Wind Turbine Dynamics Identification using Gaussian Process Machine Learning

PhD Thesis

Edward Hart

Wind and Marine Energy Systems CDT

University of Strathclyde, Glasgow

October 19, 2018

This thesis is the result of the author's original research. It has been composed by the author and has not been previously submitted for examination which has led to the award of a degree.

Acknowledgements

This thesis is dedicated to my wife, Rosie, whose love and support throughout has meant everything to me.

I would like to thank my supervisors, Bill Leithead and Julian Feuchtwang, for their support and encouragement throughout my research, along with everyone else at the Wind and Marine Energy Systems CDT. I would also like to thank the EPSRC for funding this work under project reference number EP/L016680/1.

"I'm not televising that!" - Paul 'Shanny' Horrocks

Abstract

Wind turbine controllers require dynamic information about the turbine for design and operation purposes. These dynamics are currently determined from simulation models during the turbine design stages. Hence, the dynamics for a given operational turbine will not be identical to those assumed by the controller due to manufacturing and construction variations. Furthermore, turbine dynamics are known to change over time due to environmental effects such as blade erosion. There are currently no known methods by which such information can be determined for an operational turbine. This thesis presents such a method.

The determination of sought dynamic information is formulated as a regression problem involving data available to a wind turbine controller. The nature of the dynamics identification problem is shown to necessitate a regression method which is able to process data in batches, updating predictions as new data becomes available.

Gaussian process machine learning is chosen as the regression approach best suited for application in this problem. However, a review of existing batched Gaussian process theory results in the identification of gaps in the current knowledge base which render existing methods unsuitable. A new approach to batched Gaussian process regression is therefore developed, Sufficient-Subset Gaussian process iteration, which addresses the questions for which existing theories come up short. In the process of developing this new method fundamental contributions have been made to the areas of Gaussian process polynomial regression and sparse Gaussian process approximation theory.

Sufficient-Subset Gaussian process iteration is applied to both simulated and real tur-

bine data and shown to be able to identify the sought dynamics to within a 3% error threshold. Additionally, a related regression formulation corresponding to maximum efficiency tracking is shown to present a potential method for turbine monitoring and fault detection.

Contents

List of Figures

List of Tables

Table of Symbols

A	wind turbine rotor area (m^2)
C_p	wind turbine aerodynamic efficiency coefficient
C_Q	wind turbine aerodynamic torque coefficient
J	wind turbine rotor inertia ($kg\ m^2$)
k	Gaussian process covariance function
$\mathcal{L}$	wind turbine drivetrain torque losses (Nm)
N	wind turbine gearbox ratio
P	wind turbine power (MW)
$\mathbf{P}$	Gaussian process Sufficient-Subset
Q_{aer}	wind turbine aerodynamic torque (MNm)
Q_{gen}	wind turbine generator torque (MNm)
R	wind turbine rotor radius (m)
s	frequency domain variable for Laplace transform (rad/s)
v	wind turbine rotor effective wind speed (m/s)
β	wind turbine pitch angle ($degrees$)
λ	wind turbine tip-speed ratio
Λ	covariance matrix
μ	Gaussian process mean function
ξ	noise variance hyperparameter
ϕ	Gaussian process log-hyperparameter
Φ	Sufficient-Subset Gaussian process covariance matrix
ρ	density of air (kg/m^3)
ω_r	wind turbine rotor rotational speed (rad/s)

ω_{gen} wind turbine generator rotational speed (rad/s)

Ω noise covariance matrix

$\#$ cardinality (of a set)

Chapter 1

Introduction

This thesis details the development of a technique for the identification of wind turbine dynamic information relevant to control. The development of such a technique has required fundamental questions concerning batched approaches to Gaussian process machine learning to be addressed, and the development of a novel approach suitable for application in the turbine dynamics problem.

This chapter will briefly summarise the contents of this thesis, as well as outlining its principal contributions and publications to date.

1.1 Thesis Overview

Chapter 1, provides a brief overview of the thesis and its contributions to knowledge.

In Chapter 2, relevant background information is given concerning wind energy, wind turbines and their control in various operating regions. The aerodynamic information relevant to turbine control is shown to be related to data available to the controller, and this relationship takes the form of two regression equations, loosely relating to 'maximum efficiency' and 'above rated' operation respectively. The characteristics of the available data, and requirements of potential regression approaches are discussed. A key requirement being the ability to handle batched data. This chapter concludes by introducing the turbine simulation model and real turbine data used in this thesis.

Chapter 3 gives an overview of the machine learning regression approach chosen for use in the dynamics identification problem, Gaussian process machine learning. The reasons for choosing this technique are also outlined.

Chapter 4 reviews existing approaches to Gaussian process machine learning for batched data. It is shown that existing theory fails to answer some fundamental questions concerning Gaussian process models in the batched setting. The answers to these questions are required in order to fulfill the learning criteria outlined in Chapter 2 and, hence, it is concluded that a new approach (and associated theory) is necessary. The chapter therefore concludes by outlining two new possible approaches for further study, these being *combined-posteriors* and *posterior-to-prior* iteration.

Chapter 5 consist of a comprehensive study of Gaussian process polynomial regression. This is due to one of the turbine dynamics regression equations being polynomial in form, while also representing a key case in which to understand batched approaches to GPs. It is shown that, in the polynomial case, Gaussian process machine learning can be formulated incredibly efficiently, and with linear scaling. A full maximum-likelihood approach to polynomial GPs is developed which scales linearly and includes derivative and Hessian information. The developed approach is then compared with Least-Squares regression both theoretically and experimentally. Finally the two proposed approaches to batched regression, from Chapter 4, are applied in the polynomial case. *Combined-posteriors* is shown to exhibit anomalous behaviour and so is abandoned as a possible method. The *posterior-to-prior* approach on the other hand is shown to be equivalent to standard GP regression, but, with linear scaling and without increasing computational or memory requirements as the number of processed datapoints increases.

Chapter 6 extends to the general case the lessons learned from *posterior-to-prior* iteration for polynomials. This is done through a principled consideration of covariance matrix structure. The developed theory is shown to lead to a new approach to batched Gaussian processes, Sufficient-Subset Gaussian process (SSGP) iteration. SSGPs are developed into a practical regression method. Their implementation and theoretical

properties are explored and tested with respect to accuracy and computational efficiency, SSGPs are shown to scale linearly in the number of datapoints. Comparisons between SSGPs, standard Gaussian process regression and existing batched approaches are then performed. It is shown that SSGP theory provides comprehensive answers to the questions posed in Chapter 4 for which existing theory comes up short. Finally, it is shown that there is a lot of potential for SSGP theory to be further developed beyond the point reached in the current work.

Chapter 7 applies the developed regression approaches to the turbine dynamics identification problem. In each case this is done for both simulated and measured data. First, the correct processing and conditioning of measured data is discussed. Then the regression problem for 'maximum efficiency' operation is considered. This is the polynomial regression case, and so the developed approach of Chapter 5 is applied and its performance compared to that of Least Squares regression. The Gaussian process approach is shown to be superior. The 'above rated' regression problem is considered next. In order to develop a general SSGP prior for wind turbine dynamic tables, theory for the handling of functions with varying lengthscales in the Gaussian process context is developed. Issues related to the distribution of available data across the operating envelope are shown to lead to a necessary reformulation of the regression equation in the 'above rated' region. An SSGP prior is constructed for the reformulated problem and SSGP regression tested on simulated and real data.

Chapter 8 summarises the results of the thesis, draws conclusions and discusses future work with respect to both SSGP theory and wind turbine dynamics identification.

Following this are the appendices and bibliography.

1.2 Contributions to Knowledge

The main contributions to knowledge of this thesis are:

- The development of a Gaussian process polynomial regression implementation

which scales linearly in the number of measured datapoints and includes both derivative and Hessian information for hyperparameter learning[1].

- The development of a new approach to Gaussian process machine learning for batched data, Sufficient-Subset Gaussian process iteration, the underlying theory of which provides answers to some fundamental questions concerning such models.

- An iterative regression procedure for learning a wind turbine's aerodynamic sensitivity function from data available to a wind turbine controller, allowing for predictions to be updated in time as new data becomes available.

1.3 Publications

E. Hart, W. E. Leithead and J. Feuchtwang, "Wind Turbine $C_{p,max}$ and Drivetrain-Losses Estimation using Gaussian Process Machine Learning", Journal of Physics: Conf. Series 1037 (2018) 032024.

E. Hart and W. E. Leithead, "Sufficient-Subset Gaussian Process Iterative Regression", Submitted to the Journal of Machine Learning Research - currently under revision.

[1] As will be discussed in the relevant section, some of the necessary theory for implementing such a method has been known prior to this work, however, to the best of the author's knowledge a full implementation as presented here has never been documented before now.

Chapter 2

Background, Motivation and Problem Statement

The world is facing an existential crisis due to global warming, caused by human carbon emissions. The generation of electricity is responsible for a large percentage of these emissions due to our reliance on fossil fuels such as coal and gas. The UK and other countries around the world have committed to reducing their carbon emissions in order to try and prevent further rises in global average temperatures and so avoid the catastrophic effects which rising global temperatures would have on the world and its ecosystems.

The required reductions in carbon emissions mean that electrical energy production will need to eventually be sourced almost entirely from renewable resources, and the sooner the better. Wind energy, along with solar, tidal and wave are some of the renewable energy technologies undergoing the most research currently. Wind and solar are far ahead of the rest in terms of development and uptake.

As of December 2017 there was 169 GW of installed wind energy capacity in the EU [3]. 2017 was a record year for both onshore and offshore installations, with the EU adding 12,484 MW onshore and 3,154 MW offshore [3]. Furthermore, wind energy now covers over 11.6% of the EU's electricity demand [3]. Therefore, wind energy is playing a significant role in efforts to cut carbon emissions and current trends suggest it will continue playing a significant role in the future.

With limited space and popularity for onshore wind farms, the offshore sector is going to grow considerably in the near future. This means larger turbines being placed further away from maintenance and repair capabilities and in more hostile environments. Therefore, turbines must become more reliable and have greater load tolerances along with a lower cost of energy [4]. The development of wind energy technology is therefore crucial to ensuring the success of this industry and the associated reduction in carbon emissions.

Sections 2.1-2.4 give an overview of wind turbine technology and theory relevant to this work. Section 2.5 then introduces the key motivating thesis question. In Section 2.6 it is shown that this question can be formulated as a regression problem. Learning requirements and data characteristics with respect to the developed regression equations are then discussed.

2.1 Wind Turbines

Wind energy had been used for hundreds of years in the form of windmills for milling grain [5]. Modern wind turbines which convert wind energy to electricity look similar enough to windmills that they are recognisably using the same principles to transfer kinetic energy in the wind into rotational energy in the rotor and shaft. Wind turbine technology has now standardised and nowadays wind turbines tend to be three bladed, horizontal axis machines as in Figure 2.1.

2.1.1 Types of Wind Turbines

Various modes of wind turbine operation have been developed and implemented over the years; these include methods for limiting the maximum power and loads in high wind speeds, as well as for improving aerodynamic efficiencies. These methods are combinations of Variable/Constant speed and Pitch/Stall regulation. Constant speed wind turbines are connected directly to the grid and therefore operate at synchronous speed with very little variation; Variable speed turbines are isolated from the grid by power converters, allowing for variable speed operation of the wind turbine. Variable speed operation allows for greater aerodynamic efficiency across the operating envelope

Figure 2.1: Wind turbine

of the wind turbine, as will be discussed in Section 2.4. Pitch and Stall regulation of a wind turbine are techniques for limiting the maximum power (and hence loads) generated by the wind turbine in high wind speeds. Stall regulation requires that the wind turbine blades are designed to enter stall (thus losing aerodynamic efficiency and preventing load/power increases) at the wind speed where power is to be limited. Stall regulation relies heavily on accurate design, if design/manufacturing errors are present they cannot be fixed retrospectively. Stall regulation does however benefit from being entirely passive, not requiring the input of other systems in the turbine. Pitch regulation limits power and loads via rotation of the blades at their base in order to alter the angle of attack for the blades. This requires motorized pitch actuators which can be a source of downtime if they fail. Variable Speed Pitch Regulated (VSPR) turbines have greater control capabilities and efficiencies compared to the other possible choices and so this has become the dominant wind turbine type. This work will therefore be considering VSPR wind turbines.

2.1.2 VSPR Wind Turbines

The equation describing the power extracted from the wind by a VSPR wind turbine is,

$$P = \frac{1}{2}\rho A v^3 C_P, \tag{2.1}$$

where ρ is the density of air, A is the rotor swept area, v is the rotor effective wind speed and C_P is the power coefficient [6]. The power coefficient, C_P, gives the fraction of the power in the wind that is converted to mechanical work by the turbine. The value of C_P depends on the tip speed ratio, λ, and the blade pitch angle, β. The tip speed ratio is defined as being the ratio of the blade tip speed to the incoming wind speed;

$$\lambda = \frac{\omega_r R}{v},$$

with ω_r the rotor rotational speed and R the rotor radius. We therefore have $C_P = C_P(\lambda, \beta)$, and this dependence is due to the fact that aerodynamically the lift and drag for each wind turbine blade section will depend on the apparent wind speed, being the vector sum of the incoming wind velocity and the blade velocity, seen by that section. It is λ and β which determine the apparent wind speed and aerodynamic response, and hence C_P. Generally speaking the pitch angle is only ever changed in above rated operation, once the turbine has reached its maximum or *rated* power, below this it remains fixed at the value for which the highest efficiency of the turbine can be obtained; therefore, the dependence on β for C_P is often dropped when considering below rated operation and instead we simply have $C_P = C_P(\lambda)$. Figure 2.2 shows the $C_P - \lambda$ curve for the Supergen Exemplar 5MW wind turbine model. In the figure

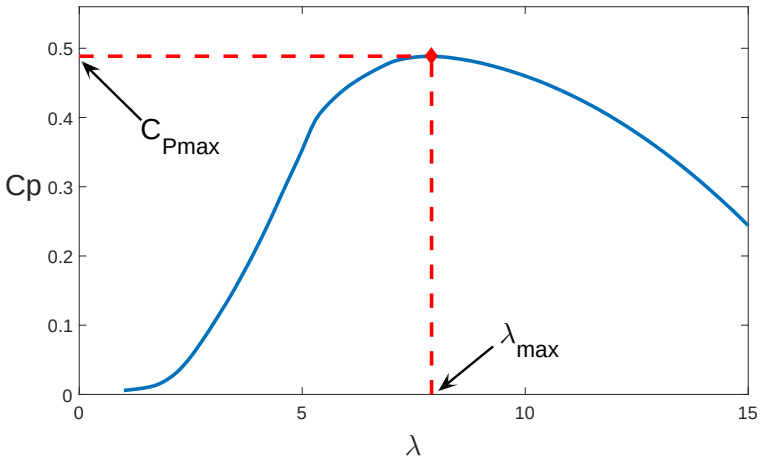

Figure 2.2: Cp-λ curve.

the maximum value of C_P is 0.4885, occurring at a tip speed ratio of 7.9. The tip

speed ratio at which C_{Pmax} occurs will be referred to as λ_{max}, as shown in the figure. Therefore, for this particular turbine $\lambda_{max} = 7.9$. There is a theoretical maximum C_P value of 0.593 (known as the Betz limit), however, in practice wind turbine C_P values tend to fall quite far below this value [6].

2.2 The Power Curve

A wind turbine is meant to operate according to its design power curve. The wind turbine's power curve relates the wind speed to the power coming out of the wind turbine. The power curve for the Supergen Exemplar 5MW wind turbine is shown in Figure 2.3; on this power curve can be seen a section where the power increases with wind speed as in Equation 2.1, with $C_P = C_{Pmax}$ throughout, followed by a short transitional section from about $11.5 - 12\ m/s$, before a final flat section where the power is kept at its rated value. The considerations from which this power curve

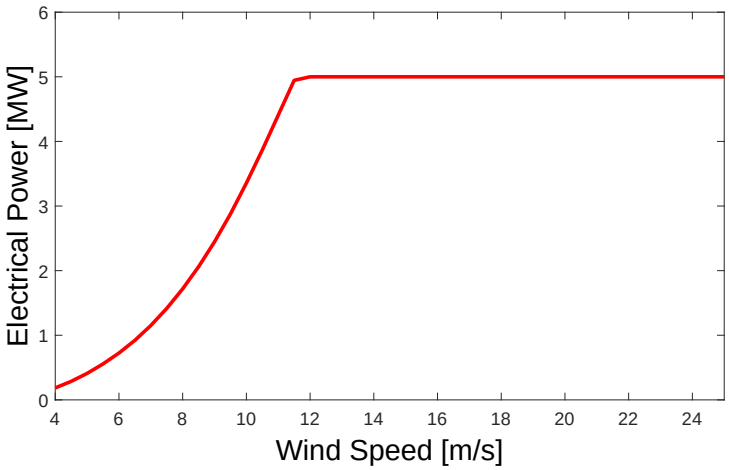

Figure 2.3: Power curve.

are designed for a given wind turbine include things such as the generator size, cost, efficiency at various wind speeds and the wind resource available in the part of the world the wind turbine will be operated. More importantly for the current work is how, having designed a power curve, can the VSPR wind turbine be made to follow it? This is discussed and expanded on in later sections.

2.3 Effective Wind Speed

Thus far the terms *wind speed* and *effective wind speed* have been used interchangeably, however, there are important distinctions to be made here and so these terms will now be put on firmer footing. Intuitively, when the wind is blowing we would refer to the wind speed as being the speed measured at a certain point by a cup anemometer or similar device. Since the wind is continuous, turbulent and spatially non-uniform, this type of wind speed should in fact be referred to as a *point wind speed* for obvious reasons. The size of a wind turbine rotor means that it is simultaneously interacting with a very large cross section of the wind field; hence point wind speeds will not be particularly meaningful. When referring to effective wind speed, or just wind speed, for a wind turbine we are loosely talking about an average wind speed across the rotor; for example considering the power curve in Figure 2.3, a turbulent wind field is considered to have an effective wind speed of 9 m/s if the amount of power it produces from the turbine is the same as for a uniform wind field with point wind speeds of 9 m/s everywhere.

2.4 Wind Turbine Operation and Control

Many industrial processes must be carefully controlled in order to ensure the process is proceeding as required, and within operational limits. Wind energy is a particularly challenging example of this. In order to connect to the grid, a wind farm must satisfy a long list of safety, performance and power quality requirements. On top of this the turbines themselves are huge financial assets exposed to very extreme conditions and loading. It is imperative that they are kept within their operational limits and shut down very quickly if something goes wrong. Apart from control for the sake of protecting the assets themselves, wind turbine operation plays a very important role in ensuring that the wind turbines are operated with maximum efficiency and behave as dictated by their design power curve; high efficiencies are important since the financial viability of this technology is key to its uptake and further development.

2.4.1 Wind Turbine Operating Regions

The design power curve does not tell the full story when it comes to a wind turbine's designed behaviour. In VSPR wind turbine operation there are in fact six distinct regions, determined by the incoming wind speed, all with different desired behaviours. These six regions are as follows:

1. **Below cut-in**. In this region the wind speeds are too low to allow for useful generation of power. The wind turbine is not operated here.

2. **First constant speed region**. In this region the wind speeds are high enough to allow for power generation but only at a rotational speed which is inefficient for the generator due to part loading. In this region the rotational speed is kept constant, as the wind speed increases the turbine's power output increases along with the aerodynamic efficiency which approaches C_{Pmax} for the given constant rotational speed.

3. **Variable speed $\mathbf{C_{P}}_{max}$ tracking region**. In this region the control system's objective is to operate the wind turbine at maximum aerodynamic efficiency, i.e. to keep the power coefficient at C_{Pmax}. This is achieved by varying rotor rotational speed in order to ensure that λ remains at λ_{max} as the wind speed fluctuates. The rotor speed is controlled by setting the generator reaction torque.

4. **Second constant speed region**. Before rated power is reached, while operating at maximum aerodynamic efficiency, a design point is chosen after which rotational speed is kept constant. As the wind speed increases further the generated power approaches rated level, while simultaneously the C_P value falls away from its maximum. The reason for leaving maximum efficiency operation before reaching rated power is mainly due to there being a maximum allowed rotational speed for the generator. From a pitch control viewpoint it is also important to consider that close to $\lambda = \lambda_{max}$ a given change in pitch angle results in a much smaller change in the aerodynamic efficiency as compared to when λ is far away from its optimal value; hence, for power regulation at rated a great deal more pitching

would be required if maximum efficiency had been maintained; this would result in poorer control and the accelerated wearing out of pitch actuators.

5. **Above rated power regulation**. Once rated power is reached, pitch control is used to prevent the power and loads from increasing above the set maximum (rated) level. The rotational speed remains the same as that of the second constant speed region.

6. **Above cut-out**. In this region the wind speeds are too high for the wind turbine to be operated within load design limits, the wind turbine is not operated here. On entering this region the turbine is shut down by fully pitching the blades in order to prevent torque generation and energy extraction.

The equation for the torque generated by a wind turbine's rotor as a function of rotor speed, effective wind speed and pitch angle is;

$$\begin{aligned} Q_{aer}(\beta, \omega_r, v) &= \frac{P}{\omega_r} \\ &= \frac{\frac{1}{2}\rho A v^3 C_P(\lambda, \beta)}{\omega_r} \qquad (2.2) \\ &= \frac{\frac{1}{2}\rho \pi R^3 v^3 C_P(\lambda, \beta)}{\omega_r R} \\ &= \frac{1}{2}\rho \pi R^3 v^2 C_Q(\lambda, \beta). \qquad (2.3) \end{aligned}$$

Where C_Q is the *torque coefficient* which is defined as,

$$C_Q(\lambda, \beta) := \frac{C_P(\lambda, \beta)}{\lambda}. \qquad (2.4)$$

These equations allow for a wind turbine's Torque-Speed diagram to be plotted, as in Figure 2.4[1], and on such a diagram the above operating regions can be clearly depicted. The Torque-Speed diagram relates the wind turbine's aerodynamic torque, Q_{aer}, to its rotational speed for different values of effective wind speed. On this diagram can be plotted the turbine's operating strategy along with other important curves such as the

[1]This Torque-Speed diagram is that of the Supergen Exemplar 5MW wind turbine.

rated power curve, maximum aerodynamic efficiency curve and stall front[2]. Figure 2.5 shows the same diagram with the above operating regions 1-5 highlighted. The filled circle as the control strategy hits the rated power curve is the point at which region 5 is entered and the use of pitch control is initiated.

