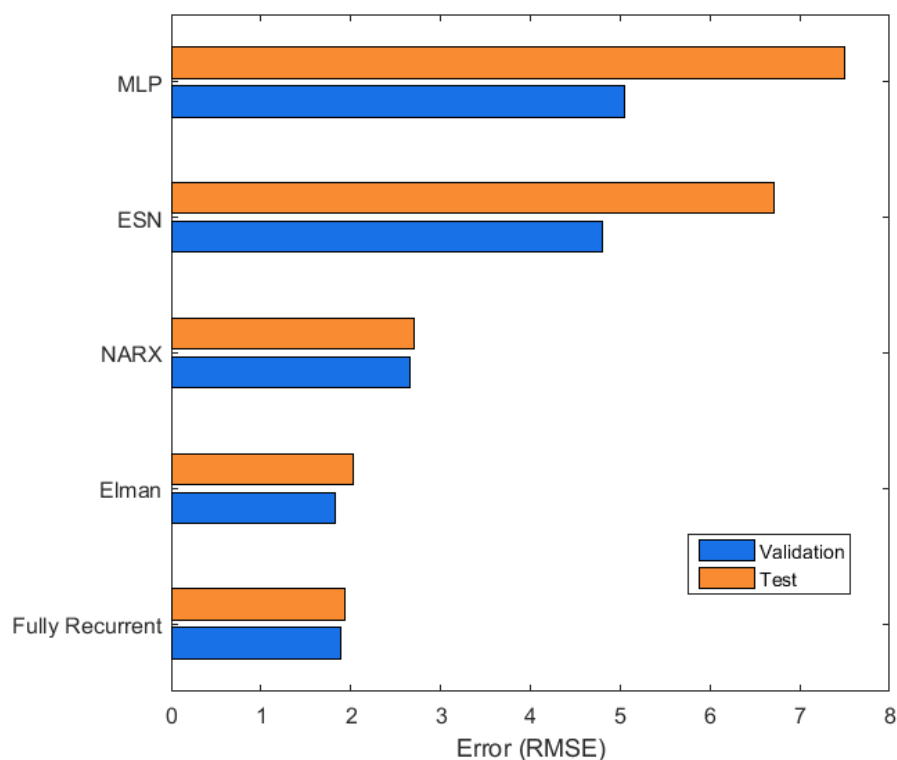


Figure 6.17: Ranking of networks architectures according to their validation errors.

7 Conclusion

This chapter reviews the motivation and purpose of the research done and sums up the developed methodology by discussing the steps from gathering the raw data to the generation and application of simulation models. Finally, the outcome of all models is analysed and a recommendation regarding their performance on this field of application is provided.

Although the underground storage of natural gas is a longstanding technology there is still a potential for growth in this business. In former days this business was driven by a seasonal varying demand of gas opposed by the nearly constant supply of gas produced and imported to Europe. The majority of long term gas delivery contracts contained a fixed gas volume per time, that would be too much in the warmer and too low in winter times. The true purpose of gas storage was then to ensure that demand could be met throughout the whole year cycle by using storage facilities to compensate for pipeline capacity. The boom of shale gas in the United States and the rapid decline of oil price, which the gas price is coupled to led to changes in the traditional supply and demand situation on the market. Economides and Wang stated that since the upsurge of U.S. shale gas in the last decade the traditional gas storage business became a lot less profitable. It was suggested to switch from long term contracts the fixed seasonal injection and withdrawal schedules to more profitable short time schemes. The proposed short time schedules were oriented at the weekly changes in supply and demand of gas and current prices at the market [4].

The ability of forecasting the UGS operation not only on a seasonal but also daily and hourly basis was the essential prerequisite to adapt these facilities to short time schedules in addition to already booked long term volumes. These short time schedules required a response time in the order of a few hours in which an optimal injection/withdrawal strategy had to be found. The concept of classic reservoir simulation demanded exceedingly high computational power to model several schedules in parallel and with appropriate accuracy.

This thesis investigated the applicability of neural networks to the task of forecasting the UGS operation. Neural networks are a purely data driven mathematical tool dedicated to classification, regression and prediction tasks. Therefore they can be used to model any existing relationship in data without nearly any knowledge about the underlying physical laws. The methodology developed to allow their application to simulate and forecast the operation of an UGS, included the steps taken in data processing and setting up the actual models.

The established data processing procedure described the gathering of dynamic and static data from internal RAG databases. The dynamic data consisted of pressure, flow rate and temperature measured at the wellhead. These measured signals from all wells were compiled into a compre-

hensive format and remedied of deficiencies like outliers, missing samples and errors coming from the measurement process itself. Several statistical plots were done in order to get a better understanding of the data at hand and to compare the quality of data coming from different wells. Thereafter the irregularly sampled data was resampled to a regular time interval using two different alternatives. In the first approach the signal was resampled using linear interpolation and the application of appropriate filters. The alternative method used a kernel regression to resample the signal to a continuous sampling interval and smoothing it in one step. The strength of the non-parametric kernel regression was the use of general parameters in order to resample the signal to any desired interval. Using the alternative of linear resampling required the careful adaption of the filtering process to the preferred sampling intervals. Albeit the second method suffered from greater complexity its greatest asset was the increased quality of the resampled signal.

There were several neural network architectures to be taken into account for simulation and forecasting of a dynamical model. The ANN model discussed in this work reached from pure feed forward types to different variants of the recurrent architecture. The basis of all models was the dynamic data coming from the wellhead measurement which had to be preprocessed before any modeling could be done with it. The first step of this preprocessing schedule consisted of picking suitable network inputs in order to predict the pressure at the wellhead. An hourly resampling interval was selected as the basis for all ANN models, as this reduced the computation time while still maintaining a satisfactory accuracy.

Several network generations of various ANN types were trained on identical data sets and equal random seeds. Finding the best network architecture and its optimal complexity required testing numerous networks with equal starting conditions and data sets. An obvious outcome from all tested ANN types clearly indicated that static feed forward networks were a poor choice when trying to model dynamic systems. This architecture did not only struggle with predictions on unseen data but also with matching the historic pressures of the much longer training interval. The recurrent architectures, particularly the fully recurrent ANN did exhibit a substantial improvement of achieved errors and generalization capability. All of these models, with the exception of ESN, managed to reproduce the trend in the training data set and also in the independent test interval. The best fully recurrent network generation reached an error of approximately ± 4 Bar in the test set. In general all tested network architectures could be ranked in the following order. The best performance was achieved by the fully recurrent ANN, followed by the Elman and NARX network that showed very similar performances. The least effective recurrent ANN proved to be ESN type which exhibited nearly as bad a performance as the feed forward MLP.

None of the tested neural network architectures could reach the imposed accuracy threshold of ± 1 Bar. The time series plots of network outputs and desired targets clearly indicated that certain types of neural networks were able to model the operation of an UGS, albeit with reduced

accuracy. Especially when analysing the performance on the independent test set it could be observed that the prediction accuracy deteriorated over time. If ANNs were used for predictions for a period less than one week a suitable accuracy could be reached. The very low computation times for predictions done with pre-trained networks compared to classical reservoir simulation proved to be a major strength of neural networks. A major advantage of neural networks, once they are trained, lay in the very short computation time for a prediction scenario. A model using neural networks could compute several times as much scenarios as a model using classic reservoir simulation. Requiring that both models had the same computational power available.

Analysing the averaged outputs in the time series plots revealed that the best network's generalization and errors were always slightly better than one of the average. It could be observed that both the best network's output and the averaged one did follow the trend in the original data set. A network ensemble could be used as sort of guideline to check the reasonableness of a network's forecast on absolutely new data. A high deviation between the predictions coming from an ensemble and a single network could indicate an erroneous forecast. This is based on the assumption that a network ensemble's output is more robust and the erroneous prediction of a single ANN in the ensemble would be alleviated by the averaging process.

In addition to the evaluation of different ANN architectures the influence of the resampling frequency on the network's generalization capability was evaluated. It could be shown that there was no negative impact inflicted on the network's performance by decreasing the resampling interval in the training data set. Therefore the short and long-term dependencies in the measured data can be resolved by the recurrent connections, regardless of the selected resampling frequency. It could even be observed that the reduced sampling interval resulted in lower errors both in the validation and independent test set.

8 Discussion and Outlook

To further increase the performance of neural networks on the task of simulating the operation of an UGS, several options could be evaluated. First of all, additional data could be used as model input if available and secondly the data could be transformed in any possible way to extract supplementary features. The adaption of the used sampling interval and the filtering process would be another option, although great care has been taken when designing the resampling process. Increasing the sampling interval of the inputs fed to the ANN could be considered as a possible way of decreasing the achievable errors. The largely increased computation time is a major drawback of this approach.

Another approach to improve the prediction capabilities of ANNs would be modifications to the discussed types. This ranges from increasing the number of individual networks in a generation and the amount of training iterations up to a sensitivity analysis off all adjustable network parameters. The MATLAB Neural Network toolbox offers the opportunity to change any network parameter and to design custom neural network architectures. Analysing the impact of changing predefined network parameters is a time consuming tasks and there is no clear indication of the possible benefits coming along with it.

Although several ANN architectures were discussed and analysed in this work, there are still numerous more advanced architectures to be tested. "The Long Short-Term Memory" (LSTM) [70, p. 1735-1780] and "Gated Recurrent Unit Network" (GRU) are two types dedicated to the modelling of sequences containing long term dependences. The general idea behind these network architectures was to replace the hidden neurons and their standard sigmoid activation function by a more complex memory cell. The purpose of this memory cells was to mitigate the issue of vanishing gradients in training of recurrent ANN and therefore enabling it to capture long term dependences. Chung et al. compared these two types and with each other and with standard recurrent network on several data sequences. It was found that both types performed very similar and were superior to the standard recurrent ANN, not only in the errors reached but also in computation times required [71].

9 References

- [1] Bundesministerium für Wissenschaft, Forschung und Wirtschaft, “Energiestatus Österreich 2015,” 31.10.2016. [Online]. Available: <http://www.bmwf.wg.at/EnergieUndBergbau/Energieeffizienz/Documents/Energiestatus%20%C3%96sterreich%202015.pdf>
- [2] D. J. Evans and R. A. Chadwick, *Underground gas storage: Worldwide experiences and future development in the UK and Europe*, ser. Geological Society special publication. London: Geological Society, 2009, vol. no. 313.
- [3] LBEG, “Untertage-gasspeicherung in deutschland: Underground gas storage in germany,” *ERDÖL ERDGAS KOHLE*, vol. 131, no. 11, pp. 398–406, 2015.
- [4] M. J. Economides and X. Wang, “A modern approach to optimizing underground natural gas storage,” in *SPE Annual Technical Conference and Exhibition*, 2013-09-30.
- [5] J. H. Henderson, J. R. Dempsey, and J. C. Tyler, “Use of numerical models to develop and operate gas storage reservoirs,” *Journal of Petroleum Technology*, vol. 20, no. 11, pp. 1239–1246, 1968.
- [6] R. A. Wattenbarger, “Maximizing seasonal withdrawals from gas storage reservoirs,” *Journal of Petroleum Technology*, vol. 22, no. 08, pp. 994–998, 1970.
- [7] D. A. McVay and J. P. Spivey, “Optimizing gas-storage reservoir performance,” *SPE Reservoir Evaluation & Engineering*, vol. 4, no. 03, pp. 173–178, 2001.
- [8] G. Zangl, M. Giovannoli, and M. Stundner, “Application of artificial intelligence in gas storage management,” in *SPE Europec/EAGE Annual Conference and Exhibition*, 2006-06-12.
- [9] K. G. Brown and D. E. Meikle, “The value of wellhead electronic flow measurement in gas storage fields,” in *SPE Eastern Regional Meeting*, 1995-09-18.
- [10] A. Tiani, G. Bartolotto, P. Strippoli, R. Latronico, and G. Spitaleri, “From real time data to updated models: The challenge of intelligent fields applied to gas storage business,” in *Europec/EAGE Conference and Exhibition*, 2008-06-09.
- [11] M. J. Economides, *Petroleum production systems*, 2nd ed. Upper Saddle River NJ u.a.: Pearson Education Inc, 2013.
- [12] J. V. Vogel, “Inflow performance relationships for solution-gas drive wells,” *Journal of Petroleum Technology*, vol. 20, no. 01, pp. 83–92, 1968.
- [13] M. J. Fetkovich, “The isochronal testing of oil wells,” in *Fall Meeting of the Society of Petroleum Engineers of AIME*, 1973-09-30.

- [14] E. Rawlins and M. Schellhardt, *Backpressure Data on Natural Gas Wells and Their Application to Production Practices*, ser. Monograph Series. Baltimore, Maryland: US Bureau of Mines, 1935, vol. 7.
- [15] L. W. Lake and J. R. Fanchi, *Petroleum Engineering Handbook: Production Operations Engineering*. Richardson, TX: Society of Petroleum Engineers, 2007.
- [16] M. Kelkar, *Natural gas production engineering*. Tulsa, Oklahoma: PennWell, 2008. [Online]. Available: <http://www.loc.gov/catdir/enhancements/fy0901/2007046605-b.html>
- [17] R. Al-Hussainy, H. J. Ramey, and P. B. Crawford, “The flow of real gases through porous media,” *Journal of Petroleum Technology*, vol. 18, no. 05, pp. 624–636, 1966.
- [18] X. Wang and M. J. Economides, “Horizontal well deliverability with turbulence effects,” in *8th European Formation Damage Conference*, 2009-05-27.
- [19] S. R. Bergin and Y. A. Shikari, “A horizontal well in gas storage: A case study,” in *SPE Gas Technology Symposium*, 1993-06-28.
- [20] X. Wang and M. J. Economides, *Advanced natural gas engineering*. Houston, Tex: Gulf Pub. Co, 2009. [Online]. Available: <http://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=655315>
- [21] D. K. Babu and A. S. Odeh, “Productivity of a horizontal well (includes associated papers 20306, 20307, 20394, 20403, 20799, 21307, 21610, 21611, 21623, 21624, 25295, 25408, 26262, 26281, 31025, and 31035),” *SPE Reservoir Engineering*, vol. 4, no. 04, pp. 417–421, 1989.
- [22] J. K. Pucknell and P. J. Clifford, “Calculation of total skin factors,” in *Offshore Europe*, 1991-09-03.
- [23] M. R. Tek, K. H. Coats, and D. L. Katz, “The effect of turbulence on flow of natural gas through porous reservoirs,” *Journal of Petroleum Technology*, vol. 14, no. 07, pp. 799–806, 1962.
- [24] A. Houpeurt, “On the flow of gases in porous media,” *Revue de L’Institut Francais du Petrole*, vol. XIV, no. 11, pp. 1468–1684, 1959.
- [25] L. W. Lake and J. R. Fanchi, *Petroleum Engineering Handbook: Reservoir Engineering and Petrophysics*. Richardson, TX: Society of Petroleum Engineers, 2007.
- [26] T. Billiter, J. Lee, and R. Chase, “Dimensionless inflow-performance-relationship curve for unfractured horizontal gas wells,” in *SPE Eastern Regional Meeting*, 2001-10-17.
- [27] F. Morrison, *An introduction to fluid mechanics*. Cambridge: Cambridge Univ. Press, 2013. [Online]. Available: <http://gbv.ebilib.com/patron/FullRecord.aspx?p=1099833>
- [28] K. L. Young, “Effect of assumptions used to calculate bottom-hole pressures in gas wells,” *Journal of Petroleum Technology*, vol. 19, no. 04, pp. 547–550, 1967.

