

Evaluating the Potential of Gaussian Process Regression for Data-driven Renewable Energy Management

by

Joachim Foster Lubbe



*Thesis presented in partial fulfilment of the requirements
for the degree of Master of Engineering (Mechanical) in the
Faculty of Engineering at Stellenbosch University*

Supervisor: Prof. TM Harms

Co-supervisor: Dr. JM Maritz

December 2019

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: Foster Lubbe - December 2019
.....

Copyright © 2019 Stellenbosch University
All rights reserved.

Abstract

Evaluating the Potential of Gaussian Process Regression for Data-driven Renewable Energy Management

JF Lubbe

*Department of Mechanical and Mechatronic Engineering,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.*

Thesis: MEng (Mech)

December 2019

The variable nature of renewable energy sources presents a challenge to the stability of the electricity grid. The probabilistic modelling of renewable energy data could assist in overcoming this challenge. In this context, the machine learning technique of Gaussian process regression is explored. It is posited that Gaussian process regression could form part of a solution to the variability problem by allowing for better-informed energy management decisions. To test this hypothesis, Gaussian process regression is applied to wind speed and solar radiation datasets in order to regress and forecast its behaviour. Weather data is acquired from the Southern African Universities Radiometric Network and the forecasts are compared to the measured values. Meter malfunction is simulated within the solar radiation dataset in order to evaluate its effect on the regression. Gaussian process regression is furthermore used in an attempt to improve the quality of interval-deficient wind speed data. Attention is given to constructing a customised kernel for the modelling of renewable energy system data. The best regression and forecasting results for solar radiation data were obtained by employing a kernel consisting of an exponential sine-squared component and a rational quadratic component. No meaningful advantage was obtained by increasing the complexity of the kernel any further. It was also found that meter failure can be bridged by employing Gaussian process regression. The success achieved in modelling solar radiation data using Gaussian process regression is encouraging and could open up new avenues in the development of an effective renewable energy management system. No significant improvement in the quality of interval deficient wind speed data was observed after applying Gaussian process regression using the Matérn kernel. Python and Matlab were used for modelling and analysis.

Uittreksel

'n Evaluering van die Potensiaal van Gaussiese Proses-regressie vir Data-gedrewe Hernubare Energiebestuur

JF Lubbe

*Departement Meganiese en Megatroniese Ingenieurswese,
Universiteit van Stellenbosch,
Privaatsak X1, Matieland 7602, Suid Afrika.*

Tesis: MIng (Meg)

Desember 2019

Die veranderlike aard van hernubare energiebronne kan die stabiliteit van die elektrisiteitsnetwerk ontwig. Ten einde 'n oplossing te vind, word die probabilistiese masjienleertegniek van Gaussiese proses-regressie verken. Hierdie tegniek kan moontlik bydra tot 'n oplossing vir die stabiliteitsprobleem deur ingeligte energiebestuursbesluite in die hand te werk. Om hierdie hipotese te toets, word Gaussiese proses-regressie op windspoed- en sonstralingsdata toegepas om soedoende die gedrag daarvan te voorspel. Weerdata is van die South African Universities Radiometric Network verkry en voorspellings is met metings vergelyk. Die wanfunksionering van metingsinstrumentasie word in die sonstralingsdata gesimuleer om die effek daarvan op die regressiemodel vas te stel. Daar is ook gepoog om Gaussiese proses-regressie te gebruik om die kwaliteit van lae-resolusie-winddata te verbeter. Aandag word geskenk aan die samestelling van 'n kovariansiefunksie vir die modellering van hernubare energie data. Die beste regressie- en voorspellingsresultate is verkry deur gebruik te maak van 'n kovariansiefunksie bestaande uit 'n eksponensieel-sinus-kwadraat komponent en 'n rasioneel-kwadratiese komponent. Geen betekinsvolle voordeel is verky deur die kompleksiteit van die kovariansiefunksie verder te verhoog nie. Daar is ook bevind dat die wanfunksionering van metingsinstrumentasie oorbrug kan word deur Gaussiese proses-regressie. Die suksesvolle modellering van sonstralingsdata is belowend en kan nuwe kanale skep vir die ontwikkeling van 'n effektiewe hernubare-energie-bestuurstelsel. Die toepassing van Gaussiese proses-regressie deur gebruik te maak van die Matérn kovariansiefunksie het nie 'n noemenswaardige verbetering in die kwaliteit van windspoeddata teweeg gebring nie. Python en Matlab is gebruik vir modellering en analise.

Acknowledgements

The financial assistance of the Manufacturing, Engineering and Related Services Seta (merSETA) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at are those of the author and are not necessarily to be attributed to merSETA.

Contents

Declaration	i
Abstract	ii
Uittreksel	iii
Acknowledgements	iv
Contents	v
List of Figures	vii
List of Tables	ix
Nomenclature	x
1 Introduction	1
1.1 Data Could Be the New Oil	1
1.2 Digitalisation and the Smart Microgrid	5
1.3 Big-data-driven Smart Energy Management	6
1.4 Trends in Early-stage Investment	7
1.5 The Need for a Well-defined Energy System Model	10
1.6 Research Objectives	12
1.7 Original Research Contribution of this Study	13
2 A Review of Machine Learning Techniques in Energy System Analysis	14
2.1 Artificial Neural Networks	15
2.2 Multilayer Perceptron	17
2.3 Extreme Learning Machines	18
2.4 Support Vector Machines	20
2.5 Wavelet Neural Networks	21
2.6 Adaptive Neuro-fuzzy Inference Systems	21
2.7 Decision Trees	21
2.8 Deep Learning	21

<i>CONTENTS</i>	vi
2.9 Ensembles	22
2.10 Hybrid Models	22
2.11 Popular Applications of Machine Learning in Energy Management	24
2.12 Gaussian Process Regression in Energy Systems Analysis	25
2.13 Summary	27
3 Gaussian Process Regression	28
3.1 The Conditional of a Gaussian	29
3.2 Supervised Learning	31
3.3 Weight-space View	34
3.4 Error Propagation in Gaussian Process Regression	35
3.5 Computational Performance of Gaussian Process Regression Algorithms	36
3.6 Summary	37
4 Kernels	38
4.1 Supervised Learning Revisited	38
4.2 The Kernel Universe	41
4.3 Complex Kernel Engineering	47
4.4 Summary	55
5 Case Studies	56
5.1 Global Horizontal Irradiance Data	57
5.2 Wind Speed Data	68
6 Conclusion and Recommendation for Future Work	77
Appendices	79
A GHI Interpolation	80
B GHI Forecasting	86
C Wind Data	91
D Instrumentation	107
References	108

List of Figures

1.1	The solar settlement	3
1.2	Borkum Riffgrund 1	4
1.3	Early-stage investment trends	8
1.4	Average transaction sizes for wave-1 and wave-2 firms	10
1.5	Power demand and solar power potential	11
2.1	Artificial neural network architecture	16
2.2	Forecast power output by artificial neural network	18
2.3	Forecast power output by extreme learning machine	19
2.4	Natural-gas spot price prediction	20
2.5	Hybrid machine learning model	23
2.6	Eskom demand side management energy savings	24
2.7	Comparison of Gaussian process regression to other regression algorithms	25
3.1	Two-dimensional joint Gaussian distribution	29
3.2	Gaussian process regression graphical model	32
3.3	Gaussian process regression example	34
4.1	Gaussian process regression flow diagram	40
4.2	Linear kernel prior and posterior	42
4.3	Radial basis function prior and posterior	43
4.4	Rational quadratic function prior and posterior	44
4.5	Periodic kernel prior and posterior	45
4.6	Matérn kernel prior and posterior	47
4.7	Structures expressible by compound kernels	48
4.8	Mauna Loa carbon dioxide concentration	49
4.9	Airline passenger data	53
4.10	Solar irradiance between 1610 and 2011	55
5.1	Sauran weather metrics	58
5.2	Gaussian process regression with no meter failure	60
5.3	Single-input Gaussian process regression result	61
5.4	Multi-input Gaussian process regression result	62
5.5	All-in Gaussian process regression result	63

LIST OF FIGURES

viii

5.6	Gaussian process forecasting best result	65
5.7	Gaussian process forecasting multiple results	66
5.8	Wind speed data	69
5.9	Impact of intermittency interval at Gobabeb	70
5.10	Interval-deficient wind speed data	72
5.11	Weibull scale parameter A	73
5.12	Weibull shape parameter k	73
D.1	Suaran weather station specifications	107

List of Tables

1.1	The top renewable-energy deals for 2018	7
1.2	Description of wave-1 and wave-2 technologies	9
3.1	Performance of CPU and GPU computation	37
4.1	Constituent kernels	50
4.2	Optimised hyperparameters	51
5.1	Sauran weather metrics	56
5.2	Number of optimiser restarts for interpolation	59
5.3	Error on interpolation	64
5.4	Number of optimiser restarts for GHI forecasting	64
5.5	Error on forecasting	67
5.6	Attempted kernels	67
5.7	Impact of intermittency interval at Stellenbosch University	74
5.8	Impact of intermittency interval: Gaussian process interpolated data	75

Nomenclature

Abbreviations

air_temp	Air temperature
ANN	Artificial neural network
AR	Autoregression
ARIMA	Autoregression integrated moving average
ARMA	Autoregression moving average
BP	Back propagation
BP	Barometric pressure
C	Constant kernel
CEM	Certified Energy Manager
CNN	Convolution neural network
CPU	Central processing unit
CSP	Concentrated solar power
DER	Distributed Energy Resources
DHI	Diffuse horizontal irradiance
DNI	Direct normal irradiance
DSM	Demand side management
EDE	Enhanced differential evolution
EDTLA	Enhanced differential teaching learning algorithm
ELM	Extreme learning machines
EMD	Empirical mode decomposition
Exp, Per	Exponential sine-squared kernel
GDP	Gross domestic product
GHI	Global horizontal irradiance
GIS	Geographic information system
GPQR	Gaussian process quantile regression
GPU	Graphics processing unit
HIMF	High-frequency intrinsic mode functions
IEC	International Electrotechnical Commission

IoT	Internet of Things
IPMVP	International Performance Measurement and Verification Protocol
IRP	Integrated Resource Plan
LFR	Low-frequency residuals
Lin	Linear kernel
MA	Moving average
MAPE	Mean absolute percentage error
Mat	Matérn kernel
MIMF	Medium-frequency intrinsic mode functions
M&V	Measurement and verification
NEX	WilderHill New Energy Global Innovation Index
NWP	Numerical weather prediction
OECD	Organization for Economic Cooperation and Development
PV	Photovoltaic
RBF, SE	Radial basis function
RES	Renewable energy source
RF	Random forest
RH	Relative humidity
RQ	Rational quadratic kernel
Sauran	Southern African Universities Radiometric Network
SVM	Support vector machines
SVR	Support vector regression
S&P	Standard and Poor's
TLBO	Teaching learning-based optimization
UVA	Long wave ultraviolet radiation
UVB	Short wave ultraviolet radiation
WD	Wind direction
WD_SD	Standard deviation in wind direction
WS	Wind speed

Subscripts

<i>i</i>	Index
*	Test set

Symbols and operators

cov	Covariance
$cholesky$	Cholesky decomposition (operator)
\mathcal{D}	Set
\mathbb{E}	Expected value
GP	Gaussian process
K_ν	Modified Bessel function
L	Cholesky decomposition (symbol)
$m()$	Mean function
\mathcal{N}	Gaussian distribution
O	Big O notation
p	Probability distribution (not necessarily Gaussian)
T	Transpose
t	Time
V, var	Variance
δ	Kronecker's delta function
$\kappa()$	Kernel function
ν	Matérn kernel parameter
$ $	Conditional
∞	Infinity
\rightarrow	Limit
\sim	Sampled from
$'$	Used to distinguish two vectors taken from the same set, for example \mathbf{x} and \mathbf{x}'

Variables

A	Weibull scale factor
k	Weibull shape factor
l	Characteristic length scale
x	Arbitrary variable
α	Scale mixture parameter
θ	Kernel hyperparameter
μ	Mean
ρ	Correlation coefficient
σ	Covariance

Vectors and Matrices

\mathbf{f}	Vector of function outputs
\mathbf{v}	Variance
\mathbf{w}	Vector of hyperparameters
\mathbf{X}	Matrix of Gaussian process inputs
\mathbf{x}	Vector of Gaussian process inputs
\mathbf{y}	Vector of Gaussian process outputs
Λ	Inverse of covariance matrix
Λ_{ii}	Sub-matrix of inverse of covariance matrix
$\boldsymbol{\mu}$	Mean vector
Σ	Covariance matrix
$\Sigma_{ii}, \mathbf{K}_{ii}$	Covariance sub-matrix

Chapter 1

Introduction

“New sources of energy system data, such as from low-cost sensors, and the ability to harness that data thanks to increases in computational power and new data science techniques, have enabled the efficient management of complex stochastic power supplies and expanded availability of information to customers and system operators. The disruptions and transformations these innovations bring to the world economy could be as significant as those that followed the rise of electricity and oil a century ago.”

David G Victor, 2018

1.1 Data Could Be the New Oil

According to the Intergovernmental Panel on Climate Change, the electricity sector is the single largest contributor to greenhouse-gas emissions. This sector represents 25% of the total global emissions (Intergovernmental Panel on Climate Change, 2014). Not only does the large-scale use of fossil fuels contribute to climate change; it is also associated with the other two known major international environmental problems: acid precipitation (due to SO₂ and NO_x pollutants) and stratospheric ozone depletion (due to the emission of chlorofluorocarbons, halons and NO_x) (Kalogirou, 2004).

The Paris Agreement, adopted on 12 December 2015, has the long-term goal of limiting the increase in the global average temperature to well below 2°C above the pre-industrial level (United Nations Framework Convention on Climate Change, 2015). This would reduce the risks and impacts of climate change significantly. To attain this target by the year 2100, global greenhouse-gas (particularly CO₂) emissions must rapidly decrease, starting from the year 2020 (Rogelj *et al.*, 2015). According to Rogelj *et al.* (2015), this will require a worldwide transition to net zero-carbon emissions between the years 2045

and 2065. As of August 2018, 194 countries – as well as the European Union – have signed the agreement.¹

Teske *et al.* (2011) provide the Energy [R]evolution 2010 scenario, which analyses the extent to which the recent trends in global energy production and demand affect the chances of reducing CO₂ emissions to 3,7 Gt/a by 2050. This CO₂ emissions target has the potential of limiting the global average temperature increase to below 2 °C. The Energy [R]evolution scenario requires renewable power generation to occupy 40 % of the global primary energy supply by the year 2030, and states that by the year 2050, 95 % of electricity can be produced from renewable energy sources (RES). The European Commission's Energy Roadmap 2050 posits that the share of RES in the gross final energy consumption in the EU could be at least 55 % (European Climate Foundation, 2010). These RES include biomass, geothermal, solar (concentrating solar power and photovoltaic) and wind.

Germany is leading the renewable energy pack in Europe with its *Energiewende* (energy transition), a transformational energy revolution that has allowed the country to obtain 35 % of its electricity from renewables in 2017 (Kunzig and Locatelli, 2015; Bundesverband der Energie- und Wasserwirtschaft, 2017). The *Energiewende* is a citizen-driven movement supported by legislation that allows producers of renewable energy to feed into the grid, earning a feed-in tariff from utilities (Kunzig and Locatelli, 2015). The Bundesverband der Energie- und Wasserwirtschaft also calls the *Energiewende* the “largest national IT project of all time”.

In South Africa, a country with a notable solar resource, net annual concentrated solar power (CSP) energy generation of 1 861 TWh could be achieved. This is at least 3,3 times more than the total electricity requirement forecast for the country by the year 2025 (Fluri, 2009).

According to Bumpus and Comello (2017), utilised RES comprised 10 % of the global electricity generation capacity in 2017. Khatib (2012), in a comment on the IEA World Energy Outlook 2011, puts the share of renewable energy resources in the global primary energy consumption at 13 %. McCrone *et al.* (2018) declare the renewable share to be 12,1 % of electricity produced worldwide in 2017, up from 11 % in 2011.

An average annual investment of US\$1 trillion is required until 2050 for renewable energy to achieve the climate goal set out in the Paris Agreement. As in 2017, only a third of this required annual investment was available (Bumpus and Comello, 2017). Teske *et al.* (2011) posit that global investments of US\$17,9 trillion would be required from 2011 to 2030 for renewables to occupy 40 % of the global primary energy supply. Much of the projected growth in renewable energy resource utilisation is underpinned by an increase in subsidies,

¹The United States of America is intent on withdrawing, but according to Article 28 of the Paris Agreement, will effectively only be allowed to withdraw after 4 November 2020 (United Nations Framework Convention on Climate Change, 2015).



Figure 1.1: The solar settlement, designed by architect Rolf Disch, is located in the solar city of Freiburg, Germany. It is a net producer of energy (Kunzig and Locatelli, 2015; Disch, 2018).

set to rise from US\$64 billion in 2010 to US\$250 billion in 2035. However, considering fiscal austerity, these subsidies cannot be taken for granted (Khatib, 2012).

It is therefore unlikely that investment in renewable energy technology alone will provide the necessary drive to reach the Paris climate goal by the end of the century (see Section 1.4). An important component of the climate mitigation strategy must therefore be energy efficiency (Teske *et al.*, 2011; Khatib, 2012). Energy efficiency also provides the possibility of reducing the financial cost associated with the use of fossil fuels (IPMVP, 2002; Sivaram *et al.*, 2018). Energy efficiency could be achieved by employing smart energy management systems that reduce the total amount of electricity consumption, as well as allow for better integration of variable renewable energy sources (such as solar and wind) into the grid. Remarkable energy savings can be achieved in this way. Building electricity demand has the potential to be reduced by 20–30%, resulting in a reduction of 96% in greenhouse-gas emissions, relative to heating using natural gas boilers (Bumpus and Comello, 2017). On a national scale, Russia could reduce its primary energy consumption by nearly one third by improving its energy efficiency to levels comparable to countries belonging to the Organisation for Economic Cooperation and Development (OECD) (Khatib, 2012).

According to David G Victor in Sivaram *et al.* (2018), new sources of en-



Figure 1.2: The Borkum Riffgrund 1 wind farm in the North Sea is part of the latest push for Germany’s *Energiewende*. It can produce enough power to supply 320 000 German households (Kunzig and Locatelli, 2015).

ergy system data, coupled with improved computational power, enable better management of complex stochastic power systems (such as energy systems containing a large RES component), which results in improved energy efficiency. Victor posits that “the disruptions and transformations these innovations bring to the world economy could be as significant as those that followed the rise of electricity and oil a century ago.”

In 2017, *The Economist* proclaimed “the world’s most valuable resource is no longer oil, but data.” This might sound like hyperbole, but it should be considered that the five most valuable listed firms in the world all deal in data: Apple, Alphabet (Google), Microsoft, Amazon and Facebook (Parkins, 2017). The Internet of Things attaches a digital trace to more and more day-to-day human activities, such as taking a run or making a cup of coffee. It is estimated that a self-driving car will generate 100 GB of data per second, and more valuable insights are extracted from big datasets such as this by machine learning algorithms (Parkins, 2017). This proliferation of data has created a data ecosystem that is almost inescapable. No wonder, therefore, that an industrial giant such as GE now markets itself as a data firm (General Electric Company, 2016).

1.2 Digitalisation and the Smart Microgrid

The drive for digitalisation in the energy sector could go hand in hand with an increase in the utilisation of RES. This could potentially lead to a global reduction in the emission of greenhouse gases (Sivaram *et al.*, 2018). Zhou *et al.* (2016) suggest that smart energy management through energy big-data analysis could provide solutions to some of the challenges currently faced by the traditional energy industry. These challenges include operational efficiency and cost control (Momoh, 2009), system stability and reliability (Amin, 2008), management of renewable energy (Altin *et al.*, 2010), energy efficiency (Zhou *et al.*, 2015), consumer engagement (Aalami *et al.*, 2010), environmental issues (Zhou *et al.*, 2015) and service improvement (Aalami *et al.*, 2010).

Furthermore, according to Lidiya Sekaric (Sivaram *et al.*, 2018), digital innovations will enable the transition from centralised electricity generation, transmission and distribution to a fully distributed grid architecture, which is more efficient and suitable for the integration of RES. Sekaric further argues that digital technologies could ensure that the deployment of distributed solar panels stabilises, rather than destabilises, the grid by the addition of smart inverters that equip the grid to actively cope with voltage variation. However, David G Victor argues that digital technologies might also reduce the cost of fossil fuel extraction, leading to greater consumption and pollution (Sivaram *et al.*, 2018).

Central to the concept of smart renewable energy management is the smart grid. Mr Louis Lagrange (2019), a certified energy manager (CEM) and Head of Engineering Sciences at the University of the Free State, states,

Smart grid energy management represents an effective digital technology to link and subsequently manage electrical load management and generation control through the pre-management of end-user patterns. As part of integrated demand management this constitutes a feasible risk-lowering strategy while simultaneously optimising both energy utilisation and capital asset usage.

Ron Melton, project director of Battelle’s Pacific Northwest Smart Grid Demonstration Project, thinks of the smart grid as “the convergence of the Internet and a lot of intelligent devices and sensors spread throughout the power system” (Groenfeldt, 2012). The smart microgrid typically contains distributed energy resources (DER), AC and DC loads and storage units, as well as control and communication modules (Ma and Ma, 2017). The smart-grid concept is therefore closely related to the internet of things (IoT) and industry 4.0.

1.3 Big-data-driven Smart Energy Management

Zhou *et al.* (2016) suggest that big-data-driven smart energy management comprises seven major steps:

- Data collection, transmission and storage
- Data cleaning and preprocessing
- Data integration and feature selection
- Data mining and knowledge discovery
- Representation, visualisation and application
- Intelligent decision-making and real-time interaction
- Smart energy management

Step 4 (data mining and knowledge discovery) is considered the key step in big-data-driven smart energy management (Zhou *et al.*, 2016). This step also allows for better integration of renewable energy resources into the energy system. Wind and solar are both variable sources of energy, as their output are affected by weather conditions. Variability of these resources can be on a minutely, hourly and annual basis (Ma and Ma, 2017). This poses a risk to the stability of energy networks. It is therefore important that reliable renewable energy availability forecasts be made in order to synchronise energy supply with demand profiles (Zhou *et al.*, 2016). Weather data (such as humidity, temperature, atmospheric pressure and wind speed) and data obtained from geographic information systems (GIS) are modelled to obtain renewable energy production forecasts. This could allow for better integration of distributed photovoltaic and wind energy sources into the electricity grid, as well as provide information that could lead to better energy management. It could also contribute to more efficient use of traditional fossil-fuel energy sources, lowering energy costs and mitigating adverse effects on the environment.

Ma and Ma (2017) classify forecasting techniques into three basic categories: physical models, statistical models and intelligent computational models. This study will make use of Gaussian process regression, which is a non-parametric statistical regression process that falls under the Bayesian paradigm (Yang *et al.*, 2018; Maritz *et al.*, 2018).

Gaussian process regression can be employed as a big-data machine learning methodology and as such its potential as a forecasting tool for photovoltaic and wind-power generation capacity as well as load profiles should be investigated. This is motivated in more detail in a subsequent section.

1.4 Trends in Early-stage Investment

A record 157 GW renewable power has been commissioned globally in 2017. This is up from 143 GW in 2016 and 69 GW in 2010 (McCrone *et al.*, 2018). McCrone *et al.* (2018) also found that global investment in renewable energy increased by 2 % in 2017 to US\$279,8 billion (US\$103 billion were invested in new fossil-fuel power-generation infrastructure in 2017). Renewable energy (excluding large hydropower) subsequently contributed 61 % to the net power-generation capacity added worldwide in 2017 (solar alone accounted for 38 %) (McCrone *et al.*, 2018). Furthermore, the WilderHill New Energy Global Innovation Index (NEX), which tracks the performance of companies that offer innovative technologies and services that focus on clean energy generation and energy efficiency has risen by 28 % in 2017, outperforming the S&P 500 (McCrone *et al.*, 2018). The NEX has, however, underperformed the S&P 500 in 2018 (Financial Times, 2019). Keep in mind that equity is a volatile asset class and index performance should be judged in the long term (at least more than five years).

According to Bloomberg New Energy Finance, in its Clean Energy Investment Trends 2018 report, global clean energy investment totalled US\$332,1 billion in 2018. Offshore wind projects comprised US\$25,7 billion of the total invested. This constitutes an increase of 14 % from 2017 (Bloomberg, 2019). Table 1.1 summarises the five largest renewable energy deals in 2018.

Table 1.1: The top renewable energy deals for 2018 (Bloomberg, 2019).

Project Name	Country	Sector	Capacity (MW)	Value (US\$ billion)
Moray Firth Offshore Wind Farm	United Kingdom	Wind	950	US\$3,3 billion
Triton Knoll Offshore Wind Farm	United Kingdom	Wind	860	US\$2,6 billion
NOORm Midelt Solar Portfolio	Morocco	Solar	800	US\$2,4 billion
Borssele III & IV Offshore Wind Farm	Netherlands	Wind	731	US\$1,5 billion
Guangdong Baolihua New Energy Shanwei Lufeng Houhu Offshore Wind Farm	China	Wind	500	US\$1,5 billion

Digital innovation has been a significant driver of change in the energy industry. Low-cost sensors and an increase in computational power have allowed for efficient management of complex stochastic energy systems. A clean-energy revolution could be imminent if digital technologies allow for the successful orchestration of clean and complex energy systems (Sivaram *et al.*, 2018). Investment trends seem to support the notion of a growing digital presence in energy systems. Worldwide, a record US\$18,5 billion have been invested in the deployment of smart electricity and gas meters in 2017. This is an increase of 35% from 2016 (McCrone *et al.*, 2018).

Bumpus and Comello (2017) define wave-1 technologies as renewable-energy supply technologies (such as PV modules) and wave-2 technologies as “distributed, control and efficiency technologies encompassing communication, analytics and automation software”. Table 1.2, adapted from Bumpus and Comello (2017), describes wave-1 and wave-2 technologies in more detail.

Bumpus and Comello (2017) measure the prevalence of investment in wave-1 and wave-2 technologies by considering the number of transactions between wave-1-focused firms, wave-2-focused firms and early stage investors. By analysing 6 769 transactions within the international energy, utilities and clean-tech sectors between 2003 and 2015, Bumpus and Comello (2017) have found that wave-1 investments have become less prevalent, giving way to wave 2-investments. It was also found that wave-2 investment transactions require, on average, 54% less capital than wave-1 investments.

Figure 1.3 shows the extent to which investment in wave-2 technologies has been gaining the lead on wave-1 investment. It can be seen that wave-2 has been receiving the majority of early-stage funding transactions since 2013.

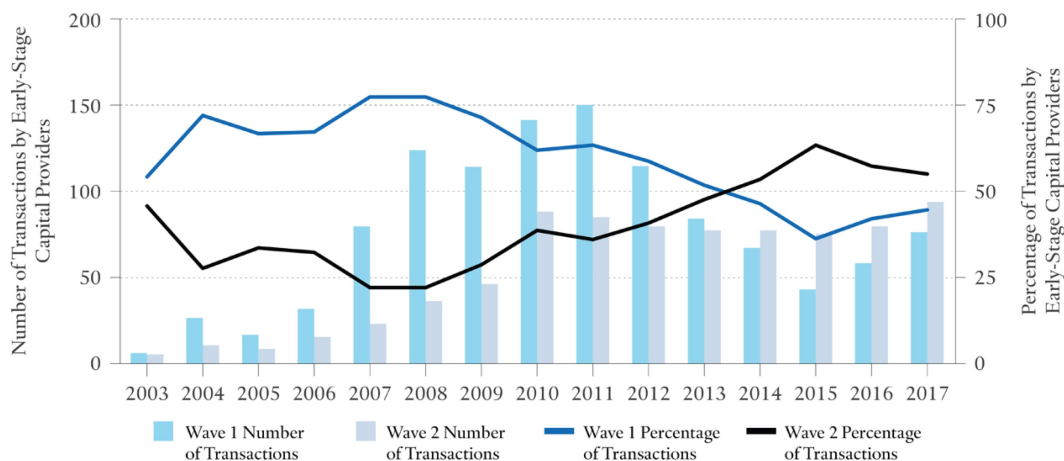


Figure 1.3: Early-stage investment trends for wave-1 and wave-2 technologies (Bumpus and Comello, 2017).

Figure 1.4 shows the average transaction size of wave-1 and wave-2 invest-

Table 1.2: Description of wave-1 and wave-2 technologies (Bumpus and Comello, 2017).

	Wave-1	Wave-2
Characterisation	Technologies focused on hardware components required for renewable energy generation and storage systems	Technologies focused on software applications and demand-side solutions
Example technologies	PV modules, wind-turbine blades, biomass digesters, battery cells, inverters	Sensors, actuators, controllers, electronics, communication, analytics, management and automation software, responsive end-use elements such as dimmable windows or shading, intelligent lighting and smart thermostats
System size (energy generation)	Megawatt to gigawatt	Watt to megawatt
Location on grid	Majority central facilities with minority distributed generation	All distributed
Owner and operator	Electric utilities and/or specialised power producers	Customers, solutions providers, specialised power producers and utilities

ments. The fact that the average transaction size for wave-2 investments is smaller than that of wave-1 investments, allows for more wave-2 solutions to be funded per unit capital employed. This is attractive to early-stage investors, since it allows for capital-light experimentation (Bumpus and Comello, 2017).

The rise in wave-2 investment seems to indicate that the innovations driving change in the energy sector are focused on digitalisation.

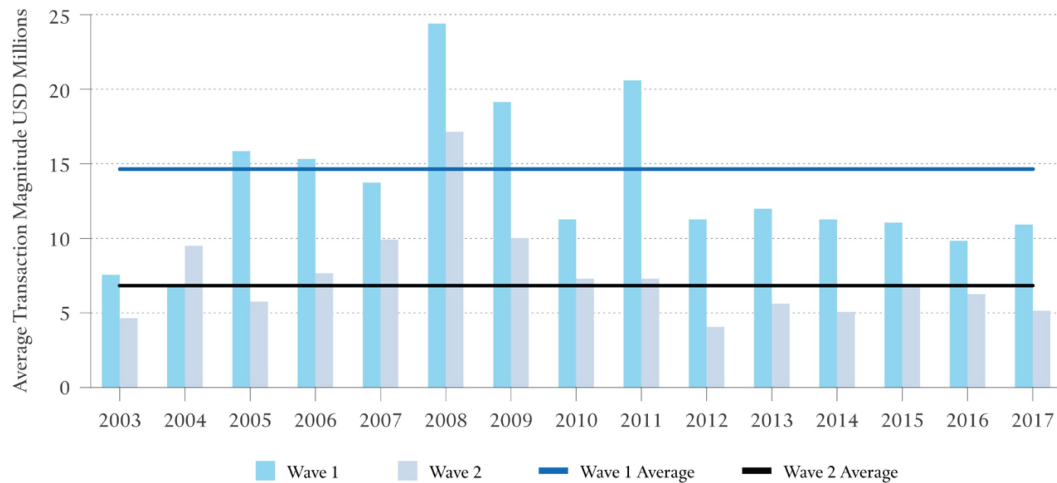


Figure 1.4: Average transaction sizes for wave-1 and wave-2 firms (Bumpus and Comello, 2017).

1.5 The Need for a Well-defined Energy System Model

The South African government's Ten-Year Plan for Science and Technology identifies the protection of the environment and access to affordable, safe, clean and reliable energy as principle energy challenges (Department of Science and Technology Republic of South Africa, 2008).