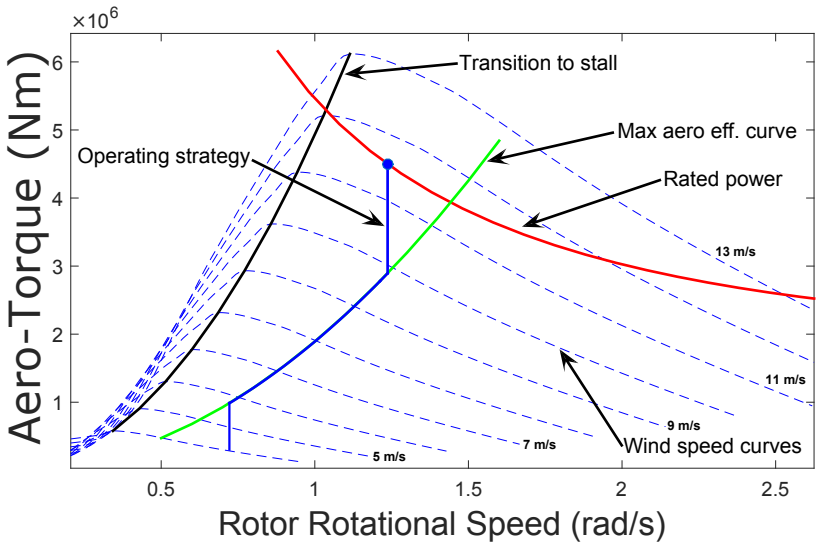

Figure 2.4: Torque-speed diagram for the Supergen Exemplar 5MW wind turbine model. The design operating strategy is also plotted.

2.4.2 Wind Turbine Control

The C_{Pmax} Tracking Region

While C_{Pmax} tracking, the generator rotational speed is measured and the generator reaction torque adjusted in order to keep the turbine operating on the design power curve of Figure 2.3 or equivalently on the maximum efficiency curve of Figure 2.4. Assuming that the wind turbine C_P (or equivalently C_Q) tables are known, the required generator torque during operations can be determined from the turbine's rotational speed as follows: consider the equation for generator torque obtained from the equation

[2]These are the points at which the wind turbine blades will start transitioning to stall and for non-stall regulated turbines are to be avoided for structural health reasons.

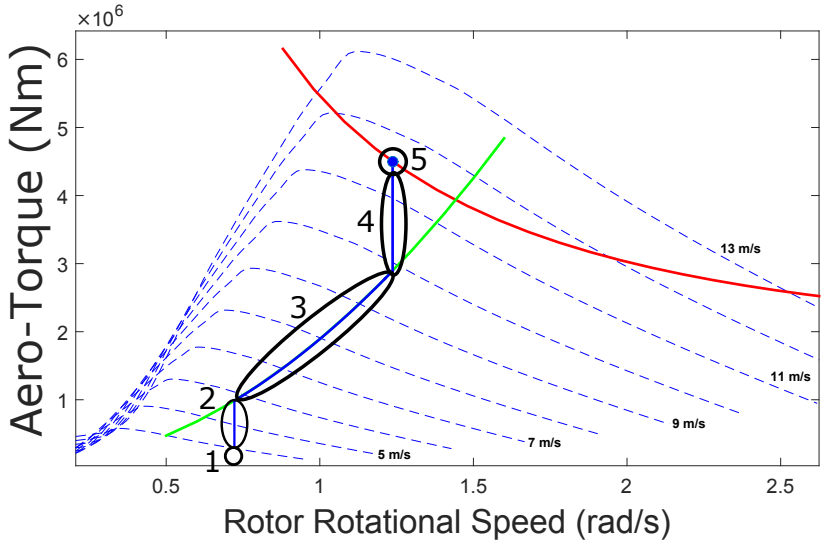

Figure 2.5: Torque-speed diagram of Figure 2.4 with operating regions 1-5 labelled.

for aerodynamic torque (Equation 2.2) while $C_P = C_{Pmax}$,

$$\begin{aligned} Q_{gen} &= \frac{Q_{aer}}{N} \\ &= \frac{\frac{1}{2}\rho A v^3 C_{Pmax}}{N\omega_r} \\ &= \left(\frac{1}{2N}\rho A R^3 \lambda_{max}^{-3} C_{Pmax}\right)\omega_r^2 \\ &= \left(\frac{1}{2N^3}\rho A R^3 \lambda_{max}^{-3} C_{Pmax}\right)\omega_{gen}^2, \end{aligned}$$

with N the turbine gearbox ratio. Hence, during C_{Pmax} operation, the required generator reaction torque is proportional to the generator speed squared,

$$Q_{gen} = a\omega_{gen}^2$$

where the constant of proportionality is,

$$a = \frac{1}{2N^3}\rho A R^3 \lambda_{max}^{-3} C_{Pmax}.$$

The C_{Pmax} tracking control strategy can therefore be implemented with the control loop shown in Figure 2.6. In practice the value of the constant a is adjusted slightly

from that shown here in order to account for the effects of drivetrain losses.

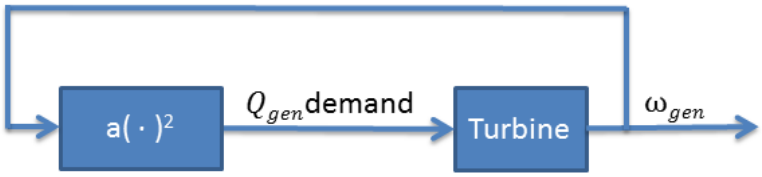

Figure 2.6: Below rated control diagram.

Figures 2.7 and 2.8 show Torque-Speed diagram trajectories while C_{Pmax} tracking. These trajectories were generated using the Supergen Exemplar 5MW wind turbine model. The trajectories are from 16 minutes of operation in wind fields of 5% and 15% turbulence intensities respectively.

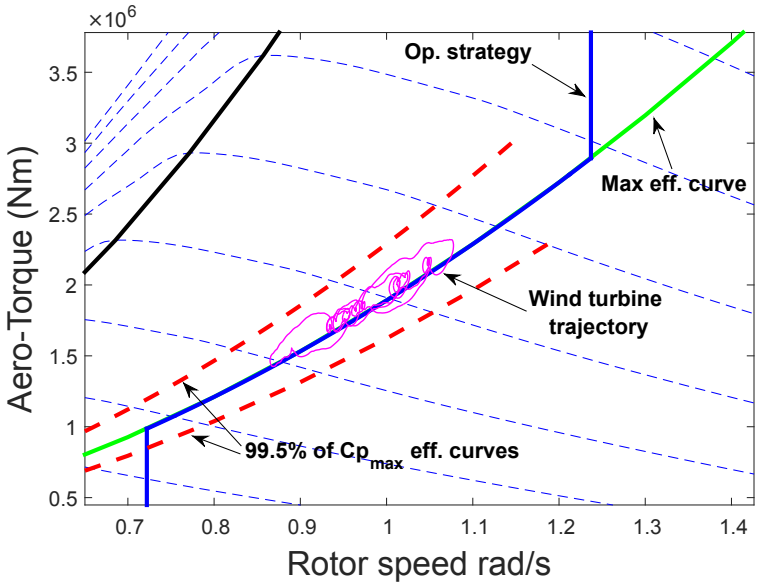

Figure 2.7: C_{Pmax} tracking trajectory on Torque-Speed diagram with 5% turbulence intensity.

It can be seen from these figures that the ability of the controller to maintain operation at C_{Pmax} is dependent on the turbulence intensity of the incident wind field, where turbulence intensity is defined to be the ratio of the wind speed's standard deviation to its mean value. As would be expected intuitively, higher turbulence levels lead to larger excursions from the maximum aerodynamic efficiency curve. Although the trajectories leave that of C_{Pmax} operation, high performance levels are still obtained

as demonstrated by the aerodynamic efficiencies remaining within 99.5% and 98% of C_{Pmax} in Figures 2.7 and 2.8 for turbulence levels of 5% and 15% respectively.

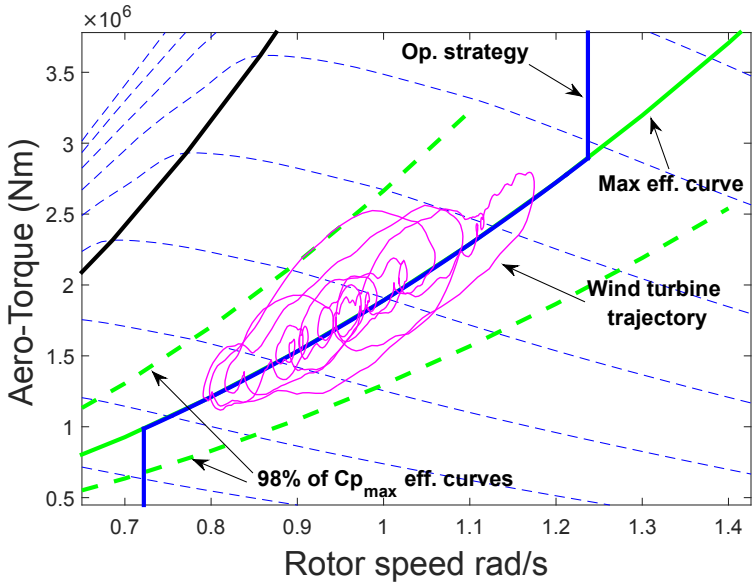

Figure 2.8: C_{Pmax} tracking trajectory on Torque-Speed diagram with 15% turbulence intensity.

Separability of Wind Turbine Aerodynamics

Before discussing above rated control of wind turbines, it is necessary to digress briefly and introduce an important property of wind turbine dynamics, that of aerodynamic separability.

There is no practical way to measure the effective wind speed seen by an operational wind turbine and although there is an anemometer present on the nacelle, this is only a point measurement and so is not going to be able to tell us what is happening across the whole rotor. This is a potential problem for above rated pitch control since the turbine dynamics depend on the effective wind speed as shown in Figure 2.9.
However, as shown in [1], it transpires that the aerodynamic torque term is in fact *separable* in the sense that it can be written as the difference of two functions;

$$Q_{aer}(\beta, \omega_r, v) = h(\beta, \omega_r) - g(v), \tag{2.5}$$

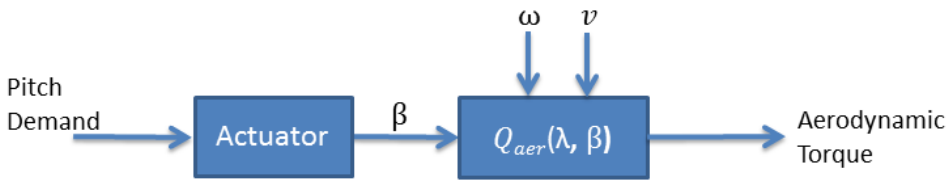

Figure 2.9: Above rated dynamics.

with the first function dependent only on the pitch angle and rotational speed, and the second only dependent on the effective wind speed. Therefore, the above rated dynamics as depicted in Figure 2.9 can be re-expressed as shown in Figure 2.10.

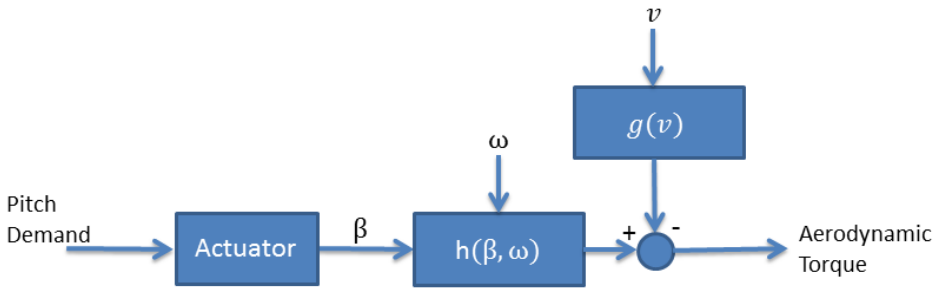

Figure 2.10: Above rated dynamics after applying separability.

The wind speed related function $g(v)$ can be well approximated as a linear function, as shown in [1] and [7]. Characteristic examples of the h and g functions are shown in Figures 2.11 and 2.12 respectively. As will be discussed, separability plays a key role in above rated wind turbine control.

Above Rated Power Regulation

Above rated wind turbine aerodynamics are highly non-linear and hence purely linear control techniques will provide poor performance. The general non-linear control approach taken for wind turbines is gain-scheduling, this technique involves designing linear controllers at some specific points of wind turbine operation and then generating a continuous family of controllers by interpolating between them based on a scheduling variable, usually pitch angle [8]. In practise this is achieved by counteracting changes in

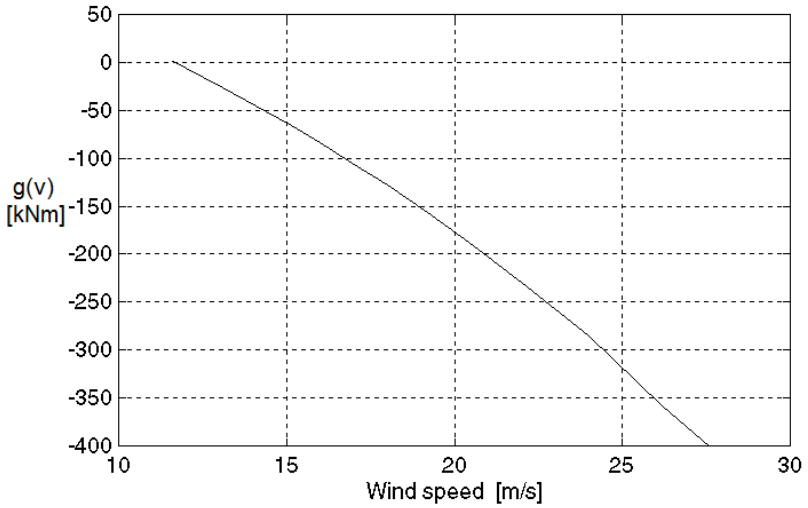

Figure 2.11: Typical example of the g separability function [1].

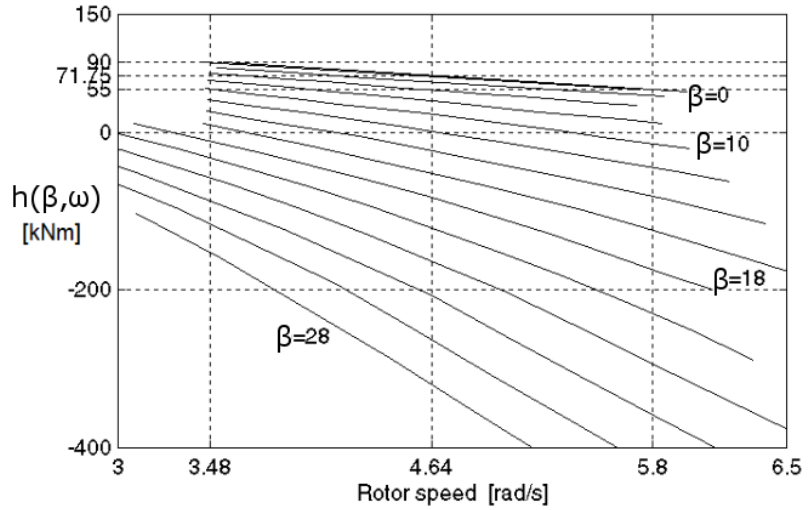

Figure 2.12: Typical examples of the h separability function at various pitch angles, taken from [1].

aerodynamic sensitivity[3], $\frac{\partial Q}{\partial \beta}$, at different operating points by including the reciprocal term, $\left(\frac{\partial Q}{\partial \beta}\right)^{-1}$, in the controller [8]. Thus the total gain of the system becomes constant and linear controller design techniques can be used from this point onwards. Figure 2.13 shows the sensitivity function for the Supergen Exemplar 5MW wind turbine model

[3]This being the main source of non-linearity since the aerodynamic sensitivities change continuously with wind speed, resulting in a continusouly changing gain in the above rated wind turbine dynamics.

along with the line of best fit. In order to ensure smooth interpolation between linear controllers, it is the reciprocal of this line of best fit which is used in the controller to counteract non-linearities.

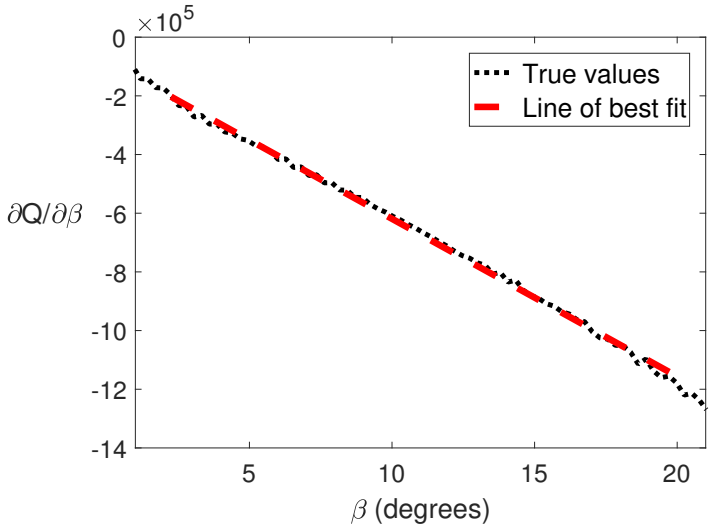

Figure 2.13: Aerodynamic sensitivity function for the Supergen Exemplar 5MW wind turbine model. The straight line best fit is also shown.

Note that implicitly in this gain-scheduling approach we are assuming that the impact of wind speed variations on the control problem is no more than an additive disturbance which can be attenuated with the correct controller design. This will of course be true close to the locus of equilibrium operating points, this being the set of values $\{(\beta, v)\}$ at which we obtain rated power[4], as follows from a Taylor expansion along the locus. However, the size of the region where this approximation holds well enough for this approach to be valid is not, based on series expansion theory alone, guaranteed. Research and testing using the above gain-scheduling approach found that, with the correct control structure, the additive relationship assumption relating wind speed to the overall aerodynamics appeared to hold over much of the turbine's operating envelope, including away from equilibrium points. It was this observation which first motivated the study of wind turbine separability, discussed in the previous section. Rather than just a local theory of separation, it has been shown that wind turbines

[4]It is this set of points along which the controller is attempting to make the turbine traverse during above rated operation, where the control task is that of maintaining rated power.

exhibit a global separability property which holds over almost the whole of a turbine's safe operating range [1, 7]. Separability theory for wind turbines therefore justifies why gain-scheduling on pitch alone is an effective control approach. More importantly, it gives what is effectively a global linearisation of wind turbine dynamics, this being the correct realisation for controller design and implementation [9].

2.4.3 Additional Controller Requirements and Advanced Control Techniques

In addition to ensuring that a wind turbine follows its design operating trajectory, wind turbine controllers are increasingly required to perform other roles relating to the extension of the turbine's lifetime and the flexibility of its operation. Some examples of these requirements are:

- Load reduction [10]
- Damping of system resonances [10, 11]
- Provision of electrical grid frequency support [12, 13]
- Ensuring stable operation during grid disturbances [14].

In order to be able to accomplish the standard controller requirements, as well as these additional ones, large numbers of new and innovative control techniques have been developed, these include:

- The 'Power Adjusting Controller' (PAC) [15]
- 'Coordinated Control' of both pitch and generator torque in above rated conditions [16]
- Individual Pitch Control and Individual Blade Control (IPC and IBC respectively) [17, 18]
- Effective wind speed estimation techniques [19]
- Control for islanded operation during grid disturbances [14].

2.5 The Value of Aerodynamic Information

In all of the control methods discussed so far, including both the standard and advanced control techniques, wind turbine aerodynamic information in the form of C_P (or equivalently C_Q) tables is required. The application of these control techniques therefore requires access to a wind turbine's aerodynamic information. This situation is problematic for two reasons; the first being that the C_P tables used to design controllers are generated using a computational model of the turbine design and so are in fact approximations, since no real-world turbine will identically match its design model. The second issue arises due to the reluctance of manufacturers to share such commercially sensitive information, a problem which becomes even less likely to be resolved when considering the retrofitting of new controllers to existing and potentially out of warranty turbines for the purposes of re-powering or lifetime extension. Furthermore, it has been well documented that a wind turbine's aerodynamic characteristics can change dramatically over its lifetime due to environmental effects. For example, rain and hail impacts can cause significant erosion to the blade leading edge, while ultraviolet light exposure can seriously degrade turbine blade coatings, leading to further possible surface damage [20, 21]. It is therefore the case that a turbine's aerodynamics are changing steadily over its 20+ years of operational lifetime and so even if the design aerodynamic's approximate well those of the real-world turbine initially, this is unlikely to remain so across the turbine's lifetime.

For these reasons it would be highly desirable to be able to learn information about a wind turbine's aerodynamics from its operational data. This would allow for controllers to be implemented using the correct aerodynamic information, and then updated as those dynamics change in time; this would both increase the effectiveness of these control techniques, while also widening their applicability to cases where no aerodynamic information is known. Furthermore, the ability to track changing aerodynamic efficiencies across a wind turbine's lifetime would allow for a deeper understanding of performance degradation which could prove useful in operation and maintenance (O&M) and lifetime extension decision making processes.

To the best of the author's knowledge, no such technique currently exists. This

thesis therefore seeks to develop such a technique by looking to answer the following question:

Can operational data available to a wind turbine controller be used in order to learn turbine C_P (equivalently C_Q) table information to within an accuracy which is useful for control applications?

Since the above question is very expansive, the answer being dependent on the intended control application, we will define a more specific scope for the current thesis which begins to tackle this problem. We therefore seek to determine:

Key Motivating Question: *Can operational data available to a wind turbine controller be used in order to learn a turbine's C_P (equivalently C_Q) tables such that the aerodynamic sensitivity function $\frac{\partial Q}{\partial \beta}$ can be calculated to within an accuracy of 3%?*

An affirmative answer to the above question could lead to improved gain-scheduling for wind turbines, tailored to individual turbines and updated as turbine dynamics change in time. The accuracy level of 3% is experiential and chosen to reflect the accepted level of confidence required in the wind industry for a technique to be considered sound.

Note that while the scope of the above question might appear narrow, it has transpired that in the course of answering it, fundamental questions arise concerning the applied machine learning technique, Gaussian process machine learning. This has led to significant advances being made concerning the applicability and usability of these methods in the course of this work.