- [29] R. D. Oden and J. W. Jennings, “Modification of the cullender and smith equation for more accurate bottomhole pressure calculations in gas wells,” in *Permian Basin Oil and Gas Recovery Conference*, 1988-03-10.
- [30] C. M. Rendeiro and C. M. Kelso, “An investigation to improve the accuracy of calculating bottomhole pressures in flowing gas wells producing liquids,” in *Permian Basin Oil and Gas Recovery Conference*, 1988-03-10.
- [31] J. Mach, E. Proano, and K. Brown, “A nodal approach for applying systems analysis to the flowing and artificial lift oil or gas well,” *Society of Petroleum Engineers*, 1979.
- [32] K. E. Brown and J. F. Lea, “Nodal systems analysis of oil and gas wells,” *Journal of Petroleum Technology*, vol. 37, no. 10, pp. 1751–1763, 1985.
- [33] J. Bellarby, *Well completion design*, reprinted. ed., ser. Developments in petroleum science. Amsterdam: Elsevier, 2010, vol. 56.
- [34] A. S. Odeh, “Reservoir simulation ...what is it,” *Journal of Petroleum Technology*, vol. 21, no. 11, pp. 1383–1388, 1969.
- [35] K. H. Coats, “A note on impes and some impes-based simulation models,” *SPE Journal*, vol. 5, no. 03, pp. 245–251, 2000.
- [36] A. G. Spillette, J. G. Hillestad, and H. L. Stone, “A high-stability sequential solution approach to reservoir simulation,” in *Fall Meeting of the Society of Petroleum Engineers of AIME*, 1973-09-30.
- [37] M. Schäfer, *Computational Engineering - Introduction to Numerical Methods*. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, 2006.
- [38] A. F. van Everdingen, “The skin effect and its influence on the productive capacity of a well,” *Journal of Petroleum Technology*, vol. 5, no. 06, pp. 171–176, 1953.
- [39] R. Gladfelter, G. Tracy, and L. Wilsey, “Selecting wells which will respond to production-stimulation treatment,” in *Drilling and Production Practice*, 1955.
- [40] H. J. Ramey, “Non-darcy flow and wellbore storage effects in pressure build-up and drawdown of gas wells,” *Journal of Petroleum Technology*, vol. 17, no. 02, pp. 223–233, 1965.
- [41] R. C. Earlougher, *Advances in well test analysis*, ser. SPE monograph series. Richardson, Tex.: Henry L. Doherty Memorial Fund of AIME Society of Petroleum Engineers, 1977, vol. 5.
- [42] R. G. Agarwal, R. Al-Hussainy, and H. J. Ramey, “An investigation of wellbore storage and skin effect in unsteady liquid flow: I. analytical treatment,” *Society of Petroleum Engineers Journal*, vol. 10, no. 03, pp. 279–290, 1970.

- [43] P. E. Oren, R. L. Lee, and M. R. Tek, “The effects of wellbore storage, skin, and turbulence intensity on early-time transient flow of real gas through porous media,” *SPE Formation Evaluation*, vol. 3, no. 03, pp. 547–554, 1988.
- [44] R. L. Lee, R. W. Logan, and M. R. Tek, “Effect of turbulence on transient flow of real gas through porous media,” *SPE Formation Evaluation*, vol. 2, no. 01, pp. 108–120, 1987.
- [45] S. Haykin, *Neural networks and learning machines*, 3rd ed. New York: Pearson, 2009.
- [46] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. de Jesús, *Neural network design*, 2nd ed. ebook, 2014.
- [47] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, “Efficient backprop,” in *Neural Networks: Tricks of the Trade*, ser. Lecture Notes in Computer Science, G. B. Orr and K.-R. Müller, Eds. Berlin and Heidelberg: Springer, 1998, vol. 1524, pp. 9–50.
- [48] L. Breiman, “Bagging predictors,” *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.
- [49] Y. Chauvin and D. E. Rumelhart, *Backpropagation: Theory, architectures, and applications*, ser. Developments in connectionist theory. Hillsdale, N.J.: Lawrence Erlbaum Associates, 1995.
- [50] Z. P. Lo, Y. Yu, and B. Bavarian, “Analysis of the convergence properties of topology preserving neural networks,” *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, vol. 4, no. 2, pp. 207–220, 1993.
- [51] J. L. Elman, “Finding structure in time,” *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.
- [52] H. T. Siegelmann, B. G. Horne, and C. L. Giles, “Computational capabilities of recurrent narx neural networks,” *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol. 27, no. 2, pp. 208–215, 1997.
- [53] Tsungnan Lin, B. G. Horne, P. Tino, and C. L. Giles, “Learning long-term dependencies in narx recurrent neural networks,” *IEEE Transactions on Neural Networks*, vol. 7, no. 6, pp. 1329–1338, 1996.
- [54] P. J. Werbos, “Backpropagation through time: What it does and how to do it,” *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.
- [55] H. Jaeger, *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the ‘echo state network’ approach*. GMD Report 159: German National Research Center for Information Technology, 2002.
- [56] —, “Echo state network,” *Scholarpedia*, vol. 2, no. 9, p. 2330, 2007.
- [57] —, “The ‘echo state’ approach to analysing and training recurrent neural networks: Gmd report 148,” *GMD - German National Research Institute for Computer Science*, 2001.

- [58] J. W. Tukey, *Exploratory data analysis*, ser. Addison-Wesley series in behavioral science Quantitative methods. Reading, Mass.: Addison-Wesley, 1997.
- [59] B. Illowsky and S. Dean, *Introductory Statistics*, Rice University, 2015. [Online]. Available: <https://openstaxcollege.org/textbooks/introductory-statistics>
- [60] Wikipedia, “Antialiasing (signalverarbeitung),” 01.08.2016. [Online]. Available: <https://de.wikipedia.org/w/index.php?oldid=155985753>
- [61] B. Muralikrishnan and J. Raja, *Computational Surface and Roundness Metrology*. London: Springer-Verlag, 2009. [Online]. Available: <http://dx.doi.org/10.1007/978-1-84800-297-5>
- [62] R. Stewart, “Median filtering: Review and a new f/k analogue design,” *J. Can. Soc. Expl. Geophys.*, vol. 21, pp. 54–63, 1985.
- [63] A. W. Bowman and A. Azzalini, *Applied smoothing techniques for data analysis: The kernel approach with S-Plus illustrations*, ser. Oxford statistical science series. Oxford: Clarendon Press, 1997, vol. 18.
- [64] H. Jaeger, “Simple and very simple matlab toolbox for echo state networks |,” 07.10.2016. [Online]. Available: <http://reservoir-computing.org/node/129>
- [65] K. Levenberg, “A method for the solution of certain non-linear problems in least square,” *Quarterly of Applied Mathematics*, vol. 2, pp. 164–168, 1944.
- [66] D. W. Marquardt, “An algorithm for least-squares estimation of nonlinear parameters,” *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.
- [67] M. T. Hagan and M. B. Menhaj, “Training feedforward networks with the marquardt algorithm,” *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, vol. 5, no. 6, pp. 989–993, 1994.
- [68] M. Lukoševičius, “A practical guide to applying echo state networks,” in *Neural Networks: Tricks of the Trade*, ser. Lecture Notes in Computer Science, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, G. Montavon, G. B. Orr, and K.-R. Müller, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, vol. 7700, pp. 659–686.
- [69] Wikipedia, “Root-mean-square deviation,” 18.09.2016. [Online]. Available: <https://en.wikipedia.org/w/index.php?oldid=731675441>
- [70] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [71] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” *arXiv preprint arXiv:1412.3555*, 2014.

-
- [72] Wikipedia, “Moody chart,” 28.09.2016. [Online]. Available: <https://en.wikipedia.org/w/index.php?oldid=706339930>

Appendices

Appendix A

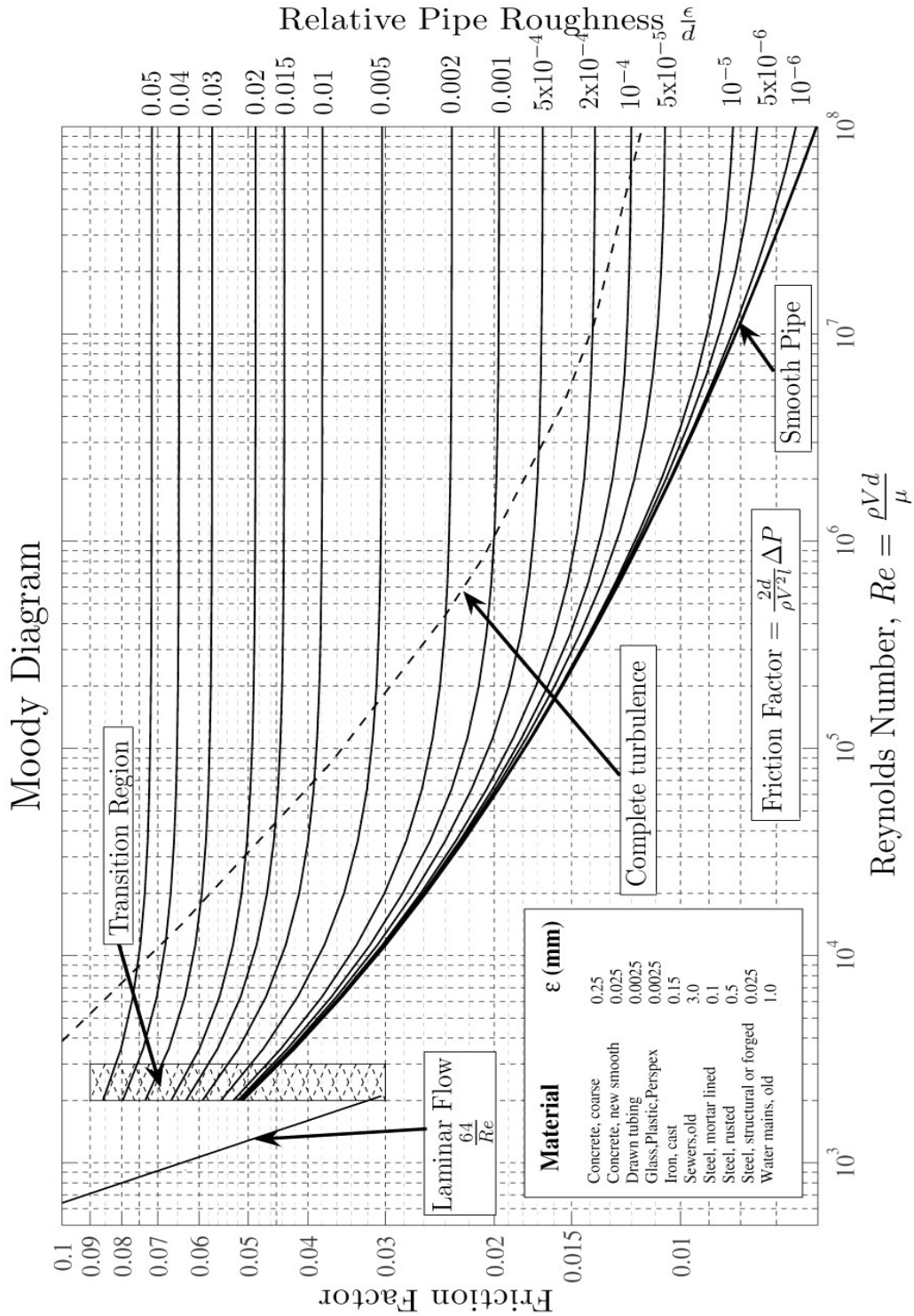


Figure A1: Moody diagram showing the Darcy-Weisbach friction factor f_D plotted against the Reynolds number N_{Re} [72]

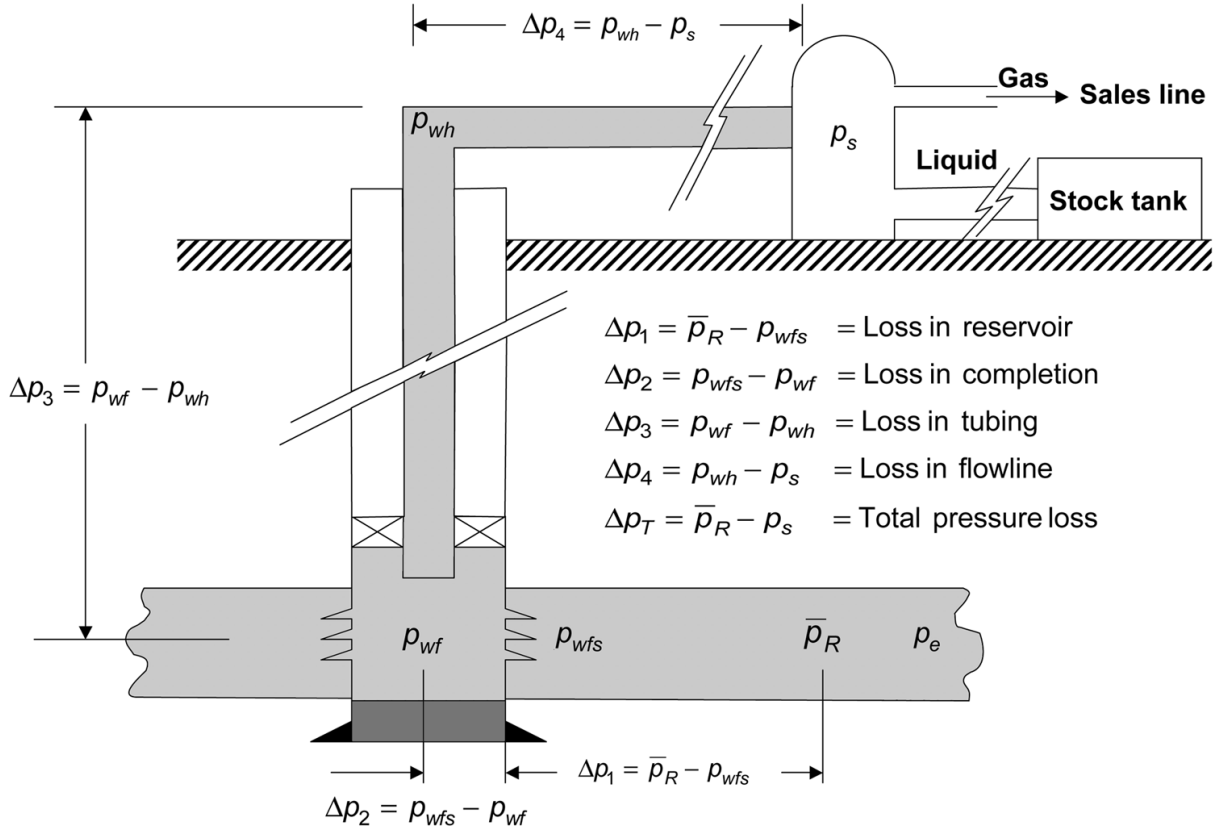


Figure A2: Overview of possible nodes used in Nodal Analysis and pressure losses in the complete system [31, p. 21]