According to the South African Department of Energy's integrated resource plan (IRP), which was released for public comment in August 2018 (the first revision has since been submitted on 6 March 2019 and is scheduled for completion in September 2019), the energy demand for the period 2010–2016 proved to be lower than anticipated. In contrast to an expected growth of 3% in electricity demand, electrical energy send-out declined at an average compound rate of 0,6% per year (Department of Energy Republic of South Africa, 2018). The IRP identifies lower-than-expected GDP growth, improved energy efficiency and an increase in embedded generation, such as rooftop photovoltaic installations as the underlying causes for this decline in electricity demand. The IRP nevertheless proposes an increase in photovoltaic power generation capacity from 1 474 MW in 2018 to 7 958 MW in 2030 and an increase in wind-power-generation capacity from 1 980 MW in 2018 to 11 442 MW in 2030. This will bring the combined component of wind and photovoltaic generation to 25,6% of the total installed capacity in South Africa in 2030, meaning that a quarter of South Africa's power generation could come from variable power sources. In light of this, the IRP requests that work be done to determine the impact that the increasing share of variable generation might have on power system operation in South Africa. The IRP therefore calls for an in-depth

analysis to find solutions to overcome grid stability issues resulting from non-synchronous and distributed generation (Department of Energy Republic of South Africa, 2018). To do this, a well-defined energy system will be required.

South Africa has an exceptional solar resource (Brooks *et al.*, 2015). Proper utilisation of this resource, however, requires seamless integration of solar energy into the national grid. This is a challenge, since the availability of solar energy is variable. The same goes for wind energy.

Figure 1.5 illustrates the availability of solar power, together with the electrical power demand at the Stellenbosch University Faculty of Engineering. Without storage elements in the power system, there are periods where the solar power is not sufficient, while there are also times when a surplus of solar power is available. To mitigate this problem, carefully considered supply-side and demand-side management is required without neglecting the importance of energy storage. If the availability of solar energy can be forecast accurately, more informed energy management decisions can be made to balance the power system better. Figure 1.5 was plotted using data from the Stellenbosch University Main Campus electricity meters, which are administrated by Terra Firma Solutions.

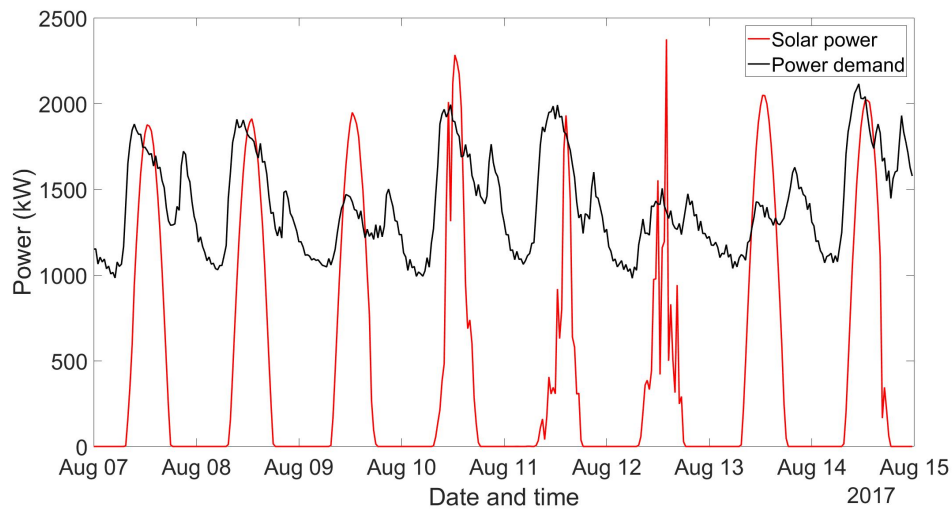


Figure 1.5: Power demand and solar power potential at Stellenbosch University Faculty of Engineering.

A well-defined statistical model of the energy system under consideration is important for accurate renewable energy supply forecasting, as well as demand-side forecasting. The quality of grid information (energy system data) has a significant impact on the success of smart-grid applications (Yang *et al.*, 2018). It is therefore important to consider the inherent uncertainty in the data when interpreting a hybrid energy system model. This uncertainty can be due to inaccurate measurements and/or incorrect assumptions in energy system models

(IPMVP, 2002). One way of mitigating uncertainty is to apply agreed-upon methodologies for measurement and verification (M&V), such as the International Performance Measurement and Verification Protocol (IPMVP, see (IPMVP, 2002)), which serves as an industry standard for energy M&V (Hamer *et al.*, 2017) (also see the IEC 61400-12-1:2005(E) standard of the International Electrotechnical Commission). Better M&V generally results in an increased level of confidence in the forecast behaviour of an energy system (IPMVP, 2002). However, Carstens *et al.* (2018) point out that the classic frequentist approach to obtaining well-defined uncertainty in data measurement can lead to a misinterpretation of the uncertainty it conveys. The use of Gaussian process regression in energy-system behaviour forecasting could provide a method for better quantification of uncertainty (Yang *et al.*, 2018). The probabilistic nature of Gaussian models means that uncertainty is well defined (Carstens *et al.*, 2018).

Determining the parameters that govern renewable energy supply models can be time consuming and expensive. This is especially true for energy systems that have high sensitivity to complex exogenous parameters and usage patterns, such as an energy system with a significant renewable energy component. In cases such as this, Gaussian process regression yields more robust predictions of energy production (Burkhart *et al.*, 2014), since complex energy modelling parameters are learned by a machine, allowing for improved confidence in energy system data (Carstens *et al.*, 2018). Another attractive property of Gaussian process regression is that little knowledge of the underlying energy model is necessary, thus reducing the induced error associated with the model (Reddy and Claridge, 2000). Within this context, the main purpose of the Gaussian process and the Bayesian paradigm is therefore to infer a mathematical structure that describes the relationships and dependencies between individual observations without prior knowledge of the parameters governing the energy system (Maritz *et al.*, 2018). These dependencies and relationships can be useful for energy system behaviour prediction. The predictive power and non-parametric nature of Gaussian process regression as a machine learning algorithm within the Bayesian paradigm could contribute to smart energy management, energy savings and renewable-energy market penetration.

1.6 Research Objectives

In light of the quest for the efficient integration of variable renewable energy sources into the grid (or micro-grid), this thesis investigates the potential of Gaussian process regression for data-driven renewable energy management. This is done by adhering to the following research structure:

- Review the use of machine learning techniques in energy system analysis.
- Study the mathematical basis of the Gaussian process regression.

- Investigate kernel structures and their suitability for different applications.
- Construct customised kernels to train Gaussian process regression algorithms on Sauran data in order to forecast the behaviour of wind and solar energy.
- Explore the effect of Gaussian process regression on the values of Weibull parameters when calculated from interval-deficient wind speed data.
- Discuss possible applications of Gaussian process regression in smart renewable energy management.

1.7 Original Research Contribution of this Study

- Wind and solar data have been translated into the Bayesian paradigm.
- Gaussian process regression has been explored as a renewable energy resource availability predictor.
- Gaussian process regression has been explored as a solution to renewable energy resource meter failure.
- A possible scenario for the application of Gaussian process regression in smart renewable energy management has been discussed.
- The application of Gaussian process regression to the calculation of Weibull parameters from interval-deficient data has been explored.

Chapter 2

A Review of Machine Learning Techniques in Energy System Analysis

Mosavi *et al.* (2019) have undertaken a systematic review of the state-of-the-art of machine learning models used in energy system modelling. The Thomson Reuters Web-of-Science and Elsevier Scopus were utilised to identify quality, original and high-impact papers on machine learning in energy systems. The identified models were then classified and reviewed. Four steps were undertaken by Mosavi *et al.* (2019) in the review process:

- Identify a database of relevant articles by using search terms such as energy system, machine learning, neural network, support vector or deep learning.
- Add to database or exclude based on abstract and keywords.
- Detailed consideration of papers.
- Categorise papers based on machine learning technique covered.

By following this process, Mosavi *et al.* (2019) identified 70 original papers for review. Ten machine learning models that are frequently used in energy system analyses were identified by Mosavi *et al.* (2019):

- artificial neural networks
- multilayer perceptron
- extreme learning machines
- support vector machines
- wavelet neural networks

- adaptive neuro-fuzzy inference systems
- decision trees
- deep learning
- ensembles
- advanced hybrid machine learning models

A summary of this review is presented and Gaussian process regression is introduced.

2.1 Artificial Neural Networks

An artificial neural network is a framework that structures machine learning techniques for processing complex data inputs. The structure of the artificial neural network mimics the human nervous system, with the fundamental unit being an artificial neuron. Artificial neural networks simplify multivariate problems and can be used for handling noisy energy system data. It has been used to optimise the generation capacity of renewable energy technology, mitigating wind power fluctuation and to predict electricity cost (Mosavi *et al.*, 2019).

Figure 2.1 illustrates the architecture of a typical artificial neural network. The network has three layers (input, hidden and output) consisting of nodes. The lines between the nodes represent the flow of information. Although information only flows from the input to the output in this example, feedback paths could be present in artificial neural network architectures. The nodes of the input layer are passive (no data modification takes place) and only serve to duplicate the data to be passed on to the first of the hidden layers. In contrast, the nodes of the hidden and output layers modify the data through mathematical functions. Any number of layers and nodes can be present in a neural network, although the three-layer architecture is the most common (Smith, 1997).

Figure 2.1 is an example of the application of an artificial neural network to forecast PV power output. The forecast was done by Nespoli *et al.* (2019) using a multilayer perceptron-based model. Multilayer perceptron is a specialised neural network (Nespoli *et al.*, 2019).

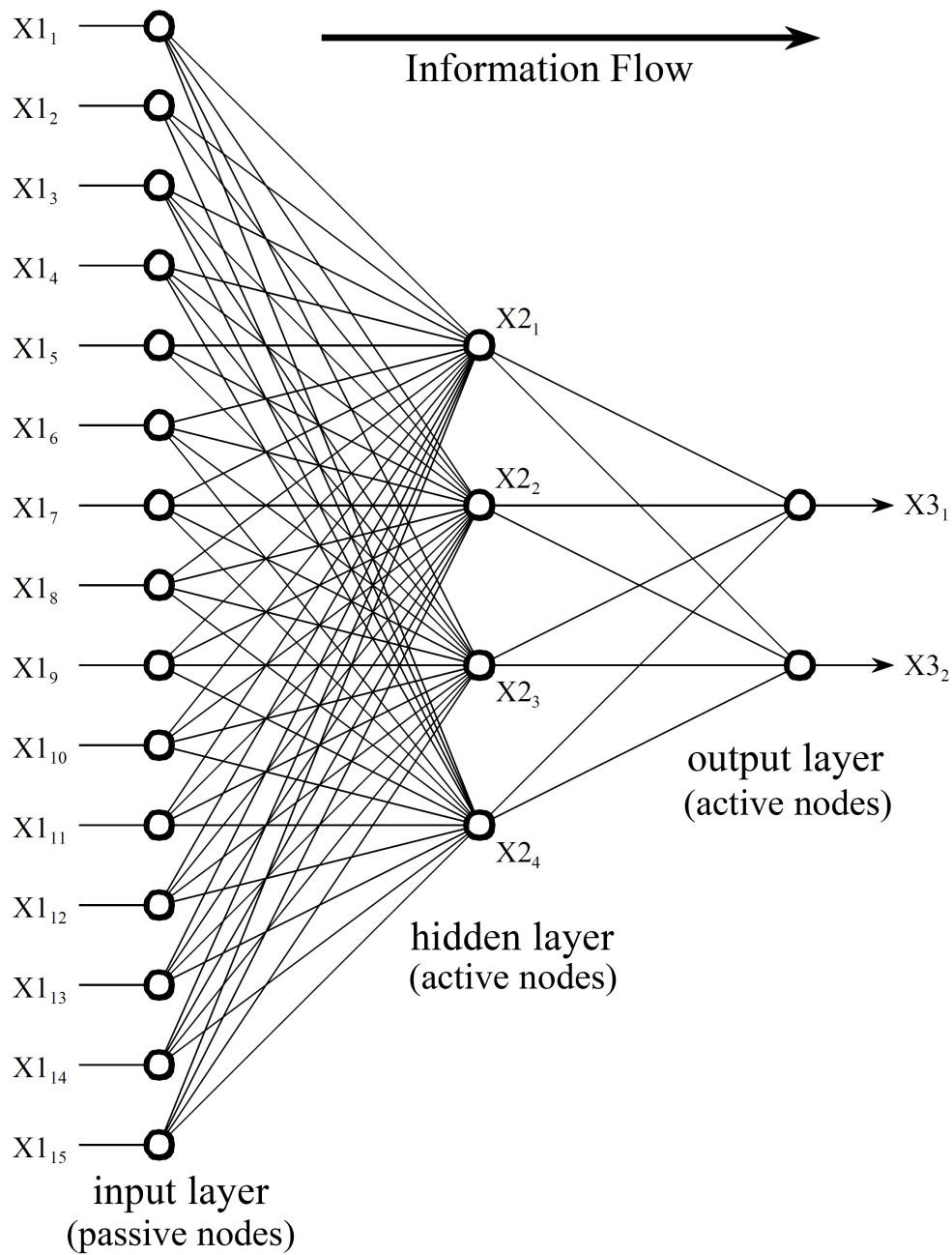


Figure 2.1: A three-layer artificial neural network architecture with full interconnection. Adopted from Smith (1997).

2.2 Multilayer Perceptron

A multilayer perceptron is a specialised neural network for application in energy systems and engineering. Important papers on the application of the multilayer perceptron are listed Mosavi *et al.* (2019).

- Ranking of different potential power plant projects (Shimray *et al.*, 2017).
- Hourly global solar radiation prediction (Loutfi *et al.*, 2017).
- Prediction of solar system power output (Kazem *et al.*, 2017).
- Electricity load forecasting (Chahkoutahi *et al.*, 2017).
- Day ahead prediction of hourly global solar radiation (Ahmad *et al.*, 2015).

Figure 2.2 is an example of the application of a multilayer perceptron for the forecasting of photovoltaic (PV) power output.

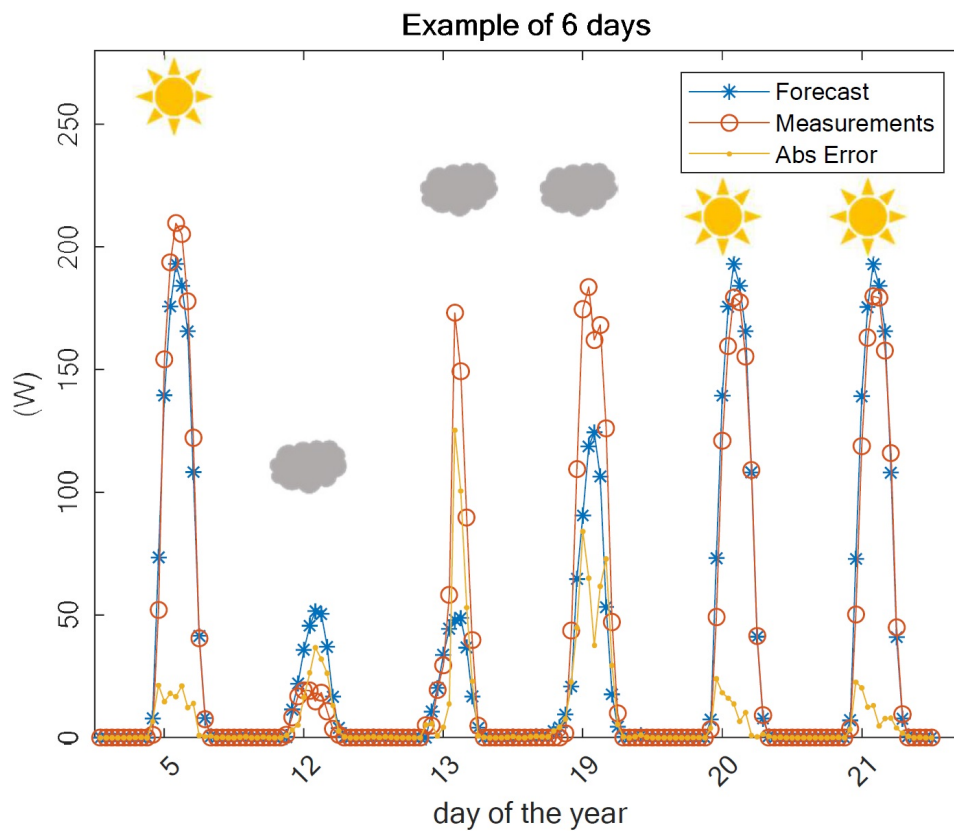


Figure 2.2: Measured and multilayer perceptron forecast PV power output. Adopted from Nespoli *et al.* (2019).

2.3 Extreme Learning Machines

Extreme learning machines employ advanced forms of artificial neural networks. These neural networks have a high speed of learning and an advanced ability to generalise (Mosavi *et al.*, 2019). Figure 2.3 illustrates solar power production forecasts obtained by training an extreme learning machine algorithm on solar irradiance and temperature data (Al-Dahidi *et al.*, 2018).

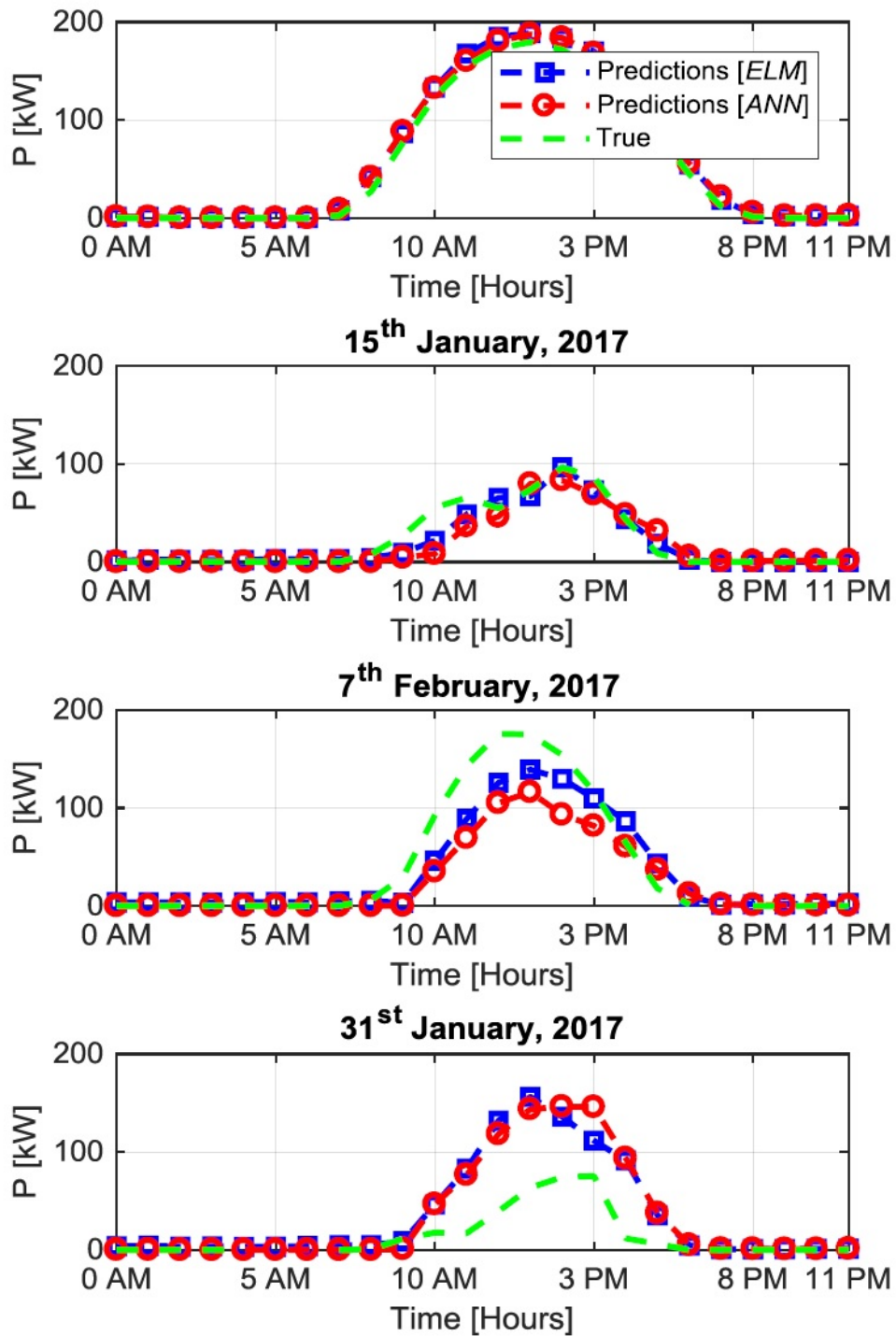


Figure 2.3: Prediction of solar power production by artificial neural network and extreme learning machine algorithms. Adopted from Al-Dahidi *et al.* (2018).

2.4 Support Vector Machines

Support vector machines outperform other methodologies in pattern recognition, classification and regression. Support vector machines have been employed successfully in load forecasting due to their ability to make generalisations. Some other notable applications include: estimation of optimum oxygen-steam ratios in gasification of coal, classification of power quality disturbance, improving the estimation accuracy of solar irradiance based on photovoltaic electrical parameters and the modelling of electricity markets (Mosavi *et al.*, 2019). Serafin and Uniejewski (2019) use probabilistic forecasting techniques to obtain day-ahead electricity price forecasts, while Su *et al.* (2019) predicts the spot price of natural gas using support vector machines (see Figure 2.4).

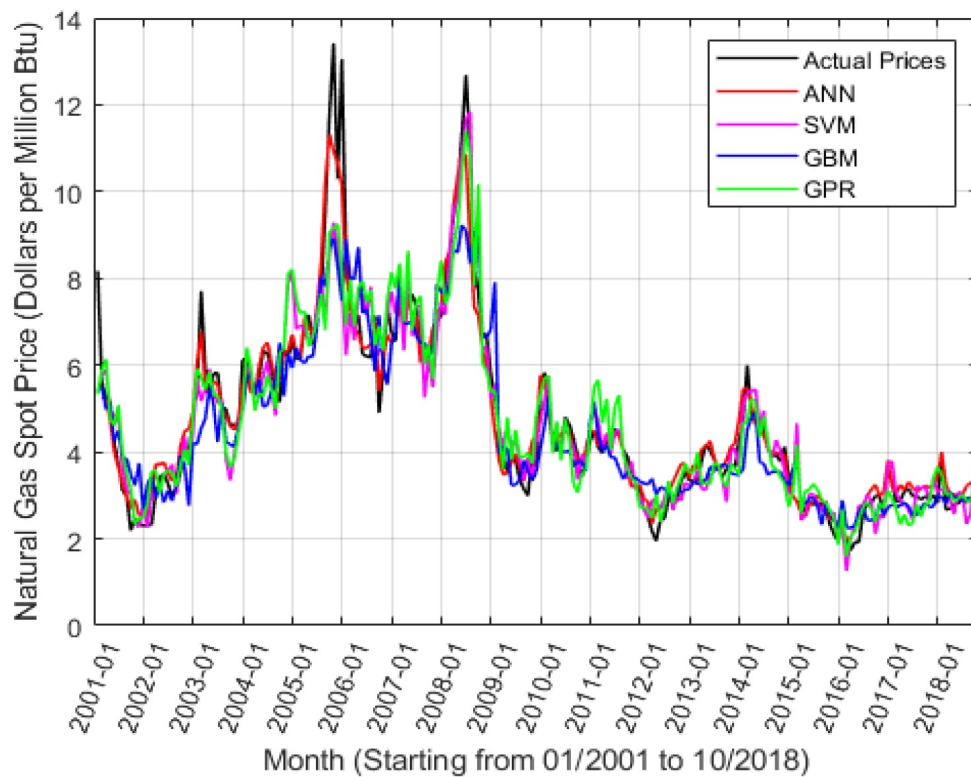


Figure 2.4: Natural-gas spot price prediction using four machine learning techniques: artificial neural network, support vector machine, gradient boosting machine and Gaussian process regression. Adopted from Su *et al.* (2019).

2.5 Wavelet Neural Networks

Wavelet neural networks exhibit quick convergence and require relatively small training datasets. Wavelet neural networks have been used to predict the behaviour of renewable energy sources as well as heat load. They were also used to predict wind power and it was shown that wavelet neural networks are more accurate than artificial neural networks for this application (Mosavi *et al.*, 2019).

2.6 Adaptive Neuro-fuzzy Inference Systems

This machine learning method combines fuzzy logic and neural networks and exhibits characteristics of both techniques. Adaptive neuro-fuzzy inference has been used to estimate the temperature of photovoltaic systems based on wind velocity and direction, environmental temperature, irradiance, atmospheric pressure, relative humidity and PV power output. A low root-mean-squared error was reported. The power demand of a factory was also forecast using adaptive neuro-fuzzy inference (Mosavi *et al.*, 2019).

2.7 Decision Trees

Decision trees are very powerful inference techniques and have been used successfully in numerous energy system applications. Some applications of decision trees in energy systems analysis are listed (Mosavi *et al.*, 2019).

- Railway electric energy systems optimal operation (Aguado *et al.*, 2018).
- Prediction of risk of a blackout in electric energy systems (Kamali *et al.*, 2017).
- Energy storage planning and energy controlling (Moutis *et al.*, 2016).
- Total cost minimisation in energy systems for the prosumers' buildings (Ottesen, 2016).

2.8 Deep Learning

Deep learning stacks multilayer information processing modules to model the hierarchical characterisation of data patterns. Deep learning has increased in popularity with the increase in computational power and the availability of large datasets. Applications of deep learning are listed (Mosavi *et al.*, 2019).

- Battery state-of-charge estimation (Chemali *et al.*, 2018).

- Household electricity demand forecasting (Coelho *et al.*, 2017).
- Estimation of the power consumption of individual appliances in the distribution system (Kim *et al.*, 2017).
- PV power forecasting (Wang *et al.*, 2017).
- Prediction of building energy consumption (Mocanu *et al.*, 2016).

2.9 Ensembles

Ensemble, in the context of machine learning, refers to a combination of multiple machine learning algorithms. Ensembles have been used in energy system modelling (Mosavi *et al.*, 2019):

- Cooling load forecasting in buildings (Fu, 2018).
- Non-linear fault features extraction (Changfeng *et al.*, 2017).
- Prediction of underground water dam level (Hasan and Twala, 2016).
- Human energy expenditure estimation (Gjoreski *et al.*, 2015).
- Forecasting of building electricity demand (Burger and Moura, 2015).

2.10 Hybrid Models

Hybrid models combine machine learning as well as artificial intelligence methods. Hybrid models usually consist of two parts: prediction and optimisation. It is especially suitable when high accuracy prediction is required (Mosavi *et al.*, 2019).

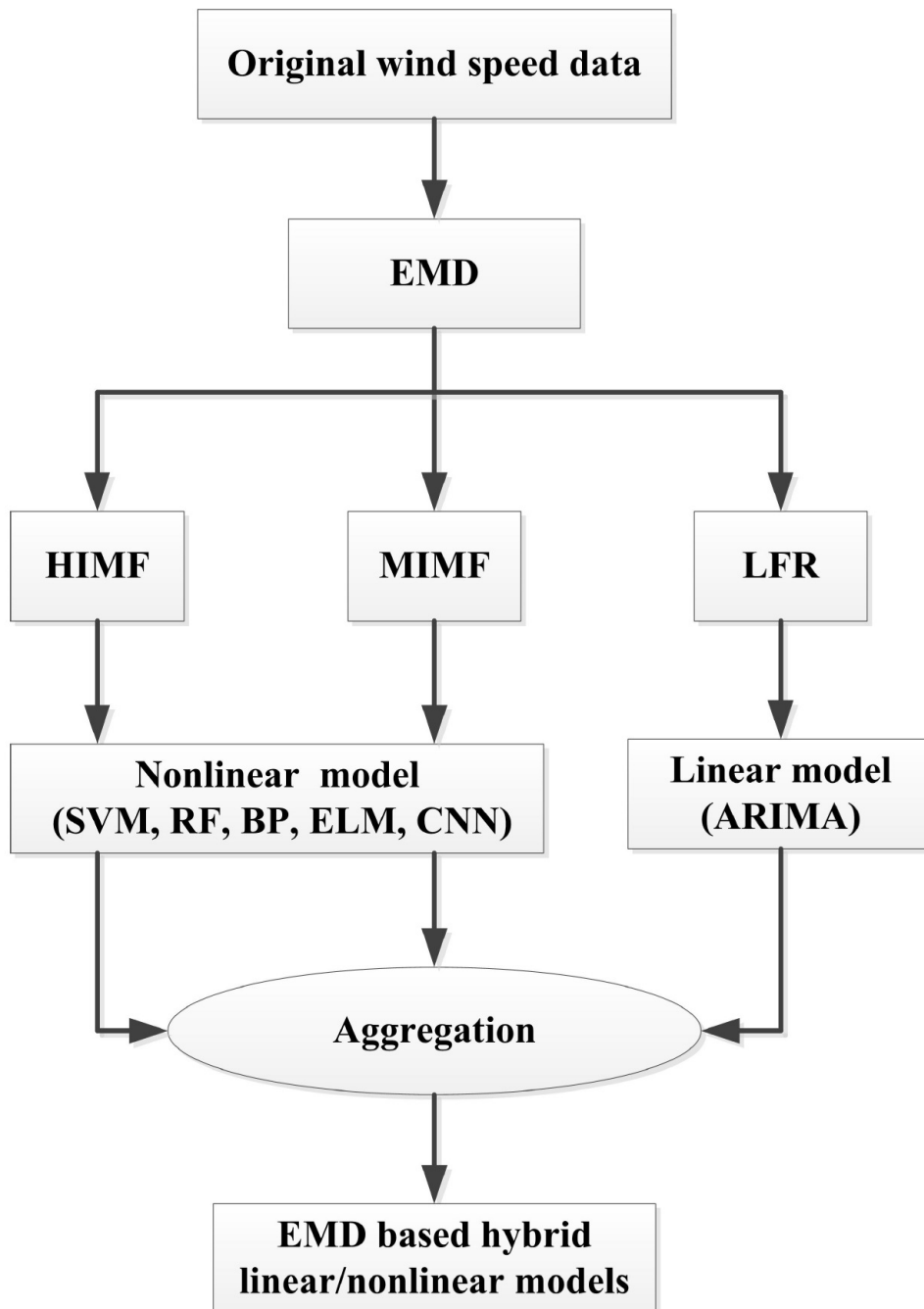


Figure 2.5: Visualisation of a hybrid machine learning model utilising empirical mode decomposition (EMD), high-frequency intrinsic mode functions (HIMF), medium-frequency intrinsic mode functions (MIMF), low-frequency residuals (LFR), support vector machines (SVM), random forest (RF), back propagation (BP), extreme learning machines (ELM), convolution neural network (CNN) and autoregressive integrated moving average (ARIMA). Adopted from Han *et al.* (2018).

2.11 Popular Applications of Machine Learning in Energy Management

Ma and Ma (2017) review state-of-the-art forecasting algorithms for power supply and load demand. Techniques such as numerical weather prediction (NWP), autoregression (AR), moving average (MA), autoregression moving average (ARMA), Kalman filter technique, artificial neural network (ANN), support vector regression (SVR) and genetic algorithm are assessed in terms their application to power generation and load forecasting within a microgrid.

Javaid *et al.* (2017) propose demand side management (DSM) that makes use of the genetic algorithm, teaching learning-based optimisation (TLBO), the enhanced differential evolution (EDE) algorithm and the enhanced differential teaching learning algorithm (EDTLA). DSM is implemented in a smart-grid context. According to South Africa's power utility, Eskom, DSM is important in promoting energy efficiency as it reduces the demand on the electricity network, delays the need for new power stations to be built and keeps electricity costs down. Eskom manages energy demand through a set of incentive based mechanisms in the residential, commercial and industrial sectors. Eskom estimates that 58 935 GWh have been saved since implementing demand side management programs (Eskom, 2016). This is illustrated in Figure 2.6.