2.6 Deriving the Regression Equations

In order to begin considering approaches to answering the question posed at the end of the previous section we start by looking at the data available to a wind turbine controller and how that relates to the information we are hoping to extract, i.e. the turbine aerodynamics. In this section it is shown that the extraction of this desired information can be formed as a regression problem.

2.6.1 Dynamics for Operation at Rated Power and in Second Constant Speed Region

For a wind turbine operating at above rated wind speeds, or in the second constant speed region, the dynamics are approximately;

$$Q_{aer} = N\hat{Q}_g + J\dot{\hat{\omega}}_r + \mathcal{L}(\hat{\omega}_r). \tag{2.6}$$

The losses function $\mathcal{L}$ accounts for drivetrain torque losses; J is rotor inertia, Q_{aer} and Q_g aerodynamic and generator torques respectively, ω_r is generator speed and N the gearbox ratio (the caret symbol is being used to indicate measured values, e.g. $\hat{\omega}_r$ denotes measured rotor speed). We can also express aerodynamic torque in terms of the wind turbine's C_Q values and its operating point;

$$Q_{aer} = Q_{aer}(\beta, \omega_r, v) = \frac{1}{2}\rho A R v^2 C_Q(\lambda, \beta), \tag{2.7}$$

as shown in Equation 2.3. The nacelle mounted anemometer measurement of wind speed, $\hat{V}$, can be thought of as the true effective wind speed corrupted by some form of noise, ζ_v;

$$\hat{V} = v + \zeta_v. \tag{2.8}$$

Given measurements, $\hat{\beta}$ and $\hat{\omega}_r$ of β and ω_r respectively, we can effectively assume these measurements to be non-noisy since any small noise quantities due to measurement errors can be readily filtered out. Applying separability of turbine dynamics it is then possible to re-express Equation 2.7 as follows;

$$\begin{aligned} Q_{aer} &= Q_{aer}(\hat{\beta}, \hat{\omega}_r, v) \\ &= h(\hat{\beta}, \hat{\omega}_r) - g(v) \\ &= h(\hat{\beta}, \hat{\omega}_r) - g(\hat{V}) + [g(\hat{V}) - g(v)] \\ &= Q_{aer}(\hat{\beta}, \hat{\omega}_r, \hat{V}) + [g(\hat{V}) - g(v)] \\ &= \frac{1}{2}\rho A R \hat{V}^2 C_Q(\hat{\lambda}, \hat{\beta}) + [g(\hat{V}) - g(v)] \end{aligned} \tag{2.9}$$

where $\hat{\lambda} = (\hat{\omega}_r R)/\hat{V}$ is the measured value of tip-speed ratio. The above formulation shows that we can express the aerodynamic torque at operating point $(\hat{\beta}, \hat{\omega}_r, v)$ as the aerodynamic torque at operating point $(\hat{\beta}, \hat{\omega}_r, \hat{V})$ corrupted by noise of the form $[g(\hat{V}) - g(v)] = [g(\hat{V}) - g(\hat{V} + \zeta_v)]$. Notice that since the separability function g is only weakly nonlinear it follows, from ζ_v being a noise term, that the above term is scaled noise of the same form.

Rearranging Equation 2.6 and dividing through by $\frac{1}{2}\rho A R^3 \hat{\omega}_r^2$ we obtain;

$$\hat{H} := \frac{N\hat{Q}_g + J\dot{\hat{\omega}}_r}{\frac{1}{2}\rho A R^3 \hat{\omega}_r^2} \tag{2.10}$$

$$= \hat{\lambda}^{-2} C_Q(\hat{\lambda}, \hat{\beta}) + \mathcal{L}^*(\hat{\omega}_g) + \frac{[g(\hat{V}) - g(v)]}{\frac{1}{2}\rho A R^3 \hat{\omega}_r^2} \tag{2.11}$$

where $\mathcal{L}^*(\hat{\omega}_r) = -\mathcal{L}(\hat{\omega}_r)/(\frac{1}{2}\rho A R^3 \hat{\omega}_r^2)$. We thus have a quantity $\hat{H}$ which can be measured using data available to the turbine control system (Equation 2.10) and which can also be expressed as the sum of two functions plus some form of noise[5], denoted η below, scaled by $\hat{\omega}_r^{-2}$ (Equation 2.11); So letting $\Theta_1 = \hat{\lambda}^{-2} C_Q(\hat{\lambda}, \hat{\beta})$, our measured data is of the form,

$$\hat{H} = \Theta_1(\hat{\lambda}, \hat{\beta}) + \mathcal{L}^*(\hat{\omega}_r) + \eta(\hat{\omega}_r^{-1})^2. \tag{2.12}$$

Importantly, if we can identify the functions Θ_1 and $\mathcal{L}^*$, then from these we can directly determine C_Q and $\mathcal{L}$.

2.6.2 Dynamics for Maximum Efficiency Tracking Region

In below rated operation, and when away from the constant speed operating states, the wind turbine control system is attempting to track the maximum power curve of the wind turbine. This corresponds to keeping the tip-speed ratio at λ_{max}, the tip-speed ratio which corresponds to C_{Pmax}, as discussed previously. No matter how sophisticated the control system, there will always be some error when attempting to track C_{Pmax},

[5] This noise being the wind speed noise scaled by the gradient of the (near) linear function g and the constant denominator terms of Equation 2.11.

with the true value of λ varying about λ_{max}. Thus,

$$\lambda = \lambda_{max} + \zeta_\lambda, \tag{2.13}$$

where ζ_λ is the error term. As in the previous section, Q_{aer} is given by equation 2.6. We also have that,

$$\begin{aligned} Q_{aer} &= \frac{1}{2}\rho A R v^2 C_Q(\lambda) \\ &= \frac{1}{2}\rho A R v^2 \frac{C_P(\lambda)}{\lambda}, \end{aligned} \tag{2.14}$$

allowing us to obtain;

$$\hat{G} := \frac{N\hat{Q}_g + J\dot{\hat{\omega}}_r}{\frac{1}{2}\rho A R^3 \hat{\omega}_r^2} \tag{2.15}$$

$$= \lambda^{-3} C_P(\lambda) + \mathcal{L}^*(\hat{\omega}_r) \tag{2.16}$$

$$\approx \lambda_{max}^{-3} C_{Pmax} + \mathcal{L}^*(\hat{\omega}_r) + \delta. \tag{2.17}$$

Where we have assumed ζ_λ is small enough to allow us to approximate the aerodynamic term $\lambda^{-3}C_P(\lambda)$ (via a first order Taylor expansion about $\lambda = \lambda_{max}$) as being the constant $\lambda_{max}^{-3}C_{Pmax}$ plus an error term $\delta = m\zeta_\lambda$ with $m = \frac{d}{d\lambda}\left(\lambda^{-3}C_P(\lambda)\right)\Big|_{\lambda=\lambda_{max}}$. It is further assumed that ζ_λ, and hence δ, is a noise term. In this case we let $\Theta_2 = \lambda_{max}^{-3}C_{Pmax}$ (the constant term from the Taylor expansion), and our measured data is then of the form,

$$\hat{G} = \Theta_2 + \mathcal{L}^*(\hat{\omega}_r) + \delta. \tag{2.18}$$

Identification of Θ_2 and $\mathcal{L}^*$ in this case allows for the losses, $\mathcal{L}$, and the value of C_{Pmax} to be estimated. We use the word estimated here since the below rated formulation contains an explicit approximation in the form of a truncated Taylor expansion.

Drivetrain Losses

The torque losses, $\mathcal{L}(\omega_r)$, present in a wind turbine drivetrain are principally caused by viscous and churning losses, gear friction and bearing friction [22]. These mecha-

nisms result in both a constant loss component and a component which increases with rotational speed. It has been shown in the literature than these losses increase with what is essentially linear behaviour with respect to rotational speed [23]. The drivetrain losses $\mathcal{L}(\omega_r)$ are therefore assumed to be a linear function of rotational speed. Note that with a linear losses assumption, the BR regression equation (Equation 2.18) becomes a quadratic polynomial regression equation in ω_r^{-1}. This is fortuitous since it will allow for possible regression approaches to first be understood in the simpler context of polynomial regression before moving to the general case.

During the constant speed operation of the first regression case, while there will still be some small fluctuations in rotor speed, there will generally not be enough variation for the losses values $\mathcal{L}(\hat{\omega}_r)$ to be determined. Instead the loss function will effectively result in the subtraction of a constant from the aerodynamic term. This leaves two possibilities. The first is for the losses function to be determined in the maximum efficiency operating region, where the rotor speed is moving through its entire range of values. The losses determined here could then be used to adjust results in the constant speed case. The second possibility is that the loss function cannot be accurately determined, resulting in vertically shifted torque predictions. This second possibility, while suboptimal, is by no means intractable. This is due to the fact that we are principally concerned with extracting derivative information, as discussed in the formulation of the key thesis question in Section 2.5, for this application a constant offset does not pose a problem since the derivatives will remain the same.

2.6.3 Data Characteristics and Learning Requirements

The application of regression techniques to the equations derived in the previous sections requires careful consideration of the characteristics of both the required data and the noise present. First, it is important to note the stochastic nature of the wind; wind speeds and turbulence intensities are hugely variable and, therefore, regression data measured at different times will potentially come from operation in very different atmospheric conditions and wind field structures. For the above rated (AR) regression case this will, in the long term, require consideration of the noise term (which is depen-

dent on the difference between anemometer measurements and the true effective wind speed) and whether it should be treated as stationary or not. Thus far a stationary and Gaussian assumption has been applied to the noise for AR regression in order to allow for the developments of the regression algorithm. The stationary part of this assumption is not unreasonable given that wind fields have been shown to be effectively stationary for periods of around 10 minutes [24, 25]. This does however necessitate a batched regression approach to this problem (this being one which can perform regression sequentially, updating predictions as new batches of data become available), allowing for each 10 minutes of collected data to be processed as it becomes available and without having to mix it with other batches that may have different statistical properties. Batching will also be necessary in order to be able to keep updating the predictions for turbine aerodynamics since new data will continue to be generated as the turbine operates over its ≈25 year lifetime.

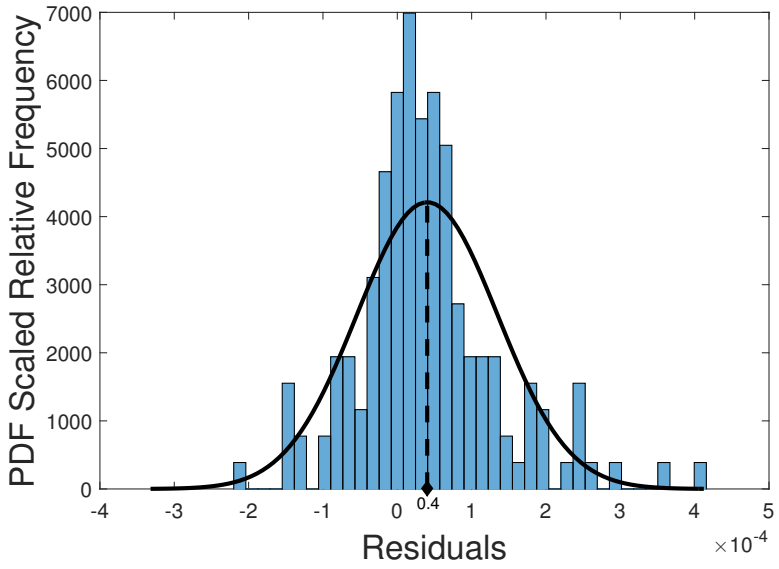

Figure 2.14: Residuals which form the noise term in the BR regression equation. A Gaussian pdf has been fitted to the data.

The measured data generated as above will also require a statistically robust method in order to allow for accurate regression results. This is due to the presence of both outliers and noise distributions which are not strictly Gaussian in nature. As an example

of this, Figure 2.14 shows residual values from simulations of the Supergen Exemplar 5MW turbine model during BR C_{Pmax} tracking. A Gaussian pdf has also been fitted. It is clear in this figure that outliers are present and that while the Gaussian approximation seems reasonable, it is by no means perfect. This is demonstrated by the fact that a Kolmogorov-Smirnov test carried out on this data returns a p value of around 0.005, thus rejecting the hypothesis that this data is Gaussian if a standard 5% cutoff probability is applied.

The final requirement being placed on any developed regression technique is that it must be fast and efficient enough to run within existing controllers, generally Programmable Logic Controllers (PLCs), for wind turbines. This is to allow for the technique to be applied without the need for additional processing power or data storage requirements. This final restriction on possible methods is to ensure maximum economic viability for such a technique; since its uptake would require implementation in each wind turbine across a wind farm, the cost of such a method must be kept to a bare minimum.

2.7 Prior Information

There is much existing information concerning wind turbine aerodynamics which could potentially prove useful in this work. For example, a set of 6 C_P tables for real wind turbines has been made available to this project. Wind turbine manufacturers and wind farm operators will likewise have access to further information. Given that this is all prior information about wind turbine dynamics, ideally we would like to harness it in order to inform or improve developed approaches to dynamics learning from measured data. However, it is not initially clear exactly how this can or should be done. This idea of the correct use of prior information will be returned to later in the thesis.

2.8 Wind Turbine Simulation Model

The Supergen Exemplar 5MW wind turbine model is used throughout this work to produce measured data on which the proposed methods can be tested. The Exemplar

Rated power	5 MW
Rotor diameter	126m
Blade number	3
Hub height	90m
Aerodynamic control	Pitch
Fixed/Variable speed	Variable
Generator type	Fully rated conv.
Gain scheduled	Yes
Controller	PI with low pass filter

Table 2.1: Summary data for Supergen Exemplar 5MW wind turbine model.

turbine was developed as a demonstration model by the Supergen Wind Hub [26] to give a benchmark model on which advanced control concepts can be tested. Full specifications are given in [15]. Table 2.1 provides summary information for this turbine.

2.9 Real Turbine Data

Approximately 150 hours of 20Hz data, collected from an operational wind turbine, has been made available to this project. The turbine is a 3 bladed VSPR wind turbine with a rated power between 1.8 and 2.8MW.

Chapter 3

Gaussian Process Machine Learning

The method selected for use in the wind turbine dynamics identification problem is Gaussian Process machine learning. The reasons for choosing this method are discussed in Section 3.10, after the relevant theory has been presented.

This chapter will provide a general overview of Gaussian process machine learning. Sections 3.1-3.4 introduce Gaussian processes in the most general sense, before showing how they can be used to model a prior over function classes. Regression is shown to follow by conditioning this prior on measured data to generate a posterior GP model. Sections 3.5-3.6 deal with practical aspects of implementing GP regression, including the selection of a prior model via maximum likelihood. This is followed by a regression example. The reasons for choosing this learning method are then discussed, as previously mentioned, in Section 3.10.

3.1 Gaussian Processes

Gaussian processes have been studied in the area of machine learning since the early 1990s, when they were found to be the limit of some infinite neural networks [27]. Their attractiveness comes from the fact that Gaussian processes, although they can be seen as a limiting case of neural networks, do not require the same type of ad hoc decisions

to be made when implementing them. They have since been applied to numerous problems with a lot of success, save the ever present issue of their $\mathcal{O}(n^3)$ complexity which tends to limit the number of measurements used to a few thousand [28].

We start by introducing Gaussian processes in the general sense.

Definition 1. *A **Gaussian process** $\{f_t\}_{t\in\mathbf{T}}$ indexed by a set $\mathbf{T}$ is a family of real-valued random variables, f_t, all defined on the same probability space, with the property that for any finite subset $\mathbf{Z} \subset \mathbf{T}$ every finite linear combination $\sum_{z\in\mathbf{Z}} c_z f_z$, with each $c_z \in \mathbb{R}$, is either identically zero or has a univariate Gaussian distribution on $\mathbb{R}$.*

The above definition was chosen to be entirely unambiguous and generalises some definitions of Gaussian processes since it includes the two possibilities that for any finite subset $\mathbf{Z} \subset \mathbf{T}$, the vector $F_{\mathbf{Z}} = [f_z]_{z\in\mathbf{Z}}$ either has a multivariate Gaussian distribution or the covariance matrix is singular. This more general definition becomes necessary when considering GPs for polynomial regression, as will be discussed later.

A Gaussian process has a mean function,

$$\begin{aligned}\mu &: \mathbf{T} \to \mathbb{R}\\ \mu(t) &= E[f_t],\end{aligned}$$

and covariance function,

$$\begin{aligned}k &: \mathbf{T} \times \mathbf{T} \to \mathbb{R}\\ k(t,t') &= E[(f_t - \mu(t))(f_{t'} - \mu(t'))].\end{aligned}$$

The mean and covariance function completely define all of the distributions of finite subsets of the f_t. Hence, the mean and covariance function together provide a complete description of the GP.

3.2 Gaussian Process Priors over Function Classes

A Gaussian process can be used to model a prior distribution over a class of functions, this follows by letting the indexing set $\mathbf{T}$ be the functional domain and, for all $t \in$

$\mathbf{T}$, interpreting the random variables f_t as representing possible function values for a function evaluated at t. Thus the GP in this context defines a prior over a subset of the functions $\{f \mid f : \mathbf{T} \to \mathbb{R}\}$. Note that $\mathbf{T}$ may have any finite dimension, meaning the modelled functions may be multivariate. The class of functions over which the prior is defined is completely determined by the mean and covariance functions of the GP in the following sense. For any finite subset $\mathbf{Z} \subset \mathbf{T}$, samples can be drawn from the distribution over the function values at these points (determined by the GP prior) as follows. Let $\mathbf{Z} = \{z_1, z_2, \ldots, z_n\}$, then the random vector $\mathbf{F_Z} = [f_{z_1}, f_{z_2}, \ldots, f_{z_n}]^T$ has mean,

$$\boldsymbol{\mu}_\mathbf{Z} = [\mu(z_1), \mu(z_2), \ldots, \mu(z_n)]^T,$$

and covariance matrix $\Lambda_{\mathbf{ZZ}}$ where,

$$[\Lambda_{XY}]_{ij} := k(X_i, Y_j).$$

Letting f_i be a realisation of the random variable f_{z_i} the following expression gives a sample drawn from the distribution of $\mathbf{F_Z}$ as determined by the GP prior,

$$[f_1, f_2, \ldots, f_n]^T = \boldsymbol{\mu}_\mathbf{Z} + \Lambda_{\mathbf{ZZ}}^{1/2}[\nu_1, \nu_2, \ldots, \nu_n]^T,$$

where the ν_i are realisations of independent Gaussian white noise random variables. In fact, for every function in the class the values taken at given input points are of this form for some vector of Gaussian white noise values. The above expression shows explicitly that the class of functions is completely determined by the mean and covariance functions of the GP.

To be clear, there is an important distinction here between a GP and GP prior. A GP is a stochastic process which may or may not be used to model functions or relationships between inputs and outputs. A GP prior on the other hand is a GP which has been specifically chosen or constructed such that it represents our prior beliefs about the object under consideration.

3.3 From Prior to Posterior

Assume we have a prior Gaussian process $\big(\{f_t\}_{t\in\mathbf{T}}, \mu, k\big)$ with which we are modelling our prior beliefs about some function. Further, assume that some new information, $\mathcal{D}$, becomes available. This new information can be used to define the **posterior Gaussian process** $\big(\{f_t\}_{t\in\mathbf{T}}, \mu^*, k^*\big)$ where,

$$\begin{aligned}&\mu^* : \mathbf{T} \to \mathbb{R}\\ &\mu^*(t) = E[f_t \mid \mathcal{D}\,],\end{aligned}$$

and,

$$\begin{aligned}&k^* : \mathbf{T}\times\mathbf{T} \to \mathbb{R}\\ &k^*(t,t') = E[(f_t - \mu^*(t))(f_{t'} - \mu^*(t')) \mid \mathcal{D}\,].\end{aligned}$$

Hence, the posterior GP is simply the prior GP conditioned on the new information $\mathcal{D}$; it can be thought of as a refined model for the function of interest in light of new information.

3.4 Gaussian Process Regression

This section discusses the application of Gaussian Process models to function regression. Assume we have a *zero mean* Gaussian process prior $\{f_t\}_{t\in\mathbf{T}}$ over a class of functions from which a sample function, $\tilde{f}$, has been drawn and at a finite number of points takes values $\mathbf{F} = [\tilde{f}_1, \tilde{f}_2, \ldots, \tilde{f}_n]^T$ corresponding to the members of $\mathbf{Z} = \{z_1, z_2, \ldots, z_n\}$. We also assume that the measurements of $\mathbf{F}$ have been corrupted by zero mean noise, thus, our measured data is $\widehat{\mathbf{F}} = \mathbf{F} + \boldsymbol{\omega}$. Where $\boldsymbol{\omega}$ is drawn from $N(\mathbf{0}, \mathbf{B})$ where $\mathbf{B}$ is the noise covariance matrix.