Dimensionless Pseudo Variables

To investigate the influence of turbulence, skin, wellbore storage and variations in the gas properties required a generalization of the problem. In order to analyse the early-time transient flow in gas wells a dedimensionalizing procedure applied to all variables was necessary. This resulted in the following set of dimensionless pseudo variables [43, p. 547-554].

$$C_D = \frac{V_w}{2\pi h \phi r_w^2} \quad (\text{A.1})$$

$$t_{pD} = F_{tpD} \cdot t_D \quad (\text{A.2})$$

$$F_{tpD} = \left(\int_0^t \frac{1}{\mu c} dt \right) / \left(\frac{t}{\mu_i c_i} \right) \quad (\text{A.3})$$

$$t_D = 2.637 \cdot 10^{-4} \frac{k_r t}{\phi r_w^2 \mu_i c_i} \quad (\text{A.4})$$

$$s_D = \frac{kh \Delta p_s}{141.2 \mu q B} \quad (\text{A.5})$$

$$N_{TD} = 1.564 \cdot 10^{-18} \frac{k_r^2 \beta \gamma_g p_{pi}}{T \mu_i r_w} \quad (\text{A.6})$$

$$q_{scD} = 1.422 \cdot 10^6 \frac{T q_{sc}}{k_r h p_{pi}} \quad (\text{A.7})$$

c	Compressibility [kPa ⁻¹]
c_i	Initial Compressibility [kPa ⁻¹]
C_D	Dimensionless Wellbore Storage Constant [-]
F_{tpD}	Pseudotime Ratio [-]
h	Formation Thickness [m]
k	Permeability [md]
k_r	Permeability in Radial Direction [md]
N_{TD}	Turbulence Intensity Number [-]
p_{pi}	Initial Pseudopressure [-]
q_{sc}	Standard Condition Flow rate [m ³ /d]
q_{scD}	Dimensionless Flow rate [-]
q_{scf}	Formation Flow rate [m ³ /d]
r_w	Wellbore Radius [m]
t_d	Dimensionless Time [-]
t_{pd}	Dimensionless Pseudotime [-]
T	Temperature [K]
V_w	Wellbore Volume [m ³]
β	Turbulence Coefficient [1/m]
μ	Viscosity [kPa · s]
μ_i	Initial Viscosity [kPa · s]










Name	Input/Output Relation	Icon	MATLAB Function
Hard Limit	$a = 0 \quad n < 0$ $a = 1 \quad n \geq 0$		hardlim
Symmetrical Hard Limit	$a = -1 \quad n < 0$ $a = +1 \quad n \geq 0$		hardlims
Linear	$a = n$		purelin
Saturating Linear	$a = 0 \quad n < 0$ $a = n \quad 0 \leq n \leq 1$ $a = 1 \quad n > 1$		satlin
Symmetric Saturating Linear	$a = -1 \quad n < -1$ $a = n \quad -1 \leq n \leq 1$ $a = 1 \quad n > 1$		satlins
Log-Sigmoid	$a = \frac{1}{1 + e^{-n}}$		logsig
Hyperbolic Tangent Sigmoid	$a = \frac{e^n - e^{-n}}{e^n + e^{-n}}$		tansig
Positive Linear	$a = 0 \quad n < 0$ $a = n \quad 0 \leq n$		poslin
Competitive	$a = 1 \quad \text{neuron with max } n$ $a = 0 \quad \text{all other neurons}$		compet

Figure A3: Overview of common ANN transfer functions and their mathematical formulation [46, p. 2-6]

Appendix B

Resamp.m

```
1 % Resampling using a Kernelsmooter or linear Interpolation und Filtering
2 % 1 Method: -Resampling with linear Interpolation
3 %           -Filtering with Gaussian filter & Median filter
4 %
5 % 2 Method: -Resampling with Kernelsmooter
6 %           -Combines resampling and filtering in one function
7 % Requires the "Kernel Smoothing Regression" from Mathworks File Exchange
8 % -----%
9
10 %% Resampling Interval & Method
11 clc
12 prompt={'Resampling Interval' , 'Resampling Method'};
13 dlg_title='Selection of Resampling Interval and Method';
14 % Resampling Interval entered in Minutes
15 num_lines=1;
16 defaultans={'10', 'Linear'};
17 answer=inputdlg(prompt,dlg_title,num_lines,defaultans,'on');
18
19 T_Samp=str2double(answer{1});
20 Method=answer{2};
21
22 %% Loading of raw data
23 load('Daten2014-2016_modified.mat');
24
25 %% Resampling & Filtering
26
27 % Checks which Method was selected
28 if strcmp(answer{2}, 'Linear')
29     Resampling_Pressure;
30     Resampling_Rate;
31 else
32
33     T=Well11_t(1):(T_Samp/1440):Well11_t(end);
34     T=T';
35     Length=length(T);
36     Resampled_TimeVector=datetime(T, 'ConvertFrom', 'datenum');
37
38     % Selection of bandwidth for kernelsmooter
39     % Greater bandwidth -> Smoother output
40     % Smaller bandwidth -> More details preserved
41
42     prompt='Select bandwidth for Kernelsmooter: \n';
43     Bandwidth=input(prompt);
44
45     % Kernelsmooter for pressure
```



```

46     Kern_THP;
47
48     % Kernelsmoother for flow rates % filling level
49     Kern_Rates;
50 end
51
52 %% Data Storage
53 % Compiles resampled data in a MATLAB array and table
54 % Saves resampled data in a .csv file
55
56 WriteTable;
57 %-----%
58 %% Cleaning up workspace
59
60 clear prompt answer defaultans dlg_title Length Bandwidth T_Samp g ...
61     Method Modus num_lines;

```

Linear Resampling Pressure.m

```

1  %% New Time Vector
2  %New Sampling Interval in seconds
3  T_Samp1=10;
4  T1=Well1_t(1):(T_Samp1/86400):Well1_t(end);
5  T1=T1';
6
7  %New Sampling Interval in Minutes
8  T_aux=60*T_Samp;
9  T=Well1_t(1):(T_aux/86400):Well1_t(end);
10 T=T';
11 Resampled_TimeVector=datetime(T, 'ConvertFrom', 'datenum');
12
13 %% 1D-Gaussfilter
14 % Selection of Filter Width
15 % Filter width should correspond with resampling interval
16 % Greater window width -> More smoothing
17 % Smaller window width -> Less smoothing
18 %-----%
19
20 %% Filtering Config
21 % Input dialog for Window width
22 prompt={'Window Width'};
23 dlg_title='Configuration of Gaussian filter Window Width';
24 num_lines=1;
25 defaultans={'120'};
26 answer=inputdlg(prompt,dlg_title,num_lines,defaultans,'on');
27
28 Window=str2double(answer{1});

```

```

29 g=gausswin(Window);
30 g=g/sum(g);
31
32 %% Upsampling
33 yWell1_THP=interp1q(Well1_t,Well1_THP,T1);
34 yWell2_THP=interp1q(Well2_t,Well2_THP,T1);
35 yWell3_THP=interp1q(Well3_t,Well3_THP,T1);
36 yWell4_THP=interp1q(Well4_t,Well4_THP,T1);
37 yWell5_THP=interp1q(Well5_t,Well5_THP,T1);
38 yWell6_THP=interp1q(Well6_t,Well6_THP,T1);
39 yOWell1_THP=interp1q(OWell1_t,OWell1_THP,T1);
40
41 %% Filtering
42 Well1_THP_filtered=filtfilt(g,1,yWell1_THP);
43 Well2_THP_filtered=filtfilt(g,1,yWell2_THP);
44 Well3_THP_filtered=filtfilt(g,1,yWell3_THP);
45 Well4_THP_filtered=filtfilt(g,1,yWell4_THP);
46 Well5_THP_filtered=filtfilt(g,1,yWell5_THP);
47 Well6_THP_filtered=filtfilt(g,1,yWell6_THP);
48 OWell1_THP_filtered=filtfilt(g,1,yOWell1_THP);
49
50 %% Resampling
51 yWell1_THP_filtered=interp1q(T1,Well1_THP_filtered,T);
52 yWell2_THP_filtered=interp1q(T1,Well2_THP_filtered,T);
53 yWell3_THP_filtered=interp1q(T1,Well3_THP_filtered,T);
54 yWell4_THP_filtered=interp1q(T1,Well4_THP_filtered,T);
55 yWell5_THP_filtered=interp1q(T1,Well5_THP_filtered,T);
56 yWell6_THP_filtered=interp1q(T1,Well6_THP_filtered,T);
57 yOWell1_THP_filtered=interp1q(T1,OWell1_THP_filtered,T);
58 yFuellstand_filtered=interp1q(Fuellstand_t,Fuellstand,T);
59
60 clear Well1_THP_filtered Well3_THP_filtered Well2_THP_filtered ...
61     Well4_THP_filtered Well6_THP_filtered OWell1_THP_filtered ...
62     Well5_THP_filtered Window g prompt answer dlg_title defaultans ...
63     num_lines;

```

Linear Resampling Rates.m

```

1 %% New Time Vector
2 %New Sampling Interval in Seconds
3 T_Samp1=10;
4 T1=Well1_t(1):(T_Samp1/86400):Well1_t(end);
5 T1=T1';
6
7 %New Sampling Interval in Minutes
8 T_aux=60*T_Samp;
9 T=Well1_t(1):(T_aux/86400):Well1_t(end);

```

```

10 T=T';
11 Resampled_TimeVector=datetime(T, 'ConvertFrom', 'datenum');
12
13 %% Upsampling
14 % Linear upsampling to smaller intervals
15 yWell1_Rate=interp1q(Well1_t,Well1_Rate,T1);
16 yWell2_Rate=interp1q(Well2_t,Well2_Rate,T1);
17 yWell3_Rate=interp1q(Well3_t,Well3_Rate,T1);
18 yWell4_Rate=interp1q(Well4_t,Well4_Rate,T1);
19 yWell5_Rate=interp1q(Well5_t,Well5_Rate,T1);
20 yWell6_Rate=interp1q(Well6_t,Well6_Rate,T1);
21
22 %% Filtering Config
23 % Input dialog for Window width
24 prompt={'Window Width'};
25 dlg_title='Configuration of Filter Window Width';
26 num_lines=1;
27 defaultans={'120'};
28 answer=inputdlg(prompt,dlg_title,num_lines,defaultans,'on');
29
30 Window=str2double(answer{1});
31
32 %% Filtering
33 yWell1_Rate_filtered=medfilt1(yWell1_Rate,Window,'truncate');
34 yWell2_Rate_filtered=medfilt1(yWell2_Rate,Window,'truncate');
35 yWell3_Rate_filtered=medfilt1(yWell3_Rate,Window,'truncate');
36 yWell4_Rate_filtered=medfilt1(yWell4_Rate,Window,'truncate');
37 yWell5_Rate_filtered=medfilt1(yWell5_Rate,Window,'truncate');
38 yWell6_Rate_filtered=medfilt1(yWell6_Rate,Window,'truncate');
39
40 %% Downsampling
41 yWell1_Rate_filtered=interp1q(T1,yWell1_Rate_filtered,T);
42 yWell2_Rate_filtered=interp1q(T1,yWell2_Rate_filtered,T);
43 yWell3_Rate_filtered=interp1q(T1,yWell3_Rate_filtered,T);
44 yWell4_Rate_filtered=interp1q(T1,yWell4_Rate_filtered,T);
45 yWell5_Rate_filtered=interp1q(T1,yWell5_Rate_filtered,T);
46 yWell6_Rate_filtered=interp1q(T1,yWell6_Rate_filtered,T);
47
48 clear yWell1_Rate yWell2_Rate yWell3_Rate yWell4_Rate ...
49     yWell5_Rate yWell6_Rate T_Samp1 T1 Window prompt dlg_title num_lines ...
50     answer defaultans;

```

Kernel THP.m

```

1 % Kernelsmoothing for pressure
2 % Bandwidth & Length are transferred from Resampling.m
3

```

```

4  %-----%
5  Test=ksr(Well1_t,Well1_THP,Bandwidth,Length);
6  T=Test.x;
7  yWell1_THP_filtered=Test.f;
8
9  Test=ksr(Well2_t,Well2_THP,Bandwidth,Length);
10 yWell2_THP_filtered=Test.f;
11
12 Test=ksr(Well3_t,Well3_THP,Bandwidth,Length);
13 yWell3_THP_filtered=Test.f;
14
15 Test=ksr(Well4_t,Well4_THP,Bandwidth,Length);
16 yWell4_THP_filtered=Test.f;
17
18 Test=ksr(Well5_t,Well5_THP,Bandwidth,Length);
19 yWell5_THP_filtered=Test.f;
20
21 Test=ksr(Well6_t,Well6_THP,Bandwidth,Length);
22 yWell6_THP_filtered=Test.f;
23
24 Test=ksr(OWell1_t(1411:end,1),...
25     OWell1_THP(1411:end,1),Bandwidth,Length);
26 yOWell1_THP_filtered=Test.f;
27
28 clear Test;