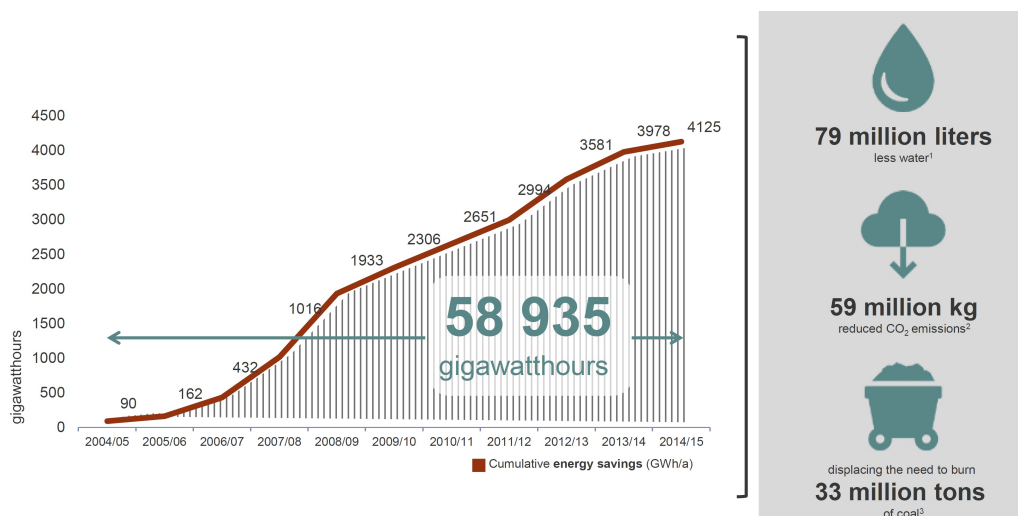


Figure 2.6: Estimated cumulative energy savings achieved since Eskom has started implementing demand-side management programs, also reducing the environmental footprint of energy generation (Eskom, 2016).

Martín *et al.* (2010) compare statistical models based on time series data as applied to the prediction of half-daily values of global solar irradiance with a temporal horizon of three days. These predictions are applied to the energy production planning of solar thermal power plants.

2.12 Gaussian Process Regression in Energy Systems Analysis

Prakash *et al.* (2018) introduce a building energy forecasting method based on Gaussian process regression. It is shown that the Gaussian process regression method produces more accurate forecasts when compared to other state-of-the-art forecasting algorithms using the mean absolute percentage error (MAPE). It has been tested using three datasets: electricity consumption data of Carnegie Mellon University as well as cooling load and lighting load data of the Y2E2 Building at Stanford University. Figure 2.7 plots the values of the mean absolute percentage error of prediction for the Carnegie Mellon University electricity consumption data, comparing the forecasts achieved by Gaussian process regression, support vector regression, random forest and autoregressive integrated moving average.

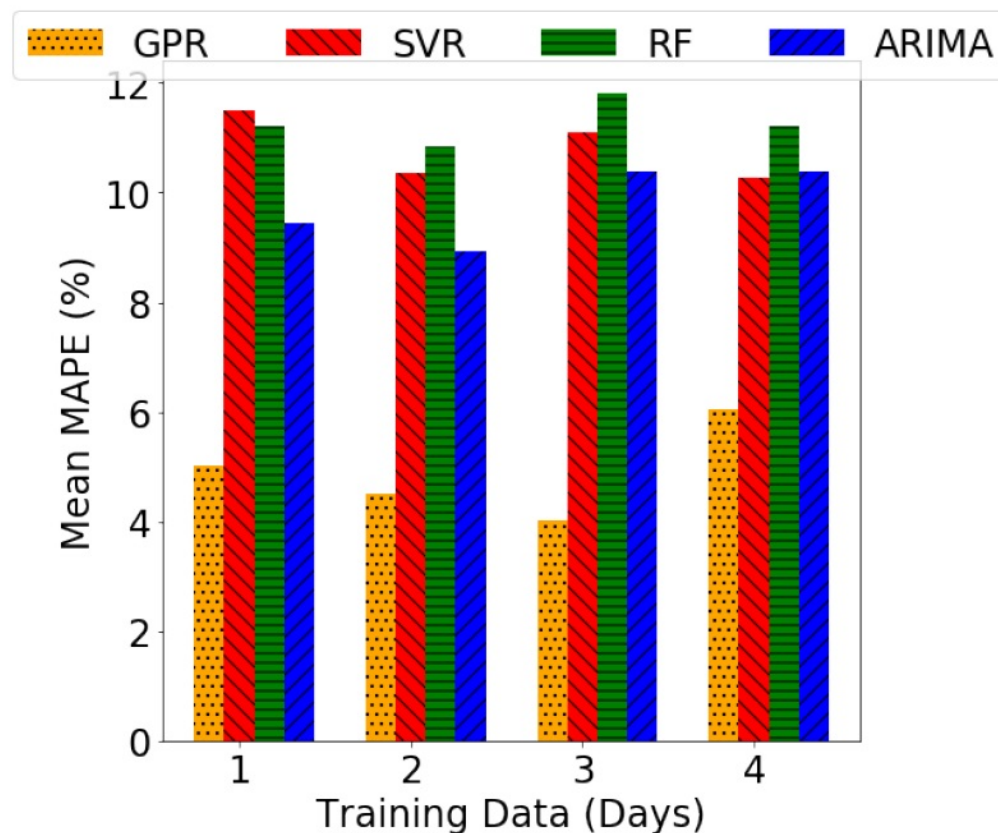


Figure 2.7: Mean absolute percentage error of prediction for Gaussian process regression, support vector regression, random forest and autoregressive integrated moving average (Prakash *et al.*, 2018).

Prakash *et al.* (2018) also investigate the characteristics of energy consumption data in order to develop robust covariance functions. Several kernels are combined to form the covariance function for the Gaussian process regression in order to model long-term building energy-load data. The kernel used by Prakash *et al.* (2018) when evaluating the performance of short-term Gaussian process regression load forecasting is a linear kernel with noise and a constant mean function, given by

$$K(x, x') = \theta_1^2 \times (x - c) \times (x' - c) + \theta_2^2 \delta_{x, x'} \quad (2.1)$$

and

$$m(x) = a \quad (2.2)$$

The hyperparameters – that are to be learned based on the dataset, are given by θ_1 and θ_2 , while δ is an independent covariance function known as Kronecker's delta function. In this case, δ represents the noise.

Equation 2.1 is an example of a kernel constructed to represent some inherent structure (signature) identified within a dataset. The better the insight into the physical structure of the data and encoding of this structure into the covariance function, the more accurate forecast results will be (Prakash *et al.*, 2018). A practical guide to kernel selection is given by Maritz *et al.* (2018).

Yang *et al.* (2018) propose Gaussian process quantile regression (GPQR) for the probabilistic formulation and prediction of the power load in a smart microgrid. Three commonly used kernel functions for the construction of Gaussian process models are also compared. The effectiveness of the GPQR method is assessed by application to a real dataset. It was found that GPQR methods achieve satisfactory prediction performance. It was also found that the squared exponential covariance function serves as an excellent kernel.

Wågberg *et al.* (2016) quantify the prediction performance of Gaussian process regression by providing a measure of the uncertainty associated with Gaussian process prediction.

Parisio *et al.* (2016) develop a microgrid management system within a stochastic framework. An experimental case study is also undertaken, which demonstrates the effectiveness of the proposed stochastic management system.

Polanco Vasquez *et al.* (2018) optimise the economic energy dispatch within a microgrid by considering known predictions of electricity demand, solar radiation and wind speed. The elements in the microgrid are considered a photovoltaic panel, a wind turbine, electric vehicles, a storage system and a point of coupling between the microgrid and the main grid. A case study, using climatological data in Almeria city, photovoltaic data and room load is done to demonstrate the effectiveness of the microgrid energy dispatch algorithm.

Gaussian Processes for Machine Learning (Rasmussen and Williams, 2006) provides a systematic and unified treatment of the application of Gaussian process models to machine learning. A wide range of connections to existing models is also covered, while detailed algorithms are presented.

Gaussian process regression in machine learning updates the machine's "belief system" as new information becomes available, by conditioning the Gaussian on observed data. This approach to learning reminds of what James Clerk Maxwell wrote in 1850 in a letter to Scottish author Lewis Campbell (Edinburgh, 3 September 1830 – Locarno, 25 October 1908):

Now, as human knowledge comes by the senses in such a way that the existence of things external is only inferred from the harmonious testimony of the different senses, Understanding, acting by the laws of right reason, will assign to different truths (or facts, or testimonies, or what shall I call them) different degrees of probability. Now, as the senses give new testimonies continually, and as no man ever detected in them any real inconsistency, it follows that the probability and *credibility* of their testimony is increasing day by day, and the more man uses them the more he believes them. He believes them. What is believing? When the probability (there is no better word found) in a man's mind of a certain proposition being true is greater than that of its being false, he believes it with a proportion of faith corresponding to the probability, and this probability may be increased or diminished by new facts.

2.13 Summary

From a preliminary analysis of the literature, it can be concluded that the use of big-data and digital technology in energy systems is on the rise. The digitalisation of the microgrid could create the opportunity for better integration of variable renewable energy sources (RES) into the power network. Various algorithms exist for use in renewable energy resource behaviour forecasting; however, the Gaussian process regression algorithm for machine learning presents unique advantages in this regard, while it seems to have been underutilised in energy systems modelling.

This project is seen as a contribution to the use of machine learning techniques to address variability in renewable energy as well as smart grid demand side management. This could have a positive effect on energy usage and cost savings. The project thus aims to assist in enhancing effective utilisation of the world's abundant renewable energy resources. It therefore aims to contribute to increasing the share of renewable energy in the world's energy mix in order to mitigate catastrophic global climate change.

Chapter 3

Gaussian Process Regression

“The actual science of Logic is conversant at present only with things either certain, impossible, or *entirely* doubtful, none of which (fortunately) we have to reason on. Therefore the true Logic for this world is the Calculus of Probabilities, which takes account of the magnitude of the probability which is, or ought to be, in a reasonable man’s mind.”

James Clerk Maxwell, 1850

Gaussian process regression is named after the German mathematician Carl Friedrich Gauss (1777–1855). Gauss is considered one of the greatest mathematicians of all time, and even after his death many novel ideas were discovered among his unpublished work, which extended his influence to well within the 20th Century (Gray, 2018).

Prediction with Gaussian processes has its roots in work by Kolmogorov (1941) and Wiener (1949). The Wiener process, or Brownian motion, is an important stochastic process that finds application in mathematics, economics, finance, physics and engineering (Siegmund, 2018). Gaussian process regression has also found great use in the field of geostatistics, where it was first used by the South African statistician and mining engineer Danie G. Krige for the valuation of new gold mines using a limited number of boreholes (Minnitt *et al.*, 2003). The method employed by Krige became known as kriging and was formalised by the French mathematician Georges Matheron (Matheron, 1973).

It was later realised that Gaussian process regression, or kriging, could be used for prediction within a more general, multivariate setting (Rasmussen and Williams, 2006). It is within this multidimensional framework that Gaussian process regression will be applied in this study.

A Gaussian process is formally defined as follows (Rasmussen and Williams, 2006):

Definition 1. A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.

A Gaussian process is a generalisation of the Gaussian probability distribution (Rasmussen and Williams, 2006). Gaussian Process regression is a stochastic process that governs the properties of functions. Imagine a function $f(x)$ to be an infinitely long vector, with each entry in the vector representing an instance of $f(x)$ at an input x . By considering the instances in the vector to be properties of a stochastic process, the properties of the function $f(x)$ can be inferred by the Gaussian Process based on only a finite number of points. This is one of the main attractions of the Gaussian Process framework (Rasmussen and Williams, 2006). Another attraction of Gaussian processes is that a variety of covariance functions can be used in combination to obtain functions with different degrees of smoothness or additive structure (Neal, 1999).

3.1 The Conditional of a Gaussian

Conditioning a multivariate Gaussian lies at the heart of Gaussian process regression.

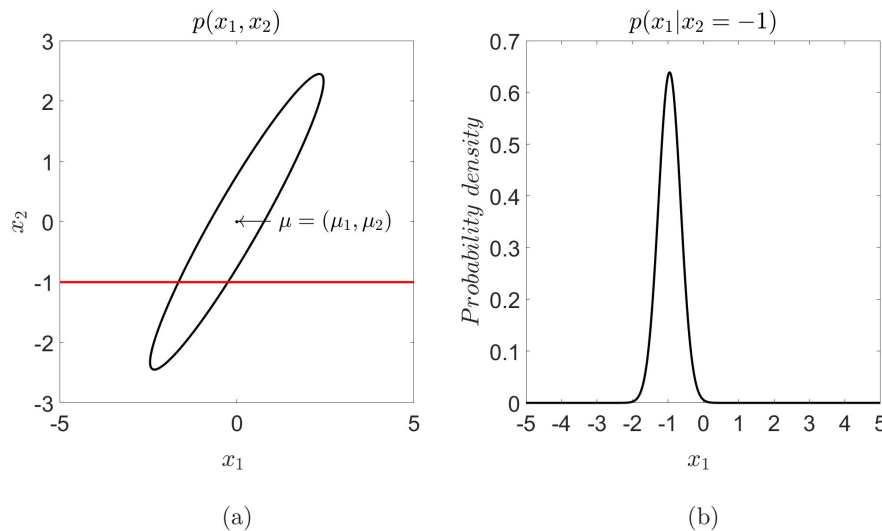


Figure 3.1: (a) Contour plot of the two-dimensional joint Gaussian distribution $p(x_1, x_2)$ with a correlation coefficient of $\rho = 0,95$ and a mean $\mu = (\mu_1, \mu_2) = (0, 0)$. (b) The conditional Gaussian distribution $p(x_1|x_2 = -1)$ generated by slicing the joint distribution in (a) at $x_2 = -1$. Figure generated by adapting `gaussCondition2Ddemo2` (Murphy, 2012).

Figure 3.1 (a) illustrates the 95 % contour plot of the two-dimensional joint Gaussian distribution $p(x_1, x_2)$. From the contour plot there is a positive correlation between the variables x_1 and x_2 : if x_1 increases, chances are good

that x_2 will also increase. Similarly, chances are good that x_2 will decrease if x_1 decreases. Since the distribution $p(x_1, x_2)$ is jointly Gaussian, there is a mean

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \quad (3.1)$$

and a covariance

$$\boldsymbol{\Sigma} = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix} \quad (3.2)$$

such that

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}\right) \quad (3.3)$$

with Σ_{12} and Σ_{21} some positive value close to unity, say 0,95, indicating a strong positive correlation between x_1 and x_2 ($\Sigma_{11} = 1$ and $\Sigma_{22} = 1$ since x_1 and x_2 correlate 100 % with themselves). Suppose now that a large dataset containing x_1 and x_2 exists, with x_1 representing an instance in time (timestamp - independent variable), and x_2 representing some time dependent variable, say solar radiation. We know that the relationship between these two variables is Gaussian, so would it be possible to say something about the value of x_2 (solar radiation) given x_1 (timestamp)? In other words: what is the distribution of x_2 given x_1 ? In order to do this, it is necessary to be able to compute the conditional of a Gaussian distribution, denoted by $p(x_2|x_1)$.

Theorem 1 allows us to do that (Murphy, 2012).

Theorem 1. *Suppose $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$ is jointly Gaussian with parameters*

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \boldsymbol{\Sigma} = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}, \boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1} = \begin{bmatrix} \Lambda_{11} & \Lambda_{12} \\ \Lambda_{21} & \Lambda_{22} \end{bmatrix}$$

then the posterior conditional is given by

$$p(x_1|x_2) = \mathcal{N}(x_1|\mu_{1|2}, \Sigma_{1|2}) \quad (3.4)$$

$$\mu_{1|2} = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2 - \mu_2) \quad (3.5)$$

$$= \mu_1 - \Lambda_{11}^{-1}\Lambda_{12}(x_2 - \mu_2) \quad (3.6)$$

$$= \Sigma_{1|2}(\Lambda_{11}\mu_1 - \Lambda_{12}(x_2 - \mu_2)) \quad (3.7)$$

$$\Sigma_{1|2} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21} \quad (3.8)$$

$$= \Lambda_{11}^{-1} \quad (3.9)$$

3.1.1 Example

The joint Gaussian distribution in Figure 3.1 has a correlation coefficient of $\rho = 0,95$. Suppose also that $\sigma_1 = \sigma_2 = 1$ and that $\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$. This leads to a covariance matrix of

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix} \quad (3.10)$$

$$= \begin{bmatrix} 1 & 0,95 \\ 0,95 & 1 \end{bmatrix} \quad (3.11)$$

Suppose that $x_2 = -1$ has been observed. The conditional $p(x_1|x_2 = -1)$ is given by Equation 3.4:

$$p(x_1|x_2 = -1) = \mathcal{N}\left(x_1|\mu_1 + \frac{\rho\sigma_1\sigma_2}{\sigma_2^2}(x_2 - \mu_2), \sigma_1^2 - \frac{(\rho\sigma_1\sigma_2)^2}{\sigma_2^2}\right) \quad (3.12)$$

$$= \mathcal{N}\left(x_1|\mu_1 + \rho(x_2 - \mu_2), \sigma^2(1 - \rho^2)\right) \quad (3.13)$$

$$= \mathcal{N}\left(x_1|\mu_1 + 0,95(x_2 - \mu_2), 0,0975\right) \quad (3.14)$$

$$= \mathcal{N}\left(x_1|0,95x_2, 0,0975\right) \quad (3.15)$$

$$= \mathcal{N}\left(x_1|-0,95, 0,0975\right) \quad (3.16)$$

This conditional distribution is illustrated in Figure 3.1 (b), with the value at which the joint distribution has been conditioned illustrated by the red line in Figure 3.1 (a). From Equation 3.16 $\mathbb{E}[x_1|x_2 = -1] = -0,95$ (this can also be seen in Figure 3.1 (b)). This makes sense, since $\rho = 0,95$ means that we believe that if x_2 decreases by 1 below its mean, x_1 will decrease by 0,95. It can also be seen from Equation 3.16 that $\text{var}[x_1|x_2 = -1] = 0,0975$. This makes sense, since the uncertainty in x_1 has gone down by observing a value for x_2 . Should the correlation coefficient be $\rho = 0$, $p(x_1|x_2) = \mathcal{N}(x_1|\mu_1, \sigma_1^2)$, which indicates that x_2 conveys no information about x_1 , as would be expected for the case of zero correlation.

3.2 Supervised Learning

In supervised learning a training dataset is assumed to contain inputs \mathbf{x}_i and outputs y_i that are related by some unknown function $f(\mathbf{x}_i)$, such that $y_i = f(\mathbf{x}_i)$. A distribution over functions, $p(f|\mathbf{X}, \mathbf{y})$, can be inferred and used to make predictions, $f_* = y_*$, given new inputs: $p(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y})$ (Murphy, 2012).

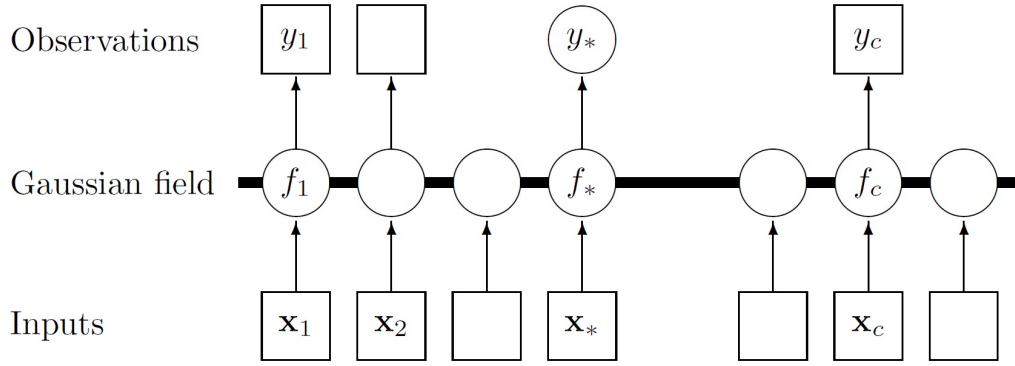


Figure 3.2: A graphical model of the function-space representation of Gaussian process regression, with squares representing observed variables and circles representing unknowns. The Gaussian field is a distribution over functions (Rasmussen and Williams, 2006).

In Gaussian process regression a prior distribution over functions is defined, which is converted into a posterior distribution over functions after observing some data. The prior over functions takes the form of the Gaussian process

$$f(\mathbf{x}) \sim GP\left(m(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}')\right) \quad (3.17)$$

where $m(\mathbf{x})$ is the mean function and $\kappa(\mathbf{x}, \mathbf{x}')$ is the kernel function, given by (Rasmussen and Williams, 2006)

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \quad (3.18)$$

$$\kappa(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))^T] \quad (3.19)$$

Within these two functions, the regression process is controlled and adapted to fit the physical problem at hand. The kernel function describes the structure in the data and captures the interrelations between datapoints in a set, with x and x' being two different function inputs (independent variables). It is used to construct the covariance matrix, Equation 3.2. The kernel function is optimised by the iterative adjustment of its hyperparameters.

Suppose a training set $\mathcal{D} = \{(\mathbf{x}_i, f_i), i = 1 : N\}$ has been observed, where $f_i = f(\mathbf{x}_i)$ is the noise-free observation of the function evaluated at \mathbf{x}_i (Murphy, 2012). Now suppose that a test set \mathbf{X}_* of size $N_* \times D$ has been supplied. The aim is to predict the function outputs \mathbf{f}_* based on the training set, \mathcal{D} , and the test points \mathbf{X}_* . In other words, we require the distribution $p(\mathbf{f}_* | \mathbf{X}_*, \mathbf{X}, \mathbf{f})$.

By definition of the joint Gaussian distribution in Section 3.1, Equation

3.3, the joint distribution over functions has the form

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu}_* \end{bmatrix}, \begin{bmatrix} \mathbf{K} & \mathbf{K}_* \\ \mathbf{K}_*^T & \mathbf{K}_{**} \end{bmatrix}\right) \quad (3.20)$$

where $\mathbf{K} = \kappa(\mathbf{X}, \mathbf{X})$ is the $N \times N$ sub-matrix of the covariances evaluated at all pairs of training points, $\mathbf{K}_* = \kappa(\mathbf{X}, \mathbf{X}_*)$ is the $N \times N_*$ sub-matrix of the covariances evaluated at all pairs of training and test points and $\mathbf{K}_{**} = \kappa(\mathbf{X}_*, \mathbf{X}_*)$ is the sub-matrix of covariances evaluated at all pairs of test points (Rasmussen and Williams, 2006).

For the purpose of predicting the function outputs \mathbf{f}_* , we condition the joint distribution, Equation 3.20, using the rules for conditioning a Gaussian, given by Equations 3.4 through 3.9 (Murphy, 2012):

$$p(\mathbf{f}_* | \mathbf{X}_*, \mathbf{X}, \mathbf{f}) = \mathcal{N}(\mathbf{f}_* | \boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*) \quad (3.21)$$

$$\boldsymbol{\mu}_* = \boldsymbol{\mu}(\mathbf{X}_*) + \mathbf{K}_*^T \mathbf{K}^{-1} (\mathbf{f} - \boldsymbol{\mu}(\mathbf{X})) \quad (3.22)$$

$$\boldsymbol{\Sigma}_* = \mathbf{K}_{**} - \mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{K}_* \quad (3.23)$$

For notational simplicity it is assumed that

$$\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (3.24)$$

therefore

$$\boldsymbol{\mu}_* = \mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{f} \quad (3.25)$$

$$\boldsymbol{\Sigma}_* = \mathbf{K}_{**} - \mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{K}_* \quad (3.26)$$

Equations 3.25 and 3.26 say that we know the mean and variance for all the points $f(x_*)$ in the set $\mathcal{D} = \{(\mathbf{x}_{*i}, f_{*i}), i = 1 : N_*\}$. This set, together with the training data, forms the interpolation function of the Gaussian process regression.

The covariances between pairs of datapoints are calculated using the kernel function. Equation 3.27 is an example of the widely used squared exponential kernel:

$$\text{cov}(f(\mathbf{x}), f(\mathbf{x}')) = \kappa(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}|\mathbf{x} - \mathbf{x}'|^2\right) \quad (3.27)$$

It can be seen by examining Equation 3.27 that

$$\lim_{\mathbf{x} \rightarrow \mathbf{x}'} \kappa(\mathbf{x}, \mathbf{x}') = 1 \quad (3.28)$$

indicating that the covariance between two function values is largest when the function inputs are close to each other.

3.2.1 Example

Figure 3.3 (a) shows a random function drawn from the Gaussian process prior constructed using the squared exponential kernel for the covariance. Figure 3.3 (b) illustrates the function drawn from the posterior after conditioning the Gaussian process on seven datapoints.

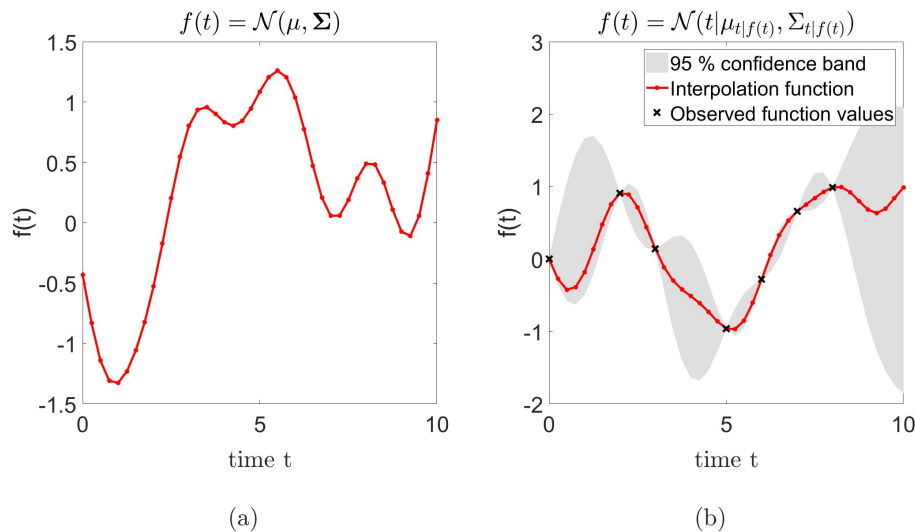


Figure 3.3: (a) A function sampled from a Gaussian process prior with squared exponential kernel. (b) A function sampled from the Gaussian process posterior after conditioning on 7 noise-free observations. The observations are given by $f(t) = \sin(t)$ and it can be seen that the posterior function distribution tends towards a sinus curve. Figure generated by adapting `gprDemoNoiseFree` (Murphy, 2012).

3.3 Weight-space View

The preceding explanation of Gaussian process regression was done by considering inference directly in function-space. An equivalent way of achieving equivalent results is by considering Gaussian process regression within the weight-space. In weight-space the kernel hyperparameters are defined in a

probabilistic manner by Bayes' rule and optimised by maximum likelihood (Rasmussen and Williams, 2006). Bayes' rule is given by Equation 3.29

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)} \quad (3.29)$$

where A and B are events with a certain probability of occurring. Applying Bayes' rule to the formulation of the hyperparameters results in Equation 3.30 (Rasmussen and Williams, 2006)

$$p(\mathbf{w}|\mathbf{y}, X) = \frac{p(\mathbf{y}|X, \mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|X)} \quad (3.30)$$

with $p(\mathbf{y}|X) = \int p(\mathbf{y}|X, \mathbf{w})p(\mathbf{w})d\mathbf{w}$ being the normalisation constant and \mathbf{w} the vector of hyperparameters (Rasmussen and Williams, 2006). Equation 3.30 defines the posterior distribution over parameters in weight-space.

3.4 Error Propagation in Gaussian Process Regression

A training dataset could contain errors due to metering inaccuracies (IPMVP, 2002). These errors find their way into the covariance matrix of the Gaussian process via the kernel hyperparameters. If the errors are measurement related, they are taken care of by the probabilistic nature of Gaussian process regression: the uncertainty will be well-defined by confidence intervals on modelled values (Carstens *et al.*, 2018).

Errors could also be modelling related (IPMVP, 2002). A large condition number for the covariance matrix (ill-conditioned matrix) could cause error in the matrix computations, where a small error in the input (kernel hyperparameter value) results in a large error in the output (Neal, 1999). The condition number is the ratio of the largest to smallest eigenvalues of the matrix. The problem of an ill-conditioned covariance matrix can be solved by adding jitter terms to the covariance function (Neal, 1999), or by solving an equivalent matrix that is well-conditioned (Farooq and Salhi, 2011).

Scikit-learn's `GaussianProcessRegressor` ensures a positive definite covariance matrix by adding a value to each entry in the diagonal of the matrix. This value can be specified by the user through the alpha-parameter (Pedregosa *et al.*, 2011).

As an example of an ill-conditioned system of linear equations, consider

the following (Farooq and Salhi, 2011):

$$\begin{aligned}x + y &= 2 \\x + 1,0001y &= 2\end{aligned}\tag{3.31}$$

This system has the solution $x = 2$ and $y = 0$. By changing the right-hand side of system 3.31 slightly to $(2,0001, 2)$, the solution changes drastically to $(3,0001, -1)$. This is a sign of an ill-conditioned system.

Poor accuracy due to ill-conditioned covariance matrices is largely avoidable (Neal, 1999).

3.5 Computational Performance of Gaussian Process Regression Algorithms

A computational burden of $O(N^3)$ for training and $O(N^2)$ for prediction is incurred with Gaussian process regression, where N is the size of the training set (Deng *et al.*, 2014). Gaussian process regression therefore becomes computationally expensive when N is large, as is the case with big-data analysis where a computational bottleneck occurs with the inversion of the $N \times N$ covariance matrix. Approximating the covariance matrix with a matrix that requires less computational power to invert could solve the problem (Park and Huang, 2016). Another approach to simplifying the covariance matrix entails partitioning the global regression domain into smaller regions, resulting in smaller covariance matrices for each region. The regressions on neighbouring regions are then stitched together to form a regression on the global domain. Care should be taken to ensure that the neighbouring regional regressions have the same values at their shared domain boundaries (Park and Huang, 2016).

Computational time for generating the covariance matrix should also be considered, since $O(N^3)$ operations are required for every iteration of the log marginal likelihood optimisation of the kernel hyperparameters (Schirru *et al.*, 2011). The computational burden of kernel hyperparameter optimisation can be reduced to $O(N)$ by employing identities derived by Schirru *et al.* (2011).

The computational burden can also be addressed by employing powerful graphics processing units (GPUs), which have been shown to deliver performance gains of up to two orders of magnitude over central processing unit (CPU) applications (Owens *et al.*, 2008). Franey *et al.* (2012) have found that employing CPU computation in conjunction with parallel GPU computation can significantly reduce the computational burden of implementing Gaussian process regression. Their results are summarised in Table 3.1.

Table 3.1: Implementation times for the Gaussian process simulation of the Goldstein-Price function using CPU implementation as well as a combination of CPU and GPU implementation (Franey *et al.*, 2012).

N	Time (s)	
	CPU implementation	CPU + GPU implementation
16	0,91	2,26
32	3,30	4,94
64	12,90	5,50
128	51,45	7,25
256	206,18	9,88
512	911,93	20,75

3.6 Summary

The underlying model existent in a dataset does not have to be fully known when employing Gaussian process regression. By making use of carefully selected kernel functions, a prior belief about the behaviour of the data can be encoded within the covariance matrix. The prior can then be conditioned on observed datapoints (the training set) to form a posterior distribution over functions in order to make predictions based on new inputs. Confidence intervals, derived from the (refined, posterior) covariance matrix, are part of the prediction output. This is a valuable attribute of Gaussian process regression. The confidence intervals provide a measure of the error propagation within the Gaussian process.