Consider the random vectors $\mathbf{F_Z} = [f_{z_1}, f_{z_2}, \ldots, f_{z_n}]^T$ (for which $\mathbf{F}$ is a realisation) and $\widehat{\mathbf{F}}_\mathbf{Z} := \mathbf{F_Z} + \boldsymbol{\Omega}$, with $\boldsymbol{\Omega} \sim N(\mathbf{0}, \mathbf{B})$, (for which $\widehat{\mathbf{F}}$ is a realisation) and $\mathbf{F}_{\mathbf{Z}^*}$ corresponding to the finite subset $\mathbf{Z}^* \subset \mathbf{T}$, the points at which we want to predict the

value taken by $\tilde{f}$. The mean and covariance functions in the GP prior determine the following multivariate Gaussian for the combined random vector,

$$\begin{bmatrix} \mathbf{F}_{\mathbf{Z}^*} \\ \mathbf{F}_{\mathbf{Z}} \end{bmatrix} \sim N\left(\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \Lambda_{\mathbf{Z}^*\mathbf{Z}^*} & \Lambda_{\mathbf{Z}^*\mathbf{Z}} \\ \Lambda_{\mathbf{Z}\mathbf{Z}^*} & \Lambda_{\mathbf{Z}\mathbf{Z}} \end{bmatrix} \right).$$

Now consider $\widehat{\mathbf{F}}_{\mathbf{Z}}$. This random vector has zero mean and covariance matrix,

$$E\left[\widehat{\mathbf{F}}_{\mathbf{Z}}\widehat{\mathbf{F}}_{\mathbf{Z}}^T\right] = E[(\mathbf{F}_{\mathbf{Z}} + \mathbf{\Omega})(\mathbf{F}_{\mathbf{Z}}{}^T + \mathbf{\Omega}^T)] = \Lambda_{\mathbf{Z}\mathbf{Z}} + \mathbf{B}.$$

Since $\mathbf{F}_{\mathbf{Z}^*}$ and $\mathbf{\Omega}$ are independent it then follows that,

$$\begin{bmatrix} \mathbf{F}_{\mathbf{Z}^*} \\ \widehat{\mathbf{F}}_{\mathbf{Z}} \end{bmatrix} \sim N\left(\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \Lambda_{\mathbf{Z}^*\mathbf{Z}^*} & \Lambda_{\mathbf{Z}^*\mathbf{Z}} \\ \Lambda_{\mathbf{Z}\mathbf{Z}^*} & \Lambda_{\mathbf{Z}\mathbf{Z}} + \mathbf{B} \end{bmatrix} \right).$$

The Gaussian conditional distribution (see Appendix A) can therefore be applied to determine the distribution of the random vector $\mathbf{F}_{\mathbf{Z}^*}$ conditioned on the noisy measurements;

$$\left(\mathbf{F}_{\mathbf{Z}^*} \mid \widehat{\mathbf{F}}_{\mathbf{Z}} = \widehat{\mathbf{F}}\right) \sim N(\boldsymbol{\mu}_{\mathbf{Z}^*}, \Sigma_{\mathbf{Z}^*})$$

where,

$$\boldsymbol{\mu}_{\mathbf{Z}^*} = \Lambda_{\mathbf{Z}^*\mathbf{Z}} \left(\Lambda_{\mathbf{Z}\mathbf{Z}} + \mathbf{B}\right)^{-1} \widehat{\mathbf{F}} \tag{3.1}$$

$$\Sigma_{\mathbf{Z}^*} = \Lambda_{\mathbf{Z}^*\mathbf{Z}^*} - \Lambda_{\mathbf{Z}^*\mathbf{Z}} \left(\Lambda_{\mathbf{Z}\mathbf{Z}} + \mathbf{B}\right)^{-1} \Lambda_{\mathbf{Z}\mathbf{Z}^*}. \tag{3.2}$$

Importantly, the distribution of the $\mathbf{F}_{\mathbf{Z}^*}$ conditioned on measurements is either Gaussian or has a singular covariance matrix for any choice of $\mathbf{Z}^* \subset \mathbf{T}$. We can therefore define the posterior Gaussian process $\left(\{f_t\}_{t\in\mathbf{T}}, \mu^*, k^*\right)$ by specifying that the mean function μ^* and covariance function k^* are those defined by Equations 3.1 and 3.2 respectively. This posterior GP is a distribution over functions which describes the probability of each function having been the one from which the noisy measurements, $\widehat{\mathbf{F}}$, were observed.

Now suppose we want to be able to predict the values taken by the function $\tilde{f}$ for

arbitrary choices of $z \in T$, given the measurements $\widehat{\mathbf{F}}$. The mean function defined by μ^* at each point is interpreted to be the best fit to the measured data. Two standard deviations, as determined by k^*, are then interpreted to be confidence intervals for the fit (which corresponds to 95% confidence intervals). It is this combination of fitted function and confidence intervals which is referred to as Gaussian process regression.

3.5 Determining a GP Prior

For a practical regression problem the underlying function has in fact not been drawn from a GP prior and so there is *a priori* no prior GP to work with. Nevertheless, in order to apply the formulations of the previous section we assume that the function did in fact arise from a GP prior and a suitable prior is determined using the measured data (as explained below); regression then follows as outlined previously.

When no information is initially known about the function the prior is assumed to have zero mean and a class of covariance functions (consisting of a covariance function model containing hyperparameters) is specified. The covariance between functional outputs is modelled as a function of the input values and the most suitable covariance function is chosen from the class by estimating the *correct* hyperparameter values from measured data. A common choice for this function class which puts a prior over smooth functions is the squared exponential covariance function [29];

$$k(z, z') = a \exp\left(-\frac{1}{2}(z - z')^T \Psi (z - z')\right), \tag{3.3}$$

where Ψ is a diagonal matrix whose values, along with a, are hyperparameters which need to be determined (as will be discussed). The parameter a determines the amplitude of the covariance structure and the diagonal values of Ψ give inverse-lengthscale values for each input dimension. This function implies a high covariance for function outputs corresponding to close inputs vectors, the covariance then decays exponentially as the inputs move further away from each other in each dimension of the input space. In this case the hyperparameters specify both the amplitude of the functional outputs as well as the length-scales over which functional outputs will remain correlated with

respect to each input dimension. Other types of covariance function models can be found in both [30] and [28]. In this thesis we will focus solely on the squared exponential (SE) covariance function, shown above, when considering non-linear regression, this is due to the fact that the SE is one of the most general covariance structures since it places a prior over all smooth functions [28]. Other covariance functions tend to have more specific structure present, for example the rational-quadratic covariance function corresponds to a sum over many squared exponential covariances with varying lengthscales [28] and the periodic covariance function corresponds to a prior over periodic functions [30]. The cases which we will considering here have not been found to require such additional structure and so other covariance functions have not been required. The noise covariance matrix, $\mathbf{B}$, is also defined by its own set of hyperparameters whose values must be determined. If the noise is assumed to be iid Gaussian then $\mathbf{B} = \xi I$ and the noise variance ξ is the only hyperparameter required to describe the noise covariance matrix.

3.6 Hyperparameter Determination

One of the advantages of working with Gaussian distributions is that the marginal distributions can be calculated easily and analytically. One such marginal is the so called log-likelihood function, this is essentially the probability of observing the measured data given a certain set of hyperparameter values. Assume that $\widehat{K} := \Lambda_{\mathbf{ZZ}} + \mathbf{B}$ is determined by some set of hyperparameter values $\underline{\mathbf{a}} = \{a_i\}$. The log-likelihood value for some set of hyperparameters, $\underline{\mathbf{a}}$, given the observed (noisy) data is then;

$$log(p(\widehat{\mathbf{F}}|\underline{\mathbf{a}})) = -\frac{1}{2}\widehat{\mathbf{F}}^T\widehat{K}^{-1}\widehat{\mathbf{F}} - \frac{1}{2}\log|\widehat{K}| - \frac{n}{2}\log 2\pi$$

Treating the above expression as a function of the hyperparameters $\underline{\mathbf{a}}$, the correct (or most likely) hyperparameter values are assumed to be those which maximise the above expression, or equivalently those which minimise the negative of the above expression. Therefore, for a given set of measurements we can perform the above optimisation and determine the most likely hyperparameter values, this completely determines the

covariance matrix $\widehat{K}$ and the prior covariance function k (hence also the GP prior). The hyperparameters for the covariance functions considered here are always non-negative and so rather than performing constrained optimisation over the a_i, we instead define $a_i = e^{\phi_i}$ and perform unconstrained optimisation over the ϕ_i instead. The ϕ_i will be referred to as *log-hyperparameters*. During the optimisation procedure we need to be wary of possible convergence to an inappropriate minimum, this and other possible issues are discussed fully in the work of [29].

Accurate and fast gradient based optimisation requires the derivatives and the Hessian matrix for the log-likelihood function with respect to log-hyperparameter values. Let $\{\phi_i\}_{i=1\ldots k}$ be our set of log-hyperparameter values. If L is the log-likelihood function defined previously, we have the following first and second derivatives with respect to log-hyperparameters [29];

$$\frac{\partial L}{\partial \phi_i} = -\frac{1}{2} trace\left[\widehat{K}^{-1}\frac{\partial \widehat{K}}{\partial \phi_i}\right] + \frac{1}{2}\widehat{\mathbf{F}}^T\widehat{K}^{-1}\frac{\partial \widehat{K}}{\partial \phi_i}\widehat{K}^{-1}\widehat{\mathbf{F}} \tag{3.4}$$

$$\begin{aligned}\frac{\partial^2 L}{\partial \phi_i \partial \phi_j} &= -\frac{1}{2} trace\left[\widehat{K}^{-1}\frac{\partial^2 \widehat{K}}{\partial \phi_i \partial \phi_j} - \widehat{K}^{-1}\frac{\partial \widehat{K}}{\partial \phi_i}\widehat{K}^{-1}\frac{\partial \widehat{K}}{\partial \phi_j}\right] \\ &+ \frac{1}{2}\widehat{\mathbf{F}}^T\widehat{K}^{-1}\left[\frac{\partial^2 \widehat{K}}{\partial \phi_i \partial \phi_j} - 2\frac{\partial \widehat{K}}{\partial \phi_i}\widehat{K}^{-1}\frac{\partial \widehat{K}}{\partial \phi_j}\right]\widehat{K}^{-1}\widehat{\mathbf{F}}.\end{aligned}$$

The quality of the hyperparameter determination optimisation is important when it comes to making accurate GP predictions; this is illustrated by the following analysis.

Assume that we are performing Gaussian process regression on noisy measurements of a function, $\tilde{f}$, drawn from a GP prior which is determined by hyperparameter values $\underline{\mathbf{a}}$. As discussed above, GP regression requires these hyperparameter values to be estimated from the measured data. The prediction is then formed using these determined hyperparameter values, $\underline{\mathbf{a}}^\dagger$, which may differ from the true values, $\underline{\mathbf{a}}$, for this class of functions. Let $\dagger$ denote terms whose hyperparameters have been determined from data; then the posterior mean function will be $\Lambda^\dagger_{z\mathbf{Z}}(\widehat{K}^\dagger)^{-1}\widehat{\mathbf{F}}$ (note that over the ensemble of

functions this is still an unbiased prediction) rather than $\Lambda_{z\mathbf{Z}}\widehat{K}^{-1}\widehat{\mathbf{F}}$ with covariance function,

$$
\begin{aligned}
& E[(\Lambda^{\dagger}_{z\mathbf{Z}}(\widehat{K}^{\dagger})^{-1}(\mathbf{F_Z}+\mathbf{\Omega})-f_z)(\Lambda^{\dagger}_{z'\mathbf{Z}}(\widehat{K}^{\dagger})^{-1}(\mathbf{F_Z}+\mathbf{\Omega})-f_{z'})] \\
= \quad & \Lambda^{\dagger}_{z\mathbf{Z}}(\widehat{K}^{\dagger})^{-1}\widehat{K}(\widehat{K}^{\dagger})^{-1}\Lambda^{\dagger}_{\mathbf{Z}z'}-\Lambda^{\dagger}_{z\mathbf{Z}}(\widehat{K}^{\dagger})^{-1}\Lambda_{\mathbf{Z}z'}-\Lambda_{z\mathbf{Z}}(\widehat{K}^{\dagger})^{-1}\Lambda^{\dagger}_{\mathbf{Z}z'}+\Lambda_{zz'} \\
= \quad & \Lambda_{zz'}-\Lambda_{z\mathbf{Z}}\widehat{K}^{-1}\Lambda_{\mathbf{Z}z'}+[\Lambda^{\dagger}_{z\mathbf{Z}}(\widehat{K}^{\dagger})^{-1}-\Lambda_{z\mathbf{Z}}\widehat{K}^{-1}]\widehat{K}[\Lambda^{\dagger}_{z'\mathbf{Z}}(\widehat{K}^{\dagger})^{-1}-\Lambda_{z'\mathbf{Z}}\widehat{K}^{-1}]^T \\
\geq \quad & \Lambda_{zz'}-\Lambda_{z\mathbf{Z}}\widehat{K}^{-1}\Lambda_{\mathbf{Z}z'} \quad \textit{when } z=z'.
\end{aligned}
$$

Thus the error variance has increased in the case where the hyperparameters differ from their true values. Note that these statistics are still over the ensemble of functions. This analysis implies that when implementing GP a careful optimisation procedure should be used in order to minimise the differences between $\underline{\mathbf{a}}$ and $\underline{\mathbf{a}}^{\dagger}$.

3.7 Gaussian Process Regression Example

The GP regression process, outlined in the previous section, was applied to noisy measurements of the nonlinear function $f(z)=\sin(z)+\frac{z}{2}$. A squared exponential covariance function was used for this regression example and the results can be seen in Figure 3.1 along with the noisy measured data.

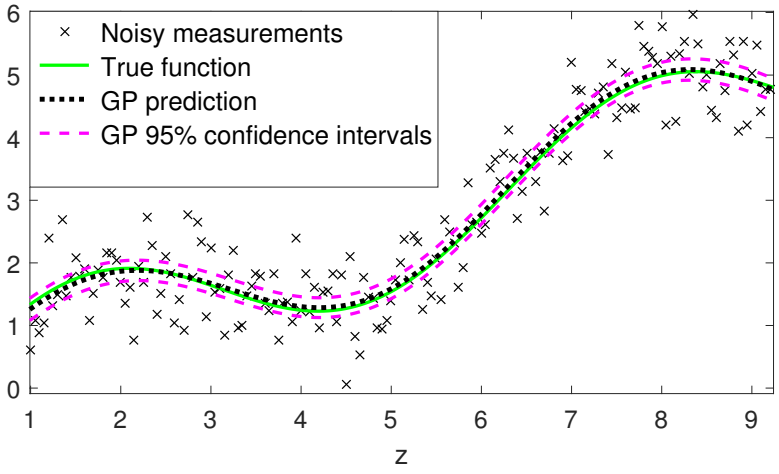

Figure 3.1: GP regression example.

3.8 Efficiency of General Gaussian Process Algorithms

GP machine learning algorithms are generally considered unsuitable for applications where the number of measured data points, n, is over a few thousand. This is due to the need to invert and find the determinant of the $n \times n$ covariance matrix (multiple times during hyperparameter determination), both of which are operations which scale as $\mathcal{O}(n^3)$, as well as the need to multiply large matrices together which can be almost as costly [28].

3.9 Transformations Applied During GP Modelling

Applying a GP model implicitly assumes that the function of interest has been drawn from the specified prior distribution. In practice, a given function might not adhere to the idealised properties of the prior, stationarity being one example, and this can degrade the performance of GP regression. Various techniques have therefore been developed which allow for a transformation of either the input values or measured data such that in the transformed coordinates the underlying assumptions are better realised. These ideas were initially explored in the context of geostatistics, for example in [31] the input space is transformed via a nonlinear function within a Bayesian framework. A Gaussian process prior is placed on the transform itself, and inference follows via a Markov chain Monte Carlo (MCMC) algorithm. Thus, a parametric form for the transform is not given, instead its posterior distribution is sampled as part of the wider learning procedure. In [32] a class of covariance functions is developed which contain in-built transformations of the input variables. These are obtained by defining mappings which are integrals over sets of positive basis functions (thus ensuring the transforms are one-to-one). The form of these basis functions is chosen by the user and contain additional hyperparameters to be learned as part of the maximum likelihood optimisation procedure. Figure 3.2 shows the application of this approach to a function which is non-stationary since it has a lengthscale which varies across the input domain, the plot is taken [32]. In [33] similar transformations are applied, but this time to the measurements themselves, in order to allow for GPs to be applied where

noise characteristics are prohibitively non-Gaussian. In this case monotonic 'warping' functions are chosen and their hyperparameters determined via optimisation as before. Because transforms are applied to the output space, in order to obtain predictions it is necessary to invert the warping functions for both mean values and any relevant percentiles (e.g. the 95% confidence intervals). This requires a numerical solutions such as Gauss-Hermite quadrature [33].

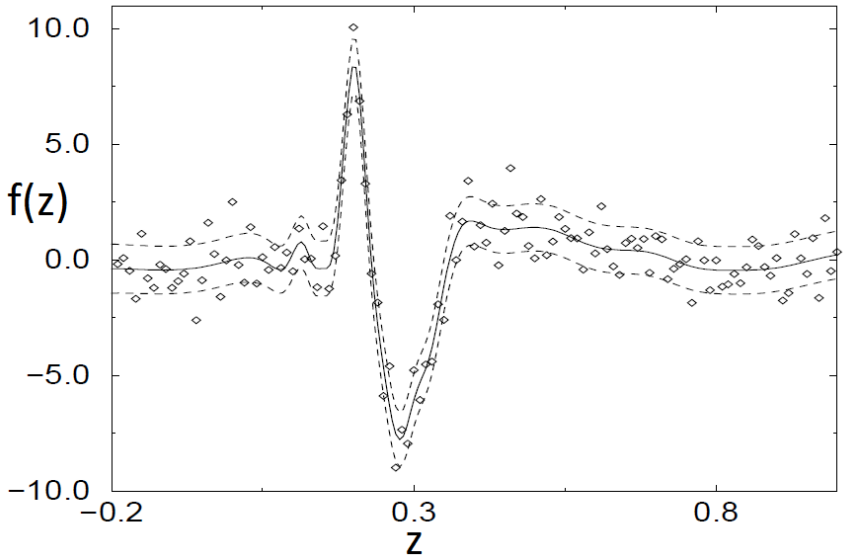

Figure 3.2: GP regression example with non-stationarity handled via an input transformation. The diamonds are measured data, the solid line is the GP mean and the dashed lines are the standard deviations about the predictions.

3.10 Why Gaussian Processes?

Other regression techniques exist which might be applied to the same problem considered in this thesis. It is therefore worth discussing why GPs have been selected over possible alternatives. The reason essentially comes down to the flexibility and robustness of the Gaussian process regression technique. In Section 2.6.3 we discussed the need to be able to handle outliers, develop a fast and generally applicable regression technique and it has also been seen that we also want to extract derivative information from our predictions. Gaussian processes have in the past been applied to wind

turbine data in order to investigate the separability functions h and g [29]. This has shown that they can handle the relevant types of data well, without the noise structure or outliers degrading their predictive abilities. Furthermore, in terms of developing a method which can be applied across a wind farm easily and simply we want to avoid as much as possible ad hoc decision making forming a part of implementation.

Of the other possible techniques, some of the most attractive alternative candidates would be Neural Networks (NNs) [34] and Random Forest (RF) [35]. NNs require the selection of numbers of so called 'hidden layers', a process which adds to the complexity of decision making when building the learning algorithms, and increases the computational burden of the procedure. Considering RF, whereas GP predictions are smooth (allowing for derivatives to be directly calculated) RF predictions are non-differentiable, and hence a smoothing function would need to be applied before derivatives are taken. This again adds complexity to the models in the form of needing to set appropriate parameters for such smoothing functions, a task which is not necessarily straightforward or easy to automate. Taking this all together, along with the fact that GPs have previously been shown to perform well for this type of problem [29], it was decided that GPs were the best candidate learning procedure on which to build these algorithms, with the results of this work forming a performance baseline against which other techniques can be compared in the future.

3.11 Other Gaussian Process Applications in Wind Energy

Although the problem under consideration in this thesis has not been studied previously, GPs have been used fairly extensively in wind energy for other applications. For example, in [36] a combination of numerical weather predictions and GP models are used for wind power forecasting, in [37] GPs are used to monitor the structural health of wind turbine assets; and, in [38] GPs are used to estimate extreme gusts. This wide range of applications again highlights the flexibility and general usefulness of GPs as a machine learning technique.

Chapter 4

Batched Approaches to Gaussian Processes

The wind turbine dynamics identification problem considered in this work requires a regression method which can be applied to batched data, as discussed in Section 2.6.3. To clarify, the term 'batched regression' is used here to mean a regression procedure which allows for the updating of predictions when new data becomes available, and hence, one which processes data in batches[1]. This chapter will therefore begin by reviewing the work done to date on Gaussian process regression formulations which can be applied to batched data. The existing techniques all falling under the general category of *sparse GP approximations*.

These various approaches are then discussed, in Section 4.5, where gaps in the current knowledge base for these techniques are identified, motivated by the requirements of the turbine dynamics identification problems. These gaps are shown to motivate the exploration of new possible techniques for batched data GP regression, based on the GP regression models themselves, rather than on purely probabilistic techniques. Two such approaches are presented in Section 4.6 to be explored in greater detail.

[1] In the current work the processing of data in batches is necessary due to the fact that more data becomes available as a wind turbine operates. Furthermore, as discussed in Section 2.6.3, turbine data can only reasonably be assumed to be stationary for roughly ten minutes.

4.1 Sparse GP Approximations

The problem of being able to apply GP regression to batched data has been previously explored in the literature, with existing methods all falling under the general category of sparse GP approximations. These various methods are presented in this current section. This is a class of methods whereby the GP posterior is approximated using the function values predicted at a set of points, often referred to as *pseudo* or *inducing* points, with order much less than the number of measurements being processed [39]. Hence, whereas standard GP regression has the predictive distribution,

$$p(f_z|\mathbf{Y_D}) = \int p(f_z|\mathbf{F_D})p(\mathbf{F_D}|\mathbf{Y_D})d\mathbf{F_D}, \tag{4.1}$$

obtained by marginalising over the posterior of $\mathbf{F_D}$, a sparse GP model will seek to approximate the above integral by way of a small set of m function values, denoted here by $\mathbf{F_m}$. Writing the predictive distribution as,

$$p(f_z|\mathbf{Y_D}) = \int p(f_z|\mathbf{F_m},\mathbf{F_D})p(\mathbf{F_D}|\mathbf{F_m},\mathbf{Y_D})p(\mathbf{F_m}|\mathbf{Y_D})d\mathbf{F_D}d\mathbf{F_m}, \tag{4.2}$$

and assuming the conditional independence of f_z and $\mathbf{F_D}$ given $\mathbf{F_m}$, approximating $p(f_z|\mathbf{Y_D})$ then comes down to determining a model for $\chi(\mathbf{F_m}) := p(\mathbf{F_m}|\mathbf{Y_D})$ [40]. In the existing techniques χ is chosen to be Gaussian and hence will be of the form,

$$\chi(\mathbf{F_m}) = \mathcal{N}(\boldsymbol{\mu}_\chi, A_\chi). \tag{4.3}$$

Once χ is known, the mean and covariance functions in the posterior for arbitrary z and z' are given by,

$$m(z) = \Lambda_{z\mathbf{m}}\Lambda_{\mathbf{mm}}^{-1}\boldsymbol{\mu}_\chi \tag{4.4}$$

$$k(z,z') = \Lambda_{zz'} - \Lambda_{z\mathbf{m}}\Lambda_{\mathbf{mm}}^{-1}\Lambda_{\mathbf{m}z'} + \Lambda_{z\mathbf{m}}\Lambda_{\mathbf{mm}}^{-1}A_\chi\Lambda_{\mathbf{mm}}^{-1}\Lambda_{\mathbf{m}z'} \tag{4.5}$$

These techniques have been applied both 'online', in a batched data context, and 'offline' where all the data is available *a priori*. A good overview of the offline case is given

in [39]. In the case of batched data $\chi(\mathbf{F_m})$ must be updated at each batching stage and in current methods this can involve changing both the number of inducing points and their locations, as well as the model's parameters.