```

Kernel Rates.m

```

1  % Kernelsmoothing for Flow rates & Filling level
2  % Bandwidth & Length are transferred from Resampling.m
3
4  %-----%
5  Test1=ksr(Well1_t,Well1_Rate,Bandwidth,Length);
6  T=Test1.x;
7  yWell1_Rate_filtered=Test1.f;
8
9  Test1=ksr(Well2_t,Well2_Rate,Bandwidth,Length);
10 yWell2_Rate_filtered=Test1.f;
11
12 Test1=ksr(Well3_t,Well3_Rate,Bandwidth,Length);
13 yWell3_Rate_filtered=Test1.f;
14
15 Test1=ksr(Well4_t,Well4_Rate,Bandwidth,Length);
16 yWell4_Rate_filtered=Test1.f;
17
18 Test1=ksr(Well5_t,Well5_Rate,Bandwidth,Length);
19 yWell5_Rate_filtered=Test1.f;

```

```
20
21 Test1=ksr(Well6_t,Well6_Rate,Bandwidth,Length);
22 yWell6_Rate_filtered=Test1.f;
23
24 Test1=ksr(Fuellstand_t(1411:end,1),...
25     Fuellstand(1411:end,1),Bandwidth,Length);
26 yFuellstand_filtered=Test1.f;
27
28 clear Test1;
```

Appendix C

TrainNarx.m

```

1 % Training of NARX Neural Networks
2 % This script allows specifying time intervals for training and testing of
3 % these NARX networks.
4 % Input dialogue for:
5 % - Number of networks in a network generation
6 % - The maximum and minimum number of hidden neurons,
7 % - The number of delays in the tapped delay line of the input and output
8 % - Random seed value
9 % - Number of training epochs
10 %-----%
11 %% Checks if data sequence is loaded in workspace
12 % Loads data into workspace from .mat file if not already available.
13
14 if exist('Daten_Resampled_Table','var') == 0
15     uiopen('matlab');
16 end
17
18 %% Loading of specified time interval for training
19 clc
20 prompt={'Start Date:', 'End Date:'};
21 dlg_title='Timerange of Network Input';
22 num_lines=1;
23 defaultans={'01-Mar-2014', '15-May-2016'};
24 answer=inputdlg(prompt,dlg_title,num_lines,defaultans);
25
26 Predict_Start=answer{2};
27
28 % TimeRange is selfmade MATLAB function which extracts the specified time
29 % interval from the complete data sequence
30 [InputMat_Narx,TargetMat_Narx,TimeVector_Narx]=TimeRange1(answer{1},...
31     answer{2},Daten_Resampled_Table);
32
33
34 % Conversion of MATLAB Array to neural network input.
35 Input=tonndata(InputMat_Narx,false,false);
36 Target=tonndata(TargetMat_Narx,false,false);
37
38 %% Configuration of networks
39     prompt={'Number of Networks','Min Number of hidden Neurons',...
40         'Max Number of hidden Neurons','Hidden Neurons Stepsize',...
41         'Input Delays','Output Delays','Training epochs','Random Seed'};
42     dlg_title='Expert Network Options';
43     num_lines=1;
44     defaultans={'10','14','14','2','3','4','25','666'};
45     answer=inputdlg(prompt,dlg_title,num_lines,defaultans);

```

```

46
47     % Specification of network parameters
48     numNN = str2double(answer(1));      % Number of networks in generation
49     Hmax = str2double(answer{3});      % Max hidden neurons
50     Hmin = str2double(answer{2});      % Min hidden neurons
51     dH = str2double(answer{4});        % Stepsize hidden neurons
52     delayi=str2double(answer{5});      % Input delays
53     delayt_Narx=str2double(answer{6}); % Target delays
54     NumEpoch=str2double(answer{7});   % Number of training epochs
55     RandomSeed=str2double(answer{8});  % Random seed value
56
57
58 %% Training of networks
59
60     HiddenNodes_Forloop_NARX
61
62 %% Training Stats
63 % Compilation of a MATLAB table of training records
64
65     TrainStats
66     openvar('Stats')
67
68 %% Plotting of results
69 % Results of the best trained network and the an average of all network
70 % outputs is plotted
71 % Input dialogue asking whether plots are required
72
73 prompt={'Sollen Ergebnisse geplottet werden Ja oder Nein'};
74 dlg_title='Plotten der Ergebnisse';
75 num_lines=1;
76 defaultans={'Ja'};
77 answer=inputdlg(prompt,dlg_title,num_lines,defaultans);
78
79 if strcmp(answer{1}, 'Ja')
80     load 'SondenNamen.mat' % Loading of well names for plot
81     NumTarget=length(TargetMat_Narx(1,:));
82
83     [~,ind]=min(Best_Network_Performance_Narx,[],1);
84     delay=max(delayt_Narx,delayi);
85
86     Plot_HM_PR(TimeVector_Narx,TargetMat_Narx,Average_Narx,...
87         Best_Narx,NumTarget,delay,Sonden,ind);
88 end
89
90 %% Cleaning workspace
91 clear answer Best1 defaultans dlg_title y2Av y2 y2c y2Avc y2AverageOutput ...
92     y2AverageOutputc y2Total y2Totalc ans E1 E2 i j k netci neti netos ...
93     Separation s timing delayt ind

```

HiddenForloopNarx.m

```
1 %% Network Initialization
2 clc
3
4 % The following variables in comment are delivered by TrainNarx.m
5 % They can be used to test the script without using TrainNarx.m
6
7 %numNN=5;    %Number of Networks
8 %NumEpoch=25;%Number of Training Epochs
9 %delayi=3;   %Input Delays
10 %delayt=4;  %Feedback Delays
11 %Hmax=14;   %Maximum Hidden Neurons
12 %Hmin=10;   %Minimum Hidden Neurons
13 %dH=2;      %Hidden Neurons Stepsize
14 Hcount=1+(Hmax-Hmin)/dH;
15 j=1;        %Internal Count
16 NumTarget=numel(Target{1});
17 delayt=delayt_Narx;
18
19 % Multiplier for Number of training epochs:
20 Multi=14;   % Standard = 5
21
22 %Preallocation of Matrices
23 perfs_closed=zeros(Hcount,numNN);
24 perfs_closed_trained = zeros(Hcount,numNN);
25 perfAverageOutput=zeros(Hcount,1);
26 perfAverageOutputc=zeros(Hcount,1);
27 Best_Network_Performance_Narx=zeros(Hcount,1);
28 Index_Best_Narx=zeros(Hcount,1);
29 nets_Narx=cell(Hcount,numNN);
30 netcs_Narx=cell(Hcount,numNN);
31 Tr_closed=cell(Hcount,numNN);
32 delay=max(delayt,delayi);
33 Best_Narx=zeros(Hcount*NumTarget,(length(Target)-delay));
34 Average_Narx=zeros(Hcount*NumTarget,(length(Target)-delay));
35 RandomSeed_Stored=cell(Hcount,numNN);
36 Separation=zeros(Hcount,1);
37 Hidden=zeros(Hcount,1);
38
39 %% Forloop over number of hidden neurons
40 % Generates a network generation with a specified number of neurons in the
41 % hidden layer
42 % All other network parameters stay the same
43
44 for H=Hmin:dH:Hmax
45 Hidden(j)=H;
46 netos=narxnet(1:delayi,1:delayt,H);
47 netos.trainFcn='trainlm';
```



```
48 netos.trainParam.epochs=NumEpoch;
49 netos.divideFcn='divideint';
50 [xo,Xio,Aio,to]=preparets(netos,Input, {},Target);
51
52 %% Network train in open loop
53 % Trains all networks in a single network generation
54
55 for i=1:numNN
56
57     if RandomSeed~=0        % Checks if user defined random seed value exists
58         rng(RandomSeed+i);% If it exists it is used for training
59     end
60
61     nets_Narx{j,i}=init(netos);
62     nets_Narx{j,i}=train(netos,xo,to,Xio,Aio);
63     netcs_Narx{j,i}=closeloop(nets_Narx{j,i});
64     RandomSeed_Stored{j,i}=rng;
65 end
66
67 %% Performance Evaluation
68 % Evaluates the performance of all networks after training in NARX open
69 % loop
70
71 y2Total=0;
72 [xc, Xic, Aic, tc]=preparets(netcs_Narx{j,1},Input, {},Target);
73
74 for i=1:numNN
75     neti=netcs_Narx{j,i};
76     y2=neti(xc,Xic,Aic);
77     perfs_closed(j,i)=mse(tc,y2);
78     y2=cell2mat(y2);
79     y2Total=y2Total+y2;
80 end
81
82
83 y2AverageOutput=y2Total/numNN;
84 y2Av=num2cell(y2AverageOutput);
85 E1=gsubtract(y2Av,tc);
86 perfAverageOutput(j)=mse(E1);
87
88 %% Closed loop training
89 % Converts all networks to the NARX closed loop representation
90 % Retrains all networks in the closed loop
91
92 for i=1:numNN
93
94     if RandomSeed~=0        % Checks if user defined random seed value exists
95         rng(RandomSeed+i);% If it exists it is used for training
96     end
97
```

```

98     netcs_Narx{j,i}.trainParam.showWindow=1;
99     netcs_Narx{j,i}.trainFcn='trainlm';
100    netcs_Narx{j,i}.trainParam.epochs=Multi*NumEpoch;
101    [netcs_Narx{j,i},Tr_closed{j,i}]=train(netcs_Narx{j,i},xc,tc,Xic,Aic);
102 end
103
104 %% Closed Loop Performance Evaluation
105 % Evaluates the performance of all networks after training in NARX closed
106 % loop
107
108 y2Totalc=0;
109 for i=1:numNN
110     netci=netcs_Narx{j,i};
111     y2c=netci(xc,Xic,Aic);
112     perfs_closed_trained(j,i)=mse(tc,y2c);
113     y2c=cell2mat(y2c);
114     y2Totalc=y2Totalc + y2c;
115 end
116
117 y2AverageOutputc=y2Totalc/numNN;
118 y2Avc=num2cell(y2AverageOutputc);
119 E2=gsubtract(y2Avc,tc);
120 perfAverageOutputc(j)=mse(E2);
121
122 [Best_Network_Performance_Narx(j,1),Index_Best_Narx(j,1)]=...
123     min(perfs_closed_trained(j,:));
124
125 k=(j-1)*NumTarget+1;
126 Best1=(netcs_Narx{j,Index_Best_Narx(j,1)}(xc,Xic,Aic));
127 Best_Narx((k:k+(NumTarget-1)),:)=cell2mat(Best1);
128 Average_Narx((k:k+(NumTarget-1)),:)=y2AverageOutputc;
129 j=j+1;
130
131 %% Checkpoint
132 % Saves a checkpoint file after a network generation is trained in open &
133 % closed loop
134
135 if mod(j,1)==0
136     s=rng;
137     tic
138     save('Checkpoint_tmp.mat');
139     if strcmp(computer,'PCWIN64') || strcmp(computer,'PCWIN')
140         %We are running on a windows machine
141         system('move /y checkpoint_tmp.mat checkpoint.mat');
142     else
143         %We are running on Linux or Mac
144         system('mv checkpoint_tmp.mat checkpoint.mat');
145     end
146     timing = toc;
147     fprintf('Checkpoint save took %f seconds\n',timing);

```

```

148
149 end
150 end
151 %% Summary
152 % Summary file of network performances for all generations
153 % This file is used by other scripts can be stored as MATLAB Array
154 summary=[Hidden, Separation, perfs_closed, perfAverageOutput, Separation, ...
155         perfs_closed_trained, perfAverageOutputc, Separation, ...
156         Best_Network_Performance_Narx, Index_Best_Narx];

```

TestNarx.m

```

1 % This script allows using trained networks to predict outputs for unseen
2 % data sequences
3 % It also enables computing the error of the network response to unseen
4 % data sequences
5
6 % An input dialogue can be used to select a time interval for testing and
7 % prediction from the whole data sequence
8
9 % Individual networks can be selected via a drop down menu
10 % An output for the selected network and an average of all networks in the
11 % specified generation is generated
12
13 % Requires the "Enhanced Input Dialog Box" from Mathworks File Exchange
14 %-----%
15
16 %% Loading of data
17 Title = 'Time Range of Prediction';
18
19 %%% SETTING DIALOG OPTIONS
20 Options.WindowStyle = 'modal';
21 Options.Resize = 'on';
22 Options.Interpreter = 'tex';
23 Options.CancelButton = 'on';
24 Options.ApplyButton = 'off';
25 Options.ButtonNames = {'Continue', 'Cancel'};
26 Option.Dim = 2; % Horizontal dimension in fields
27
28 Prompt = {};
29 Formats = {};
30 DefAns = struct([]);
31
32 Prompt(1,:) = {'Start Date:', 'start', []};
33 Formats(1,1).type = 'edit';
34 Formats(1,1).format = 'text';
35 Formats(1,1).size = 200;