The choice of kernel to construct the covariance matrix has an influence on the accuracy and computational burden of Gaussian process regression and it is therefore important to consider kernels in more detail.

Chapter 4

Kernels

“Despite its importance, choosing the structural form of the kernel in nonparametric regression remains a black art.”

(Duvenaud *et al.*, 2013)

4.1 Supervised Learning Revisited

Interpolation and prediction on solar radiation and wind data have been performed by making use of the `Scikit-learn` machine learning library for Python. The `GaussianProcessRegressor` implementation of the Gaussian process regression algorithm by Rasmussen and Williams (2006) was used. It is given by Algorithm 1 (Rasmussen and Williams, 2006).

```

1 Input:  $X$  (inputs),  $\mathbf{y}$  (targets),  $k$  (covariance function),  $\sigma_n^2$  (noise
   level),  $\mathbf{x}_*$  (test input)
2  $L := \text{cholesky}(K + \sigma_n^2 I)$ 
3  $\boldsymbol{\alpha} := L^T \backslash (L \backslash \mathbf{y})$ 
4  $\bar{f}_* := k_*^T \boldsymbol{\alpha}$ 
5  $\mathbf{v} := L \backslash \mathbf{k}_*$ 
6  $\mathbb{V}[f_*] := k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{v}^T \mathbf{v}$ 
7  $\log p(\mathbf{y}|X) := -\frac{1}{2} \mathbf{y}^T \boldsymbol{\alpha} - \sum_i \log L_{ii} - \frac{n}{2} \log 2\pi$ 
8 Return:  $\bar{f}_*$  (mean),  $\mathbb{V}[f_*]$  (variance),  $\log p(\mathbf{y}|X)$  (log marginal
   likelihood)

```

Algorithm 1: Gaussian process regression with `Scikit-learn` and `GaussianProcessRegressor`.

Equations 3.25 ($\boldsymbol{\mu}_* = \mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{f}$) and 3.26 ($\boldsymbol{\Sigma}_* = \mathbf{K}_{**} - \mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{K}_*$), giving the conditional of the joint distribution over functions, are represented by steps

4 and 6 in Algorithm 1. Step 2 computes the Cholesky decomposition of the covariance matrix required for the matrix inversion as per Equations 3.25 and 3.26.

Algorithm 1 requires the input of a covariance function, encoding some prior belief about the structure within the dataset to be modelled. The covariance function is provided in the form of a kernel and is used to construct the covariance matrix. The kernel defines the covariance between any two function values, based on the proximity of the input values to the function: $cov(y, y') = k(x, x')$ (Duvenaud *et al.*, 2013). Stated colloquially, the kernel specifies the similarity between two objects (Duvenaud, 2014). Algorithm 1 is illustrated schematically in Figure 4.1.

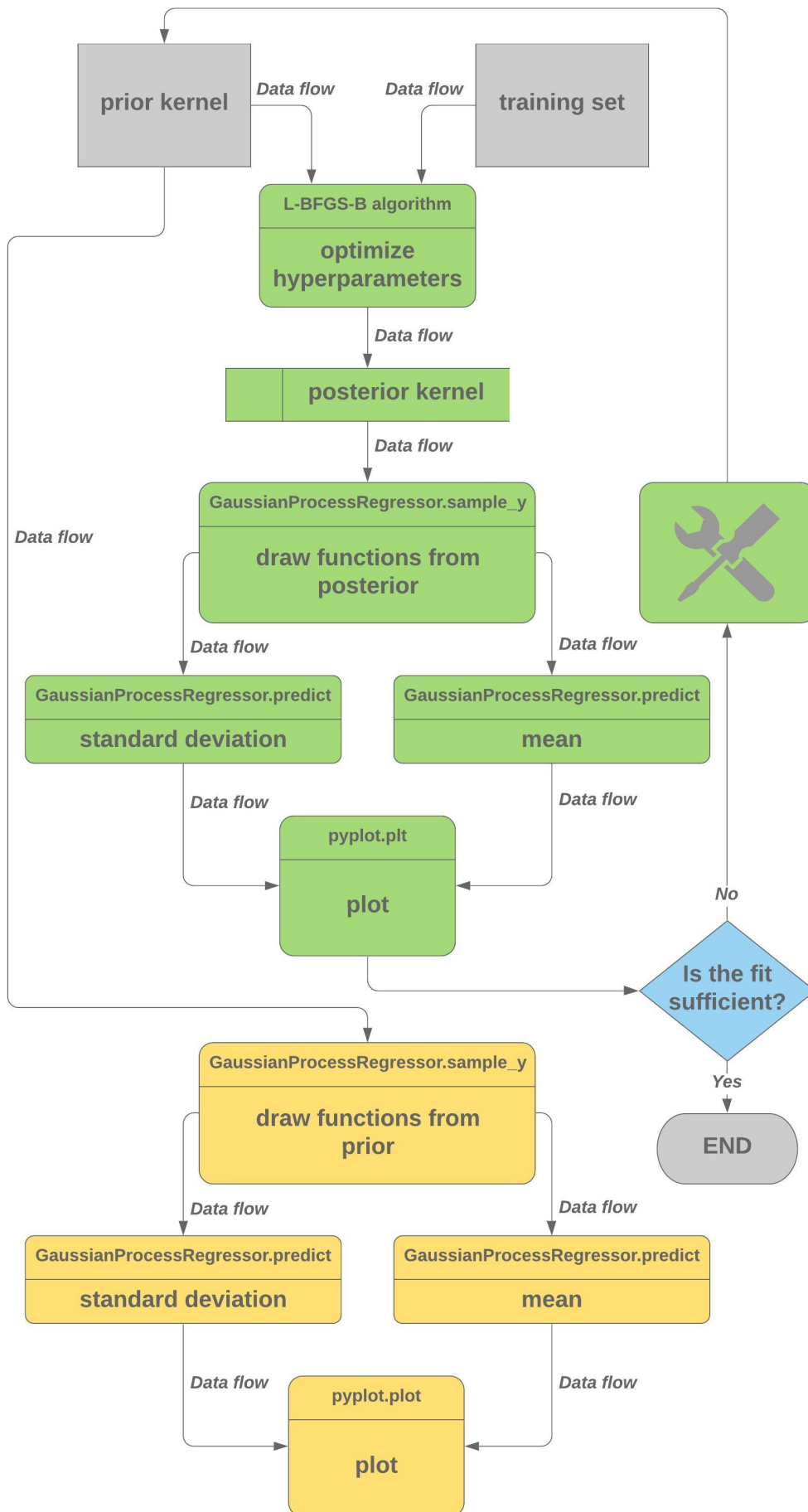


Figure 4.1: Flow diagram of Gaussian process regression as given by Algorithm 1.

4.2 The Kernel Universe

A multitude of kernels are defined in literature ((Rasmussen and Williams, 2006; Murphy, 2012; Pedregosa *et al.*, 2011)), but it is sufficient to focus on the following commonly used kernels: the linear kernel (Lin), the radial basis function (RBF), the rational quadratic kernel (RQ), the periodic kernel (Exp) and the Matérn kernel (Mat).

4.2.1 Linear kernel

The linear kernel is given by (Duvenaud, 2014)

$$k(x, x') = \sigma_f^2(x - c)(x' - c) \quad (4.1)$$

where c gives the kernel location. The linear kernel is non-stationary, as the covariance will change if the data is shifted. By contrast, the radial basis function is stationary - its value only depends on the proximity between x and x' (Duvenaud, 2014). The linear kernel is handy for encoding linear data structures (Maritz *et al.*, 2018). As will be seen in Section 4.3, the linear kernel could also be used for representing increasing variation or growing amplitude when used in combination with other kernels. Figure 4.6 illustrates a prior and posterior constructed by using the linear kernel.

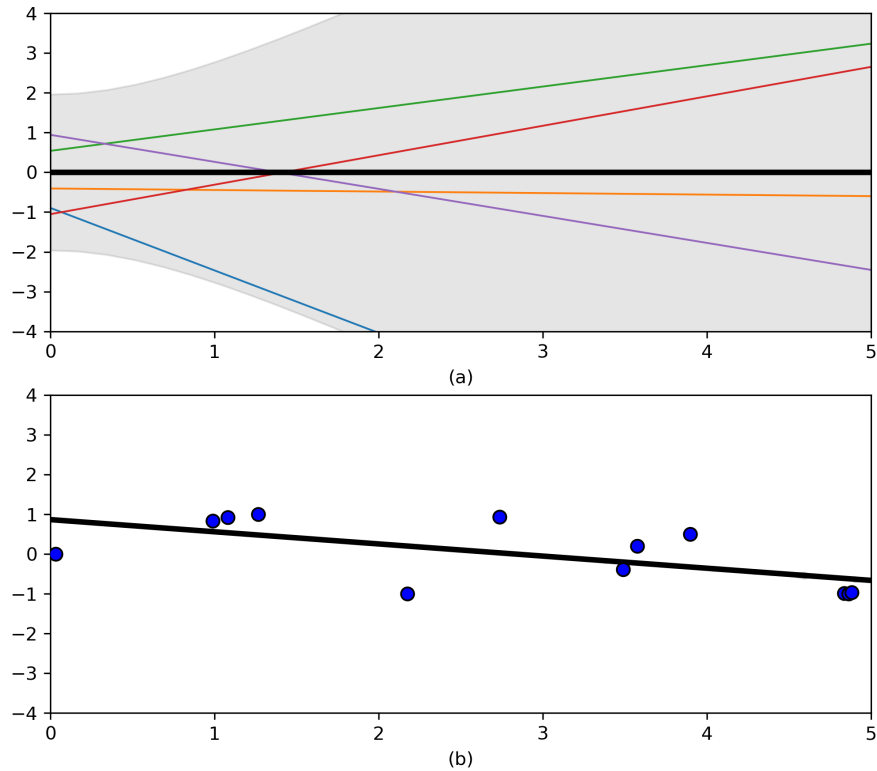


Figure 4.2: (a) Five samples from a linear kernel prior. (b) Five samples from the posterior, after conditioning on twelve noise-free datapoints obtained from $f(x) = \sin(x^2)$ (Pedregosa *et al.*, 2011).

4.2.2 Radial basis function

The radial basis function is given by (Duvenaud, 2014)

$$k(x, x') = \sigma_f^2 \exp\left(-\frac{(x - x')^2}{2l^2}\right) \quad (4.2)$$

with l being the characteristic length scale, which can be tuned to specify the precise shape of the covariance function, and σ_f^2 being a constant noise function. The radial basis function, also known as the squared exponential covariance function, is infinitely differentiable and is therefore handy when modelling the characteristic of smoothness of a function (Pedregosa *et al.*, 2011). The radial basis function could be used to represent a local variation structure within a dataset (Maritz *et al.*, 2018). It is the most widely-used

kernel (Rasmussen and Williams, 2006). Figure 4.3 illustrates a prior and posterior constructed by using the radial basis function.

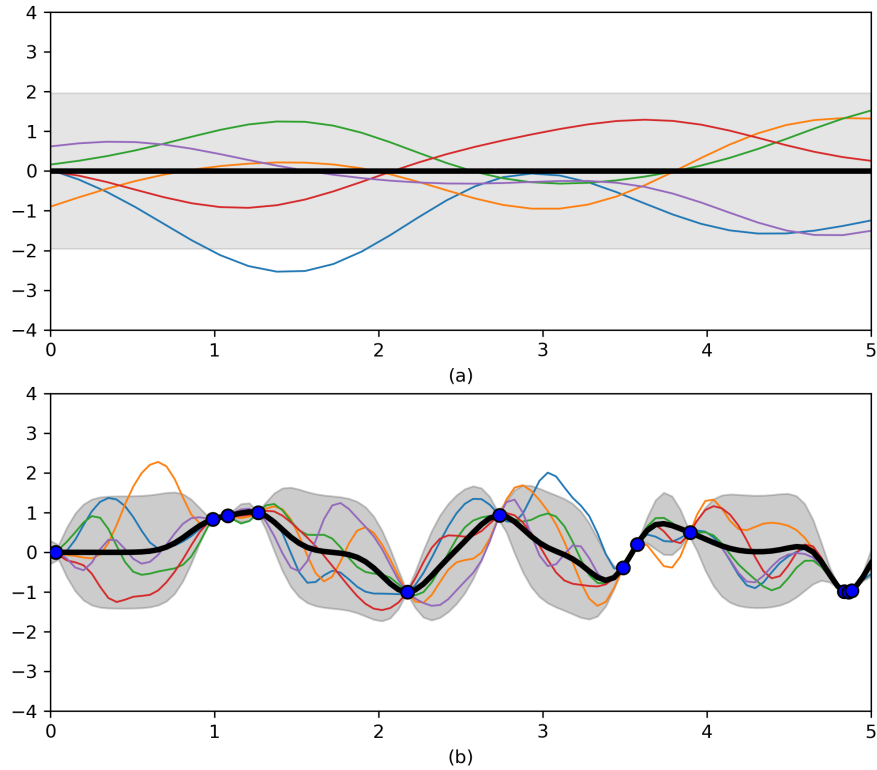


Figure 4.3: (a) Five samples from a radial basis function prior. (b) Five samples from the posterior after conditioning on twelve noise-free datapoints obtained from $f(x) = \sin(x^2)$ (Pedregosa *et al.*, 2011).

4.2.3 Rational quadratic function

The rational quadratic function is given by (Pedregosa *et al.*, 2011)

$$k(x, x') = \sigma_f^2 \left(1 + \frac{(x - x')^2}{2\alpha l^2} \right)^{-\alpha} \quad (4.3)$$

where l is the characteristic length scale. The hyperparameter α provides the scale mixture, which allows the rational quadratic function to act like an

infinite sum of radial basis function kernels with different length scales. The limit of the rational quadratic function as $\alpha \rightarrow \infty$ is the radial basis function with characteristic length scale l (Rasmussen and Williams, 2006). Figure 4.4 illustrates a prior and posterior obtained using the rational quadratic function as kernel.

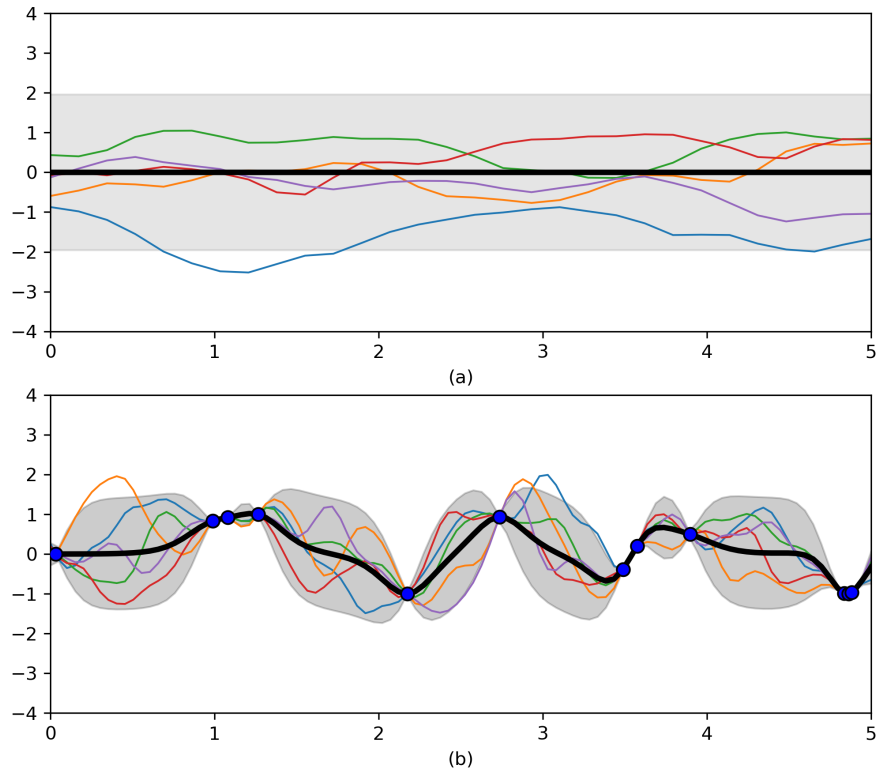


Figure 4.4: (a) Five samples from a rational quadratic function prior. (b) Five samples from the posterior after conditioning on twelve noise-free datapoints obtained from $f(x) = \sin(x^2)$ (Pedregosa *et al.*, 2011).

4.2.4 Periodic kernel

The periodic kernel, also known as the exponential sine-squared kernel, is a stationary covariance function given by (Pedregosa *et al.*, 2011)

$$k(x, x') = \sigma_f^2 \exp\left(-\frac{2\sin^2\left(\frac{\pi}{p}|x - x'|\right)}{l^2}\right) \quad (4.4)$$

with p the period and l the length scale. The periodic kernel could be used to model functions having a repetitive pattern, see Figure 4.5 (Duvenaud, 2014).

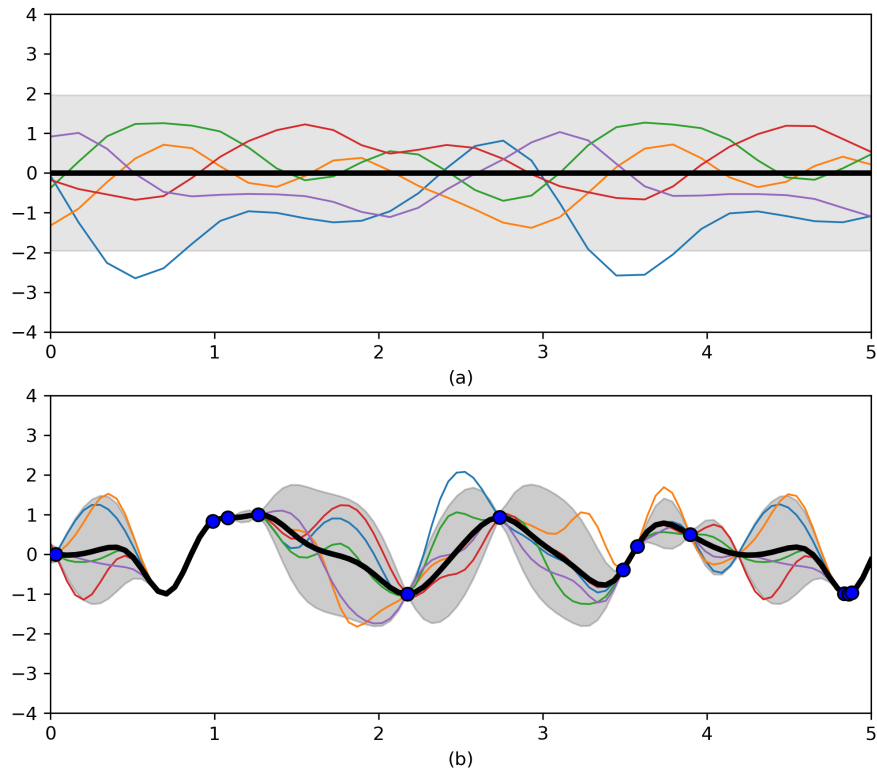


Figure 4.5: (a) Five samples from a periodic kernel prior. (b) Five samples from the posterior after conditioning on twelve noise-free datapoints obtained from $f(x) = \sin(x^2)$ (Pedregosa *et al.*, 2011).

4.2.5 Matérn kernel

The Matérn kernel refers to a class of covariance functions given by (Rasmussen and Williams, 2006)

$$k(x, x') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}|x - x'|}{l} \right)^\nu K_\nu \left(\frac{\sqrt{2\nu}|x - x'|}{l} \right) \quad (4.5)$$

where K_ν is a modified Bessel function and ν and l are positive parameters. For $\nu \rightarrow \infty$, the Matérn kernel reduces to the radial basis function (RBF). The Matérn reduces to a simple form when $\nu = p + \frac{1}{2}$ where p is a non-negative integer. In these cases the Matérn kernel is the product of an exponential function and a polynomial of order p (Rasmussen and Williams, 2006). The most popular cases of the Matérn class of kernels for machine learning are

$$k_{\nu=3/2}(x - x') = \left(1 + \frac{\sqrt{3}|x - x'|}{l}\right) \exp\left(-\frac{\sqrt{3}|x - x'|}{l}\right) \quad (4.6)$$

$$k_{\nu=5/2}(x - x') = \left(1 + \frac{\sqrt{5}|x - x'|}{l}\right) \exp\left(-\frac{\sqrt{5}|x - x'|}{l}\right) \quad (4.7)$$

where $\nu = \frac{3}{2}$ and $\nu = \frac{5}{2}$, respectively. For cases where $\nu \geq \frac{7}{2}$, the kernels of the Matérn class become difficult to distinguish (Rasmussen and Williams, 2006).

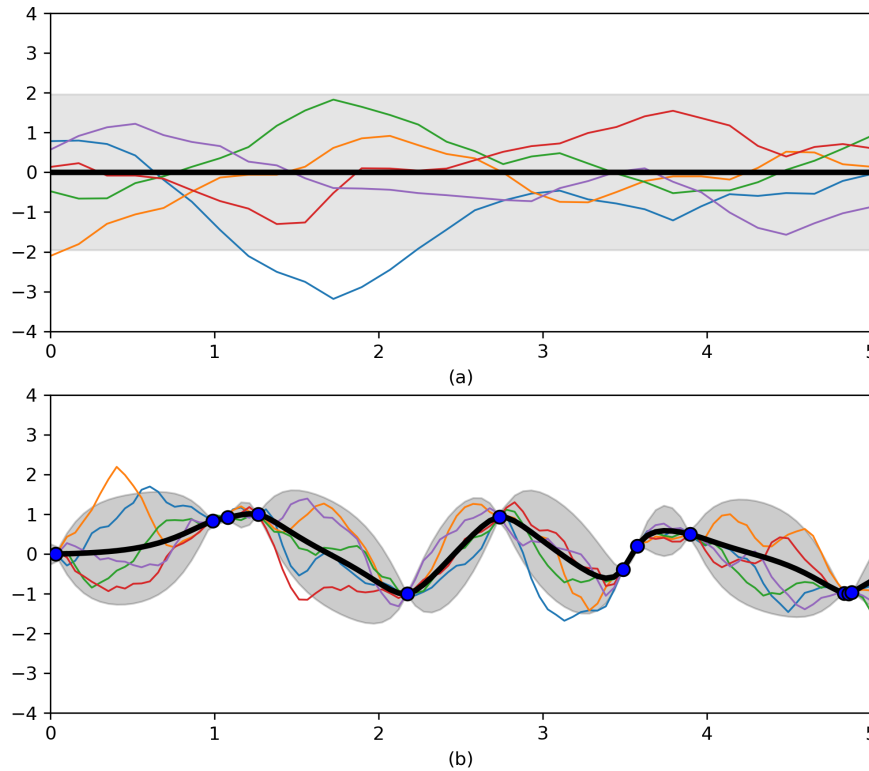


Figure 4.6: (a) Five samples from a Matérn kernel prior. (b) Five samples from the posterior after conditioning on twelve noise-free datapoints obtained from $f(x) = \sin(x^2)$ (Pedregosa *et al.*, 2011).

4.3 Complex Kernel Engineering

The basic kernels mentioned in the preceding sections can be combined by addition and/or multiplication to produce compound kernels with unique characteristics. The constituent kernels are each chosen to represent a specific characteristic of the data to be modelled and the compound kernel can be used to model data containing numerous features.

The radial basis function could be used to model local variation, a repetitive structure can be represented by the periodic kernel and linearities by the linear kernel (Maritz *et al.*, 2018). Functions that vary smoothly over numerous length scales could be modelled by the rational quadratic kernel (Duvenaud, 2014).

When multiplying kernels, the resulting kernel will indicate a high correlation between variables only if both of the base kernels produce a large value. Multiplication of base kernels can therefore be seen as an AND operation (Du-

venaud, 2014). The addition of base kernels acts like an OR operation, since a high correlation will be indicated by the compound kernel when either of the base kernels has a large value (Duvenaud, 2014).

Figure 4.7 illustrates some of the structures that can be expressed by the multiplication of different base kernels.

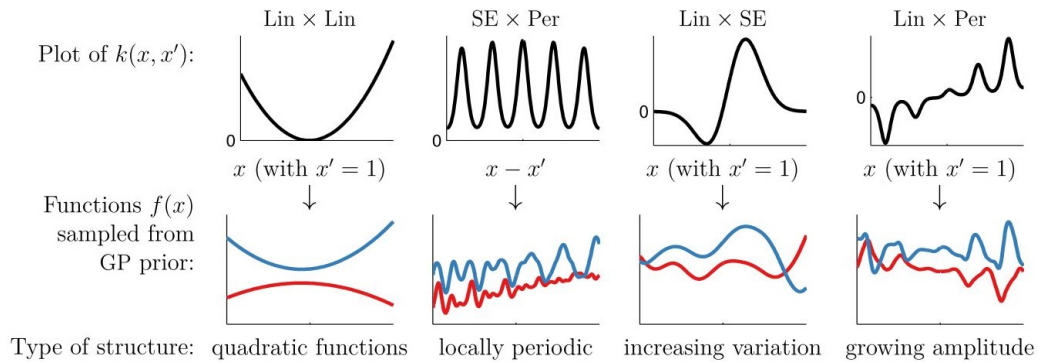


Figure 4.7: Structures expressible by the multiplication of base kernels. Adopted from Duvenaud (2014).

It should be kept in mind that the complexity of the compound kernel could increase the number of hyperparameters that have to be optimised, thus increasing the computational burden of the Gaussian process algorithm.

4.3.1 Examples

Mauna Loa carbon dioxide concentration

Having presented the ability of compound kernels, it is appropriate to analyse an example.

Figure 4.8 illustrates atmospheric CO_2 concentration measured at the Mauna Loa Observatory in Hawaii and modelled as a function of time by Gaussian process regression. The data has been sourced by investigators at the Carbon Dioxide Research Group at the University of California (Keeling *et al.*, 2009). It has been averaged monthly between the years 1958 and 2001. The Gaussian process model also predicts the CO_2 concentration 20 years into the future.

From Figure 4.8, three trends can be identified in the data:

- Seasonal variation
- Long-term increase
- Small irregularities

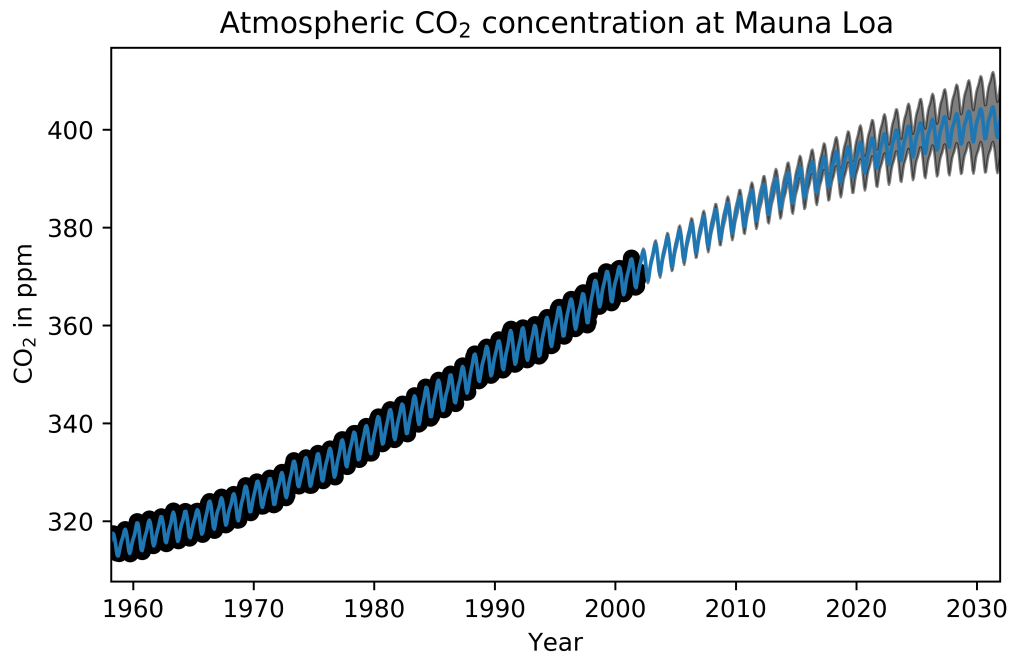


Figure 4.8: Atmospheric carbon dioxide concentration in the air at the Mauna Loa Observatory in Hawaii. Confidence intervals are indicated in grey (Pedregosa *et al.*, 2011; Rasmussen and Williams, 2006).

The prior for this data has been constructed as follows (Rasmussen and Williams, 2006):

- The seasonal variation has been modelled by an exponential sine-squared kernel with a length scale of one year, multiplied by a radial basis function. The radial basis function allows for decay away from exact periodicity.
- The smooth, long-term increasing trend has been modelled by a radial basis function with a large length scale. The fact that the trend is rising is not enforced within the kernel and the Gaussian process algorithm is free to discover this by itself.
- The small irregularities are explained by a rational quadratic kernel. The rational quadratic kernel can accommodate various length scales through its alpha-parameter and could therefore accommodate the irregularities in the data.
- A fourth component is added to the kernel: a noise term. The noise in the data is represented by a radial basis function (noise such as local weather phenomena) and a white kernel for white noise that might be present in the data.

The following covariance function (compound kernel) has been constructed by Rasmussen and Williams (2006):

$$k(x, x') = k_1(x, x') + k_2(x, x') + k_3(x, x') + k_4(x, x') \quad (4.8)$$

Table 4.1 provides more information on each of the constituent kernels of Equation 4.8.

Table 4.1: Constituent kernels used for constructing Equation 4.8 (Rasmussen and Williams, 2006).

Trend	Kernel	Description
Long-term rising	RBF	$k_1 = \theta_1^2 \exp\left(-\frac{(x-x')^2}{2\theta_2^2}\right)$
Seasonal	Exp RBF	$k_2 = \theta_3^2 \exp\left(-\frac{(x-x')^2}{2\theta_4^2} - \frac{2\sin^2(\pi(x-x'))}{\theta_5^2}\right)$
Small irregularities	RQ	$k_3 = \theta_6^2 \left(1 + \frac{(x-x')^2}{2\theta_8\theta_7^2}\right)^{-\theta_8}$
Noise	RBF White Kernel	$k_4 = \theta_9^2 \exp\left(-\frac{(x-x')^2}{2\theta_{10}^2}\right) + \theta_{11}^2 \delta_{xx'}$

Note the following: the seasonal component is caused by different rates of CO_2 uptake by plants during different seasons and it is reasonable to assume that this could change over the long term due to global warming. When fitting the data, this hypothesised trend can be removed if it turns out to be irrelevant, by letting hyperparameter θ_4 become very large (Rasmussen and Williams, 2006). This serves to illustrate that the construction of the Gaussian process prior requires some insight into possible data trends.

Table 4.2 gives the values of the hyperparameters on the posterior kernel after optimising the hyperparameters on the observed data (training the model).

The values of the optimised hyperparameters should correspond to structures found in the data. The long-term rising trend has the largest amplitude ($\theta_1 = 34,4$ ppm) and therefore dominates the model. The length scale on the long-term trend ($\theta_2 = 41,8$ years) roughly equates to the total length of the dataset (1958 – 2001). The length scale on the seasonal component ($\theta_5 = 1,44$ years) is in the order of one year. The decay time of $\theta_4 = 180$ years indicates that the seasonal trend is close to periodic with little decay. The overall noise level is very small ($\theta_9 = 0,197$ ppm and $\theta_{11} = 0,0336$ ppm). The

Table 4.2: Optimised hyperparameters (Pedregosa *et al.*, 2011).