Having chosen a model for χ, its parameters are trained in order to optimise some objective function which attempts to ensure a close fit between the approximate and true predictive distributions. Below we outline the two most popular approaches by which a model χ and its associated objective function can be developed.

4.2 Fully Independent Training Conditional (FITC)

This approach has seen several different formulations, here we follow that of [41]. Given a GP model over $\mathbf{F_m}$, it is assumed that the single point likelihood for noisy measurements is [41],

$$p(y_z|\mathbf{m}, \mathbf{F_m}) = \mathcal{N}(\Lambda_{z\mathbf{m}}\Lambda_{\mathbf{mm}}^{-1}\mathbf{F_m}, \Lambda_{zz} - \Lambda_{z\mathbf{m}}\Lambda_{\mathbf{mm}}^{-1}\Lambda_{\mathbf{m}z} + \xi), \tag{4.6}$$

with noise variance ξ. Measured data is then assumed to be generated IID, giving the complete data likelihood for measured dataset $\mathbf{D}$,

$$p(\mathbf{Y_D}|\mathbf{m}, \mathbf{F_m}) = \mathcal{N}(\Lambda_{\mathbf{Dm}}\Lambda_{\mathbf{mm}}^{-1}\mathbf{F_m}, K + \xi I), \tag{4.7}$$

where K is diagonal with $[K]_{ii} = \Lambda_{\mathbf{D}_i\mathbf{D}_i} - \Lambda_{\mathbf{D}_i\mathbf{m}}\Lambda_{\mathbf{mm}}^{-1}\Lambda_{\mathbf{mD}_i}$. Placing a Gaussian prior,

$$p(\mathbf{F_m}) = \mathcal{N}(\mathbf{0}, \Lambda_{\mathbf{mm}}), \tag{4.8}$$

over the function values at the inducing points then allows for the model, χ, to be determined via Baye's rule as,

$$\chi(\mathbf{F_m}) = \mathcal{N}(\Lambda_{\mathbf{mm}}R_{\mathbf{m}}^{-1}\Lambda_{\mathbf{mD}}(K + \xi I)^{-1}\mathbf{Y_D}, \Lambda_{\mathbf{mm}}R_{\mathbf{m}}^{-1}\Lambda_{\mathbf{mm}}), \tag{4.9}$$

with $R_{\mathbf{m}} = \Lambda_{\mathbf{mm}} + \Lambda_{\mathbf{mD}}(K + \xi I)^{-1}\Lambda_{\mathbf{Dm}}$. The parameters of the terms in χ are determined by maximising the marginal likelihood,

$$p(\mathbf{Y_D}|\mathbf{m}) = \int p(\mathbf{Y_D}|\mathbf{m}, \mathbf{F_m})p(\mathbf{F_m})d\mathbf{F_m}, \tag{4.10}$$

which is the objective function in this case. Note that inversion of $K + \xi I$ scales linearly by construction, since it is diagonal, and it is for this reason that the FITC was constructed as above.

The first formulation of a form of FITC [42] was in fact in the context of streaming data and hence is relevant to the batched setting. The set of inducing points, $\mathbf{F_m}$, is chosen to be a subset of the measured data. Scores are assigned to each new datapoint as it is processed and, once a chosen maximum subset size has been reached, whenever a new datapoint is processed the inducing point with the lowest score is deleted.

4.3 Variational Free Energy Approach (VFE)

Variational learning has also been used to generate sparse GP approximations [40]. In this case the true augmented posterior in Equation 4.2,

$$p(\mathbf{F_D}, \mathbf{F_m}|\mathbf{Y_D}) = p(\mathbf{F_D}|\mathbf{F_m}, \mathbf{Y_D})p(\mathbf{F_m}|\mathbf{Y_D}), \tag{4.11}$$

is approximated by the variational posterior,

$$q(\mathbf{F_D}, \mathbf{F_m}) := p(\mathbf{F_D}|\mathbf{F_m})\chi(\mathbf{F_m}). \tag{4.12}$$

A model χ and inducing points are then sought such that a measure of distance between the above true and modelled posteriors is minimised. This measure of distance is chosen to be the Kullback-Leibler divergence, $D_{KL}(q(\mathbf{F_D}, \mathbf{F_m})||p(\mathbf{F_D}, \mathbf{F_m}|\mathbf{Y_D}))$, which forms a premetric on the space of probability distributions and can be interpreted as a measure of information gain between statistical models. Minimising the KL divergence here has been shown to be equivalent to minimising an approximate lower bound for the true negative log-marginal likelihood, the so called 'variational free energy' [40].

Once hyperparameter values have been determined from minimisation of this objective function, it follows that the optimal model for χ is,

$$\chi(\mathbf{F_m}) = \mathcal{N}(\xi^{-1}\Lambda_{\mathbf{mm}}S_{\mathbf{m}}^{-1}\Lambda_{\mathbf{mD}}\mathbf{Y_D}, \Lambda_{\mathbf{mm}}S_{\mathbf{m}}^{-1}\Lambda_{\mathbf{mm}}), \tag{4.13}$$

with $S_{\mathbf{m}} = \left[\Lambda_{\mathbf{mm}} + \xi^{-1}\Lambda_{\mathbf{mD}}\Lambda_{\mathbf{Dm}}\right]$.

In [43] the VFE framework is extended to handle batched data. In order to avoid having to store old data, the following approach is taken to propagating information between GP approximations: Given the current approximation to the GP based on old data,

$$q_{\text{old}}(f) \approx p(f|\mathbf{Y}_{\mathbf{D}_{\text{old}}}) \propto p(f|\theta_{\text{old}})p(\mathbf{Y}_{\mathbf{D}_{\text{old}}}|f), \tag{4.14}$$

(where θ denotes hyperparameters) we wish to be able to form an updated approximation using new data,

$$q_{\text{new}}(f) \approx p(f|\mathbf{Y}_{\mathbf{D}_{\text{old}}}, \mathbf{Y}_{\mathbf{D}_{\text{new}}}) \propto p(f|\theta_{\text{new}})p(\mathbf{Y}_{\mathbf{D}_{\text{old}}}|f)p(\mathbf{Y}_{\mathbf{D}_{\text{new}}}|f). \tag{4.15}$$

Since old data is assumed to be unavailable, the $p(\mathbf{Y}_{\mathbf{D}_{\text{old}}}|f)$ term above is approximated by inverting the relationship in Equation 4.14. The VFE formulation then follows similarly to in [40], resulting in a slightly augmented objective function over which the new hyperparameters and inducing point locations can be optimised. From Equations 42 and 45 in [43] it follows that in this case the optimal model, χ, has mean,

$$\mu_\chi = \left(\Lambda_{\mathbf{mm}}^{-1} + \Lambda_{\mathbf{mm}}^{-1}[\Lambda_{\mathbf{mD}}\ \Lambda_{\mathbf{mm}_0}]\Sigma^{-1}\begin{bmatrix}\Lambda_{\mathbf{Dm}}\\ \Lambda_{\mathbf{m}_0\mathbf{m}}\end{bmatrix}\Lambda_{\mathbf{mm}}^{-1}\right)^{-1}\Lambda_{\mathbf{mm}}^{-1}[\Lambda_{\mathbf{mD}}\ \Lambda_{\mathbf{mm}_0}]\Sigma^{-1}\begin{bmatrix}\mathbf{Y}_{\mathbf{D}_{\text{new}}}\\ (\tilde{A}_\chi^{-1} - \tilde{\Lambda}_{\mathbf{m}_0\mathbf{m}_0}^{-1})^{-1}\tilde{A}_\chi^{-1}\tilde{\mu}_\chi\end{bmatrix},$$

and covariance matrix,

$$A_\chi = \left(\Lambda_{\mathbf{mm}}^{-1} + \Lambda_{\mathbf{mm}}^{-1}\begin{bmatrix}\Lambda_{\mathbf{mD}} & \Lambda_{\mathbf{mm}_0}\end{bmatrix}\Sigma^{-1}\begin{bmatrix}\Lambda_{\mathbf{Dm}}\\ \Lambda_{\mathbf{m}_0\mathbf{m}}\end{bmatrix}\Lambda_{\mathbf{mm}}^{-1}\right)^{-1}.$$

With,

$$\Sigma = \begin{bmatrix}\xi I & 0\\ 0 & (\tilde{A}_\chi^{-1} - \tilde{\Lambda}_{\mathbf{m}_0\mathbf{m}_0}^{-1})^{-1}\end{bmatrix},$$

and where $\tilde{A}_\chi$, $\tilde{\mu}_\chi$ and $\tilde{\Lambda}^{-1}_{\mathbf{m}_0\mathbf{m}_0}$ are calculated using the old model and hyperparameters, and $\mathbf{m}_0$ is the set of inducing points from the old model.

The VFE technique can then be further extended by replacing KL-divergence with the more general α-divergence [44]. This results in the same predictive equations as for VFE, except Σ becomes [43],

$$\Sigma_\alpha = \Sigma + \begin{bmatrix} \alpha \text{diag}(\Lambda_{\mathbf{DD}} - \Lambda_{\mathbf{Dm}}\Lambda^{-1}_{\mathbf{mm}}\Lambda_{\mathbf{mD}}) & 0 \\ 0 & \alpha(\Lambda_{\mathbf{m}_0\mathbf{m}_0} - \Lambda_{\mathbf{m}_0\mathbf{m}}\Lambda^{-1}_{\mathbf{mm}}\Lambda_{\mathbf{mm}_0}) \end{bmatrix}. \quad (4.16)$$

This more general formulation encompasses both FITC [45] (When $\alpha = 1$) and VFE [40, 42, 43] (when $\alpha = 0$), along with a family of other methods depending on the value of α. The possibility of unifying these various techniques under a single framework was first demonstrated in [46].

4.4 Recursive Gaussian Processes

Another approach to GP regression on batched data is Recursive GPs (RGP), developed in [47] using Kalman Filtering. While the underlying approach is somewhat different from those discussed above, fundamentally the same game is being played whereby a model for $\chi(\mathbf{F}_\mathbf{m})$ is developed. Prediction at other points then follows using the same predictive formulas as before. For known hyperparameters the model χ in Recursive GP (using the current notation) has mean,

$$\mu_\chi = \tilde{\mu}_\chi + \tilde{A}_\chi\Lambda^{-1}_{\mathbf{mm}}\Lambda_{\mathbf{mD}}\left(\Lambda_{\mathbf{DD}} - \Lambda_{\mathbf{Dm}}\Lambda^{-1}_{\mathbf{mm}}\Lambda_{\mathbf{mD}} + \Lambda_{\mathbf{Dm}}\Lambda^{-1}_{\mathbf{mm}}\tilde{A}_\chi\Lambda^{-1}_{\mathbf{mm}}\Lambda_{\mathbf{mD}} + \xi I\right)^{-1}\left(\mathbf{Y}_\mathbf{D} - \Lambda_{\mathbf{Dm}}\Lambda^{-1}_{\mathbf{mm}}\tilde{\mu}_\chi\right),$$

and covariance matrix,

$$A_\chi = \tilde{A}_\chi - \tilde{A}_\chi\Lambda^{-1}_{\mathbf{mm}}\Lambda_{\mathbf{mD}}\left(\Lambda_{\mathbf{DD}} - \Lambda_{\mathbf{Dm}}\Lambda^{-1}_{\mathbf{mm}}\Lambda_{\mathbf{mD}} + \Lambda_{\mathbf{Dm}}\Lambda^{-1}_{\mathbf{mm}}\tilde{A}_\chi\Lambda^{-1}_{\mathbf{mm}}\Lambda_{\mathbf{mD}} + \xi I\right)^{-1}\Lambda_{\mathbf{Dm}}\Lambda^{-1}_{\mathbf{mm}}\tilde{A}_\chi.$$

As before, the expressions $\tilde{\mu}_\chi$ and $\tilde{A}_\chi$ denote the mean and covariance terms with respect to the prior at each batching stage. Note that in the above expressions the term,

$$\tilde{A}_\chi\Lambda^{-1}_{\mathbf{mm}}\Lambda_{\mathbf{mD}}\left(\Lambda_{\mathbf{DD}} - \Lambda_{\mathbf{Dm}}\Lambda^{-1}_{\mathbf{mm}}\Lambda_{\mathbf{mD}} + \Lambda_{\mathbf{Dm}}\Lambda^{-1}_{\mathbf{mm}}\tilde{A}_\chi\Lambda^{-1}_{\mathbf{mm}}\Lambda_{\mathbf{mD}} + \xi I\right)^{-1},$$

is the Kalman gain.

4.5 Discussion of Existing Approaches

Three general frameworks for using GPs on batched data have been presented, these being FITC, VFE and RGP. While each of these techniques represents an important contribution to GP theory, there are still some significant gaps worth exploring.

The first of these gaps comes from the observation that these existing techniques have been developed through the manipulation of probability distributions, rather than from the underlying GP framework. In each case, the prior GP is only used in so far as to define a prior model for χ. From this point onwards these methods do not rely on the underlying relationship coming from a GP. Furthermore, for the batched regression cases the relationship between the existing methods and standard GPs has not been shown, with all comparisons having come from numerical experiments. Therefore, a pertinent question would seem to be,

Q1. *Does the underlying structure of a GP model suggest any particular approach to performing regression on batched data, and what relationship would such a method have with standard GP regression on all of the data at once?*

Secondly, there is currently no theoretical basis, or rules of thumb, for choosing the number and positions of inducing points for a GP model. In FITC and VFE they are determined either as being a subset of the measured data, or their number and placement are optimised using their objective functions. In [48] comparisons are made between FITC and VFE which do give some general results about the behavior of the two methods with respect to numbers of inducing points and their locations, but these results are focused on implementation rather than theory and all stem from numerical experiments on specific examples. RGP [47] on the other hand does not discuss the selection of these points at all, with both their number and location assumed fixed throughout. While all three methods have been shown to be able to perform GP regression with good results, there still seems to be something lacking in the underlying theoretical development when inducing point selection relies on optimisation trial and

error and random re-starts [48]. Computer implementations of these techniques will almost certainly require such measures in practise, however, it seems that there should also be some theoretical basis for choosing starting points for any optimisation. Such theory should also link back to the fact that we are considering GP models; for example, one would like to be able to answer questions such as,

Q2. *Given a GP prior, but as yet no measured data, does the GP itself indicate any information about the number and location of inducing points necessary to accurately model this GP?*

This point has general importance, but it is also directly relevant to the case of wind turbine dynamics identification in terms of the optimal and correct use of prior information. As discussed in Section 2.7, extensive prior information exists about wind turbine aerodynamics and a principled probabilistic technique should take full advantage of this by somehow incorporating it into the predictive models. A theory which provides an answer to Q2 might well also indicate how this can be done.

Thirdly, all existing techniques for batched GPs explicitly assume that the inducing-points' covariance matrix, $\Lambda_{\mathbf{mm}}$, is non-singular. Indeed, the above formulas for ϕ would be ill-defined if this were not the case. However, adding additional points to the inducing-point set is effectively adding more information into the sparse GP representation; intuitively, this should never result in degrading the ability of the sparse representation to approximate the underlying GP, and if anything should improve it. Therefore, it would seem that singular $\Lambda_{\mathbf{mm}}$ matrices should be accounted for within a sparse approach to batched GPs. In the current methods the case of singular $\Lambda_{\mathbf{mm}}$ matrices is not considered and, if the number of inducing points causes $\Lambda_{\mathbf{mm}}$ to become ill-conditioned, it is common practice to add a 'jitter' term along the diagonal to allow inversion to still be carried out [48]. Again, this seems unsatisfactory from a theoretical viewpoint.

Finally, the existing methods all require for computationally expensive optimisations to be carried out during the regression procedure in order to determine hyper-parameter values and/or inducing point numbers and locations. This is done in VFE and FITC using gradient based optimisation in large numbers of dimensions when both

hyperparamaters and inducing-points are determined. RGP updates hyperparameter values at each stage using a sigma point method. Given the processing power available to a wind turbine controller, these optimisation procedures will not be feasible for application in the problem considered in this thesis. This is another reason why a deeper understanding of the theory underlying possible techniques appears to be required, to determine,

Q3. *Can such techniques be applied without the need for demanding optimisation and, if so, when might this be possible?*

A part of answering this last point almost certainly calls again for a principled approach to incorporating prior information into the models.

4.6 Iterative Gaussian Processes

In order to explore the above ideas the specific problem we will be considering is now formalised. The two most intuitive and natural approaches to Gaussian processes for batched data, based on the GP models, are discussed. Investigations in the following chapters then consider the viability of methods related to these two approaches.

We consider the simple case of two iterates, with larger numbers obtainable from multiple applications of any given approach. The term 'GP regression' here is referring to the process of determining a posterior GP model for the function under consideration. We assume we are performing GP regression on a dataset, $\mathcal{D}$, which is the disjoint union of datasets $\mathcal{D}_1$ and $\mathcal{D}_2$, and contains noisy measurements of the function f. Further, assume that we can perform GP regression on $\mathcal{D}_1$ and $\mathcal{D}_2$ individually, but not on $\mathcal{D}$ itself; this could, for example, be due to the availability of the datasets in time, differences in noise levels for measurements in the two subsets or considerations of computational efficiency. In this context we are considering the question:

How can individual GP regression on each of $\mathcal{D}_1$ and $\mathcal{D}_2$ be used to determine a single GP model for the function, f, which takes all of the data into account?

There appear to be two overarching models for how this might be done. The first of these will be referred to as the *combined-posteriors* approach; as the name suggests,

this involves performing GP regression on $\mathcal{D}_1$ and $\mathcal{D}_2$ individually in order to obtain posteriors for both, and then somehow combining these posterior GP models. The second will be referred to as the *posterior-to-prior* approach and involves performing GP regression on one dataset, $\mathcal{D}_1$ say, in order to obtain a GP posterior model which is then used as the prior for GP regression on $\mathcal{D}_2$; the posterior obtained from this second regression will be our desired GP model which takes both $\mathcal{D}_1$ and $\mathcal{D}_2$ into account. These two approaches will be formalised and investigated in the following sections.

In doing this we are looking to develop an approach to what will be referred to as *iterative Gaussian process regression*, which we define as:

Definition 2. For a dataset $\mathcal{D} = \sqcup_{i=1}^{N} \mathcal{D}_i$, **iterative Gaussian process regression** on $\mathcal{D}$ refers to a method which sequentially generates N Gaussian Process models, $\{\mathcal{GP}_i\}_{i=1}^{N}$, where $\mathcal{GP}_j$ takes into account the data $\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_j\}$.

4.6.1 Combined Posteriors

Let $\mathbf{T}$ be the domain of f and assume we have performed GP regression on datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ obtaining the posterior GPs:

$$\mathcal{GP}_1 := (\{f_t\}_{t \in \mathbf{T}}, \mu_1, k_1),$$

and,

$$\mathcal{GP}_2 := (\{f_t\}_{t \in \mathbf{T}}, \mu_2, k_2),$$

respectively. We seek to combine these two GP models into a single refined GP model,

$$\mathcal{GP}_+ := (\{f_t\}_{t \in \mathbf{T}}, \mu_+, k_+),$$

for the function f. We must therefore define μ_+ and k_+ for the combined GP model or equivalently define the multivariate distribution over any finite subset of the f_t. Let $\mathbf{Z}^* \subset \mathbf{T}$ be such a finite subset. Ignoring singular cases for now and letting p_i denote the pdf for $\mathcal{GP}_i$ corresponding to the distribution P_i, and p the pdf of the GP prior we

have:

$$p_1(\mathbf{F}_{\mathbf{Z}^*}) = p(\mathbf{F}_{\mathbf{Z}^*} \mid \mathcal{D}_1) \text{ and } P_1(\mathbf{F}_{\mathbf{Z}^*}) = N(\boldsymbol{\mu}_1, \Lambda_1)$$

$$p_2(\mathbf{F}_{\mathbf{Z}^*}) = p(\mathbf{F}_{\mathbf{Z}^*} \mid \mathcal{D}_2) \text{ and } P_2(\mathbf{F}_{\mathbf{Z}^*}) = N(\boldsymbol{\mu}_2, \Lambda_2)$$

with $\boldsymbol{\mu}_i$ and Λ_i determined by μ_i and k_i respectively. Following the standard Bayesian approach, and assuming the independence of $\mathcal{D}_1$ and $\mathcal{D}_2$ conditionally on f we have;

$$\begin{aligned} p_+(\mathbf{F}_{\mathbf{Z}^*}) := \; p(\mathbf{F}_{\mathbf{Z}^*} \mid \mathcal{D}_1, \mathcal{D}_2) \quad &\propto \quad p(\mathbf{F}_{\mathbf{Z}^*} \mid \mathcal{D}_1) p(\mathbf{F}_{\mathbf{Z}^*} \mid \mathcal{D}_2) \\ &\propto \quad p_1(\mathbf{F}_{\mathbf{Z}^*}) p_2(\mathbf{F}_{\mathbf{Z}^*}) \end{aligned}$$

and so,

$$P_+(\mathbf{F}_{\mathbf{Z}^*}) = N(\boldsymbol{\mu}_+, \Lambda_+)$$

with,

$$\begin{aligned} \boldsymbol{\mu}_+ &:= (\Lambda_1^{-1} + \Lambda_2^{-1})^{-1}(\Lambda_1^{-1}\boldsymbol{\mu}_1 + \Lambda_2^{-1}\boldsymbol{\mu}_2) \\ &= \Lambda_2(\Lambda_1 + \Lambda_2)^{-1}\boldsymbol{\mu}_1 + \Lambda_1(\Lambda_1 + \Lambda_2)^{-1}\boldsymbol{\mu}_2 \qquad (4.17) \\ \Lambda_+ &:= (\Lambda_1^{-1} + \Lambda_2^{-1})^{-1} \\ &= \Lambda_1(\Lambda_1 + \Lambda_2)^{-1}\Lambda_2. \qquad (4.18) \end{aligned}$$

$P_+(\mathbf{F}_{\mathbf{Z}^*})$ can be seen to be a covariance weighted average of $P_1(\mathbf{F}_{\mathbf{Z}^*})$ and $P_2(\mathbf{F}_{\mathbf{Z}^*})$. These formulations were first proposed in [49] under the title of 'Bayesian Committee Machines', see also [28]. Since $\mathbf{Z}^*$ was chosen arbitrarily the above formulations can be used to define a multivariate Gaussian distribution for all finite subsets of the f_t; it is therefore logical to ask whether we can define the functions μ_+ and k_+ to be those which arise from the P_+ distributions. Unfortunately the answer is no, since, although the above formulations do indeed define multivariate Gaussian distributions for any finite subset in $\mathbf{T}$, these distributions do not marginalise consistently. More explicitly the means and covariances in the combined GP change depending on how many and which points are in $\mathbf{Z}^*$. Hence, attempting to define μ_+ and k_+ in this way would render them ill-defined. Note the fact that this approach does not result in a well-defined GP

is an important point which passes unobserved in [49].