```

```
36 DefAns(1).start = Predict_Start;
37
38 Prompt(2,:) = {'End Date:', 'end', []};
39 Formats(2,1).type = 'edit';
40 Formats(2,1).format = 'text';
41 Formats(2,1).size = 200;
42 DefAns(1).end = '15-May-2016';
43
44 for j = 1:length(netcs_Narx)
45     Cat{1,j}=['Network' ' ' num2str(j)];
46 end
47
48 Prompt(3,:) = {'Networks','List', []};
49 Formats(3,1).type = 'list';
50 Formats(3,1).style = 'popupmenu';
51 Formats(3,1).items = Cat;
52
53 Prompt(4,:) = {'Plot results ?' 'Plot', []};
54 Formats(4,1).type = 'check';
55 DefAns(1).Plot = true;
56
57 Prompt(5,:) = {'Plotting Information' 'Best', []};
58 Formats(5,1).type = 'check';
59 DefAns(1).Best = false;
60
61 Prompt(6,:) = {'Save Prediction File for Plotly' 'Save', []};
62 Formats(6,1).type = 'check';
63 DefAns(1).Save= false;
64
65 [Answer,Cancelled] = inputsdlg(Prompt,Title,Formats,DefAns,Options);
66
67 if Answer.Best == 1
68     perfs_pred=zeros(1,numNN);
69 end
70
71 % TimeRange is selfmade MATLAB function which extracts the specified time
72 % interval from the complete data sequence
73 [InputPredMat_Narx,TargetPredMat_Narx,TimeVectorPred_Narx]= ...
74     TimeRange(Answer.start,Answer.end,Daten_Resampled_Table);
75
76 % Conversion of MATLAB Array to neural network data format
77 InputPred=tonndata(InputPredMat_Narx,false,false);
78 TargetPred=tonndata(TargetPredMat_Narx,false,false);
79
80 %% Loading of network & configuration
81 % Loads the specified network and configures the input
82 Net_Narx=netcs_Narx{1,Answer.List};
83 [xP, XicP, AicP, tcP]=preparets(Net_Narx,InputPred,{},TargetPred);
84
85 % Computing network prediction
```

```

86 [Pred_Narx1, XcfP, AcfP]=Net_Narx(xP,XicP,AicP);
87 Pred_Narx=cell2mat(Pred_Narx1);
88
89 %% Average of all networks in a generation
90
91 y2Totalc=0;
92 for i=1:numNN
93     netci=netcs_Narx{1,i};
94     y2c=netci(xP,XicP,AicP);
95     if Answer.Best == 1
96         perfs_pred(1,i)=mse(tcP,y2c);
97     end
98     y2c=cell2mat(y2c);
99     y2Totalc=y2Totalc + y2c;
100 end
101 AveragePred_Narx=y2Totalc/numNN;
102 A=tonndata(AveragePred_Narx,true,false);
103
104 %% Output of testing errors in the console
105 disp(['Prediction Error - Network ' ' ' num2str(Answer.List)])
106 disp(mse(tcP,Pred_Narx1))
107
108 disp('Prediction Error - Average')
109 disp(mse(tcP,A))
110
111 if Answer.Best == 1
112     [Best_Pred_Performance_Narx(1,1),Index_Best_Pred_Narx(1,1)]=...
113         min(perfs_pred(1,:));
114     [Worst_Pred_Performance_Narx,~]=max(perfs_pred(1,:));
115     disp(['Prediction Error - Best Network ' num2str(Index_Best_Pred_Narx)])
116     disp(Best_Pred_Performance_Narx(1,1))
117     PlottingInfoPred = [Hidden,sqrt(perfs_pred(1,Index_Best_Narx))];
118
119     for i = 3:(numNN+2)
120         PlottingInfoPred(1,i)=sqrt(perfs_pred(1,i-2));
121     end
122 end
123
124
125 %% Plotting Results
126 % Plots the results of the selected network and the average of all networks
127 % in a generation
128 % Input dialogue whether plots should be generated
129
130 if Answer.Plot == 1
131     load 'SondenNamen.mat' % Loading of well names for plots
132     NumTarget=length(TargetPredMat_Narx(1,:));
133     delay=max(delayt_Narx,delayi);
134
135     Plot_HM_PR(TimeVectorPred_Narx,TargetPredMat_Narx,AveragePred_Narx,...

```

```
136         Pred_Narx, NumTarget, delay, Sonden, 1);
137 end
138
139 %% Saving prediction file
140 % Saves a .csv file with all required data to plot the results in the
141 % program R using dygraphs
142 % Input dialogue for the file name
143
144 if Answer.Save==1
145     SavePred(TargetPredMat_Narx, Pred_Narx, AveragePred_Narx, ...
146             TimeVectorPred_Narx, delay)
147 end
148 %% Cleaning workspace
149 clear y2Totalc y2c prompt i netci ind defaultans num_lines answer dgl_title ...
150     prompt Prompt Answer Formats DefAns Cat Pred_Narx1 A Net_Narx j ...
151     Cancelled Title Option Options;
```

TrainFully.m

```

1 % Training of Recurrent Neural Networks
2 % This script allows specifying time intervals for training and testing of
3 % these Recurrent networks.
4 % Input dialogue for:
5 % - Number of networks in a network generation
6 % - The maximum and minimum number of hidden neurons
7 % - Random seed value
8 % - Number of training epochs
9 %-----%
10 %% Checks if data sequence is loaded in workspace
11 % Loads data into workspace from .mat file if not already available.
12
13 if exist('Daten_Resampled_Table','var') == 0
14     uiopen('matlab');
15 end
16
17 %% Loading of specified time interval for training
18 clc
19 prompt={'Start Date:', 'End Date:'};
20 dlg_title='Timerange of Network Input';
21 num_lines=1;
22 defaultans={'01-Mar-2014', '15-May-2016'};
23 answer=inputdlg(prompt,dlg_title,num_lines,defaultans);
24
25 Predict_Start=answer{2};
26
27 % TimeRange is selfmade MATLAB function which extracts the specified time
28 % interval from the complete data sequence
29 [InputMat_Layrec,TargetMat_Layrec,F,TimeVector_Layrec]=...
30     TimeRange(answer{1},answer{2},Daten_Resampled_Table);
31
32 %F1=InputMat_Layrec(:,1)+InputMat_Layrec(:,2)+InputMat_Layrec...
33 %(:,3)+InputMat_Layrec(:,4)+InputMat_Layrec(:,5)+InputMat_Layrec(:,6);
34
35 %F3=zeros(length(F1),1);
36 %for i=2:length(F1)
37 %F3(i)=F3(i-1)+F1(i);
38 %end
39
40 %InputMat_Layrec=[InputMat_Layrec,F',F1,F3];
41
42 % Conversion of MATLAB Array to neural network input.
43 Input=tonndata(InputMat_Layrec,false,false);
44 Target=tonndata(TargetMat_Layrec,false,false);
45
46 %% Configuration of networks
47 prompt={'Zahl von Netzwerken', 'Min Nummer von Hidden nodes',...

```

```

48     'Max Nummer von Hidden nodes', 'Hidden Nodes Stepsize', ...
49     'FeedbackDelays', 'Trainingsepochen', 'RandomSeed');
50 dlg_title='Experten Netzwerkeinstellungen';
51 num_lines=1;
52 defaultans={'10', '14', '14', '2', '4', '100', '666'};
53 answer=inputdlg(prompt,dlg_title,num_lines,defaultans);
54
55 % Specification of network parameters
56 numNN = str2double(answer{1});      % Number of networks in generation
57 Hmax = str2double(answer{3});      % Max hidden nodes
58 Hmin = str2double(answer{2});      % Min hidden nodes
59 dH = str2double(answer{4});        % Stepsize hidden nodes
60 delayt_Layrec=str2double(answer{5}); % Target Delays (if required)
61 NumEpoch=str2double(answer{6});   % Number of training epochs
62 RandomSeed=str2double(answer{7});  % Random seed value
63
64 %% Training of networks
65
66     HiddenNodes_Forloop_Layrec
67
68 %% Training Stats
69 % Compilation of a MATLAB table of training records
70     TrainStats
71     openvar('Stats')
72
73 %% Plotting of results
74 % Results of the best trained network and the an average of all network
75 % outputs is plotted
76 % Input dialogue asking whether plots are required
77
78 prompt={'Sollen Ergebnisse geplottet werden Ja oder Nein'};
79 dlg_title='Plotten der Ergebnisse';
80 num_lines=1;
81 defaultans={'Ja'};
82 answer=inputdlg(prompt,dlg_title,num_lines,defaultans);
83
84 if strcmp(answer{1}, 'Ja')
85     load 'SondenNamen.mat' % Loading of well names for plot
86     NumTarget=length(TargetMat_Layrec(1, :));
87
88     [~, ind]=min(Best_Network_Performance_Layrec, [], 1);
89
90     Plot_HM_PR(TimeVector_Layrec,TargetMat_Layrec,Average_Layrec, ...
91         Best_Layrec,NumTarget,delayt_Layrec,Sonden,ind);
92 end
93
94 %% Cleaning workspace
95 clear answer Best1 defaultans dlg_title y2Av y2 y2c y2Avc y2AverageOutput ...
96     y2AverageOutputc y2Total y2Totalc ans E1 E2 i j k netci neti netos ...
97     Separation s delayt timing ind netLayrec

```


HiddenForloopFully.m

```
1 %% Network Initialization
2 clc
3
4 % The following variables in comment are delivered by TrainLayrec.m
5 % They can be used to test the script without using TrainLayrec.m
6
7 %numNN=5;    %Number of Networks
8 %NumEpoch=25;%Number of Training Epochs
9 %delayt=4;   %Feedback Delays (if required)
10 %Hmax=14;   %Maximum Hidden Neurons
11 %Hmin=10;   %Minimum Hidden Neurons
12 %dH=2;      %Hidden Neurons Stepsize
13 Hcount=1+(Hmax-Hmin)/dH;
14 j=1;        %Internal Count
15 NumTarget=numel(Target{1});
16 delayt=delayt_Layrec;
17
18 %Preallocation of Matrices
19 perfs_closed_trained = zeros(Hcount,numNN);
20 perfAverageOutputc=zeros(Hcount,1);
21 Best_Network_Performance_Layrec=zeros(Hcount,1);
22 Index_Best_Layrec=zeros(Hcount,1);
23 netcs_Layrec=cell(Hcount,numNN);
24 Tr_closed=cell(Hcount,numNN);
25 Best_Layrec=zeros(Hcount*NumTarget,(length(Target)-delayt));
26 Average_Layrec=zeros(Hcount*NumTarget,(length(Target)-delayt));
27 RandomSeed_Stored=cell(Hcount,numNN);
28 Separation=zeros(Hcount,1);
29 Hidden=zeros(Hcount,1);
30
31 %% Forloop over number of hidden neurons
32 % Generates a network generation with a specified number of neurons in the
33 % hidden layer
34 % All other network parameters stay the same
35
36 for H=Hmin:dH:Hmax
37 Hidden(j)=H;
38 netLayrec=layrecnet(1:delayt,H);
39 netLayrec.trainFcn='trainlm';
40 netLayrec.trainParam.epochs=NumEpoch;
41 netLayrec.divideMode='time';
42 netLayrec.divideFcn='divideint';
43
44 [Xc,Xic,Aic,Tc]=preparets(netLayrec,Input,Target);
45
46 %% Network train in open loop
47 % Trains all networks in a single network generation
```

```

48 for i=1:numNN
49
50     if RandomSeed~=0           % Checks if user defined random seed value exists
51         rng(RandomSeed+i); % If it exists it is used for training
52     end
53
54     netcs_Layrec{j,i}=init(netLayrec);
55     netcs_Layrec{j,i}.trainParam.epochs=NumEpoch;
56     [netcs_Layrec{j,i},Tr_closed{j,i}]=train(netcs_Layrec{j,i},Xc,Tc,Xic,Aic);
57     RandomSeed_Stored{j,i}=rng;
58 end
59
60 %% Closed Loop Performance Evaluation
61 % Evaluates the performance of all networks after training in closed loop
62
63 y2Totalc=0;
64 for i=1:numNN
65     netci=netcs_Layrec{j,i};
66     y2c=netci(Xc,Xic,Aic);
67     perfs_closed_trained(j,i)=mse(Tc,y2c);
68     y2c=cell2mat(y2c);
69     y2Totalc=y2Totalc + y2c;
70 end
71
72 y2AverageOutputc=y2Totalc/numNN;
73 y2Avc=num2cell(y2AverageOutputc);
74 E2=gsubtract(y2Avc,Tc);
75 perfAverageOutputc(j)=mse(E2);
76 [Best_Network_Performance_Layrec(j,1),Index_Best_Layrec(j,1)]=...
77     min(perfs_closed_trained(j,:));
78
79 k=(j-1)*NumTarget+1;
80 Best1=(netcs_Layrec{j,Index_Best_Layrec(j,1)}(Xc,Xic,Aic));
81 Best_Layrec((k:k+(NumTarget-1)),:)=cell2mat(Best1);
82 Average_Layrec((k:k+(NumTarget-1)),:)=y2AverageOutputc;
83 j=j+1;
84
85 %% Checkpoint
86 % Saves a checkpoint file after a network generation is trained in closed
87 % loop
88 if mod(j,1)==0
89     s=rng;
90     tic
91     save('Checkpoint_tmp.mat');
92     if strcmp(computer,'PCWIN64') || strcmp(computer,'PCWIN')
93         %We are running on a windows machine
94         system('move /y checkpoint_tmp.mat checkpoint.mat');
95     else
96         %We are running on Linux or Mac
97         system('mv checkpoint_tmp.mat checkpoint.mat');

```

```

98         end
99         timing = toc;
100        fprintf('Checkpoint save took %f seconds\n',timing);
101
102    end
103    end
104    %% Summary
105    % Summary file of network performances for all generations
106    % This file is used by other scripts can be stored as MATLAB Array
107    summary=[Hidden,Separation,perfs_closed_trained,perfAverageOutputc,...
108            Separation,Best_Network_Performance_Layrec,Index_Best_Layrec];

```

TestFully.m

```

1  % This script allows using trained networks to predict outputs for unseen
2  % data sequences
3  % It also enables computing the error of the network response to unseen
4  % data sequences
5
6  % An input dialogue can be used to select a time interval for testing and
7  % prediction from the whole data sequence
8
9  % Individual networks can be selected via a drop down menu
10 % An output for the selected network and an average of all networks in the
11 % specified generation is generated
12
13 % Requires the "Enhanced Input Dialog Box" from Mathworks File Exchange
14 %-----%
15
16 %% Loading of data
17 Title = 'Time Range of Prediction';
18
19 %%% SETTING DIALOG OPTIONS
20 Options.WindowStyle = 'modal';
21 Options.Resize = 'on';
22 Options.Interpreter = 'tex';
23 Options.CancelButton = 'on';
24 Options.ApplyButton = 'off';
25 Options.ButtonNames = {'Continue','Cancel'};
26 Option.Dim = 2; % Horizontal dimension in fields
27
28 Prompt = {};
29 Formats = {};
30 DefAns = struct([]);
31
32 Prompt(1,:) = {'Start Date:', 'start', []};
33 Formats(1,1).type = 'edit';