Hyperparameter	Interpretation	Value
θ_1	amplitude	34,4 ppm
θ_2	characteristic length scale	41,8 years
θ_3	magnitude	3,27 ppm
θ_4	decay time	180 years
θ_5	smoothness	1,44
θ_6	magnitude	0,446 ppm
θ_7	characteristic length scale	0,957 years
θ_8	shape parameter (α)	17,7
θ_9	magnitude	0,197 ppm
θ_{10}	characteristic length scale	0,138 years
θ_{11}	magnitude	0,0336 ppm

low level of noise indicates that the model explains the data well (Pedregosa *et al.*, 2011).

International airline passenger data

Duvenaud *et al.* (2013) attempted to automate the kernel selection process by searching over a space of kernel structures. By analysing monthly total airline passenger data, their algorithm suggested the following kernel to model the data:

$$kernel = RBF \times (Lin + Lin \times (Exp + RQ)) \quad (4.9)$$

By decomposing the kernel, three trends are identified: a long-term trend, annual periodicity and medium-term deviations (Duvenaud *et al.*, 2013). A noise component is also present. See Figure 4.9.

This example illustrates that by analysing the structure of a kernel, valuable insight can be gained into the patterns present in a dataset.

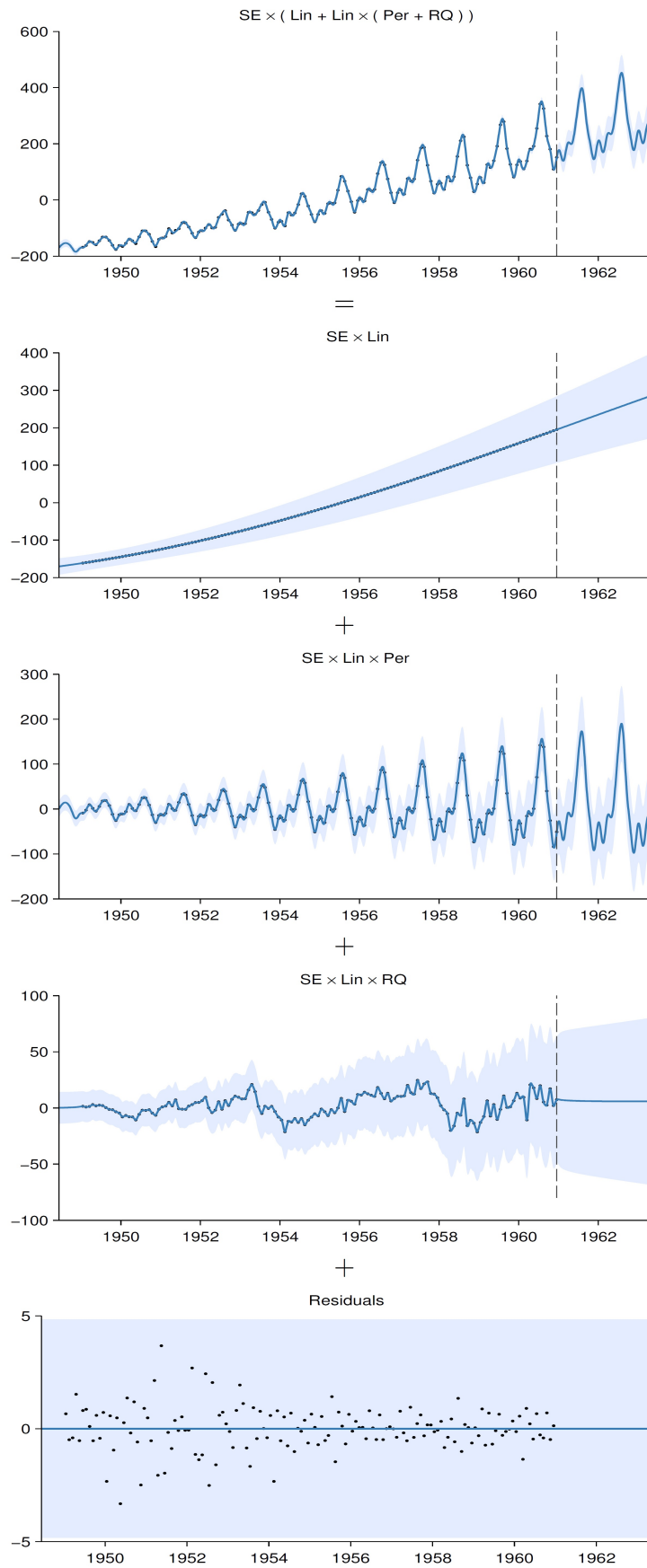


Figure 4.9: Monthly total airline passengers kernel decomposition. Adopted from Duvenaud *et al.* (2013).

Solar activity data

Annual total solar irradiance, indicating solar activity, for the period 1610 to 2011 has been modelled by Duvenaud *et al.* (2013). It is important to note that this solar irradiance dataset has been reconstructed by Lean *et al.* (1995) based on records of sunspot darkening and the brightening of solar faculae (bright spots on the sun). This record of solar activity extends over four centuries (Lean *et al.*, 1995).

Annual solar irradiance is cyclic. One of the best known of these cycles is the Milankovitch Cycle, which arises due to a periodic change in the earth's orbital geometry. This cycle has a periodicity of tens to hundreds of thousands of years and the influence thereof will therefore not be visible in the dataset (Lean and Rind, 2000). A shorter and well-documented solar cycle is the sunspot cycle, or Schwabe Cycle, which has a period of about 11 years (Lean and Rind, 2000).

Changes in total solar irradiance could result in global temperature changes such as the Little Ice Age (1550–1700) (Lean and Rind, 2000). This period of diminished solar irradiance can be seen in Figure 4.10. The Schwabe Cycle is also visible. These features of the data have been modelled by a periodic kernel multiplied by the sum of two squared exponential kernels.

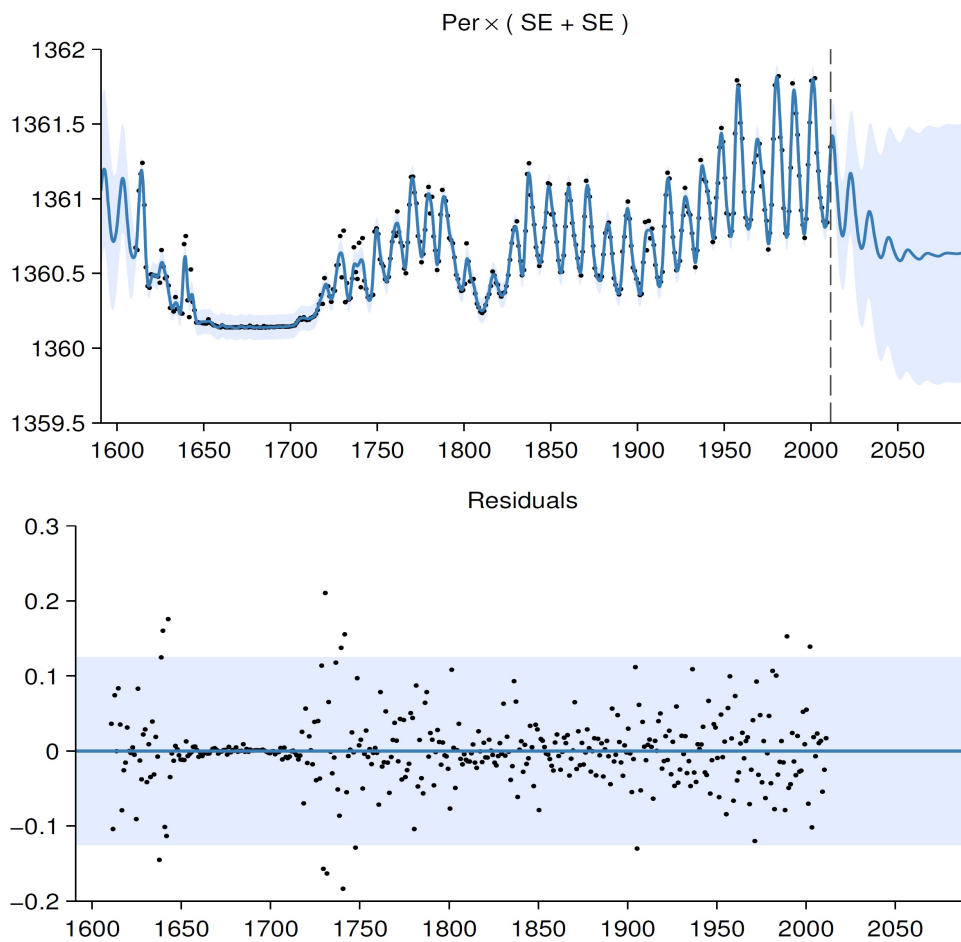


Figure 4.10: Solar irradiance measured between 1610 and 2011. The data has been modelled by Gaussian process regression and the residuals are also shown. Adopted from Duvenaud *et al.* (2013).

4.4 Summary

Kernels are at the centre of Gaussian process regression, as they describe the similarity between datapoints and in doing so control the regression process. Kernels also influence the computational performance of Gaussian process regression. Individual kernels can be used to construct compound kernels that represent the structure assumed to be in a dataset. Compound kernels have been shown to model atmospheric CO_2 levels, airline passenger data and solar activity data successfully. The kernel techniques discussed in this chapter will be put to the test in case studies on hourly solar irradiance data as well as wind speed data.

Chapter 5

Case Studies

The Southern African Universities Radiometric Network (Sauran) weather station at Stellenbosch University provides time-stamped measurements of the weather metrics provided in Table 5.1 (Brooks *et al.*, 2015).

Table 5.1: The weather metrics recorded at the Southern African Universities Radiometric Network weather station at Stellenbosch University.

Metric	Unit	Abbreviation
Global horizontal irradiance	W/m^2	GHI
Direct normal irradiance	W/m^2	DNI
Diffuse horizontal irradiance	W/m^2	DHI
Long-wave ultraviolet radiation	W/m^2	UVA
Short-wave ultraviolet radiation	W/m^2	UVB
Air temperature	$^{\circ}C$	air_temp
Barometric pressure	<i>mbar</i>	BP
Relative humidity	%	RH
Wind speed	<i>m/s</i>	WS
Wind direction	$^{\circ}$	WD
Standard deviation in wind direction	$^{\circ}$	WD_SD

For the purpose of smart renewable energy management within the context of this study, it will be attempted to use these metrics to explore the application of Gaussian process regression in the behaviour of global horizontal irradiance (GHI) and wind speed. All code was implemented on an Intel Core i7-5600U 2,60 GHz CPU with two cores and four logical processors.

Figure 5.1 illustrates the plots of each of the metrics provided by the Sauran weather station at Stellenbosch University for the period 1 February 2015 to 8 February 2015. These weather metrics were used to train the Gaussian

process regression algorithm (Algorithm 1), which is illustrated in Figure 4.1. By their nature, the following metrics are strongly correlated: GHI, DNI, DHI, UVA, UVB and air temperature. The other metrics may exhibit weak correlation. The stochastic behaviour of the parameters calls for Gaussian process regression.

The use of non-linear modelling methodologies, such as Gaussian process regression for the modelling of GHI is very limited (Tolba *et al.*, 2019) and the author is not aware of any previous application of Gaussian process regression on interval-deficient wind speed data.

5.1 Global Horizontal Irradiance Data

5.1.1 Interpolation

For the interpolation of hourly GHI data, a Gaussian process regression algorithm (Algorithm 1 and Figure 4.1) was coded in Jupyter Notebook using the Python libraries NumPy, Pandas, Scikit-learn and Matplotlib. Four kernels were tested:

- Kernel 1: Exp
- Kernel 2: RQ
- Kernel 3: $\text{Exp} \times \text{RQ}$
- Kernel 4: $\text{RBF} \times (\text{RQ} + \text{Exp})$

Refer to Section 4.2 for the characteristics of the different kernels and the data signatures they represent.

Three different sets of weather metrics from the Stellenbosch University Sauran weather station (Table 5.1) were used to train the regression algorithm:

- Dataset: hourly GHI
- Dataset 2: hourly GHI, DNI, DHI, DHI_shadowband, UVA, UVB, air_temp, RH, WS, WD, WD_SD
- Dataset 3: hourly GHI, DNI, DHI, DHI_shadowband, UVA, UVB, air_temp, RH, WS, WD, WD_SD and BP

Refer to Figure 5.1 for plots of these metrics. Note that the only difference between dataset 2 and dataset 3 is the inclusion of barometric pressure (BP) in dataset 3. The Gaussian process regression done with dataset 1 is a single-in, single-out process as the algorithm is trained on GHI data only. The training with datasets 2 and 3 are multi-in, single-out, since the Gaussian process is trained on multiple inputs.

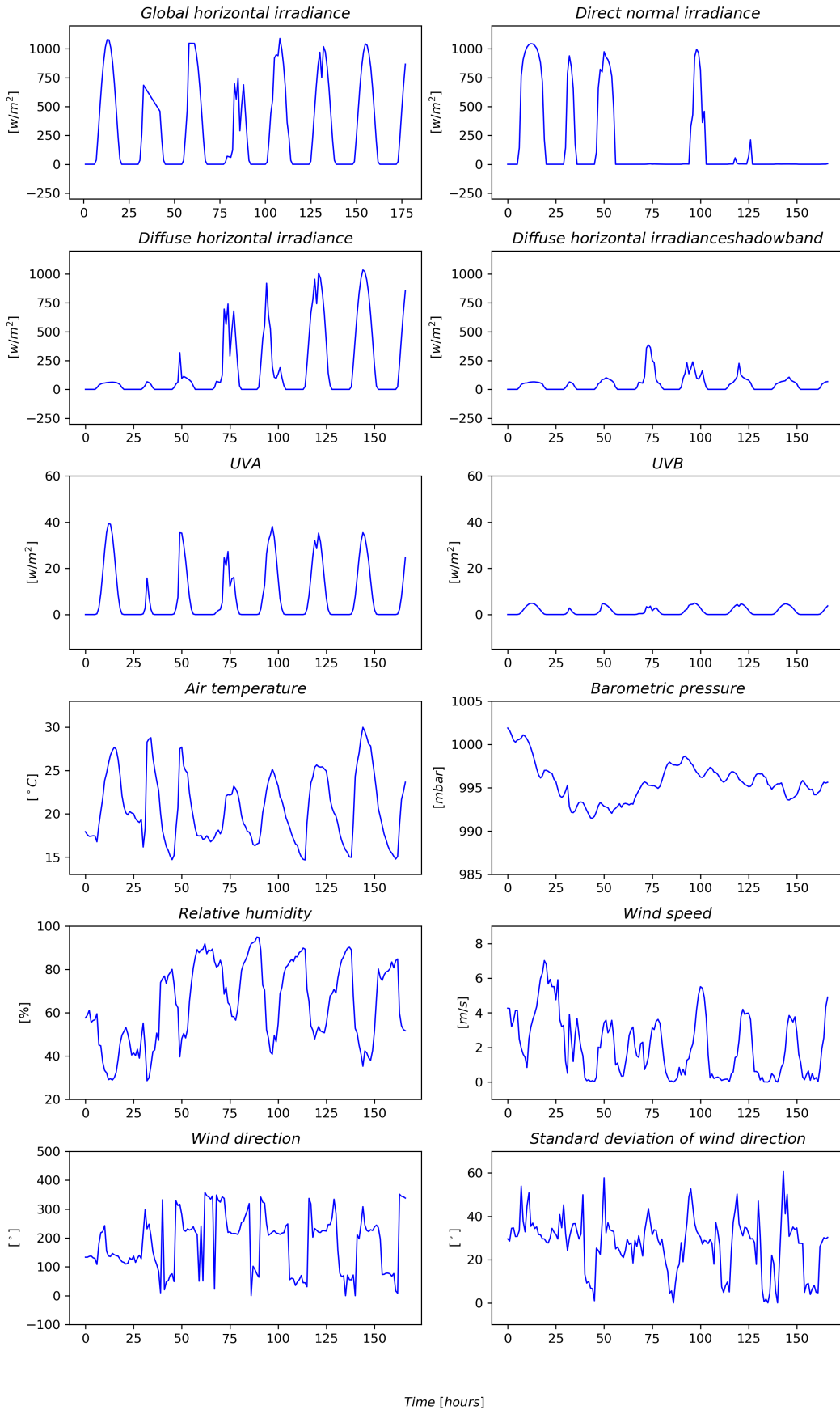


Figure 5.1: Plots of the weather metrics provided by the Sauran weather station at Stellenbosch University for the period 1 February 2015 to 8 February 2015. These metrics are used as input to the Gaussian process regression algorithm.

Weather data for the week 1 February to 8 February 2015 was downloaded in datafile format (.dat) from sauran.net and imported into a NumPy array. In order to test the interpolation capabilities of the Gaussian process in the case of meter failure, meter failure was simulated by manually removing readings from and including 2015-02-02 09:00 to 2015-02-02 16:00, as well as from and including 2015-02-03 10:00 to 2015-02-03 12:00. This is indicated in Figures 5.3, 5.4 and 5.5. The resulting training sets (datasets 1 to 3) therefore consist of 167 hourly datapoints between $t_0 = 0$ hours and $t_{end} = 177$ hours. For dataset 1 each datapoint is two-dimensional (time and GHI), for dataset 2, each datapoint is twelve-dimensional – $[1 \times 12]$ – and for dataset 3, each datapoint is thirteen-dimensional – $[1 \times 13]$.

The number of restarts of the built-in optimiser for refining the kernel parameters had to be set at different values, depending on the kernel and training set. The number of restarts is given in Table 5.2.

Table 5.2: The number of restarts of the Scikit-learn hyperparameter optimiser for the interpolation of GHI data.

	Dataset 1	Dataset 2	Dataset 3
Exp	100	4	4
RQ	100	4	4
Exp \times RQ	1000	100	100
RBF \times (RQ + Exp)	100	4	4

The Python code for the interpolation of the GHI can be found in Appendix A.

5.1.2 Root-mean-square error

The Gaussian process model was trained and used to produce 10 739 interpolation points evenly spaced in time between $t_0 = 0$ hours and $t_{end} = 177$ hours. The reason for specifying this specific amount of interpolation points is that in order to compare the interpolation over hourly data with the minutely dataset for the same period by means of root-mean-square error, the vector of interpolation points and minutely datapoints must have the same first dimension. The first dimension of the minutely dataset over the period 1 February to 8 February 2015 is 10 739, i.e. there are 10 739 data points in this set.

The interpolated points were compared to minutely data in order to evaluate the Gaussian process regression algorithm’s ability to predict the GHI values between hourly observations. Calculating the root-mean-square error based on comparison to hourly data would be futile, as the algorithm is trained

on hourly data and the interpolated GHI values would therefore pass through the hourly observed GHI datapoints, resulting in a root-mean-square error of zero.

5.1.3 Results

Figure 5.3 shows the result of interpolation by training the four kernels on dataset 1, Figure 5.4 shows the result of interpolation by training the four kernels on dataset 2, while Figure 5.5 shows the result of interpolation by training the four kernels on dataset 3.

In order to assess the effect that meter failure has on the goodness-of-fit of the regression model, the Gaussian process model was also trained on a set with no simulated meter failure. Figure 5.2 illustrates the result of this regression. The inlay in Figure 5.2 illustrates how the interpolated points mesh in time with the minutely measured datapoints (observations) in order to allow for root-mean-square error evaluation.

Also important to note in the inlay, is that the Gaussian process regression is a vector of discrete points. If the resolution on the interpolation mesh goes to infinity, the Gaussian process regression will become a continuous function.

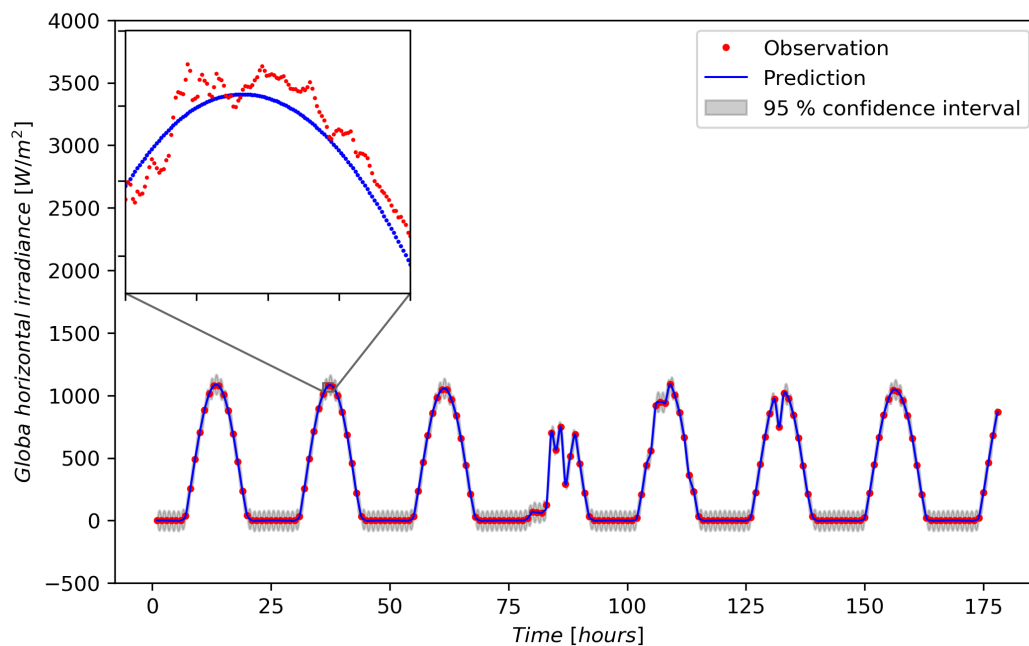


Figure 5.2: Interpolation using dataset 1 with no simulated meter failure as training set and the $\text{Exp} \times \text{RQ}$ kernel. Inlay: The discrete points constituting the Gaussian process regression, together with minutely observations of GHI.

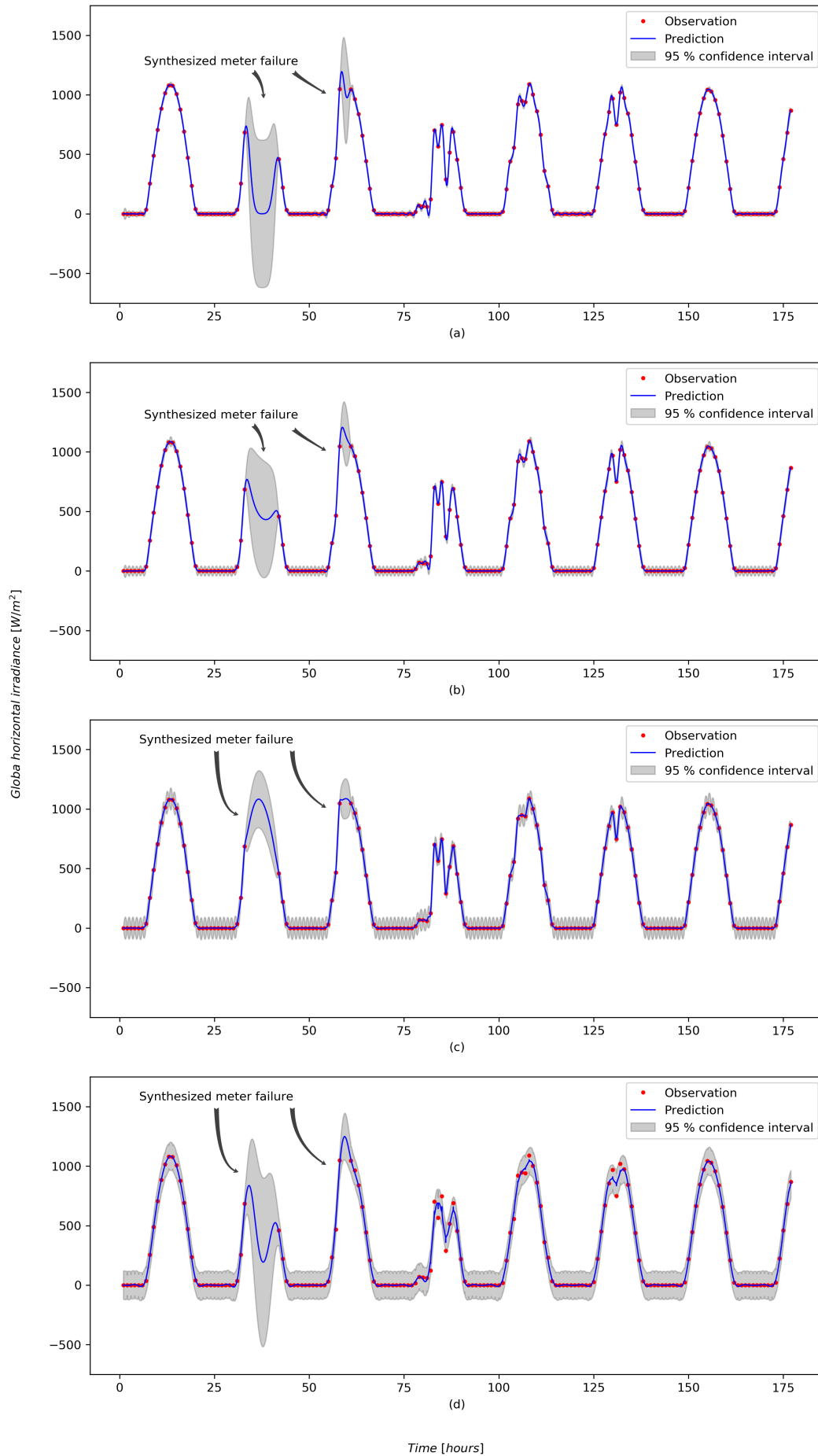


Figure 5.3: Interpolation using GHI as single-input training set with kernels (a) Exp, (b) RQ, (c) Exp × RQ and (d) RBF × (RQ + Exp).

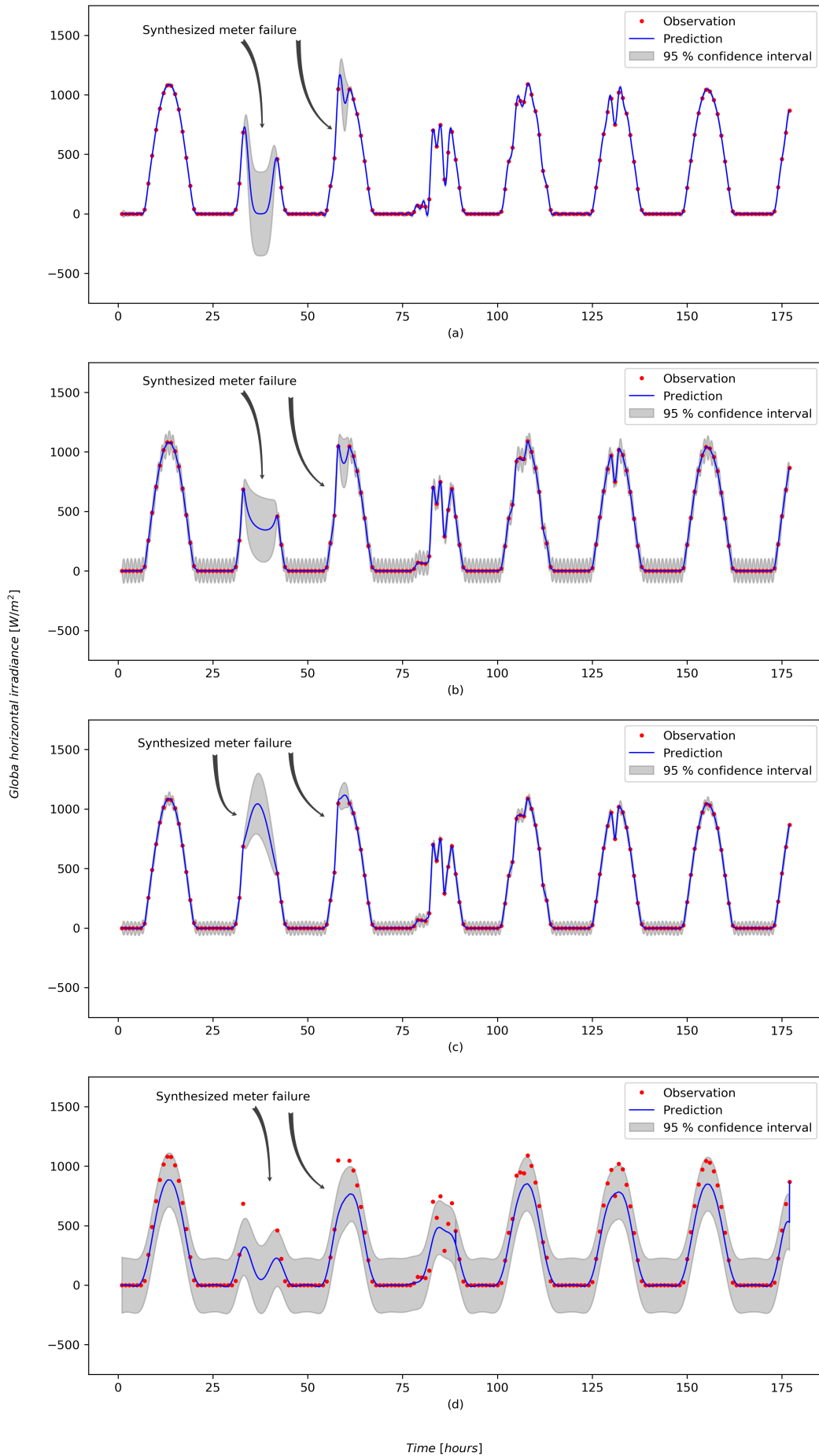


Figure 5.4: Interpolation using dataset 2 as multi-input training set with kernels (a) Exp, (b) RQ, (c) Exp \times RQ and (d) RBF \times (RQ + Exp).

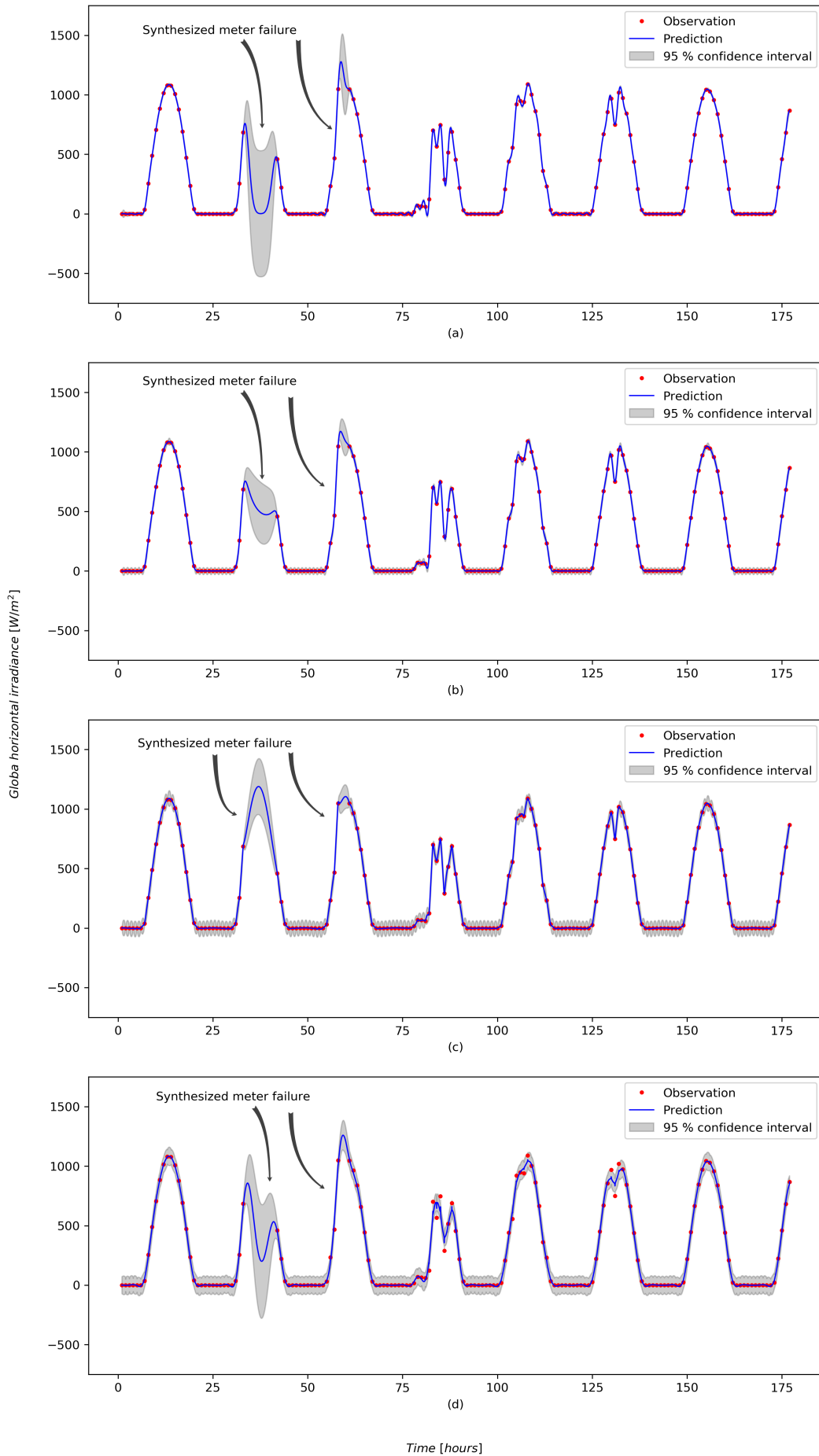


Figure 5.5: Interpolation using dataset 3 as multi-input training set with kernels (a) Exp, (b) RQ, (c) Exp \times RQ and (d) RBF \times (RQ + Exp).