Thus, this approach is unsuitable for generating a combined GP model for the function f; and while there may be other possible approaches to *combined-posteriors* iterative GPs, the one presented here is certainly the most straightforward and natural and it seems likely that any similar approach will suffer from the same issues of ill-definedness.

4.6.2 A Pragmatic Approach - Combining Predictive Statistics

We have seen in the previous section that a straightforward Bayesian approach to *combined-posteriors* iterative GPs does not work if we are looking to define our combined GP. This will therefore not be a suitable method by which to develop iterative Gaussian Processes.

However, from a practical viewpoint it is interesting to consider where such an approach will lead. Also, one could argue that often when applying GP models we are not really interested in the whole GP model; instead we tend to use the GP to make predictions at specific input points and determine confidence intervals around these predictions. For many cases of functional regression it would therefore be sufficient to pre-determine a set of input points (distributed finely enough such that values taken at interior points can be interpolated with negligible error) at which we would like to make our predictions and determine confidence intervals. This being the case we can let $\mathbf{Z}^*$ be our now fixed set of chosen input points; then the distribution $P_+(\mathbf{F}_{\mathbf{Z}^*})$ can be used to define predicted function values and confidence intervals at each of our chosen points. Since we are now only dealing with finite Gaussian distributions for fixed function output points there are no longer issues of ill-definedness. It is important to stress that determining function predictions and confidence intervals this way means we are combining the predictive statistics rather than the GP models themselves. However, this theoretically results in an iterative method[2] of updating the predicted function values and confidence intervals while taking into account all available data. This method of combining the predictive statistics may therefore be enough for many purposes and

[2]It should be clear that the formula for $P_+(\mathbf{F}_{\mathbf{Z}^*})$ can be used to keep updating iteratively the combined distribution as new data becomes available.

so this case will be investigated further. This technique will be henceforth referred to as the *Combined-Statistics* approach.

4.6.3 Posterior-to-Prior Approach

This section considers the other most obvious approach to iterative GP regression. We again consider our two datasets $\mathcal{D}_1$ and $\mathcal{D}_2$. Performing GP regression on $\mathcal{D}_1$ yields the posterior GP, $\mathcal{GP}_1 := (\{f_t\}_{t\in\mathbf{T}}, \mu_1, k_1)$. This current approach to iterative GPs will combine the information from both datasets by using the posterior GP from $\mathcal{D}_1$ as a prior for GP regression on $\mathcal{D}_2$.

Let $\mathcal{D}_2 = \{(z_i, y_i)\}_{i=1}^n$, where the z_i are inputs and y_i the corresponding measured outputs (which are corrupted by noise). We denote by $\mathbf{Z}$ and $\mathbf{Y}$ the vectors of these quantities respectively. The formulations of Section 3.4 assume a prior of zero everywhere, our current prior will generally be non-zero and so in order to avoid having to re-write the GP regression process for a non-zero prior mean we simply observe that the data $\widetilde{\mathcal{D}}_2 := \{(z_i, y_i - \mu_1(z_i))\}_{i=1}^n$ is now zero mean and with the same covariance structure. We therefore work with this adjusted data, which will be denoted $\widetilde{\mathbf{Y}}$, and finally add the mean value at each predicted point back on in the posterior to recover the GP predictions which include the non-zero prior. For a set of prediction points $\mathbf{Z}^*$ we then have the following multivariate Gaussian and posterior equations for the mean and covariance matrix:

$$\begin{bmatrix} \mathbf{F}_{\mathbf{Z}^*} \\ \widetilde{\mathbf{Y}} \end{bmatrix} \sim N\left(\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \Lambda^1_{\mathbf{Z}^*\mathbf{Z}^*} & \Lambda^1_{\mathbf{Z}^*\mathbf{Z}} \\ \Lambda^1_{\mathbf{Z}\mathbf{Z}^*} & \Lambda^1_{\mathbf{Z}\mathbf{Z}} + \Omega \end{bmatrix} \right), \tag{4.19}$$

$$\boldsymbol{\mu}^{1\&2}_{\mathbf{Z}^*} = \boldsymbol{\mu}^1_{\mathbf{Z}^*} + \Lambda^1_{\mathbf{Z}^*\mathbf{Z}} \left(\Lambda^1_{\mathbf{Z}\mathbf{Z}} + \Omega\right)^{-1} \widetilde{\mathbf{Y}} \tag{4.20}$$

$$\Lambda^{1\&2}_{\mathbf{Z}^*\mathbf{Z}^*} = \Lambda^1_{\mathbf{Z}^*\mathbf{Z}^*} - \Lambda^1_{\mathbf{Z}^*\mathbf{Z}} \left(\Lambda^1_{\mathbf{Z}\mathbf{Z}} + \Omega\right)^{-1} \Lambda^1_{\mathbf{Z}\mathbf{Z}^*}, \tag{4.21}$$

where the $\boldsymbol{\mu}^1{}_{(\cdot)}$ and $\Lambda^1_{(\cdot)(\cdot)}$ terms are given by the functions μ_1 and k_1 of $\mathcal{GP}_1$ respectively. In general, the covariance structure of the noise will not be the same for the two datasets and so the noise term Ω will have to be determined via maximum like-

lihood as before. However, this will require training of a much smaller number of hyperparameters and so this process should be comparatively faster than training hyperparameters to determine both the noise and the covariance structure of the data. In the common case that the noise is assumed to be iid zero mean and Gaussian, the noise hyperparameter training will only require the training of a single parameter, the noise variance ξ. Thus, we can use the above equations to define the combined GP model $\mathcal{GP}_{1\&2} := (\{f_t\}_{t\in\mathbf{T}}, \mu^{1\&2}, k^{1\&2})$ which has been formed from both $\mathcal{D}_1$ and $\mathcal{D}_2$. This process can then be continued as new datasets become available; for example if we now obtain the dataset $\mathcal{D}_3$ then we can form the combined GP model $\mathcal{GP}_{1\&2\&3}$ via GP regression on $\mathcal{D}_3$ with a prior determined by $\mathcal{GP}_{1\&2}$ and so on.

For general Gaussian Process regression it can be shown that posterior-to-prior iteration is in fact equivalent to GP regression on the entire dataset, more formally:

Theorem 1. *Assume we have a data set $\mathcal{D} = \mathcal{D}_1 \sqcup \mathcal{D}_2 \sqcup \ldots \sqcup \mathcal{D}_n$ and a prior GP, $\mathcal{GP}_{pr} = (\{f_t\}_{t\in\mathbf{T}}, \mu^0, k^0)$. The GP obtained by performing standard GP regression on the whole dataset $\mathcal{D}$, with prior $\mathcal{GP}_{pr}$, is identical to that obtained by performing posterior-to-prior iteration on each of the datasets $\mathcal{D}_i$ sequentially, starting from an initial prior of $\mathcal{GP}_{pr}$.*

Proof. We prove this for the case $n = 2$. The general case follows from this inductively. Terms relating to standard GP regression on the whole of $\mathcal{D} = \mathcal{D}_1 \sqcup \mathcal{D}_2$ are denoted by the superscript $1 + 2$, whereas those relating to posterior-to-prior iterative regression on $\mathcal{D}_1$ and then $\mathcal{D}_2$ are denoted by the superscript 1&2 as above. Terms relating to regression on $\mathcal{D}_1$ only are denoted by the superscript 1. Let $\widetilde{\mathbf{Y}}$ denote measured data adjusted by the relevant prior mean and Q denote covariance matrices with the noise term added.

Following the standard GP regression formulas, the mean at given input points $\mathbf{Z}^*$ from regression on $\mathcal{D}$ is,

$$\boldsymbol{\mu}^{1+2}_{\mathbf{Z}^*} = \boldsymbol{\mu}^0_{\mathbf{Z}^*} + \begin{bmatrix}\Lambda^0_{\mathbf{Z}^*\mathbf{Z}_1} & \Lambda^0_{\mathbf{Z}^*\mathbf{Z}_2}\end{bmatrix} \begin{bmatrix} Q_{\mathbf{Z}_1\mathbf{Z}_1} & \Lambda^0_{\mathbf{Z}_1\mathbf{Z}_2} \\ \Lambda^0_{\mathbf{Z}_2\mathbf{Z}_1} & Q_{\mathbf{Z}_2\mathbf{Z}_2}\end{bmatrix}^{-1} \begin{bmatrix}\mathbf{Y}_{\mathbf{Z}_1} - \boldsymbol{\mu}^0_{\mathbf{Z}_1} \\ \mathbf{Y}_{\mathbf{Z}_2} - \boldsymbol{\mu}^0_{\mathbf{Z}_2}\end{bmatrix}.$$

Applying the partitioned matrix inverse formula, and letting $X = \left(Q_{\mathbf{Z}_2\mathbf{Z}_2} - \Lambda^0_{\mathbf{Z}_2\mathbf{Z}_1} Q^{-1}_{\mathbf{Z}_1\mathbf{Z}_1} \Lambda^0_{\mathbf{Z}_1\mathbf{Z}_2}\right)^{-1}$, this mean can be re-expressed as follows:

$$\begin{aligned}
\boldsymbol{\mu}^{1+2}_{\mathbf{Z}^*} &= \boldsymbol{\mu}^0_{\mathbf{Z}^*} + \begin{bmatrix}\Lambda^0_{\mathbf{Z}^*\mathbf{Z}_1} & \Lambda^0_{\mathbf{Z}^*\mathbf{Z}_2}\end{bmatrix} \begin{bmatrix} Q_{\mathbf{Z}_1\mathbf{Z}_1} & \Lambda^0_{\mathbf{Z}_1\mathbf{Z}_2} \\ \Lambda^0_{\mathbf{Z}_2\mathbf{Z}_1} & Q_{\mathbf{Z}_2\mathbf{Z}_2}\end{bmatrix}^{-1} \begin{bmatrix}\widetilde{\mathbf{Y}}_{\mathbf{Z}_1} \\ \widetilde{\mathbf{Y}}_{\mathbf{Z}_2}\end{bmatrix} \\
&= \boldsymbol{\mu}^0_{\mathbf{Z}^*} + \begin{bmatrix}\Lambda^0_{\mathbf{Z}^*\mathbf{Z}_1} & \Lambda^0_{\mathbf{Z}^*\mathbf{Z}_2}\end{bmatrix} \begin{bmatrix} Q^{-1}_{\mathbf{Z}_1\mathbf{Z}_1} + Q^{-1}_{\mathbf{Z}_1\mathbf{Z}_1}\Lambda^0_{\mathbf{Z}_1\mathbf{Z}_2} X \Lambda^0_{\mathbf{Z}_2\mathbf{Z}_1} Q^{-1}_{\mathbf{Z}_1\mathbf{Z}_1} & -Q^{-1}_{\mathbf{Z}_1\mathbf{Z}_1}\Lambda^0_{\mathbf{Z}_1\mathbf{Z}_2} X \\ -X\Lambda^0_{\mathbf{Z}_2\mathbf{Z}_1} Q^{-1}_{\mathbf{Z}_1\mathbf{Z}_1} & X\end{bmatrix} \begin{bmatrix}\widetilde{\mathbf{Y}}_{\mathbf{Z}_1} \\ \widetilde{\mathbf{Y}}_{\mathbf{Z}_2}\end{bmatrix} \\
&= \boldsymbol{\mu}^1_{\mathbf{Z}^*} + \left[\Lambda^0_{\mathbf{Z}^*\mathbf{Z}_2} - \Lambda^0_{\mathbf{Z}^*\mathbf{Z}_1} Q^{-1}_{\mathbf{Z}_1\mathbf{Z}_1} \Lambda^0_{\mathbf{Z}_1\mathbf{Z}_2}\right] X \left[\widetilde{\mathbf{Y}}_{\mathbf{Z}_2} - \left(\boldsymbol{\mu}^0_{\mathbf{Z}_2} + \Lambda^0_{\mathbf{Z}_2\mathbf{Z}_1} Q^{-1}_{\mathbf{Z}_1\mathbf{Z}_1} \widetilde{\mathbf{Y}}_{\mathbf{Z}_1}\right)\right] \\
&= \boldsymbol{\mu}^1_{\mathbf{Z}^*} + \Lambda^1_{\mathbf{Z}^*\mathbf{Z}_2} \left[\Lambda^1_{\mathbf{Z}_2\mathbf{Z}_2} + \left(Q_{\mathbf{Z}_2\mathbf{Z}_2} - \Lambda^0_{\mathbf{Z}_2\mathbf{Z}_2}\right)\right]^{-1} \left[\widetilde{\mathbf{Y}}_{\mathbf{Z}_2} - \boldsymbol{\mu}^1_{\mathbf{Z}_2}\right] \\
&= \boldsymbol{\mu}^{1\&2}_{\mathbf{Z}^*}.
\end{aligned}$$

Similarly, for the covariance matrices in the posterior,

$$\begin{aligned}
\Lambda^{1+2}_{\mathbf{Z}^*\mathbf{Z}^*} &= \Lambda^0_{\mathbf{Z}^*\mathbf{Z}^*} - \begin{bmatrix}\Lambda^0_{\mathbf{Z}^*\mathbf{Z}_1} & \Lambda^0_{\mathbf{Z}^*\mathbf{Z}_2}\end{bmatrix} \begin{bmatrix} Q_{\mathbf{Z}_1\mathbf{Z}_1} & \Lambda^0_{\mathbf{Z}_1\mathbf{Z}_2} \\ \Lambda^0_{\mathbf{Z}_2\mathbf{Z}_1} & Q_{\mathbf{Z}_2\mathbf{Z}_2}\end{bmatrix}^{-1} \begin{bmatrix}\Lambda^0_{\mathbf{Z}_1\mathbf{Z}^*} \\ \Lambda^0_{\mathbf{Z}_2\mathbf{Z}^*}\end{bmatrix} \\
&= \Lambda^0_{\mathbf{Z}^*\mathbf{Z}^*} - \begin{bmatrix}\Lambda^0_{\mathbf{Z}^*\mathbf{Z}_1} & \Lambda^0_{\mathbf{Z}^*\mathbf{Z}_2}\end{bmatrix} \begin{bmatrix} Q^{-1}_{\mathbf{Z}_1\mathbf{Z}_1} + Q^{-1}_{\mathbf{Z}_1\mathbf{Z}_1}\Lambda^0_{\mathbf{Z}_1\mathbf{Z}_2} X \Lambda^0_{\mathbf{Z}_2\mathbf{Z}_1} Q^{-1}_{\mathbf{Z}_1\mathbf{Z}_1} & -Q^{-1}_{\mathbf{Z}_1\mathbf{Z}_1}\Lambda^0_{\mathbf{Z}_1\mathbf{Z}_2} X \\ -X\Lambda^0_{\mathbf{Z}_2\mathbf{Z}_1} Q^{-1}_{\mathbf{Z}_1\mathbf{Z}_1} & X\end{bmatrix} \begin{bmatrix}\Lambda^0_{\mathbf{Z}_1\mathbf{Z}^*} \\ \Lambda^0_{\mathbf{Z}_2\mathbf{Z}^*}\end{bmatrix} \\
&= \Lambda^1_{\mathbf{Z}^*\mathbf{Z}^*} - \left[\Lambda^0_{\mathbf{Z}^*\mathbf{Z}_2} - \Lambda^0_{\mathbf{Z}^*\mathbf{Z}_1} Q^{-1}_{\mathbf{Z}_1\mathbf{Z}_1} \Lambda^0_{\mathbf{Z}_1\mathbf{Z}_2}\right] X \left[\Lambda^0_{\mathbf{Z}_2\mathbf{Z}^*} - \Lambda^0_{\mathbf{Z}_2\mathbf{Z}_1} Q^{-1}_{\mathbf{Z}_1\mathbf{Z}_1} \Lambda^0_{\mathbf{Z}_1\mathbf{Z}^*}\right] \\
&= \Lambda^1_{\mathbf{Z}^*\mathbf{Z}^*} - \Lambda^1_{\mathbf{Z}^*\mathbf{Z}_2} \left[\Lambda^1_{\mathbf{Z}_2\mathbf{Z}_2} + \left(Q_{\mathbf{Z}_2\mathbf{Z}_2} - \Lambda^0_{\mathbf{Z}_2\mathbf{Z}_2}\right)\right]^{-1} \Lambda^1_{\mathbf{Z}_2\mathbf{Z}^*} \\
&= \Lambda^{1\&2}_{\mathbf{Z}^*\mathbf{Z}^*}.
\end{aligned}$$

Since posterior GPs are entirely determined by their mean and covariance functions it follows that the two GPs, $\mathcal{GP}_{1+2}$ and $\mathcal{GP}_{1\&2}$, are equivalent. □

This approach therefore seems very attractive since the above theorem shows that it is an iterative way of processing batches of data into an overall GP model in a way which is equivalent to performing GP regression on all of the data at once. This demonstrates that the posterior-to-prior approach does consistently propagate a GP model, and in such a way as to give meaningful predictive results which preserve the underlying structure of the GP function model; in fact, giving the same resultant GP model as a single GP regression on all of the data.

However, in order to iterate, the covariances and mean values for input points corresponding to new measurements must be calculated at each stage. Following the procedure above this unfortunately requires the measured data, covariances and mean values from every dataset to be saved for future use. Very quickly this iterative procedure, which was intended to be both fast and economical in terms of data storage, will be neither of these things and with enough iterates will be just as burdensome as applying a single GP to all of the data. This should not be surprising since Theorem 1 tells us that these techniques are equivalent and so, specifics of implementations aside, we should expect them to be similarly computationally expensive. Therefore, in general this approach will not be practical (even in the case where the data has to be batched, e.g. if each dataset has different noise levels or for temporal considerations) since the computational times and required data storage involved will increase rapidly as the number of iterates increases.

However, given that the above iterative scheme is equivalent to a full GP regression on all the data, it would seem to be a good candidate upon which to base any new method. Furthermore, it is shown in the next chapter that for certain covariance structures the posterior-to-prior approach can be reformulated very efficiently, and without increasing memory requirements. This proves to be a useful insight when moving to the general case.

4.7 Discussion

The current chapter has reviewed existing approaches to GP regression on batched data. Gaps in the current knowledge base were identified which have motivated the consideration of approaches to batched GPs which are based on the underlying GP models, rather than solely on distributions over inducing point sets. It was also shown that due to expensive optimisation procedures, the current methods are unsuitable for the wind turbine dynamics problem, necessitating the development of new techniques and a deeper theoretical understanding of batched data GPs. Some key questions we hope to answer relate to updating methods based on GP regression equations, the selection of inducing points based on the GP model alone and the theoretical

implications of singular covariance matrices. Two general approaches to iterative GPs, on which a new batched GP approach might be based, were also outlined.

In the following chapters we investigate the possibility for an iterative GP framework with a performance which is in-line with current state of the art techniques, while allowing for implementation in PLCs and also which addresses the posed theoretical questions of Section 4.5. Chapter 5 considers this in the simplified case of GP polynomial regression (after also developing efficient implementations of GP polynomial regression). This is initially motivated by the fact that the maximum efficiency operating region regression problem (Section 2.6.2) is polynomial in form. However, in light of the fact that existing techniques for batched GPs are all based on building a probabilistic function model using a distribution over a compact subset of domain points, it seems pertinent to initially consider these methods for a case where this assumption is true. Polynomial regression is such a case, as demonstrated by the fact that polynomial functions of degree d are completely determined by a subset of function values of cardinality $d+1$. Chapter 6 then applies the lessons learned in the polynomial case to develop a new approach to iterative GPs in the general case which addresses the gaps in existing theory.

Chapter 5

Gaussian Process Polynomial Regression

This chapter considers the application of Gaussian processes to polynomial regression. The reasons for studying polynomial regression in this work are two-fold. Firstly, it was shown in Section 2.6.2 that the below rated wind turbine regression equation (Equation 2.18) is polynomial in form. Secondly, polynomial regression is a simplified case of Gaussian Process regression for which the behaviour with respect to batched regression approaches has proved useful in developing knowledge and intuition which is then applied fruitfully in the general case. As discussed in Section 4.7, a key motivating factor for studying polynomial regression in this context is the fact that polynomial functions are entirely determined by their values taken on a small subset of input points, a property which the sparse GP approaches of Chapter 4 are essentially approximating with respect to the set of inducing points.

Section 5.1 introduces the relevant covariance structure for performing GP polynomial regression. After a reformulation of this covariance structure, fast implementations of GP polynomial regression are developed in Section 5.1.2 which are shown to scale linearly in the number of measured datapoints. Examples of implementation and a performance assessement are carried out in Section 5.1.3. Section 5.2 compares GP polynomial regression performance with that of Least Squares (LS) regression both theoretically and experimentally. The proposed approaches to Batched GP regression

of Sections 4.6.2 and 4.6.3, combined statistics and posterior-to-prior iteration respectively, are then investigated for the case of polynomial regression in Section 5.4.

While Gaussian process machine learning is a relatively recent technique, Least Squares polynomial regression on the other hand has been studied for around two hundred years and still tends to be the most common polynomial regression technique due to it being well known, easy to understand and easy to implement. For example, it is used in Microsoft Excel for trendline fitting. The method of least squares itself has been described as "the most widely used nontrivial technique of modern statistics" [50].

Note that while the inversion formula (Lemma 1), which will form a part of the implementations discussed here, has been documented previously, a full implementation of polynomial GP regression requires more than just this; including determinant evaluations, Maximum-Likelihood gradient based optimisation (via derivative and Hessian information) and evaluation of the conditional distribution. To the best of the author's knowledge a full implementation of GP polynomial regression which scales linearly in the number of data points has never been documented before now.

LS regression has always had the upper hand for polynomial and, more generally linear, regression due to its superior speed and scaling properties. Therefore, detailed comparisons between GP and LS in these cases have not been performed to date. The GP implementation presented here for polynomial regression is fast enough, and scales well enough, that comparisons of the two methods on the basis of accuracy become meaningful for any number of data points. Furthermore, regression is often performed on data which may have non-ideal noise characteristics, either being non-Gaussian or containing outliers. These cases are also investigated.