```

```
34 Formats(1,1).format = 'text';
35 Formats(1,1).size = 200; % automatically assign the height
36 DefAns(1).start = Predict_Start;
37
38 Prompt(2,:) = {'End Date:', 'end', []};
39 Formats(2,1).type = 'edit';
40 Formats(2,1).format = 'text';
41 Formats(2,1).size = 200; % automatically assign the height
42 DefAns(1).end = '15-May-2016';
43
44 for j = 1:length(netcs_Layrec)
45     Cat{1,j}=['Network' ' ' num2str(j)];
46 end
47
48 Prompt(3,:) = {'Networks', 'List', []};
49 Formats(3,1).type = 'list';
50 Formats(3,1).style = 'popupmenu';
51 Formats(3,1).items = Cat;
52
53 Prompt(4,:) = {'Plot results ?' 'Plot', []};
54 Formats(4,1).type = 'check';
55 DefAns(1).Plot = true;
56
57 Prompt(5,:) = {'Plotting Information' 'Best', []};
58 Formats(5,1).type = 'check';
59 DefAns(1).Best = false;
60
61 Prompt(6,:) = {'Save Prediction File for Plotly' 'Save', []};
62 Formats(6,1).type = 'check';
63 DefAns(1).Save= false;
64
65 [Answer,Cancelled] = inputsdlg(Prompt,Title,Formats,DefAns,Options);
66
67 if Answer.Best == 1
68     perfs_pred=zeros(1,numNN);
69 end
70
71 % TimeRange is selfmade MATLAB function which extracts the specified time
72 % interval from the complete data sequence
73 [InputPredMat_Layrec,TargetPredMat_Layrec,TimeVectorPred_Layrec]= ...
74     TimeRange(Answer.start,Answer.end,Daten_Resampled_Table);
75
76 % Conversion of MATLAB Array to neural network data format
77 InputPred=tonndata(InputPredMat_Layrec,false,false);
78 TargetPred=tonndata(TargetPredMat_Layrec,false,false);
79
80 %% Loading of network & configuration
81 % Loads the specified network and configures the input
82
83 Net_Layrec=netcs_Layrec{1,Answer.List};
```

```
84 [xP, XicP, AicP, tcP]=preparets(Net_Layrec,InputPred,TargetPred);
85
86 % Computing network prediction
87 [Pred_Layrec1, XcfP, AcfP]=Net_Layrec(xP,XicP,AicP);
88 Pred_Layrec=cell2mat(Pred_Layrec1);
89
90 %% Average of all networks in a generation
91
92 y2Totalc=0;
93 for i=1:numNN
94     netci=netcs_Layrec{1,i};
95     y2c=netci(xP,XicP,AicP);
96     if Answer.Best == 1
97         perfs_pred(1,i)=mse(tcP,y2c);
98     end
99     y2c=cell2mat(y2c);
100    y2Totalc=y2Totalc + y2c;
101 end
102
103 AveragePred_Layrec=y2Totalc/numNN;
104 A=tonndata(AveragePred_Layrec,true,false);
105
106 %% Output of testing erros in the console
107 disp(['Prediction Error - Netzwerk' ' ' num2str(Answer.List)])
108 disp(mse(tcP,Pred_Layrec1))
109
110 disp('Prediction Error - Average')
111 disp(mse(tcP,A))
112
113 if Answer.Best == 1
114     [Best_Pred_Performance_Layrec(1,1),Index_Best_Pred_Layrec(1,1)]=...
115         min(perfs_pred(1,:));
116     [Worst_Pred_Performance_Layrec,~]=max(perfs_pred(1,:));
117     disp(['Prediction Error - Bestes Netzwerk ' num2str(Index_Best_Pred_Layrec)])
118     disp(Best_Pred_Performance_Layrec(1,1))
119     PlottingInfoPred = [Hidden,sqrt(perfs_pred(1,Index_Best_Layrec))];
120
121     for i = 3:(numNN+2)
122         PlottingInfoPred(1,i)=sqrt(perfs_pred(1,i-2));
123     end
124 end
125
126
127 %% Plotting Results
128 % Plots the results of the selected network and the average of all netowrks
129 % in a generation
130 % Input dialogue whether plots should be generated
131
132 if Answer.Plot == 1
133     load 'SondenNamen.mat' % Loading of well names for plots
```

```
134     NumTarget=length(TargetPredMat_Layrec(1,:));
135     delay=max(delayt_Layrec);
136
137     Plot_HM_PR(TimeVectorPred_Layrec,TargetPredMat_Layrec,AveragePred_Layrec,...
138         Pred_Layrec,NumTarget,delay,Sonden,1);
139 end
140
141 %% Saving prediction file
142 % Saves a .csv file with all required data to plot the results in the
143 % program R using dygraphs
144 % Input dialogue for the file name
145
146 if Answer.Save==1
147     SavePred(TargetPredMat_Layrec,Pred_Layrec,AveragePred_Layrec,...
148         TimeVectorPred_Layrec,delayt_Layrec)
149 end
150 %% Cleaning workspace
151 clear y2Totalc y2c prompt i netci ind defaultans num_lines answer dgl_title ...
152     prompt Prompt Answer Formats DefAns Cat Pred_Layrec1 A Net_Layrec j ...
153     Cancelled Title Option Options;
```

TrainElman.m

```
1 % Training of Elman Neural Networks
2 % This script allows specifying time intervals for training and testing of
3 % these Elman networks.
4 % Input dialogue for:
5 % - Number of networks in a network generation
6 % - The maximum and minimum number of hidden neurons
7 % - Random seed value
8 % - Number of training epochs
9 %-----%
10 %% Checks if data sequence is loaded in workspace
11 % Loads data into workspace from .mat file if not already available.
12
13 if exist('Daten_Resampled_Table','var') == 0
14     uiopen('matlab');
15 end
16
17 %% Loading of specified time interval for training
18 clc
19 prompt={'Start Date:','End Date:'};
20 dlg_title='Timerange of Network Input';
21 num_lines=1;
22 defaultans={'01-Mar-2014','15-May-2016'};
23 answer=inputdlg(prompt,dlg_title,num_lines,defaultans);
24
25 Predict_Start=answer{2};
26
27 % TimeRange is selfmade MATLAB function which extracts the specified time
28 % interval from the complete data sequence
29 [InputMat_Elman,TargetMat_Elman,TimeVector_Elman]=TimeRange1(answer{1},...
30     answer{2},Daten_Resampled_Table);
31
32
33 % Conversion of MATLAB Array to neural network input.
34 Input=tonndata(InputMat_Elman,false,false);
35 Target=tonndata(TargetMat_Elman,false,false);
36
37 %% Configuration of networks
38
39     prompt={'Zahl von Netzwerken','Min Nummer von Hidden nodes',...
40         'Max Nummer von Hidden nodes','Hidden Nodes Stepsize',...
41         'FeedbackDelays','Trainingsepochen','RandomSeed'};
42     dlg_title='Experten Netzwerkeinstellungen';
43     num_lines=1;
44     defaultans={'10','14','14','2','4','100','666'};
45     answer=inputdlg(prompt,dlg_title,num_lines,defaultans);
46
47     % Specification of network parameters
```

```
48     numNN = str2double(answer(1));           % Number of networks in generation
49     Hmax = str2double(answer{3});           % Max hidden nodes
50     Hmin = str2double(answer{2});           % Min hidden nodes
51     dH = str2double(answer{4});            % Stepsize hidden nodes
52     delayt_Elman=str2double(answer{5});    % Target delays
53     NumEpoch=str2double(answer{6});       % Number of training epochs
54     RandomSeed=str2double(answer{7});      % Random seed value
55
56
57 %% Training of networks
58
59     HiddenNodes_Forloop_Elman
60
61 %% Training Stats
62 % Compilation of a MATLAB table of training records
63     TrainStats
64     openvar('Stats')
65
66 %% Plotting of results
67 % Results of the best trained network and the an average of all network
68 % outputs is plotted
69 % Input dialogue asking whether plots are required
70
71 prompt={'Sollen Ergebnisse geplottet werden Ja oder Nein'};
72 dlg_title='Plotten der Ergebnisse';
73 num_lines=1;
74 defaultans={'Ja'};
75 answer=inputdlg(prompt,dlg_title,num_lines,defaultans);
76
77 if strcmp(answer{1},'Ja')
78     load 'SondenNamen.mat' % Loading of well names for plot
79     NumTarget=length(TargetMat_Elman(1,:));
80
81     [~,ind]=min(Best_Network_Performance_Elman,[],1);
82
83     Plot_HM_PR(TimeVector_Elman,TargetMat_Elman,Average_Elman,...
84         Best_Elman,NumTarget,delayt_Elman,Sonden,ind);
85 end
86
87 %% Cleaning workspace
88 clear answer Best1 defaultans dlg_title y2Av y2 y2c y2Avc y2AverageOutput ...
89     y2AverageOutputc y2Total y2Totalc ans E1 E2 i j k netci neti netos ...
90     Separation s timing delayt ind
```


HiddenForloopElman.m

```
1 %% Network Initialization
2 clc
3
4 % The following variables in comment are delivered by TrainElman.m
5 % They can be used to test the script without using TrainElman.m
6
7 %numNN=5;    %Number of Networks
8 %NumEpoch=25;%Number of Training Epochs
9 %delayt=4;   %Feedback Delays
10 %Hmax=14;   %Maximum Hidden Neurons
11 %Hmin=10;   %Minimum Hidden Neurons
12 %dH=2;      %Hidden Neurons Stepsize
13
14 Hcount=1+(Hmax-Hmin)/dH;
15 j=1;        %Internal Count
16 NumTarget=numel(Target{1});
17 delayt=delayt_Elman;
18
19 %Preallocation of Matrices
20 perfs_closed_trained = zeros(Hcount,numNN);
21 perfAverageOutputc=zeros(Hcount,1);
22 Best_Network_Performance_Elman=zeros(Hcount,1);
23 Index_Best_Elman=zeros(Hcount,1);
24 netcs_Elman=cell(Hcount,numNN);
25 Tr_closed=cell(Hcount,numNN);
26 Best_Elman=zeros(Hcount*NumTarget,(length(Target)-delayt));
27 Average_Elman=zeros(Hcount*NumTarget,(length(Target)-delayt));
28 RandomSeed_Stored=cell(Hcount,numNN);
29 Separation=zeros(Hcount,1);
30 Hidden=zeros(Hcount,1);
31
32 %% Forloop over number of hidden neurons
33 % Generates a network generation with a specified number of neurons in the
34 % hidden layer
35 % All other network parameters stay the same
36
37 for H=Hmin:dH:Hmax
38 Hidden(j)=H;
39 netElman=elmannet(1:delayt,H);
40 netElman.trainFcn='trainlm';
41 netElman.trainParam.epochs=NumEpoch;
42 netElman.divideMode='time';
43 netElman.divideFcn='divideint';
44
45 [Xc,Xic,Aic,Tc]=preparets(netElman,Input,Target);
46
47 %% Network train in closed loop
```

```

48 % Trains all networks in a single network generation
49 for i=1:numNN
50
51     if RandomSeed~=0           % Checks if user defined random seed value exists
52         rng(RandomSeed+i); % If it exists it is used for training
53     end
54
55     netcs_Elman{j,i}=init(netElman);
56     netcs_Elman{j,i}.trainParam.epochs=NumEpoch;
57     [netcs_Elman{j,i},Tr_closed{j,i}]=train(netcs_Elman{j,i},Xc,Tc,Xic,Aic);
58     RandomSeed_Stored{j,i}=rng;
59 end
60
61 %% Closed Loop Performance Evaluation
62 % Evaluates the performance of all networks after training in NARX closed
63 % loop
64
65
66 y2Totalc=0;
67 for i=1:numNN
68     netci=netcs_Elman{j,i};
69     y2c=netci(Xc,Xic,Aic);
70     perfs_closed_trained(j,i)=mse(Tc,y2c);
71     y2c=cell2mat(y2c);
72     y2Totalc=y2Totalc + y2c;
73     %[j,i]
74 end
75
76 y2AverageOutputc=y2Totalc/numNN;
77 y2Avc=num2cell(y2AverageOutputc);
78 E2=gsubtract(y2Avc,Tc);
79 perfAverageOutputc(j)=mse(E2);
80 [Best_Network_Performance_Elman(j,1),Index_Best_Elman(j,1)]=...
81     min(perfs_closed_trained(j,:));
82
83 k=(j-1)*NumTarget+1;
84 Best1=(netcs_Elman{j,Index_Best_Elman(j,1)}(Xc,Xic,Aic));
85 Best_Elman(k:k+(NumTarget-1),:)=cell2mat(Best1);
86 Average_Elman(k:k+(NumTarget-1),:)=y2AverageOutputc;
87 j=j+1;
88
89 %% Checkpoint
90 % Saves a checkpoint file after a network generation is trained in
91 % closed loop
92 if mod(j,1)==0
93     s=rng;
94     tic
95     save('Checkpoint_tmp.mat');
96     if strcmp(operating_system,'PCWIN64') || strcmp(operating_system,'PCWIN')
97         %We are running on a windows machine

```

```

98         system( 'move /y checkpoint_tmp.mat checkpoint.mat' );
99     else
100         %We are running on Linux or Mac
101         system( 'mv checkpoint_tmp.mat checkpoint.mat' );
102     end
103     timing = toc;
104     fprintf('Checkpoint save took %f seconds\n',timing);
105
106 end
107 end
108 %% Summary
109 % Summary file of network performances for all generations
110 % This file is used by other scripts can be stored as MATLAB Array
111 summary=[Hidden,Separation,perfs_closed_trained,perfAverageOutputc,...
112         Separation,Best_Network_Performance_Elman,Index_Best_Elman];

```

TestElman.m

```

1 % This script allows using trained networks to predict outputs for unseen
2 % data sequences
3 % It also enables computing the error of the network response to unseen
4 % data sequences
5
6 % An input dialogue can be used to select a time interval for testing and
7 % prediction from the whole data sequence
8
9 % Individual networks can be selected via a drop down menu
10 % An output for the selected network and an average of all networks in the
11 % specified generation is generated
12
13 % Requires the "Enhanced Input Dialog Box" from Mathworks File Exchange
14 %-----%
15
16 %% Loading of data
17 Title = 'Time Range of Prediction';
18
19 %%% SETTING DIALOG OPTIONS
20 Options.WindowStyle = 'modal';
21 Options.Resize = 'on';
22 Options.Interpreter = 'tex';
23 Options.CancelButton = 'on';
24 Options.ApplyButton = 'off';
25 Options.ButtonNames = {'Continue','Cancel'};
26 Option.Dim = 2; % Horizontal dimension in fields
27
28 Prompt = {};
29 Formats = {};