The goodness of fit, as expressed by root-mean-square error in relation to minutely data, is given in Table 5.3 as a function of kernel and training set. Dataset 1 was chosen as training set for the case of no meter failure since it gave the best fit in the case of simulated meter failure.

Table 5.3: The root-mean-square error in relation to minutely GHI data for the different kernels and training sets.

	Dataset 1	Dataset 2	Dataset 3
Exp	197,3 W/m^2	198,3 W/m^2	194,9 W/m^2
RQ	133,9 W/m^2	148,6 W/m^2	128,8 W/m^2
Exp \times RQ	92,8 W/m^2	93,5 W/m^2	94,1 W/m^2
RBF \times (RQ + Exp)	152,5 W/m^2	217,4 W/m^2	151,3 W/m^2
No simulated meter failure			
Exp	83,3 W/m^2		
RQ	82,7 W/m^2		
Exp \times RQ	82,2 W/m^2		
RBF \times (RQ + Exp)	87,0 W/m^2		

5.1.4 Forecasting

An attempt was made to forecast the behaviour of the GHI. The algorithm was trained on dataset 3 for the period from and including 2015-02-01 00:00 to 2015-02-10 05:00, while the test set comprised the rest of the GHI data up to 2015-02-14 23:00, concluding a two-week training and test set. No meter failure was simulated with the forecasting. The number of restarts for the hyperparameter optimiser is given in Table 5.4.

Table 5.4: The number of restarts of the Scikit-learn hyperparameter optimiser for the forecasting of GHI data.

	Dataset 3
Exp	100
RQ	100
Exp \times RQ	1 000
RBF \times (RQ + Exp)	100

For the interpolation of the GHI data in Section 5.1.1, the root-mean-square error was computed by comparing the interpolated points to the minutely GHI data of the same time period. This gave an indication of how well the Gaussian process regression algorithm predicted the GHI values between hourly observations. However, in order to judge the success of the forecasting it is sufficient to compare the forecast GHI values to the hourly observed GHI values over the forecast period. This is because the forecast period does not include the training set, as was the case for the interpolation.

The Python code for the forecasting of the GHI can be found in Appendix B.

5.1.5 Results

Figure 5.7 shows the result of forecasting by training four kernels on dataset 3. Figure 5.6 illustrates the best forecast result obtained by by the $\text{Exp} \times \text{RQ}$ kernel.

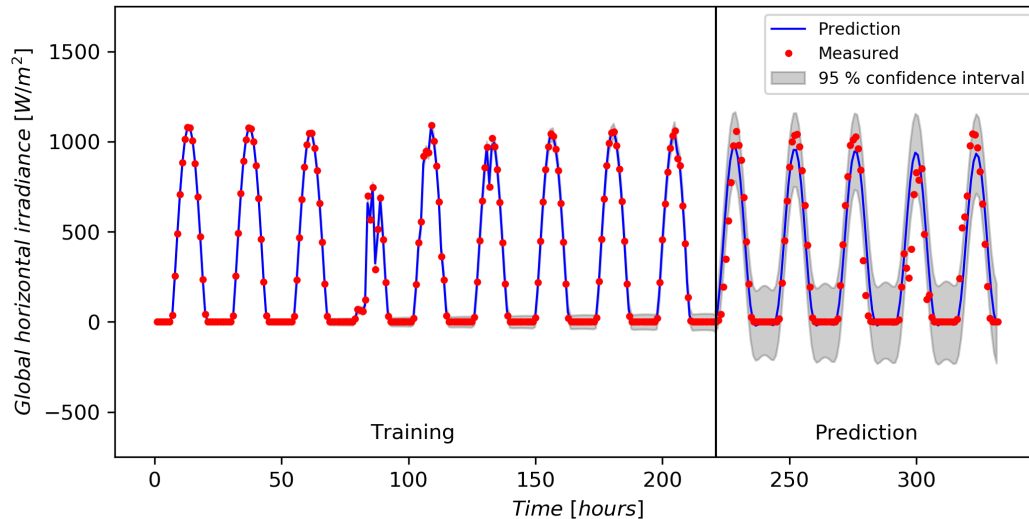


Figure 5.6: Gaussian process forecast using exponential sine-squared, times rational quadratic kernel.

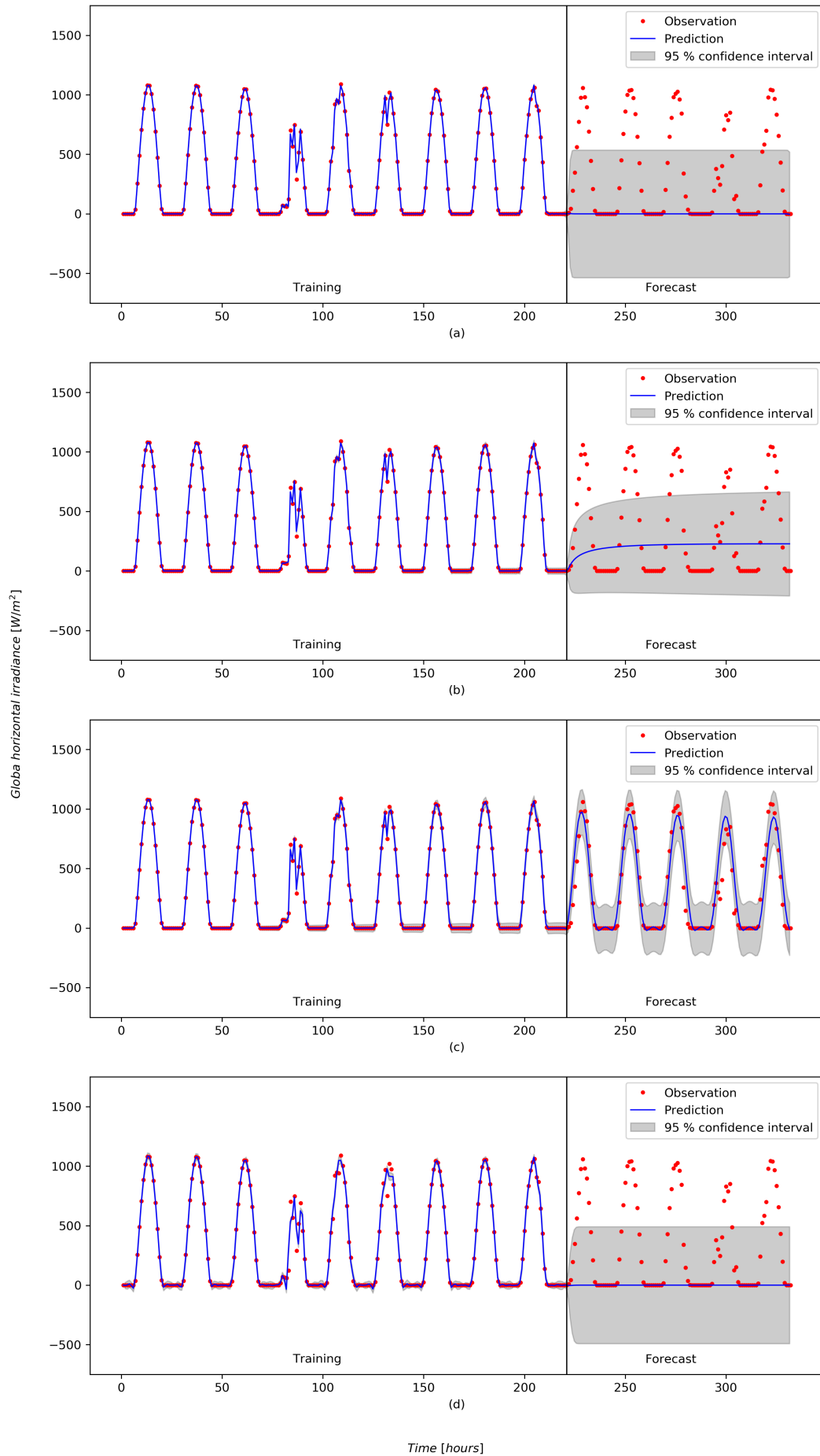


Figure 5.7: Gaussian process forecasting using dataset 3 as multi-input training set with kernels (a) Exp, (b) RQ, (c) Exp × RQ and (d) RBF × (RQ + Exp).

The root-mean-square error, in relation to hourly observed GHI data over the forecast time period, is given in Table 5.5 as a function of the kernel.

Table 5.5: The root-mean-square error on the forecast GHI-values for the different kernels.

Dataset 3	
Exp	$506,7 W/m^2$
RQ	$406,5 W/m^2$
Exp \times RQ	$151,1 W/m^2$
RBF \times (RQ + Exp)	$506,7 W/m^2$

5.1.6 Discussion

The process of interpolation of GHI data through Gaussian process regression is not trivial and clear answers on the reasons for certain kernels performing better than others are not apparent. Before deciding on the final kernels used, other kernels were constructed and tested. These are given in Table 5.6.

Table 5.6: Some kernels that were attempted. Only the kernels constructed analogous to Tolba *et al.* (2019) were found to be effective.

Kernels derived by trial and error
$RBF \times Exp + WhiteKernel$
$C + RBF \times RQ \times Exp$
$RBF \times RBF \times RQ \times Exp$
$0,00000001Exp + RBF \times RBF \times RQ \times Exp$
Kernels derived by fundamental analysis of data patterns
$C \times RBF \times Exp + Exp + RQ + RBF + WhiteKernel$
$Exp + RQ + RBF + WhiteKernel$
Kernels constructed analogous to Tolba <i>et al.</i> (2019)
$C \times Exp + RQ$
$C \times Exp \times RQ$

It was found that no meaningful advantage was attained by increasing the complexity of the kernel. For both interpolation and forecasting, the

$Exp \times RQ$ kernel consistently gave the best fit, as quantified by the root-mean-square error. The second-best fit was delivered by the RQ kernel, while the $RBF \times (RQ + Exp)$ and Exp kernels allowed for the third- and fourth-best fits, respectively. This ranking of kernels are based on the average root-mean-square error that the kernels delivered over the three training sets.

For interpolation, the best fit was delivered by the combination of training dataset 1 and the $Exp \times RQ$ kernel, with a root-mean-square error of $92,8 W/m^2$. For the forecasting, the best fit was achieved by set 3 and the $Exp \times RQ$ kernel with a root-mean-squared error of $151,1 W/m^2$.

Although it is sensible to employ the root-mean-square error to compare the efficacy of the kernel-training-set combinations to each other, caution should be employed when interpreting the error. The results in Table 5.3 indicate that the simulation of meter failure, by omitting 10 datapoints from the training set, results in a 12 % increase in the root-mean-square error. In this case, therefore, a 6 % reduction in training set size results in double the increase the root-mean-square error. The root-mean-square error is used to judge the Gaussian process fit over the whole time period of one week and the effect of meter failure during a relatively small period of time is, unfairly, carried over to the total time period. In other words, a localised error is used to conclude something about the goodness-of-fit during other periods within the time frame. A better approach to quantifying the goodness-of-fit for the Gaussian process regression would be to look at the 95 % confidence intervals on the regression points. This does not give an average error, but rather allows for a localised evaluation of the goodness-of-fit. Note that the confidence interval expands significantly during the periods of simulated meter failure in Figures 5.3, 5.4 and 5.5. It is recommended that the confidence intervals should be used as gauge of success for the Gaussian process regression and forecast.

It is worth scrutinising the two individual kernels that make up the best-performing kernel. The Exp kernel is known to be appropriate for modelling functions that have a repetitive pattern (refer to Chapter 4) – which is indeed visible in the GHI data – while the RQ kernel allows for the incorporation of different length scales – accounting for the irregular variations in the GHI data. It seems, therefore, that these two kernels are sufficient for capturing most of the phenomena present in the data.

An attempt was made to train the Gaussian process regression algorithm on minutely averaged solar radiation data, but the processor proved insufficient to handle the computational burden resulting from the large dataset.

5.2 Wind Speed Data

Wind speed is an important metric in smart renewable energy management. Wind speed data from the Sauran weather station at Stellenbosch University for the week 1 February to 8 February 2015 was used to train a Gaussian

process regression algorithm, utilising the Matérn kernel with $\nu = \frac{3}{2}$. Based on the 95% confidence intervals in the training section of Figure 5.8, the interpolation was successful. An attempt to forecast the wind behaviour failed – see the confidence intervals in the forecast section of Figure 5.8. Success in wind speed forecasting could possibly be achieved by constructing a compound kernel, as the Matérn kernel seems to fail to capture the non-periodic nature of the wind speed over short time intervals.

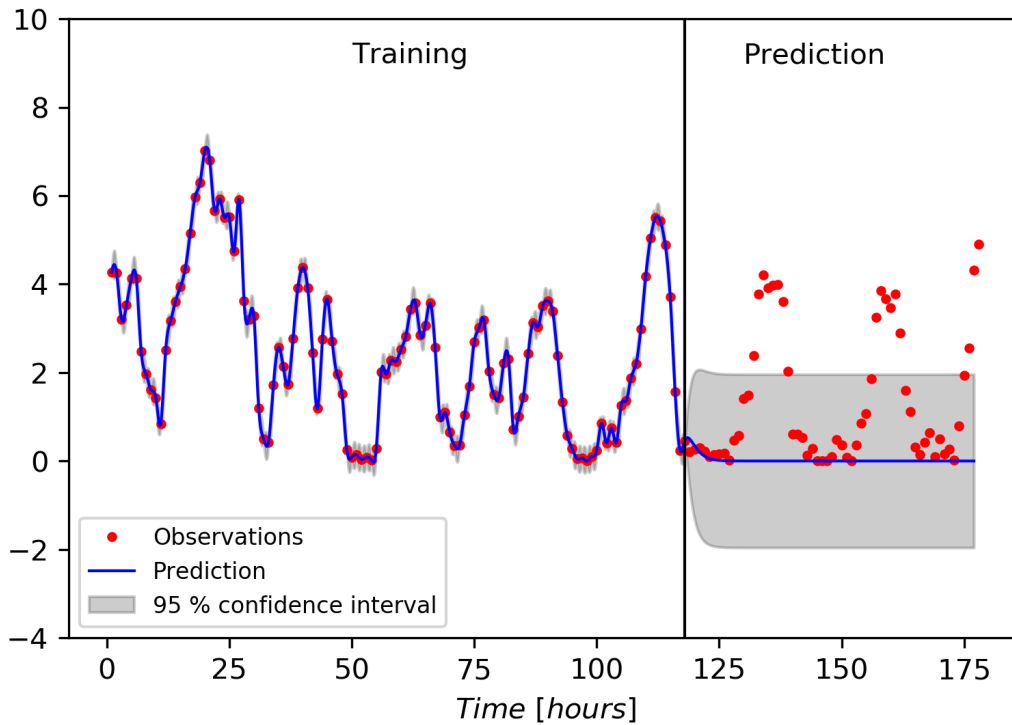


Figure 5.8: Applying Gaussian process regression to wind speed data using the Matérn kernel with $\nu = \frac{3}{2}$.

5.2.1 Dealing with Interval Deficiency

The IEC 61400-12-1:2005(E) standard of the International Electrotechnical Commission sets the requirements for wind speed measurements for the power performance evaluation of wind turbines. The standard requires wind speed data to be sampled at a sampling rate of 1 Hz. Means, standard deviations, minima and maxima must be averaged and stored every 10 minutes (Siepker and Harms, 2017). Data that does not adhere to this standard is called interval-deficient.

The wind energy resource, as a function of wind speed, is described by the Weibull shape and scale parameters. These parameters define the probability

distribution of wind speed. By investigating the behaviour of the Weibull parameters as the interval between wind speed measurements is increased, the effect of interval deficiency on the validity of the wind speed dataset could be gauged.

Siepkner and Harms (2017) have shown that interval-deficient wind speed data might still be useful in wind energy applications. Figure 5.9, adopted from Siepkner and Harms (2017), indicates that the errors in the Weibull parameters become more severe as the intervals between wind speed measurements increase. The wind speed dataset was produced by two anemometers at the Gobabeb Research and Training Centre in Namibia, mounted on a mast at 11,66 m and 27,53 m from the ground (Siepkner and Harms, 2017).

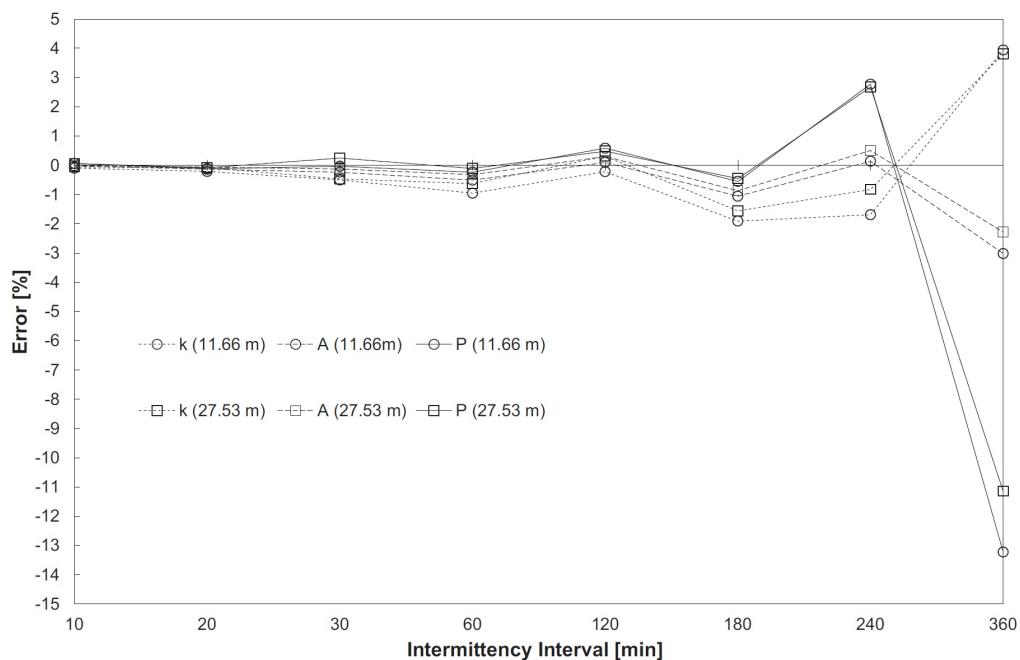


Figure 5.9: Errors in Weibull scale and shape parameters as a function of intermittency interval for wind speed data from the Gobabeb Research and Training Centre. Adopted from (Siepkner and Harms, 2017).

The true Weibull shape and scale parameters, calculated by using the wind speed dataset for the period 1 February to 8 February 2015 that adheres to the IEC 61400-12-1:2005(E) standard, have been found to be 0,6634 and 1,8701 m/s, respectively. 24 wind speed datasets with increasing interval deficiency have been simulated and the Weibull parameters have been calculated for each of these datasets. Refer to Table 5.7.

Table 5.8 explores the possibility of whether Weibull parameters calculated from interval-deficient wind speed datasets could be closer to the true values if

the interval-deficient datasets were interpolated by the use of Gaussian process regression before calculating the Weibull parameters.

5.2.2 Results

Figure 5.10 shows twelve wind speed datasets of increasing interval deficiency, interpolated by Gaussian process regression using the Matérn kernel with $\nu = \frac{3}{2}$. Table 5.7 provides the Weibull shape (k) and scale (A) parameters as a function of interval deficiency. The Weibull parameters were calculated from non-interpolated data. Table 5.8 provides the Weibull shape (k) and scale (A) parameters as a function of interval deficiency, calculated from Gaussian process interpolated data. The true values of the Weibull scale and shape parameters are $1,8701 \text{ m/s}$ and $0,6634$ respectively. Figure 5.11 is a plot of the Weibull scale parameter as a function of interval deficiency. The figure compares the scale parameters calculated from interpolated and non-interpolated data. Figure 5.12 is a plot of the Weibull shape parameter as a function of interval deficiency. The figure compares the shape parameters calculated from interpolated and non-interpolated data.

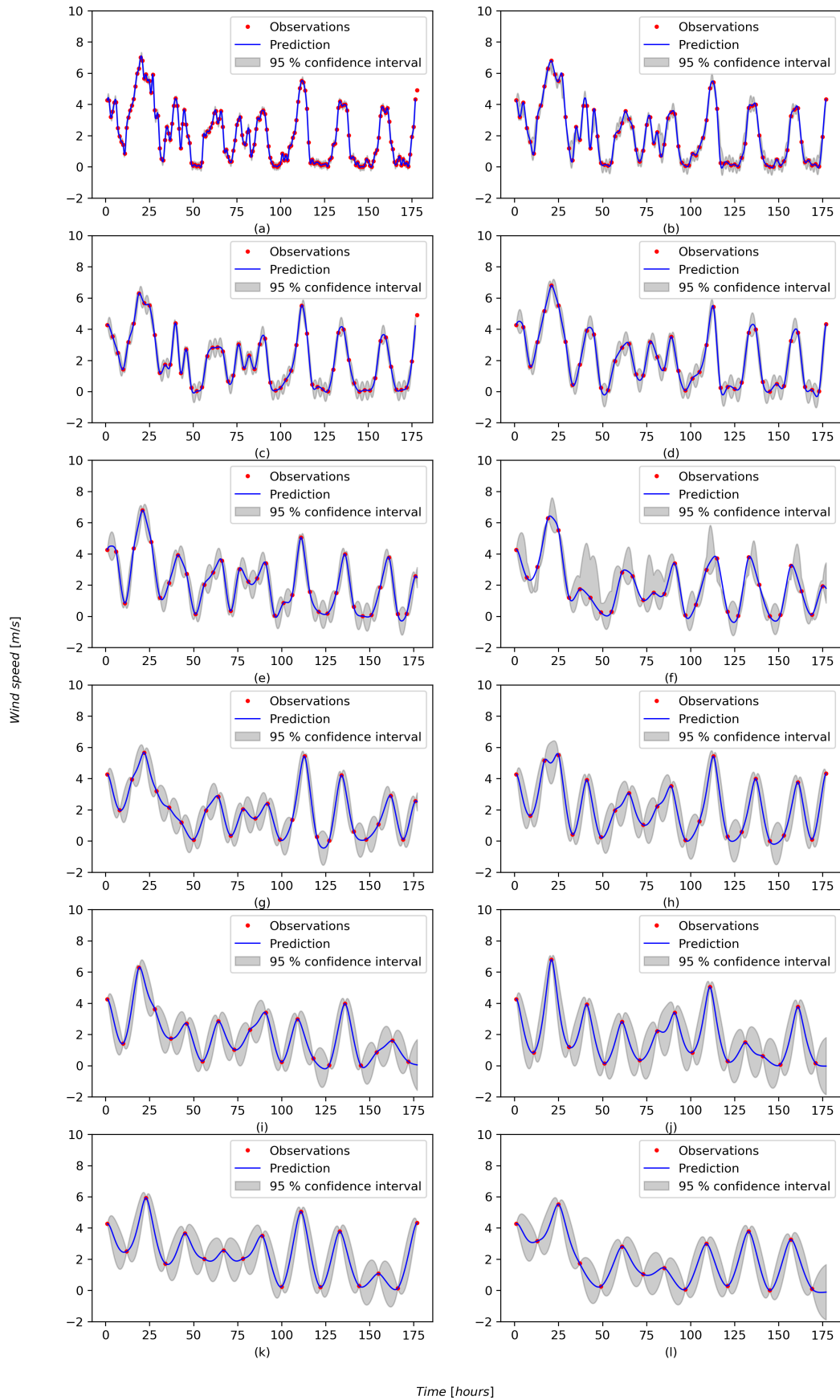


Figure 5.10: Applying Gaussian process regression to interval-deficient wind speed data with intermittency intervals of (a) 1 hour (b) 2 hours (c) 3 hours (d) 4 hours (e) 5 hours (f) 6 hours (g) 7 hours (h) 8 hours (i) 9 hours (j) 10 hours (k) 11 hours and (l) 12 hours.

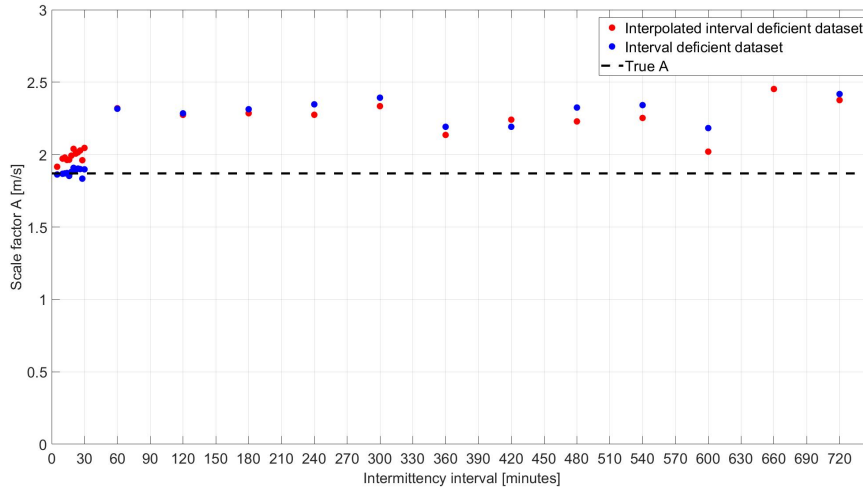


Figure 5.11: Weibull scale parameter A , calculated as a function of interval deficiency using interpolated and non-interpolated data

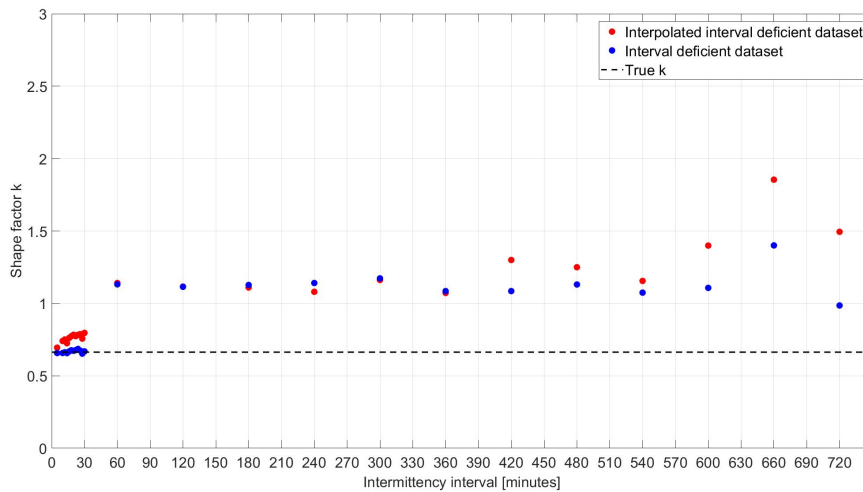


Figure 5.12: Weibull shape parameter k , calculated as a function of interval deficiency using interpolated and non-interpolated data

Table 5.7: Impact of intermittency interval on the values of the Weibull scale and shape parameters.

<i>Interval</i> [minute]	<i>k</i>	<i>error(k)</i> [%]	<i>A</i> [m/s]	<i>error(A)</i> [%]
5	0,6574	-0,9122	1,8625	-0,4059
10	0,6575	-0,8885	1,8673	-0,1489
12	0,6626	-0,1217	1,8702	0,0045
14	0,6569	-0,9836	1,8726	0,1338
16	0,6683	0,7365	1,8526	-0,9403
18	0,6762	1,9283	1,8807	0,5668
20	0,6727	1,3968	1,9085	2,0538
22	0,6775	2,1225	1,8950	1,3305
24	0,6857	3,3639	1,9034	1,7813
26	0,6747	1,7044	1,8999	1,5936
28	0,6532	-1,5457	1,8338	-1,9453
30	0,6688	0,8119	1,8981	1,4954
60	1,1316	70,5634	2,3164	23,8635
120	1,1147	68,0155	2,2847	22,1692
180	1,1272	69,9056	2,3134	23,7026
240	1,1408	71,9589	2,3470	25,5004
300	1,1729	76,8017	2,3929	27,9516
360	1,0849	63,5251	2,1919	17,2079
420	1,0849	63,5251	2,1919	17,2079
480	1,1305	70,4096	2,3248	24,3118
540	1,0744	61,9405	2,3416	25,2088
600	1,1071	66,8796	2,1831	16,7344
660	1,4004	111,0873	2,5840	38,1694
720	0,9857	48,5704	2,4181	29,3017

Table 5.8: Impact of intermittency interval on the values of the Weibull scale and shape parameters. The Weibull parameters were calculated using Gaussian process interpolated data.

<i>Interval</i> [minute]	<i>k</i>	<i>error(k)</i> [%]	<i>A</i> [m/s]	<i>error(A)</i> [%]
5	0,6945	4,6875	1,9159	2,4486
10	0,7405	11,6251	1,9719	5,4441
12	0,7509	13,1850	1,9801	5,8774
14	0,7250	9,2821	1,9631	4,9727
16	0,7616	14,7986	1,9643	5,0335
18	0,7735	16,5848	1,9927	6,5553
20	0,7834	18,0828	2,0397	9,0682
22	0,7738	16,6436	2,0069	7,3127
24	0,7826	17,9631	2,0156	7,7762
26	0,7878	18,7448	2,0280	8,4424
28	0,7568	14,0722	1,9611	4,8661
30	0,7961	19,9958	2,0459	9,3977
60	1,1409	71,9692	2,3199	24,0510
120	1,1160	68,2210	2,2746	21,6255
180	1,1097	67,2751	2,2853	22,2014
240	1,0802	62,8143	2,2748	21,6395
300	1,1621	75,1625	2,3345	24,8311
360	1,0723	61,6339	2,1357	14,2006
420	1,2996	95,8996	2,2411	19,8364
480	1,2499	88,3937	2,2290	19,1896
540	1,1551	74,1160	2,2530	20,4717
600	1,3993	110,9269	2,0202	8,0260
660	1,8543	179,5066	2,4529	31,1628
720	1,4945	125,2772	2,3763	27,0681

5.2.3 Discussion

The percentage deviation of the Weibull parameters from their true values increases as the interval deficiency increases, as would be expected. The increase in deviation is not linear, as can be seen in Figures 5.11 and 5.12. This is in accordance with Siepker and Harms (2017). Both the scale and shape factors deviated more from the true values when calculated from interpolated data. This is somewhat surprising, as one might expect the interpolated data to provide more datapoints and thus decrease interval deficiency. It could be worthwhile to test kernels other than Matérn to test the impact of Gaussian process regression on interval deficiency. The behaviour of the Weibull parameters are nevertheless equivalent whether calculated from interpolated or non-interpolated data. Both the frequentist (non-interpolated) and probabilistic (interpolated) datasets exhibit the same behaviour. This could be seen as a validation of the probabilistic approach. Although the Gaussian process regressed data provided less accurate Weibull parameters, the advantage of this probabilistic approach is that uncertainty is well defined. In a previous work, the author has suggested a procedure that could be followed to find the true Weibull parameters from an interval-deficient wind speed dataset. An article on this study is currently under peer review (Lubbe *et al.*, 2019).