5.1 A Covariance Function for Polynomial GPs

Polynomial regression can be performed within a GP framework. In order to do this, it is necessary to use a covariance function which restricts the class of predicted functions to being polynomial in form. This polynomial covariance function is derived below.

Given a polynomial function $P(x) = \sum_{k=0}^{d} \alpha_k x^k$, of degree d and with unknown constant coefficients α_i, we can model $P(x)$ probabilistically by letting its coefficients

be *independent* Gaussian random variables, $\alpha_k \sim N(0, \sigma_k)$ for $k = 0 \dots d$ (Note this adheres to our usual prior assumption that the function is zero), the function's covariance for two given values of x is then;

$$\begin{aligned} Cov(P(x_i), P(x_j)) &= E\left[\left(\sum_{k=0}^{d} \alpha_k x_i^k\right)\left(\sum_{k=0}^{d} \alpha_k x_j^k\right)\right] \\ \textit{By linearity and independence} &= \sum_{k=0}^{d} E\left[\alpha_k^2\right] x_i^k x_j^k \\ &= \sum_{k=0}^{d} \sigma_k^2 x_i^k x_j^k. \end{aligned}$$

Hence we define an **independent coefficients polynomial covariance function** with hyperparameters $\{\gamma_k\}_{k=0}^{d}$ as,

$$k_P(x_i, x_j) = \sum_{k=0}^{d} \gamma_k x_i^k x_j^k.$$

Performing Gaussian process regression with such a covariance function is equivalent to polynomial regression in the sense that the predicted mean function will be a degree d polynomial. Using such a covariance function is essentially restricting the function space from which we are making predictions.

5.1.1 Reformulating the Polynomial Covariance Terms

Consider the covariance matrix generated by k_P,

$$[K_P]_{ij} := k_P(z_i, z_j),$$

which corresponds to a set of measurements, $\{z_i, P(z_i))\}_{i=1}^{n}$, of our polynomial function P. This matrix has inbuilt structure introduced by the polynomial covariance function which allows us to re-express K_P as follows,

$$K_P = V_{\mathbf{z}} \Sigma V_{\mathbf{z}}^T,$$

where $\mathbf{z} := [z_1, z_2, \ldots, z_n]^T$ contains the scalar input values, Σ is the diagonal matrix,

$$\Sigma = \begin{bmatrix} \gamma_d & 0 & \ldots & 0 \\ 0 & \gamma_{d-1} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \gamma_0 \end{bmatrix} \tag{5.1}$$

and,

$$V_{\mathbf{z}} = \begin{bmatrix} z_1^d & z_1^{d-1} & \ldots & z_1 & 1 \\ z_2^d & z_2^{d-1} & \ldots & z_2 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ z_n^d & z_n^{d-1} & \ldots & z_n & 1 \end{bmatrix} \tag{5.2}$$

Σ is a $(d+1) \times (d+1)$ matrix and $V_{\mathbf{z}}$ is $n \times (d+1)$ (recall that d is the degree of the polynomial and n is the number of measured data points used for regression).

Remark 1. *Note that removing the assumption that the coefficients are independent results in a covariance matrix of the form $K = V_{\mathbf{z}} \Delta V_{\mathbf{z}}^T$ where Δ is symmetric and positive definite, but not necessarily diagonal. The independence assumption is applied in the prior as there is no reason initially to believe the coefficients are dependent on one another, this assumption also allows us to avoid having large numbers of hyperparameters whose values need to be estimated. The fact that K as defined here is in fact the general form for a polynomial covariance matrix will be important when considering the posterior GP, since after conditioning on data the polynomial coefficients will no longer be independent and hence the resulting Δ is not diagonal. This general form is discussed further in Section 5.2.2.*

Remark 2. *The general polynomial covariance structure, $K = V_{\mathbf{z}} \Delta V_{\mathbf{z}}^T$, necessarily yields a singular covariance matrix whenever the number of measurements exceeds $d+1$, where d is the degree of the polynomial (as should always be the case when performing polynomial regression). The singularity can be seen by considering the dimensions of the matrices from which K is composed. Thus, the more general definition of Gaussian process, as presented in this paper, is required in this case.*

The full covariance matrix, $\widehat{K}_P$, for the measurements also contains the noise term; the noise is assumed to be iid Gaussian and so $\mathbf{B} = \xi I$ and,

$$\widehat{K}_P = K_P + \xi I = V_{\mathbf{z}} \Sigma V_{\mathbf{z}}^T + \xi I,$$

where ξ is the noise variance hyperparameter.

As discussed in Section 3.8, the main bottleneck in the efficiency of Gaussian process algorithms lies in the need to invert and find the determinants of $n \times n$ matrices where n is the number of data points being used. We now show that having formulated the covariance matrices as above for polynomial regression, we can in fact avoid having to invert large matrices when performing Gaussian process polynomial regression.

Lemma 1. The matrix $\widehat{K}_P = V_{\mathbf{z}} \Sigma V_{\mathbf{z}}^T + \xi I$ has inverse,

$$\widehat{K}_P^{-1} = \frac{1}{\xi}[I_n - V_{\mathbf{z}}[\xi I_{(d+1)} + \Sigma V_{\mathbf{z}}^T V_{\mathbf{z}}]^{-1} \Sigma V_{\mathbf{z}}^T],$$

where the dimensions of the two identity matrices have been given in subscripts.

Proof.

$$\begin{aligned}
\left(V_{\mathbf{z}} \Sigma V_{\mathbf{z}}^T + \xi I\right)^{-1} &= \frac{1}{\xi}\left[I_n - \frac{1}{\xi} V_{\mathbf{z}} \left[\Sigma^{-1} + \frac{1}{\xi} V_{\mathbf{z}}^T V_{\mathbf{z}}\right]^{-1} V_{\mathbf{z}}^T\right] \\
&= \frac{1}{\xi}\left[I_n - V_{\mathbf{z}} \left[\Sigma^{-1}\left(\xi I_{(d+1)} + \Sigma V_{\mathbf{z}}^T V_{\mathbf{z}}\right)\right]^{-1} V_{\mathbf{z}}^T\right] \\
&= \frac{1}{\xi}\left[I_n - V_{\mathbf{z}} \left[\xi I_{(d+1)} + \Sigma V_{\mathbf{z}}^T V_{\mathbf{z}}\right]^{-1} \Sigma V_{\mathbf{z}}^T\right]
\end{aligned}$$

where the first equality uses the Sherman-Morrison-Woodbury formula [51]. □

Lemma 2. The matrix $\widehat{K}_P = V_{\mathbf{z}} \Sigma V_{\mathbf{z}}^T + \xi I$ has determinant,

$$\det\left(V_{\mathbf{z}} \Sigma V_{\mathbf{z}}^T + \xi I\right) = \xi^n \det\left(I_{(d+1)} + \frac{1}{\xi} V_{\mathbf{z}}^T V_{\mathbf{z}} \Sigma\right).$$

Proof. This follows directly from the generalised Matrix Determinant Lemma. □

The preceding results are very significant for Gaussian process polynomial regression

and have been previously noted by various authors such as in [28] and [52] in the context of Bayesian linear regression; finding the inverse and determinant of the $n \times n$ covariance matrix now only requires us to invert or find determinants of $(d+1) \times (d+1)$ matrices (where d is the degree of the polynomial) **irrespective of the number of data points being used**. This removes the usual barrier of having to invert large matrices and find their determinants; however, these observations alone will not result in a full Gaussian process regression implementation which scales linearly in the number of measurements, to achieve this further work is necessary. In the literature to date this further work has not been performed, with the above identities being a stopping point for theoretical development. For example this is the case in both [28] and [52]. In the following section the existing theory is therefore extended, resulting in a complete GP polynomial regression implementation which scales linearly and includes both derivative and Hessian information for fast hyperparameter optimisation.

5.1.2 Efficient Implementation of Gaussian Process Polynomial Regression

The formulations of the previous section allow us to implement GP polynomial regression with complexity $\mathcal{O}(n)$ (for a given degree of polynomial), but, this can only be achieved with further work. Even with the inversion and determinant formulas of Lemmas 1 and 2, an inefficient implementation could still see evaluations of the log-likelihood function and its derivative vector and Hessian matrix as having complexities of $\mathcal{O}(n^2)$ or even $\mathcal{O}(n^3)$, since matrix multiplication can (at its crudest) be an $\mathcal{O}(n^3)$ operation. We will derive the linear-scaling formulas for efficient evaluation of the log-likelihood function and its derivative vector here; the Hessian equations are derived similarly, however they are considerably more involved and so the final formulas only are given in Appendix B.

Log-likelihood function

For the purposes of optimisation we maximise the log-likelihood function by minimising the negative log-likelihood function. Constants and scaling factors are removed and

hence we are minimising the following function;

$$L^* = \widehat{\mathbf{F}}^T \widehat{K}_P^{-1} \widehat{\mathbf{F}} + \log |\widehat{K}_P|, \tag{5.3}$$

notice that if we simply applied our matrix inversion formula of Lemma 1 to evaluate $\widehat{K}(\mathbf{Z}, \mathbf{Z})^{-1}$, we would then have to multiply an $n \times n$ matrix by a $1 \times n$ on its left or by a $n \times 1$ on its right and either one of these multiplications has an $\mathcal{O}(n^2)$ complexity. Instead, if the formula for the inverse is unpacked and the measurement vectors multiplied into it we obtain the following expression,

$$\widehat{\mathbf{F}}^T \widehat{K}_P^{-1} \widehat{\mathbf{F}} = \frac{1}{\xi} \left(\widehat{\mathbf{F}}^T \widehat{\mathbf{F}} - (\widehat{\mathbf{F}}^T V_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^T \widehat{\mathbf{F}}) \right), \tag{5.4}$$

where,

$$D := \xi I_{(d+1)} + \Sigma V_{\mathbf{z}}^T V_{\mathbf{z}}, \tag{5.5}$$

and the parentheses show the order of multiplication. All the expressions on the right hand side of Equation 5.4 have dimensions of size n appearing only as the central dimensions of matrix multiplications, thus each multiplication now has complexity $\mathcal{O}(n)$.

From Lemma 2 the log-determinant term in Equation 5.3 is;

$$\begin{aligned} \log |\widehat{K}_P| &= \log \left(\xi^n \det \left(I_{(d+1)} + \frac{1}{\xi} V_{\mathbf{z}}^T V_{\mathbf{z}} \Sigma \right) \right) \\ &= n \log(\xi) + \log \left(\det \left(I_{(d+1)} + \frac{1}{\xi} V_{\mathbf{z}}^T V_{\mathbf{z}} \Sigma \right) \right), \end{aligned}$$

the latter expression should be used for implementation since it is numerically more stable. The right hand side of the above equation has complexity $\mathcal{O}(n)$.

Since its constituent parts have complexity $\mathcal{O}(n)$ it follows that evaluating L^* for a given degree of polynomial is also of complexity $\mathcal{O}(n)$.

Derivatives of log-likelihood function

Let ϕ_i be the log-hyperparameter corresponding to γ_i (for $i = 0, ..., d$) and let ϕ_ξ be the log-hyperparameter of the noise variance ξ. The derivative of L^* with respect to a given

log-hyperparameter is simply the negative of Equation 3.4 with the scaling removed;

$$\frac{\partial L^*}{\partial \phi_i} = trace\left[\widehat{K}_P^{-1}\frac{\partial \widehat{K}_P}{\partial \phi_i}\right] - \widehat{\mathbf{F}}^T\widehat{K}_P^{-1}\frac{\partial \widehat{K}_P}{\partial \phi_i}\widehat{K}_P^{-1}\widehat{\mathbf{F}}. \tag{5.6}$$

Re-expressing the noisy polynomial covariance matrix as;

$$\widehat{K}_P = \sum_{i=0}^{d}\gamma_i V_i V_i^T + \xi I,$$

with $V_i := [z_1^i, z_2^i, \ldots, z_n^i]^T$ it follows that,

$$\frac{\partial \widehat{K}_P}{\partial \phi_\eta} = \begin{cases} \gamma_i V_i V_i^T & \eta = i \\ \xi I & \eta = \xi. \end{cases}$$

We then formulate the two terms in Equation 5.6 for the case $\eta = i$ as shown below;

$$\begin{aligned} trace\left[\widehat{K}_P^{-1}\frac{\partial \widehat{K}_P}{\partial \phi_i}\right] &= trace\left[\frac{1}{\xi}[I_n - V_{\mathbf{z}}D^{-1}\Sigma V_{\mathbf{z}}^T]\gamma_i V_i V_i^T\right] \\ &= \frac{\gamma_i}{\xi}trace\left[V_i V_i^T - V_{\mathbf{z}}D^{-1}\Sigma V_{\mathbf{z}}^T V_i V_i^T\right] \\ &= \frac{\gamma_i}{\xi}\Big\{V_i^T V_i - trace\left[(D^{-1}\Sigma)(V_{\mathbf{z}}^T V_i)(V_i^T V_{\mathbf{z}})\right]\Big\}, \end{aligned}$$

where we have used linearity of the trace function and invariance of trace under cyclic permutations; $V_i^T V_i$ is equal to its trace since it is scalar. For the second term we obtain,

$$\begin{aligned} \widehat{\mathbf{F}}^T\widehat{K}_P^{-1}\frac{\partial \widehat{K}_P}{\partial \phi_i}\widehat{K}_P^{-1}\widehat{\mathbf{F}} = \frac{\gamma_i}{\xi^2}\Big\{&(\widehat{\mathbf{F}}^T V_i)(V_i^T\widehat{\mathbf{F}}) \\ &-(\widehat{\mathbf{F}}^T V_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^T V_i)(V_i^T\widehat{\mathbf{F}}) \\ &-(\widehat{\mathbf{F}}^T V_i)(V_i^T V_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^T\widehat{\mathbf{F}}) \\ &+(\widehat{\mathbf{F}}^T V_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^T V_i)(V_i^T V_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^T\widehat{\mathbf{F}})\Big\}. \end{aligned}$$

Applying the same reasoning as in Section 5.1.2, the complexities for the above expressions can be seen to have $\mathcal{O}(n)$ complexity. The derivative when $\eta = \xi$, along with

the Hessian formulas, the derivations of which are rather more involved, are given in Appendix B. As with evaluating L^*, the above formulations show that we can evaluate the derivatives $\frac{\partial L^*}{\partial \phi_\eta}$ with algorithmic complexity $\mathcal{O}(n)$ and the same is true for the Hessian terms. Hence, each evaluation of the log-likelihood function (or rather L^* which we use for determining hyperparameters) along with its first derivatives and Hessian, if formulated as above, scales as $\mathcal{O}(n)$ for a given degree of polynomial.

From a coding perspective it should be noted that many of the terms in the above equations are repeated, or are transposes of each other, thus there are relatively few multiplications that need to be performed which have central dimensions of n. It is also important to formulate the prediction equations (Equations 3.1 and 3.2) efficiently, these reformulations are given in Appendix B in a form which explicitly shows that the predictions are polynomial functions with covariance matrix of the general form discussed in Remark 1. The reformulation can be done following the same approach as above, first unpacking the various terms and multiplying through before gathering adjacent terms so dimensions of size n only occur as central dimensions in matrix multiplication. Various terms needed for hyperparameter determination also appear in the prediction equations and so don't need to be re-calculated.

In the current implementation the initial log-hyperparameter values for the optimisation process are chosen randomly from the standard normal distribution. Very occasionally the D matrix of Equation 5.5 can become ill-conditioned, leading to numerical instabilities in the optimisation; it has been found that this problem is related to the initial values used for the optimisation procedure. This problem is solved by simply monitoring the condition number of D during optimisation, if the condition number becomes too large (for example $> 1 \times 10^9$) then the optimisation is restarted from a new randomly chosen initial set of log-hyperparameter values. This problem occurs only rarely and while investigating it only single restarts were generally required to remove the ill-conditioning problems. This last point highlights an important next step in developing the GP polynomial regression algorithms. The Gaussian process formulations have been optimised in terms of the number of operations required for each part of the regression process; however, work is currently ongoing to understand

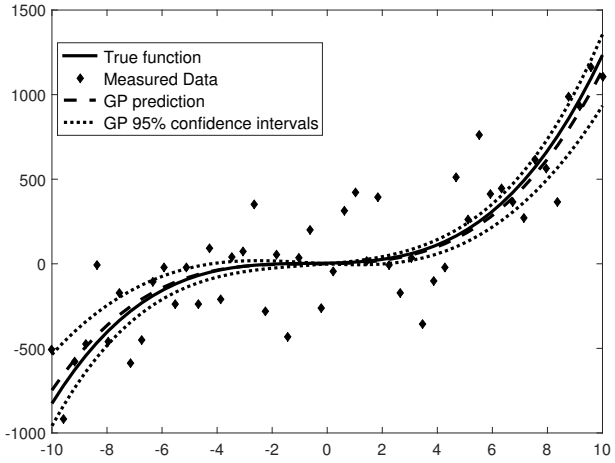

Figure 5.1: GP polynomial regression for noisy measurements of the cubic polynomial $p(x) = x^3 + 2x^2 + 3x + 4$; where the noise is iid Gaussian with standard deviation 250. The regression here was performed on 50 measurements. Note that the confidence intervals are also cubic in form; they noticeably reduce around 0 since the only variance here comes from noise and the single 'degree zero' hyperparameter.

how best to implement these algorithms from a numerical point of view. Function tolerances for optimisation, condition numbers and initial hyperparameter values have all been seen to play a role in the stability of the optimisation procedure. In order to properly optimise the implementation and thus the performance of GP polynomial regression, all of these factors need to be considered.

5.1.3 Examples of Implementation

All computations were performed on a Dell Optiplex 7010 desktop computer with an Intel Core i7-3770 processor and 8 GB of RAM.

Figure 5.1 shows the result of applying the above regression implementation to 50 noisy measurements of a cubic polynomial; the complete process of hyperparameter determination and function prediction takes about 0.3 seconds for this example. In the figure can also be seen the confidence intervals of the fit which are cubic in form.

The current algorithms are in fact able to handle substantially larger quantities of data than in this example. Figure 5.2 shows computational times for evaluating L^* and its derivative vector and Hessian matrix (for the same cubic polynomial as above)

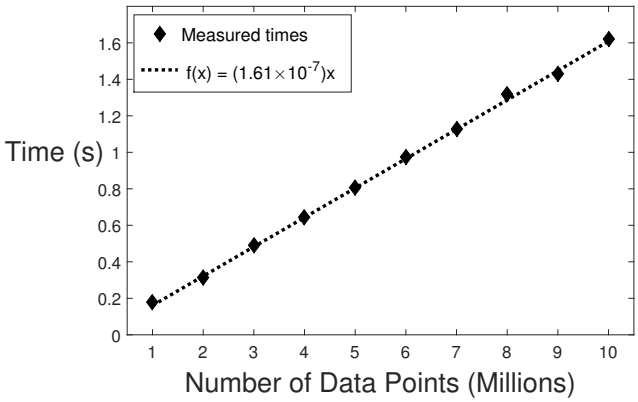

Figure 5.2: Computational times for evaluating the log-likelihood function along with its derivative vector and Hessian matrix are plotted against the number of measured data points used. Note that the numbers of data points are in millions and each quantity of input data consists of noisy measurements of the same cubic function as in Figure 5.1. The computational time for this optimised implementation can be seen to be increasing as $\mathcal{O}(n)$, i.e. linearly in n, as demonstrated by the function $f(x) = (1.61 \times 10^{-7})x$ which is plotted for comparison.

against the number of input data points and demonstrates the $\mathcal{O}(n)$ scaling of this algorithm. Note that the numbers of input points are in millions and, as can be seen in the figure, this implementation is very fast; for example we see that for 10 million input data points these evaluations take roughly 1.6 seconds. Full GP polynomial regression will take longer since in order to determine the hyperparameter values multiple evaluations of L^* and its derivatives will be required, however, the number of steps in the optimisation is usually between about 5 and 15 and is independent of the number of data points so there is not much increase from the numbers seen here. For example, Figure 5.3 shows the result of performing GP polynomial regression with one million noisy measurements of a linear polynomial; in this case the whole regression process, including making the function prediction, takes between 1 and 5 seconds. This regression code is currently programmed in Matlab and so further improvements of the time results here could certainly be achieved by implementing the code in C Mex or C.

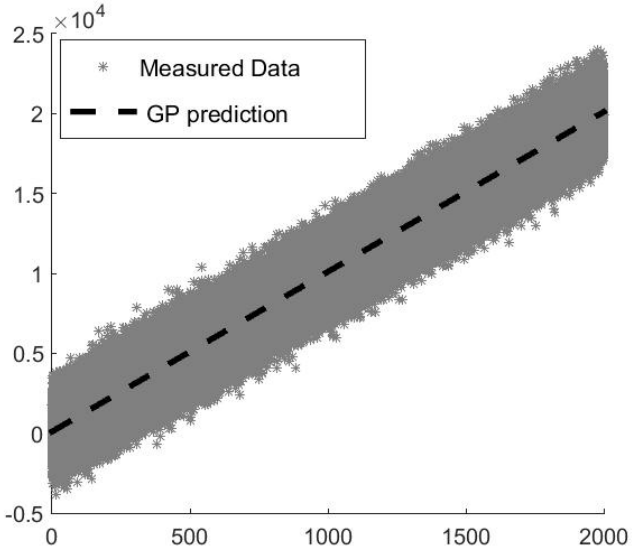

Figure 5.3: Gaussian process prediction example for polynomial regression using one million noisy measurements of the function $q(x) = 10x + 100$; the noise standard deviation on the measurements is 1000. The entire process, i.e. both hyperparameter determination and function prediction, takes between 1 and 5 seconds depending on how many steps are taken in the optimisation procedure. The prediction is lying right on top of the true underlying function, the confidence intervals for the GP fit are not shown as they have become too small to see owing to the sheer density of data being used.

5.2 Comparing Gaussian Processes and Least Squares for Polynomial Regression

Having shown that polynomial regression with GPs can be performed fast and efficiently with linear scaling in the number of data points, we now investigate whether the GP approach has any advantages over Least Squares. This present section compares the Gaussian process and Least Squares methods of polynomial regression both theoretically and experimentally. A proper analysis requires further discussion of the underlying theory of both techniques.