```

```
30 DefAns = struct([]);
31
32 Prompt(1,:) = {'Start Date:', 'start', []};
33 Formats(1,1).type = 'edit';
34 Formats(1,1).format = 'text';
35 Formats(1,1).size = 200; % automatically assign the height
36 DefAns(1).start = Predict_Start;
37
38 Prompt(2,:) = {'End Date:', 'end', []};
39 Formats(2,1).type = 'edit';
40 Formats(2,1).format = 'text';
41 Formats(2,1).size = 200; % automatically assign the height
42 DefAns(1).end = '15-May-2016';
43
44 for j = 1:length(netcs_Elman)
45     Cat{1,j}=['Network' ' ' num2str(j)];
46 end
47
48 Prompt(3,:) = {'Networks','List', []};
49 Formats(3,1).type = 'list';
50 Formats(3,1).style = 'popupmenu';
51 Formats(3,1).items = Cat;
52
53 Prompt(4,:) = {'Plot results ?' 'Plot', []};
54 Formats(4,1).type = 'check';
55 DefAns(1).Plot = true;
56
57 Prompt(5,:) = {'Plotting Information' 'Best', []};
58 Formats(5,1).type = 'check';
59 DefAns(1).Best = false;
60
61 Prompt(6,:) = {'Save Prediction File for Plotly' 'Save', []};
62 Formats(6,1).type = 'check';
63 DefAns(1).Save= false;
64
65 [Answer,Cancelled] = inputdlg(Prompt,Title,Formats,DefAns,Options);
66
67 if Answer.Best == 1
68     perfs_pred=zeros(1,numNN);
69 end
70
71 % TimeRange is selfmade MATLAB function which extracts the specified time
72 % interval from the complete data sequence
73 [InputPredMat_Elman,TargetPredMat_Elman,TimeVectorPred_Elman]= ...
74     TimeRange(Answer.start,Answer.end,Daten_Resampled_Table);
75
76 % Conversion of MATLAB Array to neural network data format
77 InputPred=tonndata(InputPredMat_Elman,false,false);
78 TargetPred=tonndata(TargetPredMat_Elman,false,false);
79
```

```
80 %% Loading of network & configuration
81 % Loads the specified network and configures the input
82 Net_Elman=netcs_Elman(1,Answer.List);
83 [xP, XicP, AicP, tcP]=preparets(Net_Elman,InputPred,TargetPred);
84
85 % Computing network prediction
86 [Pred_Elman1, XcfP, AcfP]=Net_Elman(xP,XicP,AicP);
87 Pred_Elman=cell2mat(Pred_Elman1);
88
89 %% Average of all networks in a generation
90
91 y2Totalc=0;
92 for i=1:numNN
93     netci=netcs_Elman(1,i);
94     y2c=netci(xP,XicP,AicP);
95     if Answer.Best == 1
96         perfs_pred(1,i)=mse(tcP,y2c);
97     end
98     y2c=cell2mat(y2c);
99     y2Totalc=y2Totalc + y2c;
100 end
101 AveragePred_Elman=y2Totalc/numNN;
102 A=tonndata(AveragePred_Elman,true,false);
103
104 %% Output of testing erros in the console
105 disp(['Prediction Error - Netzwerk' ' ' num2str(Answer.List)])
106 disp(mse(tcP,Pred_Elman1))
107
108 disp('Prediction Error - Average')
109 disp(mse(tcP,A))
110
111 if Answer.Best == 1
112     [Best_Pred_Performance_Elman(1,1), Index_Best_Pred_Elman(1,1)]=min(perfs_pred(1,:));
113     [Worst_Pred_Performace_Elman,~]=max(perfs_pred(1,:));
114     disp(['Prediction Error - Bestes Netzwerk ' num2str(Index_Best_Pred_Elman)])
115     disp(Best_Pred_Performance_Elman(1,1))
116     PlottingInfoPred = [Hidden,sqrt(perfs_pred(1,Index_Best_Elman))];
117
118     for i = 3:(numNN+2)
119         PlottingInfoPred(1,i)=sqrt(perfs_pred(1,i-2));
120     end
121 end
122
123
124 %% Plotting Results
125 % Plots the results of the selected network and the average of all netowrks
126 % in a generation
127 % Input dialogue whether plots should be generated
128
129 if Answer.Plot == 1
```

```
130     load 'SondenNamen.mat' % Loading of well names for plots
131     NumTarget=length(TargetPredMat_Elman(1,:));
132     delay=max(delayt_Elman);
133
134     Plot_HM_PR(TimeVectorPred_Elman,TargetPredMat_Elman,AveragePred_Elman,...
135         Pred_Elman,NumTarget,delay,Sonden,1);
136 end
137
138 %% Saving prediction file
139 % Saves a .csv file with all required data to plot the results in the
140 % program R using dygraphs
141 % Input dialogue for the file name
142
143 if Answer.Save==1
144     SavePred(TargetPredMat_Elman,Pred_Elman,AveragePred_Elman,...
145         TimeVectorPred_Elman,delayt_Elman)
146 end
147
148 %% Cleaning workspace
149 clear y2Totalc y2c prompt i netci ind defaultans num_lines answer dgl_title ...
150     prompt Prompt Answer Formats DefAns Cat Pred_Layrec1 A Net_Layrec j ...
151     Cancelled Title Option Options;
```

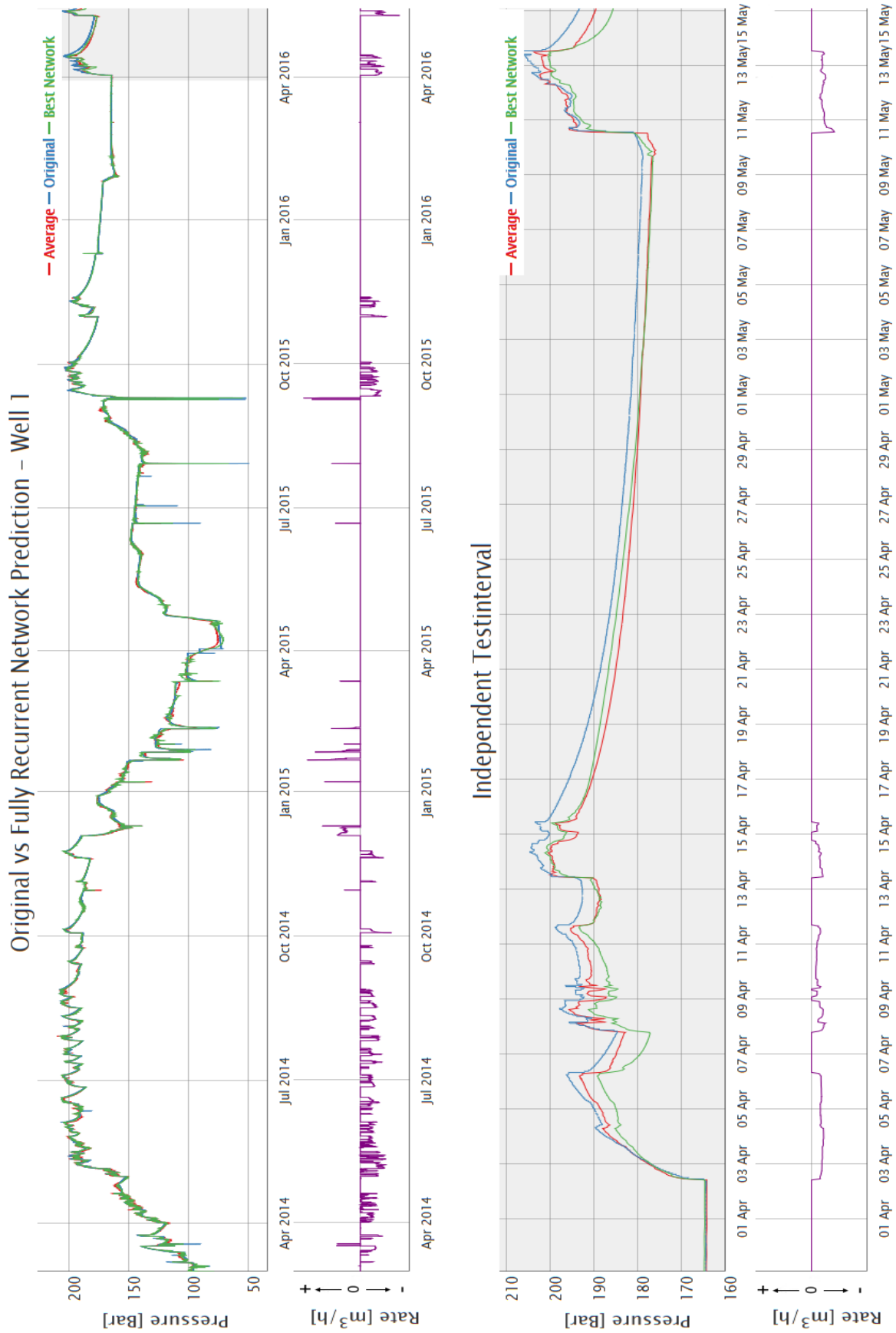


Figure C1: Comparison of Original & fully recurrent network (Generation 9 with 13 hidden neurons) Output for Well 1 in a time series plot. The upper graph depicts the whole 27 months period. The independent test set is shaded in grey.

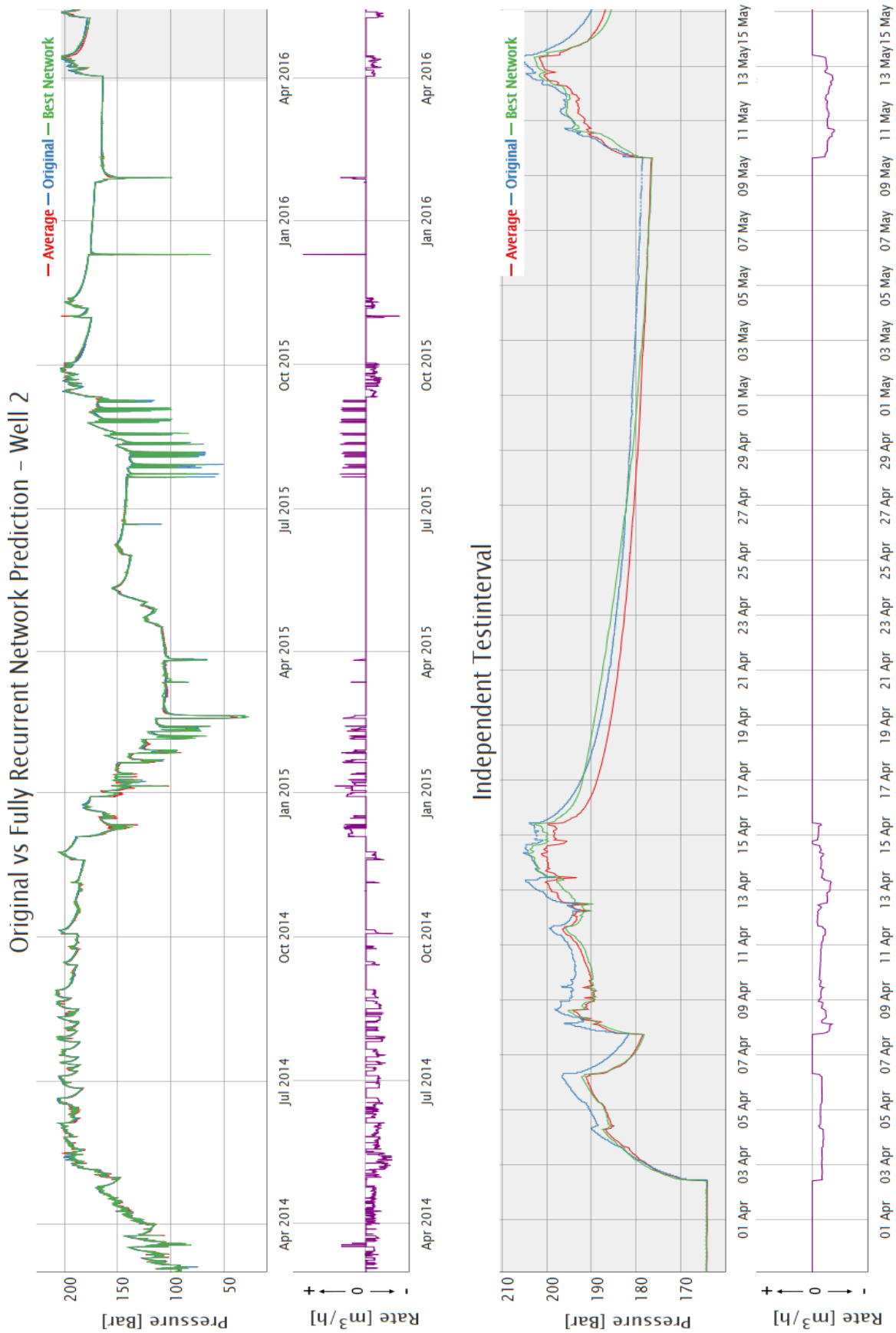


Figure C2: Comparison of Original & fully recurrent network (Generation 9 with 13 hidden neurons) Output for Well 2 in a time series plot. The upper graph depicts the whole 27 months period. The independent test set is shaded in grey.