The processor managed to train the Gaussian process regression algorithm on five-minutely averaged wind speed data. Higher-resolution wind speed data proved to be too computationally expensive and the machine crashed.

A successful probabilistic model of solar radiation and wind speed data could be incorporated in the following hypothetical smart-campus scenario. Consider a university campus deriving energy from centralised PV, wind, hydrogen and biogas plants. The generational capacity of this energy system is dependent on weather conditions and an energy manager would require a control system that incorporates weather and demand information in order to schedule energy consumption and storage. Meter failure could also occur in the system and influence the data. Using Gaussian process regression the energy system could be modelled in a robust and probabilistic manner, allowing the energy manager to dispatch energy based on projected energy system behaviour. It could even be possible to forecast periods of excess energy availability, which could then be traded in a smart-grid environment on a future contract basis. If, for example, a prosumer with solar panels on her roof agrees to sell electricity to be dispatched within an hour's time, she might not be sure (due to the intermittent nature of solar power) whether the electricity will be available. A Gaussian process model will be able to attach a certain probability to the availability of solar power within an hour's time. The prosumer might then decide to increase the price of her solar power in accordance with the probability, as she runs the risk of not being able to deliver on the contract.

Chapter 6

Conclusion and Recommendation for Future Work

The research presented in this thesis set out to investigate whether Gaussian process regression could add value to the field of renewable energy by allowing for better utilisation of energy system data within the context of smart-grid technology. Economic literature was explored and it was concluded that smart-grid based energy economics are on the rise.

The variable nature of solar and wind energy poses a challenge to grid stability and this was identified as an area where data-driven smart renewable energy management could provide solutions. It was discovered that the application of Gaussian process regression to wind and solar data has been limited to date and in an attempt to add to the existing literature, Gaussian process regression was applied to solar and wind data from the Sauran weather station at Stellenbosch University.

It was found that a relatively simple kernel, consisting of $Exp \times RQ$, provides good interpolation and prediction capabilities on solar radiation data. This kernel was arrived at after several kernels (constructed by means of fundamental analysis and trial and error) were tested. No meaningful advantage was attained by increasing the complexity of the kernel. It was found that the root-mean-square error is disproportionately affected by a reduction in the training set. Caution should therefore be employed when interpreting the root-mean-square error. A better approach to quantifying the goodness-of-fit for the Gaussian process regression would be to interpret the 95% confidence intervals.

For wind data, the Matérn kernel provided smooth interpolation capabilities; however, failed to provide meaningful forecasts. An investigation into the behaviour of the Weibull parameters as interval deficiency increases, revealed that the Matérn kernel does not provide any benefit when attempting to improve data quality. Future work investigating other kernels should therefore

be undertaken. The Intel Core i7-5600U 2,60 GHz CPU processor that was used proved sufficient for handling the computational requirements of training on an hourly averaged, thirteen-dimensional training dataset. With a two-dimensional training set, the processor was able to handle five-minutely averaged data.

The success achieved in forecasting the behaviour of solar radiation data using Gaussian process regression is encouraging. It is also valuable to note that Gaussian process regression with a Matérn kernel did not improve the quality of interval-deficient wind speed data.

It can be concluded that the application of Gaussian process regression in smart renewable energy management holds promise. It is recommended that future work focus specifically on four areas:

- Finding a rules-based method for solar and wind kernel construction.
- Attempting to use larger real-time datasets for Gaussian process training. This will require better computational power.
- From theory to practice: incorporating Gaussian process regression into a energy management system. This will likely require lessening the computational burden of the Gaussian process regression (ways of doing this have been discussed in Section 3.5).
- From theory to practice: incorporating Gaussian process regression into smart-grid energy trading based on the entire energy ecosystem. Such a project could be undertaken within a smart campus set-up with localised energy storage and embedded generation.

Appendices

Appendix A

GHI Interpolation

Please note: due to page restriction requirements, only a representative amount of code is given in the appendix.

```
##This script tests the interpolation capabilities of four
↳ different kernels on hourly ghi data.

import os
os.system('clear')
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

from mpl_toolkits.axes_grid1.inset_locator import
↳ zoomed_inset_axes, mark_inset
from mpl_toolkits.axes_grid1.anchored_artists import
↳ AnchoredSizeBar

from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import RBF, ConstantKernel
↳ as C, RationalQuadratic as RQ, WhiteKernel,
↳ ExpSineSquared as Exp

np.random.seed(1)

df = pd.read_csv(r'C:\Users\lubbejf\Documents\MEng\Data\2015
↳ _2017_SUN\20150201_SUN_H.dat', sep=',')
df_array = np.asarray(df)

date = df_array[:,0]
```

```

rec_num = df_array[:,1]
ghi = df_array[:,2]
DNI = df_array[:,3]
DHI = df_array[:,4]
DHI_shadowband = df_array[:,5]
UVA = df_array[:,6]
UVB = df_array[:,7]
air_temp = df_array[:,8]
BP = df_array[:,9]
RH = df_array[:,10]
WS = df_array[:,11]
WD = df_array[:,12]
WD_SD = df_array[:,13]

y = np.asarray([ghi]).T

#with no meter failuer
df_no_fail = pd.read_csv(r'C:\Users\lubbejf\Documents\MEng\Data
    ↪ \2015_2017_SUN\20150201_SUN_H_test.dat', sep=',')
df_array_no_fail = np.asarray(df_no_fail)
ghi_no_fail = df_array_no_fail[:,2]
y_no_fail = np.asarray([ghi_no_fail]).T

X = np.atleast_2d
    ↪ ([1.,2.,3.,4.,5.,6.,7.,8.,9.,10.,11.,12.,13.,14.,15.,16.,
17.,18.,19.,20.,21.,22.,23.,24.,25.,26.,27.,28.,29.,30.,31.,32.,
33.,42.,43.,44.,45.,46.,47.,48.,49.,50.,51.,52.,53.,54.,55.,56.,
57.,58.,61.,62.,63.,64.,65.,66.,67.,68.,69.,70.,71.,72.,73.,74.,
75.,76.,77.,78.,79.,80.,81.,82.,83.,84.,85.,86.,87.,88.,89.,90.,
91.,92.,93.,94.,95.,96.,97.,98.,99.,100.,101.,102.,103.,104.,
105.,106.,107.,108.,109.,110.,111.,112.,113.,114.,115.,116.,
117.,118.,119.,120.,121.,122.,123.,124.,125.,126.,127.,128.,
129.,130.,131.,132.,133.,134.,135.,136.,137.,138.,139.,140.,
141.,142.,143.,144.,145.,146.,147.,148.,149.,150.,151.,152.,
153.,154.,155.,156.,157.,158.,159.,160.,161.,162.,163.,164.,
165.,166.,167.,168.,169.,170.,171,172.,173.,174.,175.,176.,
177.]).T

XX = np.atleast_2d
    ↪ ([1.,2.,3.,4.,5.,6.,7.,8.,9.,10.,11.,12.,13.,14.,
15.,16.,17.,18.,19.,20.,21.,22.,23.,24.,25.,26.,27.,28.,29.,30.,
31.,32.,33.,34.,35.,36.,37.,38.,39.,40.,41.,42.,43.,44.,45.,46.,
47.,48.,49.,50.,51.,52.,53.,54.,55.,56.,57.,58., 59., 60.,
61.,62.,63.,64.,65.,66.,67.,68.,69.,70.,71.,72.,73.,74.,75.,

```

```

76.,77.,78.,79.,80.,81.,82.,83.,84.,85.,86.,87.,88.,89.,90.,
91., 92.,93.,94.,95.,96.,97.,98.,99.,100.,101.,102.,103.,104.,
105., 106.,107.,108.,109.,110.,111.,112.,113.,114.,115.,116.,
117., 118.,119.,120.,121.,122.,123.,124.,125.,126.,127.,128.,
129., 130.,131.,132.,133.,134.,135.,136.,137.,138.,139.,140.,
141., 142.,143.,144.,145.,146.,147.,148.,149.,150.,151.,152.,
153., 154.,155.,156.,157.,158.,159.,160.,161.,162.,163.,164.,
165., 166.,167.,168.,169.,170.,171.,172.,173.,174.,175.,176.,
177.,178]).T

#-----

x = np.atleast_2d(np.linspace(1, 177, 10739)).T

# Instantiate a Gaussian Process model
kernel = C()*Exp(length_scale=24,periodicity=1)
gp = GaussianProcessRegressor(kernel=kernel,
    ↪ n_restarts_optimizer=100)

# Fit to data using Maximum Likelihood Estimation of the
    ↪ parameters
gp.fit(X, y)

# Make the prediction on the meshed x-axis (ask for MSE as well
    ↪ )
y_pred_1, sigma_1 = gp.predict(x, return_std=True)

kernel = C()*Exp(length_scale=24,periodicity=1)
gp = GaussianProcessRegressor(kernel=kernel,
    ↪ n_restarts_optimizer=100)
xx = np.atleast_2d(np.linspace(1, 178, 10739)).T
gp.fit(XX, y_no_fail)
y_pred_no_fail_1, sigma_no_fail_1 = gp.predict(xx, return_std=
    ↪ True)

kernel = C()*RQ(length_scale=24, alpha=1)
gp = GaussianProcessRegressor(kernel=kernel,
    ↪ n_restarts_optimizer=100)
gp.fit(XX, y_no_fail)
y_pred_no_fail_2, sigma_no_fail_2 = gp.predict(xx, return_std=
    ↪ True)

#Plot figure
fig = plt.figure(num=1, figsize=(11,0.8), dpi=300, facecolor='w'

```



```

    ↪ , edgecolor='k')
fig.text(0.5, -1, '$Time\ [hours]$', ha='center')
fig.text(0.04, 10, '$Globo\ horizontal\ irradiance\ [W/m^2]$',
    ↪ va='center', rotation='vertical')

plt.subplot(4,1,1)
plt.plot(X, y, 'r.', markersize=5, label=u'Observation')
plt.plot(x, y_pred_1, 'b-', linewidth=1, label=u'Prediction')

plt.fill_between(x[:,0], y_pred_1[:,0] - 1.96*sigma_1, y_pred_1
    ↪[:,0] + 1.96*sigma_1,
alpha = 0.2, color='k', label=u'95%\ confidence\ interval')
plt.xlabel('(a)')
plt.legend(loc='upper\ right', fontsize=10)
plt.ylim(-750,1750)
plt.annotate('Synthesized\ meter\ failure', xy=(38, 950,),
    ↪ xycoords='data',
        xytext=(-100, 30), textcoords='offset\ points',
        arrowprops=dict(arrowstyle="fancy",
            fc="0.25", ec="none",
            connectionstyle="angle3,angleA=0,angleB
                ↪=-90"))
plt.annotate('', xy=(55, 1000,), xycoords='data',
        xytext=(-30, 21), textcoords='offset\ points',
        arrowprops=dict(arrowstyle="fancy",
            fc="0.25", ec="none",
            connectionstyle="angle3,angleA=90,
                ↪ angleB=150"))

plt.subplots_adjust(top = 20)

plt.savefig('four_kernels_hourly.png', bbox_inches='tight')
plt.show('four_kernels_hourly.png')

#calculate root mean square error, comparing interpolated
    ↪ values to minutely ghi data.

df_M = pd.read_csv(r'C:\Users\lubbejf\Documents\MEng\Data\2015
    ↪ _2017_SUN\20150201_SUN_M_week.dat', sep=',')
df_array_M = np.asarray(df_M)

#observed values
ghi_M = df_array_M[:,2]
y_M = np.asarray([ghi_M]).T

```

```

difference = y_pred_1 - y_M
squared = np.square(difference)
average = np.sum(squared)/(np.shape(squared)[0])
rmse_1 = np.sqrt(average)

difference = y_pred_2 - y_M
squared = np.square(difference)
average = np.sum(squared)/(np.shape(squared)[0])
rmse_2 = np.sqrt(average)

print(rmse_1)
print(rmse_2)
print(rmse_3)
print(rmse_4)

print(rmse_no_fail_1)
print(rmse_no_fail_2)
print(rmse_no_fail_3)
print(rmse_no_fail_4)

#no meter failure
fig, ax = plt.subplots(num=1, figsize=(8,5), dpi=300, facecolor=
    ↪ 'w', edgecolor='k')
ax.plot(XX, y_no_fail, 'r.', markersize=5, label=u'Observation')
ax.plot(xx, y_pred_no_fail_3, 'b-', linewidth=1, label=u'
    ↪ Prediction')
ax.fill_between(xx[:,0], y_pred_no_fail_3[:,0] - 1.96*
    ↪ sigma_no_fail_3, y_pred_no_fail_3[:,0]
+ 1.96*sigma_no_fail_3, alpha = 0.2, color='k',label=u'95%
    ↪ confidence interval')

plt.legend(loc='upper_right', fontsize=10)
plt.ylabel('$Global horizontal irradiance [W/m^2]$')
plt.xlabel('$Time [hours]$')
plt.ylim([-500, 4000])

ttt = np.linspace(36,38,120)
axins = zoomed_inset_axes(ax, 30, loc=2)
axins.plot(ttt, y_pred_no_fail_3[2160:2280:1], 'b.', linewidth=1,
    ↪ markersize=2, label=u'Prediction')
axins.plot(ttt, y_M[2160:2280:1], 'r.', markersize=2, label=u'
    ↪ Observation')
x1, x2, y1, y2 = 36, 38, 1030, 1100 # specify the limits

```

```
axins.set_xlim(x1, x2) # apply the x-limits
axins.set_ylim(y1, y2) # apply the y-limits
plt.yticks(visible=False)
plt.xticks(visible=False)
mark_inset(ax, axins, loc1=3, loc2=4, fc='none', ec='0.4')

plt.savefig('no_fail.png', bbox_inches='tight')
plt.show('no_fail.png')
```

Appendix B

GHI Forecasting

Please note: due to page restriction requirements, only a representative amount of code is given in the appendix.

```
##This script tests the forecasting capabilities of four
  ↪ different kernels on hourly ghi data, with all input.

import os
os.system('clear')
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import RBF, ConstantKernel
  ↪ as C,
RationalQuadratic as RQ, WhiteKernel, ExpSineSquared as Exp

np.random.seed(1)

df = pd.read_csv(r'C:\Users\lubbejf\Documents\MEng\Data\2015
  ↪ _2017_SUN\20150201_SUN_H_test_forecasting.dat', sep=',')
df_array = np.asarray(df)

date = df_array[:,0]
rec_num = df_array[:,1]
ghi = df_array[:,2]
DNI = df_array[:,3]
DHI = df_array[:,4]
DHI_shadowband = df_array[:,5]
UVA = df_array[:,6]
UVB = df_array[:,7]
```

```

air_temp = df_array[:,8]
BP = df_array[:,9]
RH = df_array[:,10]
WS = df_array[:,11]
WD = df_array[:,12]
WD_SD = df_array[:,13]

parameters = np.asarray([ghi,DNI,DHI,DHI_shadowband,UVA,UVB,
    ↪ air_temp,RH,WS,WD,WD_SD,BP]).T

#slice parameters in half to create training and test set.

end = np.int((np.shape(parameters)[0])/1.5)
Y = parameters[:end,:] #training set y
ghi = parameters[:,0] #test set y
X_ghi = np.atleast_2d(np.linspace(1,np.shape(ghi)[0],np.shape(
    ↪ ghi)[0])).T
X = np.atleast_2d(np.linspace(1,np.shape(Y)[0],np.shape(Y)[0])).
    ↪ T #training set x
x = np.atleast_2d(np.linspace(1, 1.5*end, 332)).T #test set x

#-----

# Instantiate a Gaussian Process model

kernel = C()*Exp(length_scale=24,periodicity=1)
gp = GaussianProcessRegressor(kernel=kernel,
    ↪ n_restarts_optimizer=100)

# Fit to data using Maximum Likelihood Estimation of the
    ↪ parameters
gp.fit(X, Y)

# Make the prediction on the meshed x-axis (ask for MSE as well
    ↪ )
y_pred_1, sigma_1 = gp.predict(x, return_std=True)

#-----

kernel = C()*RQ(length_scale=24, alpha=1)
gp = GaussianProcessRegressor(kernel=kernel,
    ↪ n_restarts_optimizer=100)

gp.fit(X, Y)

```

```

y_pred_2, sigma_2 = gp.predict(x, return_std=True)

#-----

kernel = C()*Exp(length_scale=24,periodicity=1)
*RQ(length_scale=24, alpha=0.5, length_scale_bounds=(1e-05, 2),
alpha_bounds=(1e-05, 100000.0))
gp = GaussianProcessRegressor(kernel=kernel,
    ↪ n_restarts_optimizer=1000)

gp.fit(X, Y)

y_pred_3, sigma_3 = gp.predict(x, return_std=True)

#-----

kernel = C()*RBF(length_scale=24, length_scale_bounds=(1e-5, 2))
*(RQ(length_scale=1, alpha=0.5, length_scale_bounds=(1e-05, 2),
alpha_bounds=(1e-05, 100000.0)) + Exp(length_scale=24,
    ↪ periodicity=1))

gp = GaussianProcessRegressor(kernel=kernel,
    ↪ n_restarts_optimizer=100)

gp.fit(X, Y)

y_pred_4, sigma_4 = gp.predict(x, return_std=True)

#-----

#Plot figure

fig = plt.figure(num=1, figsize=(11,0.8), dpi=300, facecolor='w'
    ↪ , edgecolor='k')
fig.text(0.5, -1, '$Time\_\_[hours]$', ha='center')
fig.text(0.04, 10, '$Globo\_\_horizontal\_\_irradiance\_\_[W/m^2]$',
    ↪ va='center', rotation='vertical')

plt.subplot(4,1,1)
plt.plot(X_ghi, ghi, 'r.', markersize=5, label=u'Observation')
plt.plot(x, y_pred_1[:,0], 'b-',linewidth=1, label=u'Prediction'
    ↪ )
plt.axvline(x = end, linewidth=1, color='k')

```

```

plt.annotate('Training', xy=(85, -650))
plt.annotate('Forecast', xy=(260, -650))
plt.fill_between(x[:,0], y_pred_1[:,0] - 1.96*sigma_1, y_pred_1
    ↪[:,0] + 1.96*sigma_1,
alpha = 0.2, color='k', label=u'95% confidence interval')
plt.xlabel('(a)')
plt.legend(loc='upper_right', fontsize=10)
plt.ylim(-750,1750)

plt.subplots_adjust(top = 20)

plt.savefig('forecast_four_kernels.png', bbox_inches='tight')
plt.show('forecast_four_kernels.png')

#-----

plt.figure(num=1, figsize=(8,4), dpi=300, facecolor='w',
    ↪ edgecolor='k')

plt.plot(x[:,0], y_pred_3[:,0], 'b-', linewidth=1, label=u'
    ↪ Prediction')
plt.plot(X_ghi,ghi,'r.',markersize = 5,label=u'Measured')
plt.axvline(x = end, linewidth=1, color='k')
plt.annotate('Training', xy=(85, -650))
plt.annotate('Prediction', xy=(260, -650))
plt.fill_between(x[:,0], y_pred_3[:,0] - 1.96*sigma_3, y_pred_3
    ↪[:,0] + 1.96*sigma_3,
alpha = 0.2, color='k',label=u'95% confidence interval')
plt.xlabel('$Time [hours]$')
plt.ylabel('$Global horizontal irradiance [W/m^2]$')
plt.legend(loc='upper_right', fontsize=10)
plt.ylim(-750,1750)

plt.savefig('forecast.png', bbox_inches='tight')
plt.show('forecast.png')

difference = y_pred_1[end::1,0] - ghi[end::1]
squared = np.square(difference)
average = np.sum(squared)/(np.shape(squared)[0])
rmse_1 = np.sqrt(average)

difference = y_pred_2[end::1,0] - ghi[end::1]
squared = np.square(difference)

```

```
average = np.sum(squared)/(np.shape(squared)[0])
rmse_2 = np.sqrt(average)

plt.plot(ghi[end::1], 'r-')
plt.plot(y_pred_3[end::1,0], 'b-')

print(rmse_1)
print(rmse_2)
print(rmse_3)
print(rmse_4)
```


Appendix C

Wind Data

Please note: due to page restriction requirements, only a representative amount of code is given in the appendix.

```
##This script tests the interpolation capabilities of four
  ↪ different kernels on hourly wind data, with multiple
  ↪ input.

#imports
import os
os.system('clear')
import numpy as np
import pandas as pd
from scipy.stats import exponweib
from matplotlib import pyplot as plt
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import RBF, ConstantKernel
  ↪ as C, RationalQuadratic as RQ, WhiteKernel,
ExpSineSquared as Exp, Matern as Mtrn

np.random.seed(1)

df = pd.read_csv(r'C:\Users\lubbejf\Documents\MEng\Data\2015
  ↪ _2017_SUN\20150201_SUN_H_test.dat', sep=',')
df_array = np.asarray(df)

date = df_array[:,0]
rec_num = df_array[:,1]
ghi = df_array[:,2]
DNI = df_array[:,3]
DHI = df_array[:,4]
```

```

DHI_shadowband = df_array[:,5]
UVA = df_array[:,6]
UVB = df_array[:,7]
air_temp = df_array[:,8]
BP = df_array[:,9]
RH = df_array[:,10]
WS = df_array[:,11]
WD = df_array[:,12]
WD_SD = df_array[:,13]

parameters = np.asarray([ghi,DNI,DHI,DHI_shadowband,UVA,UVB,
    ↪ air_temp,RH,WS,WD,WD_SD,BP]).T

end = np.int((np.shape(parameters)[0])/1.5)
Y = parameters[:end,:] #training set y
X_ws = np.atleast_2d(np.linspace(1,np.shape(WS)[0],np.shape(WS)
    ↪ [0])).T

X = np.atleast_2d(np.linspace(1,np.shape(Y)[0],np.shape(Y)[0])).
    ↪ T #training set x
x = np.atleast_2d(np.linspace(1, 1.5*end, 2000)).T #test set x

#-----

# Instantiate a Gaussian Process model
kernel = Mtrn()
gp = GaussianProcessRegressor(kernel=kernel,
    ↪ n_restarts_optimizer=80)

# Fit to data using Maximum Likelihood Estimation of the
    ↪ parameters
gp.fit(X, Y)

# Make the prediction on the meshed x-axis (ask for MSE as well
    ↪ )
y_pred_1, sigma_1 = gp.predict(x, return_std=True)

#-----

kernel = RQ(length_scale=24, alpha=1)
gp = GaussianProcessRegressor(kernel=kernel,
    ↪ n_restarts_optimizer=10)

gp.fit(X, Y)

```

```

y_pred_2, sigma_2 = gp.predict(x, return_std=True)

#-----

kernel = Exp(length_scale=24,periodicity=1)*RQ(length_scale=24,
    ↪ alpha=0.5, length_scale_bounds=(1e-05, 2),
alpha_bounds=(1e-05, 100000.0))

gp = GaussianProcessRegressor(kernel=kernel,
    ↪ n_restarts_optimizer=10)

gp.fit(X, Y)

y_pred_3, sigma_3 = gp.predict(x, return_std=True)

#-----

kernel = RBF(length_scale=24, length_scale_bounds=(1e-5, 2))*(RQ
    ↪ (length_scale=1, alpha=0.5,
length_scale_bounds=(1e-05, 2), alpha_bounds=(1e-05, 100000.0))*
    ↪ Exp(length_scale=24,periodicity=1))

gp = GaussianProcessRegressor(kernel=kernel,
    ↪ n_restarts_optimizer=10)

gp.fit(X, Y)

y_pred_4, sigma_4 = gp.predict(x, return_std=True)

#-----

#Plot figure
fig = plt.figure(num=1, figsize = [11,3],dpi=300, facecolor='w',
    ↪ edgecolor='k')

plt.subplot(2,2,1)
plt.plot(X_ws, WS, 'r.', markersize=5, label=u'Observations')
plt.plot(x, y_pred_1[:,8], 'b-',linewidth=1, label=u'Prediction'
    ↪ )
plt.axvline(x = end, linewidth=1, color='k')
plt.annotate('Training', xy=(50, 9))
plt.annotate('Prediction', xy=(130, 9))
plt.fill_between(x[:,0], y_pred_1[:,8] - 1.96*sigma_1, y_pred_1
    ↪[:,8] + 1.96*sigma_1,

```

```

alpha = 0.2,
    color='k', label=u'95% confidence interval')
plt.xlabel('(a)')
plt.legend(loc='best', fontsize=8)
plt.ylim(-4,10)

plt.subplots_adjust(top = 2)

plt.savefig('wind_2.png', bbox_inches='tight')
plt.show('wind_2.png')

#-----

df_minutely = pd.read_csv(r'C:\Users\lubbejf\Documents\MEng\Code
    ↪ \notebook\20150201_SUN_M.dat', sep=',')
df_array_minutely = np.asarray(df_minutely)
WS_minutely = df_array_minutely[:,11]
kernel_1 = Mtrn()

#-----

#every 5 minutes
WS_5_minutes = list(WS_minutely[0::5])
np.savetxt('WS_5_minutes.txt', WS_5_minutes)
X_5_minutes = np.atleast_2d(np.linspace(1, np.shape(WS_5_minutes)
    ↪ [0], np.shape(WS_5_minutes)[0])).T
end_5_minutely = np.int((np.shape(WS_5_minutes)[0])/1.5)

xx = np.atleast_2d(np.linspace(1, 1.5*end_5_minutely, 4000)).T
gp = GaussianProcessRegressor(kernel=kernel_1,
    ↪ n_restarts_optimizer=1)

gp.fit(X_5_minutes, WS_5_minutes)
y_pred_5_minutely, sigma_5_minutely = gp.predict(xx, return_std=
    ↪ True)
np.savetxt('ypred_5_minutely.txt', y_pred_5_minutely)
np.savetxt('sigma_5_minutely.txt', sigma_5_minutely)

#-----

#every 10 minutes
WS_10_minutes = list(WS_minutely[0::10])
np.savetxt('WS_10_minutes.txt', WS_10_minutes)
X_10_minutes = np.atleast_2d(np.linspace(1, np.shape(

```

```

    ↪ WS_10_minutes)[0],np.shape(WS_10_minutes)[0])).T
end_10_minutely = np.int((np.shape(WS_10_minutes)[0])/1.5)

xx = np.atleast_2d(np.linspace(1, 1.5*end_10_minutely, 4000)).T
gp = GaussianProcessRegressor(kernel=kernel_1,
    ↪ n_restarts_optimizer=1)

np.savetxt('WS_10_minutes.txt', WS_10_minutes)
gp.fit(X_10_minutes, WS_10_minutes)
y_pred_10_minutely, sigma_10_minutely = gp.predict(xx,
    ↪ return_std=True)
np.savetxt('ypred_10_minutely.txt',y_pred_10_minutely)
np.savetxt('sigma_10_minutely.txt',sigma_10_minutely)

#every 12 minutes
WS_12_minutes = list(WS_minutely[0::12])
np.savetxt('WS_12_minutes.txt',WS_12_minutes)
X_12_minutes = np.atleast_2d(np.linspace(1,np.shape(
    ↪ WS_12_minutes)[0],np.shape(WS_12_minutes)[0])).T
end_12_minutely = np.int((np.shape(WS_12_minutes)[0])/1.5)

xx = np.atleast_2d(np.linspace(1, 1.5*end_12_minutely, 4000)).T
gp = GaussianProcessRegressor(kernel=kernel_1,
    ↪ n_restarts_optimizer=1)

np.savetxt('WS_12_minutes.txt', WS_12_minutes)
gp.fit(X_12_minutes, WS_12_minutes)
y_pred_12_minutely, sigma_12_minutely = gp.predict(xx,
    ↪ return_std=True)
np.savetxt('ypred_12_minutely.txt',y_pred_12_minutely)
np.savetxt('sigma_12_minutely.txt',sigma_12_minutely)

#every 14 minutes
WS_14_minutes = list(WS_minutely[0::14])
np.savetxt('WS_14_minutes.txt',WS_14_minutes)
X_14_minutes = np.atleast_2d(np.linspace(1,np.shape(
    ↪ WS_14_minutes)[0],np.shape(WS_14_minutes)[0])).T
end_14_minutely = np.int((np.shape(WS_14_minutes)[0])/1.5)

xx = np.atleast_2d(np.linspace(1, 1.5*end_14_minutely, 4000)).T
gp = GaussianProcessRegressor(kernel=kernel_1,
    ↪ n_restarts_optimizer=1)

np.savetxt('WS_14_minutes.txt', WS_14_minutes)

```

```

gp.fit(X_14_minutes, WS_14_minutes)
y_pred_14_minutely, sigma_14_minutely = gp.predict(xx,
    ↪ return_std=True)
np.savetxt('ypred_14_minutely.txt',y_pred_14_minutely)
np.savetxt('sigma_14_minutely.txt',sigma_14_minutely)

#-----

#plot interval deficient interpolations up to half hourly
plt.figure(10)
plt.plot(xx, y_pred_5_minutely, 'r-', linewidth=1)
plt.figure(11)
plt.plot(xx, y_pred_10_minutely, 'r-', linewidth=1)
plt.figure(12)
plt.plot(xx, y_pred_12_minutely, 'r-', linewidth=1)
plt.figure(13)
plt.plot(xx, y_pred_14_minutely, 'r-', linewidth=1)
plt.figure(14)

#-----

#synthesize interval deficient wind speed data and fit Gaussian
    ↪ process regression. Also write data to .txt-file
#for import to matlab.

x = np.atleast_2d(np.linspace(1, 1.5*end, 4000)).T #test set x
kernel = Exp(length_scale=24,periodicity=0.5)
kernel_1 = Mtrn()
gp = GaussianProcessRegressor(kernel=kernel_1,
    ↪ n_restarts_optimizer=80)

#every 1 hour
WS_1 = list(WS)
np.savetxt('WS_1.txt',WS_1)
X_ws1 = X_ws
gp.fit(X_ws1, WS_1)
y_pred_1, sigma_1 = gp.predict(x, return_std=True)
np.savetxt('ypred_1.txt',y_pred_1)
np.savetxt('sigma_1.txt',sigma_1)

#every 2 hours
WS_2 = list(WS[0::2])
np.savetxt('WS_2.txt',WS_2)
X_ws2 = X_ws[0::2]