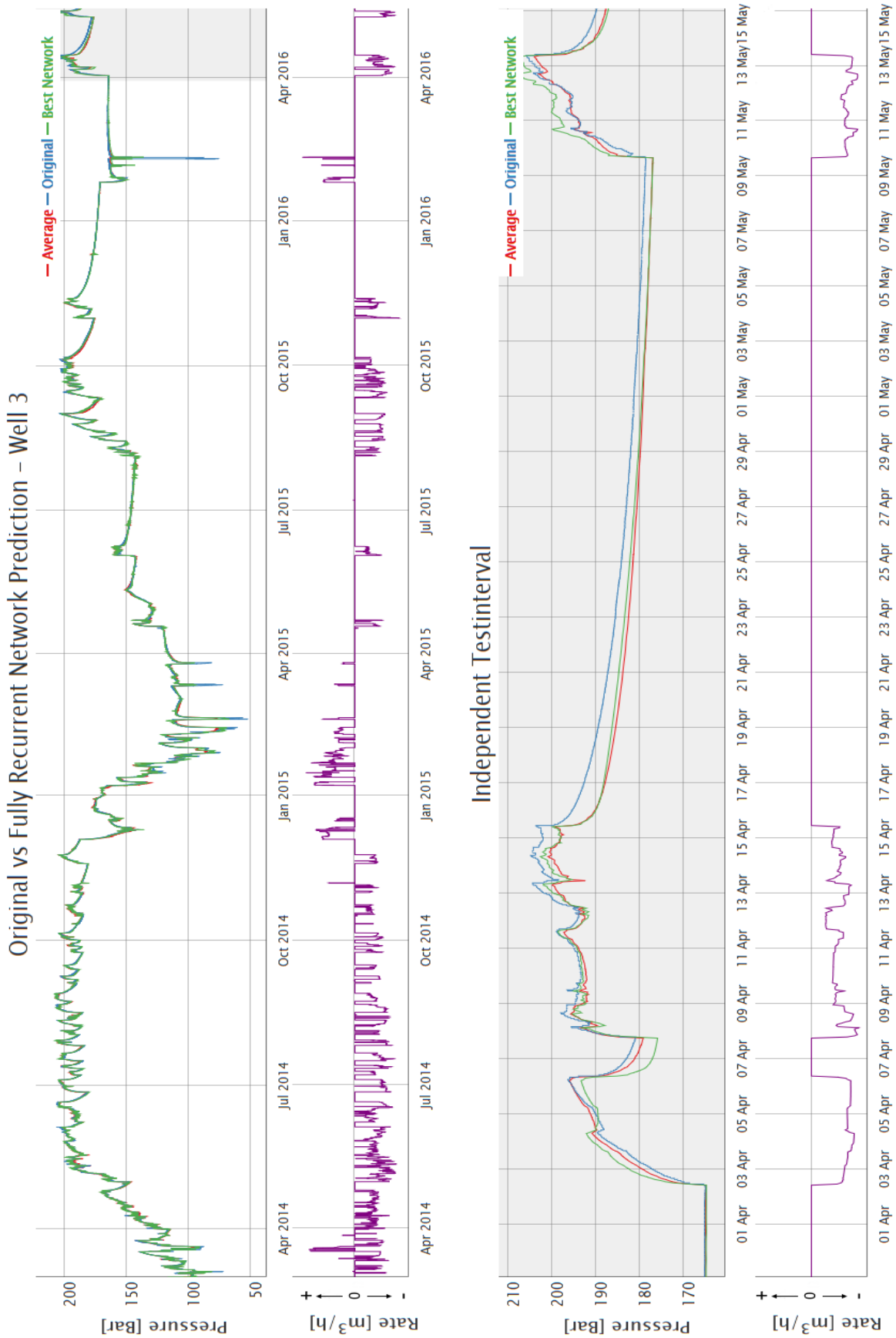


Figure C3: Comparison of Original & fully recurrent network (Generation 9 with 13 hidden neurons) Output for Well 3 in a time series plot. The upper graph depicts the whole 27 months period. The independent test set is shaded in grey.

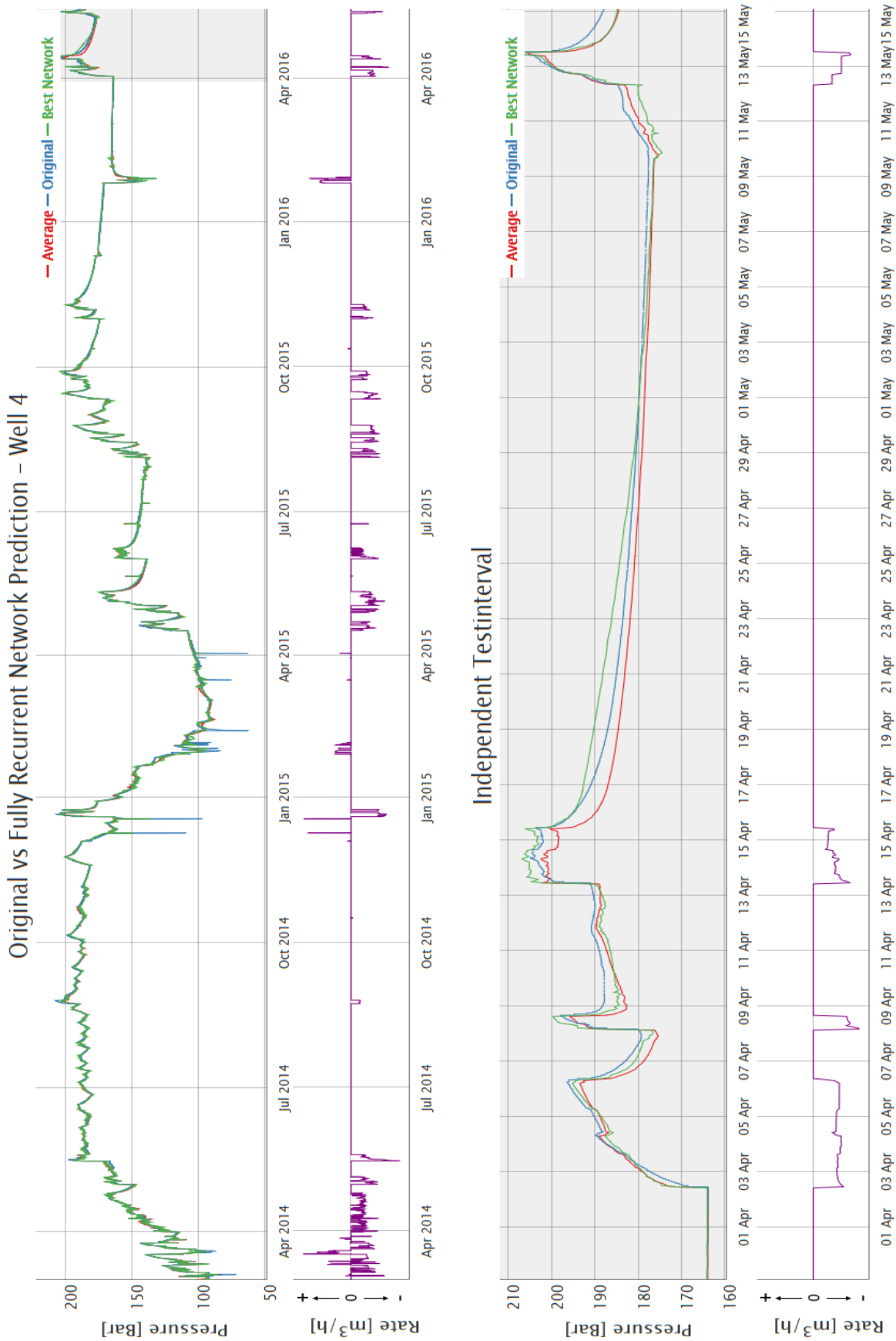


Figure C4: Comparison of Original & fully recurrent network (Generation 9 with 13 hidden neurons) Output for Well 4 in a time series plot. The upper graph depicts the whole 27 months period. The independent test set is shaded in grey.

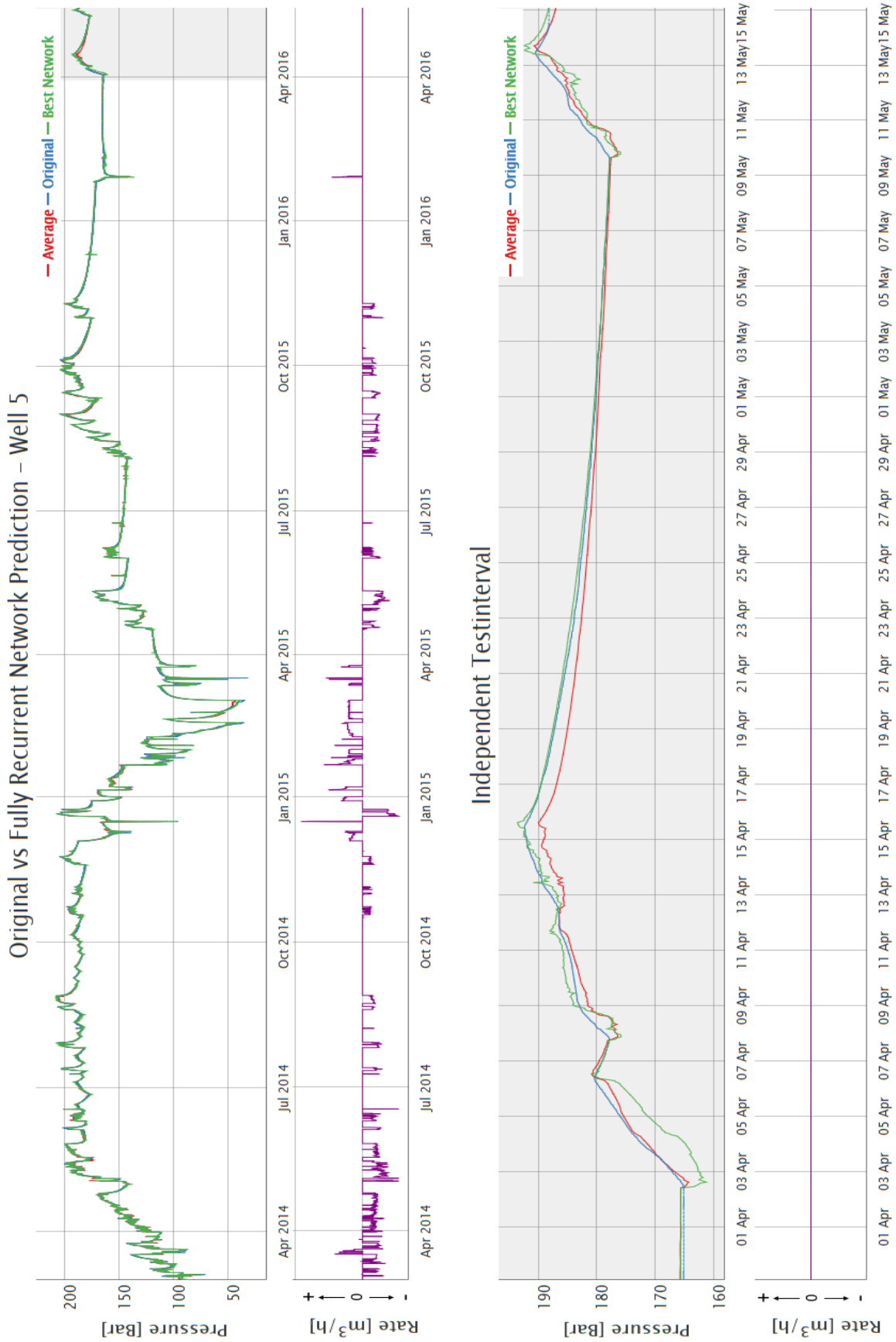


Figure C5: Comparison of Original & fully recurrent network (Generation 9 with 13 hidden neurons) Output for Well 5 in a time series plot. The upper graph depicts the whole 27 months period. The independent test set is shaded in grey.

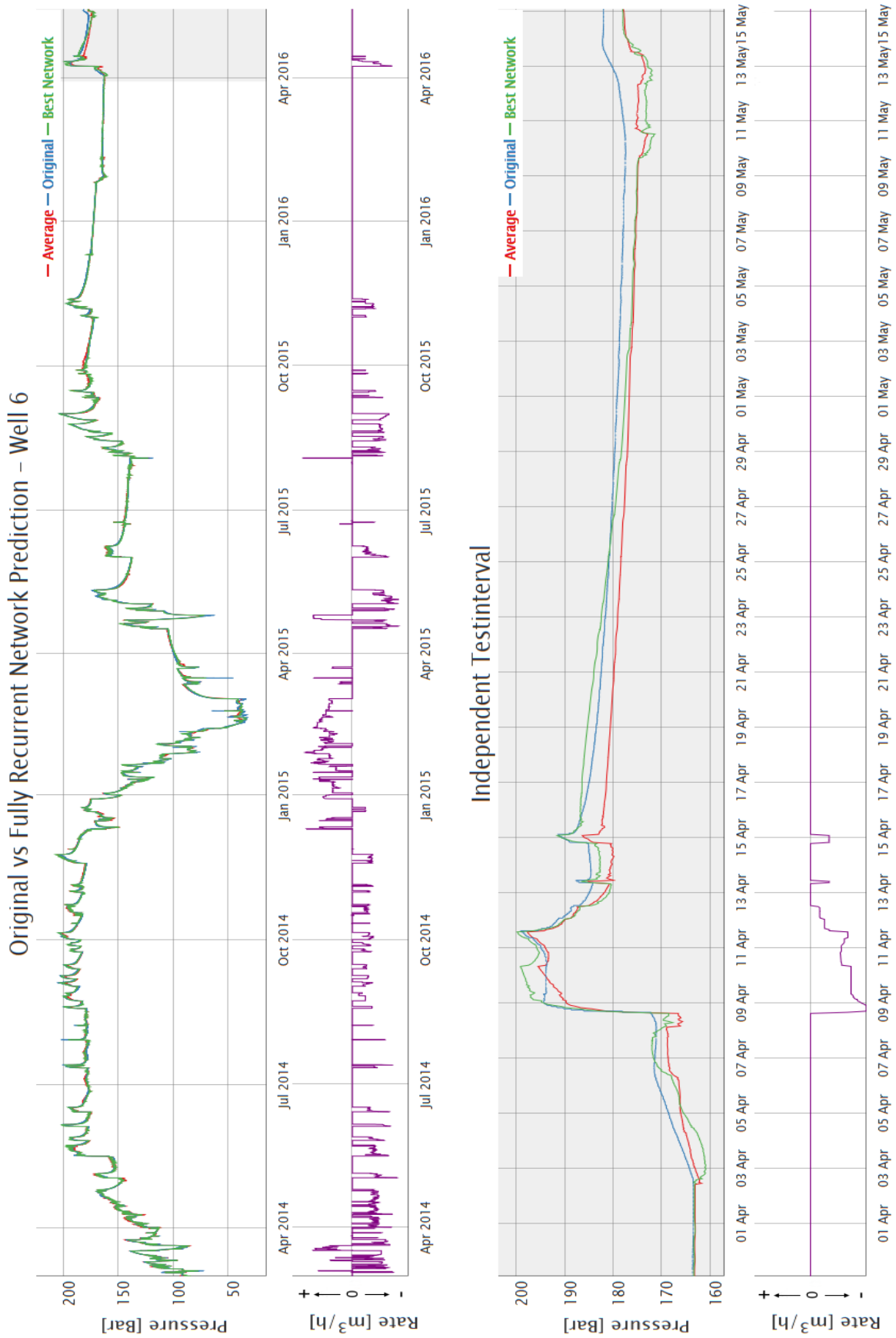


Figure C6: Comparison of Original & fully recurrent network (Generation 9 with 13 hidden neurons) Output for Well 6 in a time series plot. The upper graph depicts the whole 27 months period. The independent test set is shaded in grey.