```

```

gp.fit(X_ws2, WS_2)
y_pred_2, sigma_2 = gp.predict(x, return_std=True)
np.savetxt('ypred_2.txt',y_pred_2)
np.savetxt('sigma_2.txt',sigma_2)

#every 3 hours
WS_3 = list(WS[0::3])
np.savetxt('WS_3.txt',WS_3)
X_ws3 = X_ws[0::3]
gp.fit(X_ws3, WS_3)
y_pred_3, sigma_3 = gp.predict(x, return_std=True)
np.savetxt('ypred_3.txt',y_pred_3)
np.savetxt('sigma_3.txt',sigma_3)

#-----

#plot interval deficient wind speed data
fig = plt.figure(num=1, figsize=(11,0.75), dpi=300, facecolor='w
    ↪ ', edgecolor='k')
fig.text(0.5, -1, '$Time\_[hours]$', ha='center')
fig.text(0.04, 10, '$Wind\_[speed]\_[m/s]$', va='center', rotation
    ↪ ='vertical')

plt.subplot(6,2,1)
plt.plot(X_ws1, WS_1, 'r.', markersize=5, label=u'Observations')
plt.plot(x, y_pred_1, 'b-', linewidth=1, label=u'Prediction')
plt.fill_between(x[:,0], y_pred_1 - 1.96*sigma_1, y_pred_1 +
    ↪ 1.96*sigma_1, alpha = 0.2,
color='k',label=u'95\_%\_[confidence]\_[interval]')
plt.legend(loc='upper\_[right]', fontsize=8)
plt.xlabel('(a)')
plt.ylim(-2,10)

plt.subplots_adjust(top = 20)

plt.savefig('wind_deficient.png', bbox_inches='tight')
plt.show('wind_deficient.png')

```

```

%% import data
%interpolated data

filename = 'C:\Users\lubbejf\Documents\MEng\Code\notebook\
    ↪ ypred_5_minutely.txt';
delimiter = {' '};

```

```

formatSpec = '%f%[\n\r]';
fileID = fopen(filename,'r');
dataArray = textscan(fileID, formatSpec, 'Delimiter', delimiter,
    ↪ 'TextType', 'string', 'EmptyValue', NaN, 'ReturnOnError',
    ↪ false);
fclose(fileID);
ypred5minutely = [dataArray{1:end-1}];
clearvars filename delimiter formatSpec fileID dataArray ans;

filename = 'C:\Users\lubbejf\Documents\MEng\Code\notebook\
    ↪ ypred_1.txt';
delimiter = {' '};
formatSpec = '%f%[\n\r]';
fileID = fopen(filename,'r');
dataArray = textscan(fileID, formatSpec, 'Delimiter', delimiter,
    ↪ 'TextType', 'string',
    ↪ 'EmptyValue', NaN, 'ReturnOnError', false);
fclose(fileID);
ypred1 = [dataArray{1:end-1}];
clearvars filename delimiter formatSpec fileID dataArray ans;

%% create upper and lower confidence arrays

upper_1 = ypred1 + 1.96*sigma1;
lower_1 = ypred1 - 1.96*sigma1;

%-----

figure(1)
hold on
plot(upper_12)
plot(ypred12,'LineWidth',2)
plot(lower_12)
legend('Upper_95%_confidence_bound', 'Predicted_wind_speed', '
    ↪ Lower_95%_confidence_bound')
hold off

for i = 1:length(lower_1)
    if lower_1(i) <= 0
        lower_1(i)=0.001;
    end
end

for i = 1:length(lower_2)

```



```

    if lower_2(i) <= 0
        lower_2(i)=0.001;
    end
end
end

figure(6)
hold on
plot(upper_12)
plot(ypred12,'LineWidth',2)
plot(lower_12)
ylim([-2 7])
legend('Upper_95%_confidence_bound','Predicted_wind_speed','
    ↪ Lower_95%_confidence_bound')
hold off

%
%%

%-----

%non-interpolated data
filename = 'C:\Users\lubbejf\Documents\MEng\Code\notebook\
    ↪ WS_5_minutes.txt';
delimiter = {' '};
formatSpec = '%f%[\n\r]';
fileID = fopen(filename,'r');
dataArray = textscan(fileID, formatSpec, 'Delimiter', delimiter,
    ↪ 'TextType', 'string', 'ReturnOnError', false);
fclose(fileID);
WS5minutes = [dataArray{1:end-1}];
clearvars filename delimiter formatSpec fileID dataArray ans;

filename = 'C:\Users\lubbejf\Documents\MEng\Code\notebook\
    ↪ WS_10_minutes.txt';
delimiter = {' '};
formatSpec = '%f%[\n\r]';
fileID = fopen(filename,'r');
dataArray = textscan(fileID, formatSpec, 'Delimiter', delimiter,
    ↪ 'TextType', 'string', 'ReturnOnError', false);
fclose(fileID);
WS10minutes = [dataArray{1:end-1}];
clearvars filename delimiter formatSpec fileID dataArray ans;

%-----

```



```

        end
    end
    % Convert numeric text to numbers.
    if ~invalidThousandsSeparator
        numbers = textscan(char(strrep(numbers, ',', '')),
            ↪ '%f');
        numericData(row, col) = numbers{1};
        raw{row, col} = numbers{1};
    end
catch
    raw{row, col} = rawData{row};
end
end
end

R = cellfun(@(x) ~isnumeric(x) && ~islogical(x),raw); % Find non
    ↪ -numeric cells
raw(R) = {NaN}; % Replace non-numeric cells

SUNM = cell2mat(raw);

clearvars filename delimiter formatSpec fileID dataArray ans raw
    ↪ col numericData rawData
row regexstr result numbers invalidThousandsSeparator
    ↪ thousandsRegExp R;

WS_minutely = SUNM(:,12);

%
%% remove all negative values

%interpolated data
for i = 1:length(ypred5minutely)
    if ypred5minutely(i) <= 0
        ypred5minutely(i)=0.001;
    end
end

for i = 1:length(ypred10minutely)
    if ypred10minutely(i) <= 0
        ypred10minutely(i)=0.001;
    end
end
end

```

```

%-----
%non-interpolated data

for i = 1:length(WS5minutes)
    if WS5minutes(i) <= 0
        WS5minutes(i)=0.001;
    end
end

for i = 1:length(WS10minutes)
    if WS10minutes(i) <= 0
        WS10minutes(i)=0.001;
    end
end

%-----

%% calculate true Weibull parameters from minutely data

xxxx = linspace(1,length(WS_minutely),332);
pdWeibull_true = fitdist(WS_minutely,'weibull'); %Find Weibull
    ↪ parameters
                                %using MAXIMUM LIKELIHOOD
                                ↪ method.
A_true = pdWeibull_true.A; %Calculate Weibull A
k_true = pdWeibull_true.B; %Calculate Weibull k

%-----

figure(2)
plot(WS_minutely)
hold on
plot(xxxx,WS1,'r-','LineWidth',2)

A_true
k_true
%

%-----

%% calculate Weibull parameters from interpolated data

A = zeros([1 24]); %Prelocation to be populated with Weibull

```

```

    ↪ parameter A
k = zeros([1 24]); %Prelocation to be populated with Weibull
    ↪ parameter k
%A_upper = zeros([1 12]);
%k_upper = zeros([1 12]);
%A_lower = zeros([1 12]);
%k_lower = zeros([1 12]);

pdWeibull_5_minutely = fitdist(ypred5minutely,'weibull'); %Find
    ↪ Weibull parameters
A(1) = pdWeibull_5_minutely.A; %Calculate Weibull A
k(1) = pdWeibull_5_minutely.B; %Calculate Weibull k

pdWeibull_10_minutely = fitdist(ypred10minutely,'weibull'); %
    ↪ Find Weibull parameters
A(2) = pdWeibull_10_minutely.A; %Calculate Weibull A
k(2) = pdWeibull_10_minutely.B; %Calculate Weibull k

pdWeibull_12_minutely = fitdist(ypred12minutely,'weibull'); %
    ↪ Find Weibull parameters
A(3) = pdWeibull_12_minutely.A; %Calculate Weibull A
k(3) = pdWeibull_12_minutely.B; %Calculate Weibull k

pdWeibull_14_minutely = fitdist(ypred14minutely,'weibull'); %
    ↪ Find Weibull parameters
A(4) = pdWeibull_14_minutely.A; %Calculate Weibull A
k(4) = pdWeibull_14_minutely.B; %Calculate Weibull k

%-----

A_interpolate = A(1:24);
k_interpolate = k(1:24);

error_A_interpolate = zeros([1 24]);
error_k_interpolate = zeros([1 24]);

for j = 1:24
    error_A_interpolate(j) = ((A_interpolate(j)-A_true)/A_true)
        ↪ *100;
    error_k_interpolate(j) = ((k_interpolate(j)-k_true)/k_true)
        ↪ *100;
end

A_interpolate

```

```

k_interpolate
error_A_interpolate
error_k_interpolate

%-----

%
%% calculate Weibull parameters from deficient wind speed data

A = zeros([1 24]); %Prelocation to be populated with Weibull
    ↪ parameter A
k = zeros([1 24]); %Prelocation to be populated with Weibull
    ↪ parameter k

pdWeibull_5_minutely = fitdist(WS5minutes,'weibull'); %Find
    ↪ Weibull parameters
                                %using MAXIMUM LIKELIHOOD
                                ↪ method.
A(1) = pdWeibull_5_minutely.A; %Calculate Weibull A
k(1) = pdWeibull_5_minutely.B; %Calculate Weibull k

pdWeibull_10_minutely = fitdist(WS10minutes,'weibull'); %Find
    ↪ Weibull parameters
                                %using MAXIMUM LIKELIHOOD
                                ↪ method.
A(2) = pdWeibull_10_minutely.A; %Calculate Weibull A
k(2) = pdWeibull_10_minutely.B; %Calculate Weibull k

pdWeibull_12_minutely = fitdist(WS12minutes,'weibull'); %Find
    ↪ Weibull parameters
                                %using MAXIMUM LIKELIHOOD
                                ↪ method.
A(3) = pdWeibull_12_minutely.A; %Calculate Weibull A
k(3) = pdWeibull_12_minutely.B; %Calculate Weibull k

pdWeibull_14_minutely = fitdist(WS14minutes,'weibull'); %Find
    ↪ Weibull parameters
                                %using MAXIMUM LIKELIHOOD
                                ↪ method.
A(4) = pdWeibull_14_minutely.A; %Calculate Weibull A
k(4) = pdWeibull_14_minutely.B; %Calculate Weibull k

%-----

```

```

A_deficient = A(1:24);
k_deficient = k(1:24);

for t = 1:24
    error_A_deficient(t) = ((A_deficient(t)-A_true)/A_true)*100;
    error_k_deficient(t) = ((k_deficient(t)-k_true)/k_true)*100;
end

A_deficient
k_deficient
error_A_deficient
error_k_deficient

%-----

%% plot Weibull parameters
interval = [5 10 12 14 16 18 20 22 24 26 28 30 60 120 180 240
    ↪ 300 360 420 480 540 600 660 720]
figure(3)
hold on
grid on
box on
plot(interval, A_interpolate,'r.','MarkerSize',30)
ax1 = gca;
ax1.XAxis.Limits = [0 750];
set(ax1,'XTick',(0:30:725),'FontSize',20)
ax1.YAxis.Limits = [0 3];
xlabel('Intermittency_interval [minutes]','FontSize',20)
ylabel('Scale_factor_A [m/s]','FontSize',20)
plot(interval, A_deficient,'b.','MarkerSize',30)
plot([0, 750], [A_true, A_true],'k--','LineWidth',3)
lgd = legend('Interpolated_interval_deficient_dataset','Interval
    ↪ _deficient_dataset','True_A');
lgd.FontSize = 20;
hold off

figure(4)
hold on
grid on
box on
plot(interval, k_interpolate,'r.','MarkerSize',30)
ax2 = gca;
ax2.XAxis.Limits = [0 750];
set(ax2,'XTick',(0:30:725),'FontSize',20)

```

```
ax2.YAxis.Limits = [0 3];
xlabel('Intermittency_interval [minutes]', 'FontSize', 20)
ylabel('Shape_factor_k', 'FontSize', 20)
plot(interval, k_deficient, 'b.', 'MarkerSize', 30)
plot([0, 750], [k_true, k_true], 'k--', 'LineWidth', 2)
lgd = legend('Interpolated_interval_deficient_dataset', 'Interval
    ↪ _deficient_dataset', 'True_k');
lgd.FontSize = 20;
hold off
```


Appendix D

Instrumentation

Figure D.1 lists the technical specifications of the Sauran weather station at Stellenbosch University that was used to collect data for the study.



SAURAN Station Details: SUN Stellenbosch University, Stellenbosch, Western Cape Province, South Africa

The SAURAN SUN station is located at Stellenbosch University, in Stellenbosch, South Africa. The instruments are positioned on the roof of an Engineering building with very good solar exposure. The horizons are partially obscured by surrounding mountains. The station was installed on 24 May 2010.

SAURAN station code: SUN	Stellenbosch University, Stellenbosch, Western Cape, South Africa	Latitude: -33.9281° (E) Longitude: 18.8654° (S) Elevation: 119 m AMSL		
Database notation:	Measurement:	Instrument:	Serial number:	Last calibration:
GHI_CMP11	Global horizontal irradiance in [W/m ²]	Kipp & Zonen CMP11 unshaded	114667	28 Jul 2011
DNI_CHP1	Direct normal irradiance in [W/m ²]	Kipp & Zonen CHP1 on a SOLYS tracker	100235	18 Jul 2012
DHI_CMP11	Diffuse horizontal irradiance in [W/m ²]	Kipp & Zonen CMP11 under a shading ball on a SOLYS tracker	114668	28 Jul 2011
DHI_CMP11_ShadowBand	Diffuse horizontal irradiance in [W/m ²]	Kipp & Zonen CMP11 under a shadowband		
UVA_Avg	UVA in [W/m ²]	Kipp & Zonen UVS-AB-T	120113	24 Oct 2012
UVB_Avg	UVB in [W/m ²]	Kipp & Zonen UVS-AB-T	120113	24 Oct 2012
Air_Temp	Air temperature in [°C]	Campbell Scientific CS215 sensor		
BP	Barometric pressure in [mbar]	Vaisala PTB110 sensor		
RH	Relative humidity in [%]	Campbell Scientific CS215 sensor		
WS	Wind speed in [m/s]	R.M.Young 03001 sensor		
WD	Wind direction in [°]	R.M.Young 03001 sensor		
WD_SD	Standard deviation of the wind direction in [°]	R.M.Young 03001 sensor		

Note that some gaps are present in the data record for SUN, including the following:

17/02/2014 11:18 – 12:05 08/09/2014 15:12 – 09/09/2014 14:04
 20/02/2014 12:17 – 13:04 04/12/2014 06:17 – 08/01/2015 08:54
 21/02/2014 08:17 – 09:14
 21/02/2014 09:21 – 19:33
 18/06/2014 21:18 – 22:05

Figure D.1: The technical specifications of the Sauran weather station at Stellenbosch University (Brooks *et al.*, 2015).

References

- Aalami, H.A., Moghaddam, M.P. and Yousefi, G.R. (2010). Modeling and prioritizing demand response programs in power markets. *Electric Power Systems Research*, vol. 80, no. 4, pp. 426–435.
- Al-Dahidi, S., Ayadi, O., Adeeb, J., Alrbai, M. and Qawasmeh, B.R. (2018). Extreme learning machines for solar photovoltaic power predictions. *Energies*, vol. 11, no. 10. Available at: <https://www.mdpi.com/1996-1073/11/10/2725>[Accessed July 1st, 2019].
- Altin, M., Göksu, Ö., Teodorescu, R., Rodriguez, P., Bak-Jensen, B. and Helle, L. (2010). Overview of recent grid codes for wind power integration. In: *Proceedings of the 12th International Conference on Optimization of Electrical and Electronic Equipment, OPTIM 2010*, pp. 1152 – 1160.
- Amin, M. (2008). Challenges in reliability, security, efficiency, and resilience of energy infrastructure: toward smart self-healing electric power grid. In: *2008 IEEE Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century*, pp. 1–5.
- Bloomberg (2019). Clean energy investment trends, 2018. Tech. Rep.. Available at: <https://data.bloomberglp.com/professional/sites/24/BNEF-Clean-Energy-Investment-Trends-2018.pdf> [Accessed March 21st, 2019].
- Brooks, M.J., du Clou, S., van Niekerk, J.L., Gauche, P., Leonard, C., Mouzouris, M.J., Meyer, A.J., van der Westhuizen, N., van Dyk, E. and Vorster, F.J. (2015). SAURAN: A new resource for solar radiometric data in Southern Africa. *Journal of Energy in Southern Africa*, vol. 26, no. 1, pp. 2–10.
- Bumpus, A. and Comello, S. (2017). Emerging clean energy technology investment trends. *Nature Climate Change*, vol. 7, no. 6, pp. 382–385.
- Bundesverband der Energie- und Wasserwirtschaft (2017). Erneuerbare Energien decken 35 Prozent des Strombedarfs. Tech. Rep.. Available at: <https://www.bdew.de/presse/presseinformationen/pi-erneuerbare-energien-decken-35-prozent-des-strombedarfs/> [Accessed September 18th, 2018].
- Burkhart, M.C., Heo, Y. and Zavala, V.M. (2014). Measurement and verification of building systems under uncertain data: a Gaussian process modeling approach. *Energy and Buildings*, vol. 75, pp. 189–198.

- Carstens, H., Xia, X. and Yadavalli, S. (2018). Bayesian energy measurement and verification analysis. *Energies*, vol. 11, no. 2. Available at: <https://www.mdpi.com/1996-1073/11/2/380> [Accessed July 30th, 2018].
- Deng, Y., Khon, E., Liang, Y., Lim, T., Ng, J.W. and Yin, L. (2014). GPU implementation of Gaussian processes. Tech. Rep.. Available at: <http://www.doc.ic.ac.uk/~mpd37/theses/2014-msc-group-project-gpgpu.pdf> [Accessed August 20th, 2019].
- Department of Energy Republic of South Africa (2018). Integrated resource plan 2018 final draft for public input. Available at: <http://www.energy.gov.za/IRP/irp-update-draft-report2018/IRP-Update-2018-Draft-for-Comments.pdf> [Accessed September 18th, 2018].
- Department of Science and Technology Republic of South Africa (2008). Innovation towards a knowledge-based economy: ten-year plan for South Africa (2008 - 2018). Tech. Rep.. Available at: <http://unpan1.un.org/intradoc/groups/public/documents/CPSI/UNPAN027810.pdf> [Accessed August 7th, 2018].
- Disch, R. (2018). Rolf Disch solar architektuur. Available at: <http://www.rolfdisch.de/en/projects/the-solar-settlement/> [Accessed September 17th, 2018].
- Duvenaud, D.K. (2014). *Automatic model construction with Gaussian processes*. Ph.D. thesis, University of Cambridge.
- Duvenaud, D.K., Lloyd, J.R., Grosse, R., Tenenbaum, J.B. and Ghahramani, Z. (2013). Structure discovery in nonparametric regression through compositional kernel search. *arXiv e-prints*. Available at: <https://arxiv.org/abs/1302.4922> [Accessed March 19, 2019].
- Eskom (2016). Effective management of energy demand. Available at: <http://www.eskom.co.za/sites/idm/Business/Documents/EskomAdvisoryService14042016.pdf> [Accessed August 24th, 2019].
- European Climate Foundation (2010). Roadmap 2050 - volume 2 policy report. Available at: http://www.roadmap2050.eu/attachments/files/Volume2_Policy.pdf [Accessed August 7th, 2018].
- Farooq, M. and Salhi, A. (2011). Improving the solvability of ill-conditioned systems of linear equations by reducing the condition number of their matrices. *Journal of the Korean Mathematical Society*, vol. 48, no. 5, pp. 939–952.
- Financial Times (2019). Markets data. Available at: <https://markets.ft.com/data/indices/tearsheet/summary?s=INX:IOM> [Accessed March 26th, 2019].
- Fluri, T.P. (2009). The potential of concentrating solar power in South Africa. *Energy Policy*, vol. 37, no. 12, pp. 5075–5080.

- Franey, M., Ranjan, P. and Chipman, H. (2012). A short note on Gaussian process modeling for large datasets using graphics processing units. *arXiv e-prints*. Available at: <https://arxiv.org/abs/1203.1269> [Accessed August 20th, 2019].
- General Electric Company (2016). Become a digital industrial company. Available at: https://www.ge.com/digital/sites/default/files/download_assets/Become-digital-industrial-company-GE-Digital-overview.pdf [Accessed July 8th, 2018].
- Gray, J.J. (2018). Carl Friedrich Gauss. Available at: <https://www.britannica.com/biography/Carl-Friedrich-Gauss> [Accessed November 19th, 2018].
- Groenfeldt, T. (2012). Big data meets the smart electrical grid. *Forbes*. Available at: <https://www.forbes.com/sites/tomgroenfeldt/2012/05/09/big-data-meets-the-smart-electrical-grid/#5e462ba840ea> [Accessed July 28th, 2018].
- Hamer, W., Booysen, W. and Mathews, E. (2017). A practical approach to managing uncertainty in the measurement and verification of energy efficiency savings. *South African Journal of Industrial Engineering*, vol. 28, no. November, pp. 128–146.
- Han, Q., Wu, H., Hu, T. and Chu, F. (2018). Short-term wind speed forecasting based on signal decomposing algorithm and hybrid linear/nonlinear models. *Energies*, vol. 11, no. 11. Available at: <https://www.mdpi.com/1996-1073/11/11/2976/htm> [Accessed August 24th, 2019].
- Intergovernmental Panel on Climate Change (2014). Mitigation of climate change: summary for policymakers and technical summary. Tech. Rep.. Available at: https://www.ipcc.ch/site/assets/uploads/2018/03/WGIIIAR5_SPM_TS_Volume-3.pdf [Accessed August 7th, 2018].
- IPMVP (2002). Concepts and options for determining energy and water savings. Available at: <https://www.nrel.gov/docs/fy02osti/31505.pdf> [Accessed August 7th, 2018].
- Javaid, N., Hussain, S.M., Ullah, I., Noor, M.A., Abdul, W., Almogren, A. and Alamri, A. (2017). Demand side management in nearly zero energy buildings using heuristic optimizations. *Energies*, vol. 10, no. 8. Available at: <https://www.mdpi.com/1996-1073/10/8/1131> [Accessed August 7th, 2018].
- Kalogirou, S.A. (2004). Solar thermal collectors and applications. *Progress in Energy and Combustion Science*, vol. 30, no. 3, pp. 231–295.
- Keeling, R., Piper, S., Bollenbacher, A. and Walker, J. (2009). Atmospheric CO₂ records from sites in the SIO air sampling network. Tech. Rep., Carbon Dioxide Information Analysis Center, Oak Ridge National Laboratory, U.S Department of Energy. Available at <https://cdiac.ess-dive.lbl.gov/trends/co2/sio-mlo.html> [Accessed April 8th, 2019].

- Khatib, H. (2012). IEA world energy outlook 2011 - a comment. *Energy Policy*, vol. 48, pp. 737–743.
- Kolmogorov, A.N. (1941). Interpolation und extrapolation von stationären zufälligen folgen. *Izvestiya Academy of Sciences of the USSR*, vol. 5, no. 1, pp. 3–14.
- Kunzig, R. and Locatelli, L. (2015). Germany could be a model for how we'll get power in the future. *National Geographic*. Available at: <https://www.nationalgeographic.com/magazine/2015/11/germany-renewable-energy-revolution/#close> [Accessed September 18th, 2018].
- Lagrange, L.F. (2019). personal communication.
- Lean, J., Beer, J. and Bradley, R. (1995). Reconstruction of solar irradiance since 1610: implications for climate change. *Geophysical Research Letters*, vol. 22, no. 23, pp. 3195 – 3198.
- Lean, J. and Rind, D. (2000). The sun and climate. Tech. Rep., U.S. Geological Survey. Available at: <https://pubs.er.usgs.gov/publication/fs09500> [Accessed April 17th, 2019].
- Lubbe, J., Maritz, J. and Harms, T. (2019). A Statistical Analysis of Wind Data. *Journal of Energy in Southern Africa*. Manuscript submitted for publication.
- Ma, J. and Ma, X. (2017). State-of-the-art forecasting algorithms for microgrids. In: *2017 23rd International Conference on Automation and Computing (ICAC)*, pp. 1–6.
- Maritz, J., Lubbe, F. and Lagrange, L. (2018). A practical guide to gaussian process regression for energy measurement and verification within the bayesian framework. *Energies*, vol. 11, no. 4. Available at: <https://www.mdpi.com/1996-1073/11/4/935> [Accessed August 7th, 2018].
- Martín, L., Zarzalejo, L.F., Polo, J., Navarro, A., Marchante, R. and Cony, M. (2010). Prediction of global solar irradiance based on time series analysis: application to solar thermal power plants energy production planning. *Solar Energy*, vol. 84, no. 10, pp. 1772–1781.
- Matheron, G. (1973). The intrinsic random functions and their applications. *Advances in Applied Probability*, vol. 5, no. 3, pp. 439–468.
- McCrone, A., Moslener, U., D'Estais, F. and Grüning, C. (2018). Global trends in renewable energy investment 2018. Tech. Rep., FS-UNEP Collaborating Centre. Available at: http://www.iberglobal.com/files/2018/renewable_trends.pdf [Accessed August 16th, 2018].
- Minnitt, R., Assibey-Bonsu, W. and Camisani-Calzolari, F. (2003). Keynote address: a tribute to Prof. D. G. Krige for his contributions over a period of more than half a century. *Operations Research*, pp. 405–408.

- Momoh, J.A. (2009). Smart grid design for efficient and flexible power networks operation and control. In: *2009 IEEE/PES Power Systems Conference and Exposition*, pp. 1–8.
- Mosavi, A., Salimi, M., Faizollahzadeh Ardabili, S., Rabczuk, T., Shamshirband, S. and Varkonyi-Koczy, A. (2019). State of the art of machine learning models in energy systems, a systematic review. *Energies*, vol. 12, no. 7. Available at: <https://www.mdpi.com/1996-1073/12/7/1301> [Accessed April 8th, 2019].
- Murphy, K.P. (2012). *Machine learning: a probabilistic perspective*. MIT Press, Cambridge, Massachusetts.
- Neal, R.M. (1999). Regression and classification using Gaussian process priors. *Bayesian Statistics*, vol. 6, pp. 475–501.
- Nespoli, A., Ogliari, E., Leva, S., Massi Pavan, A., Mellit, A., Lughi, V. and Dolara, A. (2019). Day-ahead photovoltaic forecasting: a comparison of the most effective techniques. *Energies*, vol. 12, no. 9. Available at: <https://www.mdpi.com/1996-1073/12/9/1621> [Accessed May 29th, 2019].
- Owens, J.D., Houston, M., Luebke, D., Green, S., Stone, J.E. and Phillips, J.C. (2008). Gpu computing. *Proceedings of the IEEE*, vol. 96, no. 5, pp. 879–899.
- Parisio, A., Rikos, E. and Glielmo, L. (2016). Stochastic model predictive control for economic/environmental operation management of microgrids: an experimental case study. *Journal of Process Control*, vol. 43, pp. 24–37.
- Park, C. and Huang, J.Z. (2016). Efficient computation of Gaussian process regression for large spatial data sets by patching local Gaussian processes. *Journal of Machine Learning Research*, vol. 17, pp. 1–29.
- Parkins, D. (2017). The world’s most valuable resource is no longer oil, but data. *The Economist*. Available at: <https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data> [Accessed August 7th, 2018].
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011). Scikit-learn: machine learning in Python. *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830.
- Polanco Vasquez, L.O., Carreño Meneses, C.A., Pizano Martínez, A., López Redondo, J., Pérez García, M. and Álvarez Hervás, J.D. (2018). Optimal energy management within a microgrid: a comparative study. *Energies*, vol. 11, no. 8. Available at: <https://www.mdpi.com/1996-1073/11/8/2167> [Accessed August 21st, 2018].

- Prakash, A.K., Xu, S., Rajagopal, R. and Noh, H.Y. (2018). Robust building energy load forecasting using physically-based kernel models. *Energies*, vol. 11, no. 4. Available at: <https://www.mdpi.com/1996-1073/11/4/862> [Accessed August 7th, 2018].
- Rasmussen, C.E. and Williams, C.K.I. (2006). *Gaussian processes for machine learning*. MIT Press, Cambridge, Massachusetts.
- Reddy, T.A. and Claridge, D.E. (2000). Uncertainty of measured energy savings from statistical baseline models. *HVAC&R Research*, vol. 6, no. 1, pp. 3–20.
- Rogelj, J., Luderer, G., Pietzcker, R.C., Kriegler, E., Schaeffer, M., Krey, V. and Riahi, K. (2015). Energy system transformations for limiting end-of-century warming to below 1.5 °C. *Nature Climate Change*, vol. 5, no. 6, pp. 519–527.
- Schirru, A., Pampuri, S., De Nicolao, G. and McLoone, S. (2011). Efficient marginal likelihood computation for Gaussian process regression. *arXiv e-prints*. Available at <https://arxiv.org/abs/1110.6546> [Accessed August 22nd, 2019].
- Serafin, T. and Uniejewski, B. (2019). Averaging predictive distributions across calibration windows for day-ahead electricity price forecasting. *Energies*, vol. 12, no. 13. Available at: <https://www.mdpi.com/1996-1073/12/13/2561> [Accessed August 18th, 2019].
- Siegmund, D. (2018). Probability theory. Available at: <https://www.britannica.com/science/probability-theory> [Accessed November 22nd, 2018].
- Siepkner, E.B.J. and Harms, T.M. (2017). On the potential value of interval deficient wind data. *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 165, pp. 100–114.
- Sivaram, V., Comello, S.D., Victor, D.G., Sekaric, L., Hertz-Hagel, B., Fox-Penner, P., Aggarwalla, R.T., Bradbury, K., Garg, S., Ibrahim, E., Scott, J., O’Leary, J., Kauffman, R. and Ho, H.K. (2018). *Digital decarbonization: promoting digital innovations to advance clean energy systems*. Maurice R. Greenberg Centre for Geoeconomic Studies, New York, New York.
- Smith, S.W. (1997). Chapter 26: neural networks (and more!). In: *The Scientist and Engineer’s Guide to Digital Signal Processing*, pp. 451–480.
- Su, M., Zhang, Z., Zhu, Y., Zha, D. and Wen, W. (2019). Data driven natural gas spot price prediction models using machine learning methods. *Energies*, vol. 12, no. 9. Available at: <https://www.mdpi.com/1996-1073/12/9/1680> [Accessed July 1st, 2019].
- Teske, S., Pregger, T., Simon, S., Naegler, T., Graus, W. and Lins, C. (2011). Energy [r]evolution 2010—a sustainable world energy outlook. *Energy Efficiency*, vol. 4, no. 3, pp. 409–433.

- Tolba, H., Dkhili, N., Nou, J., Eynard, J., Thil, S., Tolba, H., Dkhili, N., Nou, J., Eynard, J. and Thil, S. (2019). GHI forecasting using Gaussian process regression. In: *IFAC Workshop on Control of Smart Grid and Renewable Energy Systems*. Jeju, South Korea.
- United Nations Framework Convention on Climate Change (2015). Paris agreement. In: *Conference of the parties on its twenty-first session*. Available at: <https://unfccc.int/resource/docs/2015/cop21/eng/109r01.pdf> [Accessed August 7th, 2019].
- Wågberg, J., Zachariah, D., Schön, T.B. and Stoica, P. (2016). Prediction performance after learning in Gaussian process regression. vol. 54.
- Wiener, N. (1949). *Extrapolation, interpolation, and smoothing of stationary time series with engineering applications*. MIT Press, Cambridge, Massachusetts.
- Yang, Y., Li, S., Li, W. and Qu, M. (2018). Power load probability density forecasting using Gaussian process quantile regression. *Applied Energy*, vol. 213, pp. 499–509.
- Zhou, K., Fu, C. and Yang, S. (2016). Big data driven smart energy management: from big data to big insights. *Renewable and Sustainable Energy Reviews*, vol. 56, pp. 215 – 225.
- Zhou, K., Yang, S., Shen, C., Ding, S. and Sun, C. (2015). Energy conservation and emission reduction of China's electric power industry. *Renewable and Sustainable Energy Reviews*, vol. 45, pp. 10–19.