5.2.1 Least Squares Polynomial Regression

As above we consider the polynomial function $P(z) = \sum_{k=0}^{d} \alpha_k z^k$, of degree d with unknown constant coefficients α_i. Noisy measurements, $Y = [y_1, y_2, \ldots, y_n]^T$ with

$y_i = P(z_i) + \omega_i$, are taken for a number of input values giving the following system of equations;

$$Y = V_{\mathbf{z}} A + \boldsymbol{\omega},$$

where $V_{\mathbf{z}}$ is defined as in Equation 5.2, $A = [\alpha_d, \alpha_{d-1}, \ldots, \alpha_0]^T$ and $\boldsymbol{\omega} = [\omega_1, \omega_2, \ldots, \omega_n]^T$. Estimates, $\tilde{A}$, for the coefficients are determined by minimising the following cost function,

$$J = [Y - V_{\mathbf{z}}\tilde{A}]^T [Y - V_{\mathbf{z}}\tilde{A}],$$

which corresponds to solving,

$$\frac{1}{2}\frac{\partial J}{\partial \tilde{A}} = -[Y - V_{\mathbf{z}}\tilde{A}]^T V_{\mathbf{z}} = \mathbf{0} \implies \tilde{A} = (V_{\mathbf{z}}^T V_{\mathbf{z}})^{-1} V_{\mathbf{z}}^T Y = (V_{\mathbf{z}}^T V_{\mathbf{z}})^{-1} V_{\mathbf{z}}^T [V_{\mathbf{z}} A + \boldsymbol{\omega}]$$

It follows that $E[\tilde{A}] = A$ and $E[(\tilde{A} - A)(\tilde{A} - A)^T] = \xi (V_{\mathbf{z}}^T V_{\mathbf{z}})^{-1}$, where ξ is the noise variance. Then for any given value of z and with $v := [z^d, z^{d-1}, ..., 1]$ we have,

$$E[v(\tilde{A} - A)] = 0, \quad E[v(\tilde{A} - A)(\tilde{A} - A)^T v^T] = \xi v (V_{\mathbf{z}}^T V_{\mathbf{z}})^{-1} v^T.$$

Thus the least squares estimates of both A and the function values $P(\cdot)$ at any z are unbiased with confidence intervals which are the same for any polynomial of the same order.

5.2.2 Polynomial GP Theory

In the general polynomial regression case we have $\Lambda_{\mathbf{ZZ}} = V_{\mathbf{z}} \Delta V_{\mathbf{z}}^T$, where Δ is symmetric and positive definite (as discussed in Remark 1). It is now shown explicitly that samples drawn from a zero mean GP with this covariance structure are polynomials, furthermore, there is shown to be a 1-1 correspondence between polynomials of degree d and the samples drawn from the GP.

Drawing a sample polynomial from the GP prior can be simplified using the following square root formula for $\Lambda_{\mathbf{ZZ}}$;

$$\Lambda_{\mathbf{ZZ}}^{1/2} = (V_{\mathbf{z}} \Delta V_{\mathbf{z}}^T)^{1/2} = V_{\mathbf{z}} (V_{\mathbf{z}}^T V_{\mathbf{z}})^{-1/2} \left[(V_{\mathbf{z}}^T V_{\mathbf{z}})^{1/2} \Delta (V_{\mathbf{z}}^T V_{\mathbf{z}})^{1/2} \right]^{1/2} (V_{\mathbf{z}}^T V_{\mathbf{z}})^{-1/2} V_{\mathbf{z}}^T,$$

this can easily be verified by squaring the above expression. Then for $\boldsymbol{\nu} = [\nu_1, \nu_2, \ldots, \nu_n]^T$, a vector of independent Gaussian white noise values, observe that,

$$\begin{aligned} E[\left((V_{\mathbf{z}}^T V_{\mathbf{z}})^{-1/2} V_{\mathbf{z}}^T \boldsymbol{\nu}\right) \left((V_{\mathbf{z}}^T V_{\mathbf{z}})^{-1/2} V_{\mathbf{z}}^T \boldsymbol{\nu}\right)^T] &= (V_{\mathbf{z}}^T V_{\mathbf{z}})^{-1/2} V_{\mathbf{z}}^T E[\boldsymbol{\nu}\boldsymbol{\nu}^T] V_{\mathbf{z}} (V_{\mathbf{z}}^T V_{\mathbf{z}})^{-1/2} \\ &= (V_{\mathbf{z}}^T V_{\mathbf{z}})^{-1/2} V_{\mathbf{z}}^T V_{\mathbf{z}} (V_{\mathbf{z}}^T V_{\mathbf{z}})^{-1/2} \\ &= I. \end{aligned}$$

Note the commutativity of the transpose operator with the inversion and square root operators was used. Hence, $(V_{\mathbf{z}}^T V_{\mathbf{z}})^{-1/2} V_{\mathbf{z}}^T \boldsymbol{\nu}$ is also a Gaussian white noise vector with $d+1$ entries where d is the degree of the polynomials in the GP prior. Therefore, samples drawn from the polynomial GP prior are all of the form,

$$p(z) = v(V_{\mathbf{z}}^T V_{\mathbf{z}})^{-1/2} \left[(V_{\mathbf{z}}^T V_{\mathbf{z}})^{1/2} \Delta (V_{\mathbf{z}}^T V_{\mathbf{z}})^{1/2}\right]^{1/2} [\eta_1, \eta_2, \ldots, \eta_{d+1}]^T,$$

where,

$$v = [z^d, z^{d-1}, \ldots, z, 1],$$

and the η_i are drawn from independent Gaussian white noise. This formulation of drawing from the GP prior improves on computational speed and accuracy since square rooting is only required for $(d+1)\times(d+1)$ matrices rather than the full $n \times n$ covariance matrix. Letting,

$$\mathbf{q} := (V_{\mathbf{z}}^T V_{\mathbf{z}})^{-1/2} \left[(V_{\mathbf{z}}^T V_{\mathbf{z}})^{1/2} \Delta (V_{\mathbf{z}}^T V_{\mathbf{z}})^{1/2}\right]^{1/2} [\eta_1, \eta_2, \ldots, \eta_{d+1}]^T \tag{5.7}$$

then $\mathbf{q}$ is a vector with (d+1) entries, $\mathbf{q} = [q_d, q_{d-1}, \ldots, q_0]^T$,

$$[\eta_1, \eta_2, \ldots, \eta_{d+1}]^T = \left[(V_{\mathbf{z}}^T V_{\mathbf{z}})^{1/2} \Delta (V_{\mathbf{z}}^T V_{\mathbf{z}})^{1/2}\right]^{-1/2} (V_{\mathbf{z}}^T V_{\mathbf{z}})^{1/2} \mathbf{q}, \tag{5.8}$$

and,

$$p(z) = v\mathbf{q} = \sum_{i=0}^{d} q_i z^i. \tag{5.9}$$

Equations 5.7 and 5.8 give the 1-1 correspondence between polynomials of degree d and

samples drawn from the GP; while Equation 5.9 demonstrates explicitly that functions drawn from this GP prior are polynomials, with $\mathbf{q}$ containing the generated coefficients. Furthermore, from Equation 5.7 it is clear that the q_i are all linear combinations of normally distributed random variables and hence are themselves normally distributed and with means of zero. The result,

$$E[\mathbf{q}\mathbf{q}^T] = \Delta,$$

follows easily. For the case $\Delta = \Sigma$ this shows the q_i to be independent with q_i having variance γ_i (where γ_i is the corresponding element of the diagonal matrix Σ). Therefore, drawing polynomial coefficients from the independent coefficients GP prior using Equation 5.7 is equivalent to drawing them from independent normal distributions with relevant variances.

The error comparisons of the next section require the following reformulation of the posterior GP covariance function as defined by Equation 3.2,

$$\begin{aligned}
k^*(z, z') = & \quad v\Sigma v'^T - v\Sigma V_{\mathbf{z}}^T (V_{\mathbf{z}} \Sigma V_{\mathbf{z}}^T + \xi I)^{-1} V_{\mathbf{z}} \Sigma v'^T \\
= & \quad v\Sigma \left(I - V_{\mathbf{z}}^T (V_{\mathbf{z}} \Sigma V_{\mathbf{z}}^T + \xi I)^{-1} V_{\mathbf{z}} \Sigma \right) v'^T \\
= & \quad v\Sigma \left(I - (V_{\mathbf{z}}^T V_{\mathbf{z}} \Sigma + \xi I)^{-1} V_{\mathbf{z}}^T V_{\mathbf{z}} \Sigma \right) v'^T \\
= & \quad v\Sigma (V_{\mathbf{z}}^T V_{\mathbf{z}} \Sigma + \xi I)^{-1} \left(V_{\mathbf{z}}^T V_{\mathbf{z}} \Sigma + \xi I - V_{\mathbf{z}}^T V_{\mathbf{z}} \Sigma \right) v'^T \\
= & \quad v\Sigma (I + \frac{1}{\xi} V_{\mathbf{z}}^T V_{\mathbf{z}} \Sigma)^{-1} v'^T.
\end{aligned}$$

Where the third line uses the following identity.

Lemma 3. $X(YX + \epsilon I)^{-1} = (XY + \epsilon I)^{-1} X$ for $\epsilon \in \mathbb{R}$.

Proof. Clearly $(XY + \epsilon I)X = X(YX + \epsilon I)$, the result follows by taking appropriate inverses. □

5.2.3 Comparing Expected Errors

Error covariances have been derived for both Least Squares and Gaussian process polynomial regression techniques. These determine the error variance in the function pre-

dictions at a given point (when $z = z'$). For LS and GP respectively the error variances of the prediction at a given point, z, are,

$$E_{LS} = \xi v(V_{\mathbf{z}}^T V_{\mathbf{z}})^{-1} v^T, \quad E_{GP} = v\Sigma(I + \frac{1}{\xi} V_{\mathbf{z}}^T V_{\mathbf{z}} \Sigma)^{-1} v^T,$$

where $v = [z^d, z^{d-1}, \ldots, z, 1]$. These two error variances have very different behavior as ξ increases,

$$\lim_{\xi \to \infty} E_{LS} = \infty$$

$$\lim_{\xi \to \infty} E_{GP} = v\Sigma v^T,$$

hence, while the Least squares error variance becomes infinite as noise variance increases, the GP error variance can be seen to tend to the variance of the GP prior. The following theorem shows that GP will in fact always have the lower expected error of the two methods.

Theorem 2. *Gaussian process predictions have smaller expected error than Least Squares in the case of polynomial regression; i.e.* $E_{GP} < E_{LS}\ \forall z$.

Proof. Let $G = \frac{1}{\xi}(V_{\mathbf{z}}^T V_{\mathbf{z}})$ and $H = \frac{1}{\xi}(V_{\mathbf{z}}^T V_{\mathbf{z}}) + \Sigma^{-1}$, so $H = G + \Sigma^{-1}$. Note that G is positive definite by construction as $V_{\mathbf{z}}$ clearly has full column rank [53]; it follows that H is also positive definite since Σ^{-1} is the inverse of a positive definite matrix. Then for any nonzero column vector x,

$$\begin{aligned} x^T(H - G)x &= x^T(G + \Sigma^{-1} - G)x \\ &= x^T\Sigma^{-1}x \\ &> 0, \end{aligned}$$

hence, $H > G$. Since both matrices are positive definite, their inverses are also positive definite and satisfy $G^{-1} > H^{-1}$ [53]. Observe that,

$$G^{-1} = \xi(V_{\mathbf{z}}^T V_{\mathbf{z}})^{-1},$$

and,

$$\begin{aligned} H^{-1} &= \left(\frac{1}{\xi}(V_{\mathbf{z}}^T V_{\mathbf{z}}) + \Sigma^{-1}\right)^{-1} \\ &= \left(\left(\frac{1}{\xi}(V_{\mathbf{z}}^T V_{\mathbf{z}})\Sigma + I\right)\Sigma^{-1}\right)^{-1} \\ &= \Sigma(I + \frac{1}{\xi}V_{\mathbf{z}}^T V_{\mathbf{z}}\Sigma)^{-1}. \end{aligned}$$

Therefore, for all nonzero x,

$$x^T\xi(V_{\mathbf{z}}^T V_{\mathbf{z}})^{-1}x > x^T\Sigma(I + \frac{1}{\xi}V_{\mathbf{z}}^T V_{\mathbf{z}}\Sigma)^{-1}x.$$

Setting $x^T = v$ completes the proof. □

Remark 3. *Note that the above proof still holds if* Σ *is replaced by* Δ *(symmetric and positive definite but not necessarily diagonal). Hence, this result is still valid in the more general case where the polynomial coefficients are not independent of each other.*

5.2.4 Numerical Experiments

In the previous section it is shown that GP will on average outperform LS for a cohort of polynomials with coefficients drawn from normal distributions. The proof assumes that the hyperparameters, i.e. the variances for the coefficients and the noise, are known. In practice the hyperparameters will not be known and are inferred from the data in each regression case. This results in an increase in the expected errors whenever the inferred hyperparameter values differ from their true values as was shown explicitly in Section 3.6. Thus, whether or not the assertion that GP will outperform LS stands up in general will depend on the quality of the training process for the hyperparameters.

The performance of the current implementation of GP polynomial regression was compared with the Matlab least squares fitting function *polyfit*. Two cohorts of 1000 quartic polynomial functions were generated from probabilistic coefficients with means of 0 and variances of 1 using Equation 5.7. These will be referred to as Cohorts 1 and 2 and they each contain *different* randomly generated polynomial functions. For each polynomial in the two cohorts, noise was added at 30 measured function points

across the polynomial to form noisy measurements, Cohort 1 had added noise with a standard deviation of 5 and Cohort 2 had added noise with a standard deviation of 7. For the polynomials in each cohort both GP and LS polynomial regression is performed (full GP hyperparameter training was used for each polynomial to determine the most likely hyperparameter values based on the noisy data), the root-mean-square-error (RMSE) and mean-absolute-error (MAE) between the predicted function values and the true underlying polynomial values was calculated for both GP and LS. The error statistics for GP and LS regression are given in Tables 5.1 and 5.2 for Cohorts 1 and 2 respectively. The Kolmogorov-Smirnov test was used in each case to determine whether the differences between the error distributions for GP and LS were statistically significant, in all cases the p-value was found to be of order less than 10^{-10} and so we conclude that the differences are statistically significant.

Regression Method	Error Type	Mean	Standard Deviation
GP	RMSE	1.66	0.60
LS	RMSE	1.94	0.65
GP	MAE	1.28	0.49
LS	MAE	1.57	0.54

Table 5.1: Error statistics for GP and LS regression across the 1000 quartic polynomials in Cohort 1 with measurement noise standard deviation of 5.

Regression Method	Error Type	Mean	Standard Deviation
GP	RMSE	2.19	0.85
LS	RMSE	2.71	0.89
GP	MAE	1.66	0.67
LS	MAE	2.19	0.76

Table 5.2: Error statistics for GP and LS regression across the 1000 quartic polynomials in Cohort 2 with measurement noise standard deviation of 7.

These results reflect the theoretical findings of Section 5.2.3. Over both cohorts of functions GP can be seen to have a lower mean error than LS in both RMSE and MAE. In all cases GP also has a lower standard deviation of error. Furthermore, the asymptotics of Section 5.2.3 suggest that as noise increases, the margin by which GP is better than LS should also increase and again this behavior is seen in the experimental

results; observe that for Cohort 1 (the lower noise case) the margins by which GP has lower error are 0.28 and 0.29 for RMSE and MAE respectively, whereas for Cohort 2 (the higher noise case) the margins by which GP has lower error are respectively 0.52 and 0.53. Thus, an increase in noise level increased the margin by which GP has lower mean error than LS in both RMSE and MAE.

The increase in error variance due to hyperparameter training, discussed in Section 3.6, was also analysed numerically during the above experiments. Figure 5.4 shows the standard deviations of error at each point across the polynomials in Cohort 1 for both GP and LS, the theoretical error standard deviations from Section 5.2.3 are also shown.

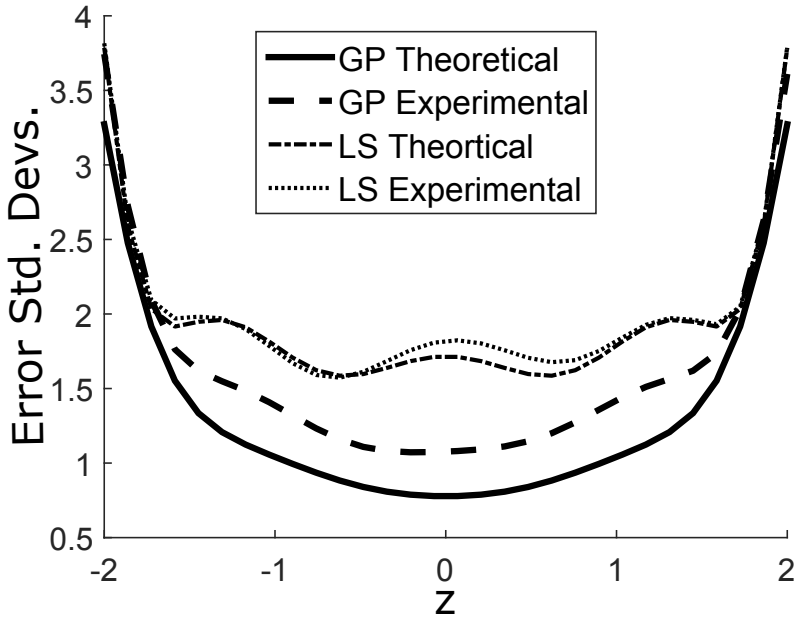

Figure 5.4: Theoretical and experimental error standard deviations for GP and LS. The predicted increase in standard deviation for GP, due to hyperparameter training, is present. However, GP still has much lower experimental error standard deviations than LS.

Note that the large increases in error standard deviations on either side of the figure are due to there being less information at the edges of the interval in which measurements were taken. The LS experimental errors in Figure 5.4 match their theoretical values very closely. For GP the predicted increase in error standard deviations from the use of estimated hyperparameters is present, however, even with this increase the GP can

still be seen to have much lower experimental error standard deviations than LS across most of the measurement space, with the errors for the two methods becoming similar (with GP still slightly smaller) towards the edges.

The results of the experimental analyses in this section show GP to be the better method for polynomial regression, even with the training of hyperparameter values.

Data with Outliers Present

In many applications of polynomial regression the measured data will contain outliers. The presence of outliers is known to cause issues for LS polynomial regression [54] and although many techniques exist to remove outliers from data, they often require manual intervention and thus in automated systems some outliers may remain. It is therefore desirable to be using techniques which are as robust as possible to the presence of outliers. In this section the performance of GPs for polynomial regression is compared to that of LS for data with outliers present. In order to model outliers in the data, t-distributed noise with one degree of freedom is used. As in the previous analysis, a cohort of 1000 polynomials was generated, noisy measurements were taken at 30 points (in this case corrupted by t-distributed noise) and both GP and LS polynomial regression performed for each set of measurements. RMSE and MAE between the predicted function values and the true polynomial values were again recorded for each of the 1000 cases. Figure 5.5 shows an example case, and the error statistics across the cohort are given in Table 5.3.

Regression Method	Error Type	Mean	Standard Deviation
GP	RMSE	3.8	10.3
LS	RMSE	9.8	47.4
GP	MAE	2.6	5.4
LS	MAE	7.1	34.4

Table 5.3: Error statistics for GP and LS regression across 1000 quartic polynomials with measurements corrupted by t-distributed noise. This noise distribution allows for the modeling of outliers in the measured data.

The results in Table 5.3 show that in the case of t-distributed noise (i.e. when outliers are present), the margin by which GP gives better results has increased considerably.

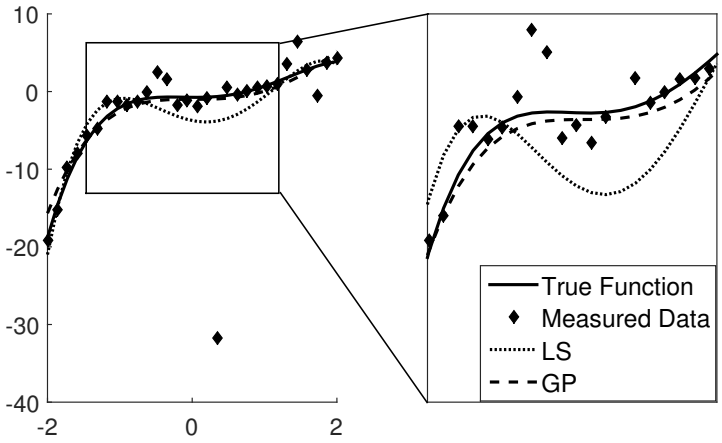

Figure 5.5: An example case of GP and LS polynomial regression in the presence of t-distributed noise. This noise distribution is being used to model data with outliers present. Even with the very large outlier present in this example the GP prediction can be seen to approximate the true function well; LS regression on the other hand has resulted in a much poorer fit.

Observe that the LS mean errors over the cohort are over twice the size of the GP errors in both RMSE and MAE; furthermore, the LS standard deviations of error are over 5 and 6 times the values for GP respectively for RMSE and MAE. These results imply that GP is far more robust to the presence of outliers in the data than LS in the case of polynomial regression.

Non-Gaussian Noise

Regression techniques are often applied in cases where the noise on the data does not, or at least cannot be known to, conform to the structure assumed by the theory underlying the regression. It is therefore necessary to consider how a given technique performs for different noise types. In the current case it is therefore necessary to consider non-Gaussian noise types. It has generally been found that GP regression appears to be robust and well-behaved in the presence of non-Gaussian noise. We have, in fact, already seen an example of this in the previous section. While t-distributed noise was used primarily to understand the comparative behaviours of GP and LS regression in the presence of outliers, this was also a comparison of the two methods in the presence of 'heavy-tailed' noise; and, as noted above, GP regression outperformed LS

regression in numerical experiments for this case. We also present results of numerical

Regression Method	Error Type	Mean	Standard Deviation
GP	RMSE	1.69	0.60
LS	RMSE	2.01	0.61
GP	MAE	1.32	0.49
LS	MAE	1.64	0.53

Table 5.4: Error statistics for GP and LS regression across 1000 quartic polynomials with measurements corrupted by uniform noise of standard deviation 5.

Regression Method	Error Type	Mean	Standard Deviation
GP	RMSE	2.23	0.82
LS	RMSE	2.75	0.83
GP	MAE	1.70	0.67
LS	MAE	2.24	0.73

Table 5.5: Error statistics for GP and LS regression across 1000 quartic polynomials with measurements corrupted by uniform noise of standard deviation 7.

experiments comparing GP and LS regression in the presence of non-Gaussian noise which, in contrast to t-distributed noise, has no weight in the tails. Tables 5.4 and 5.5 give the error statistics across cohorts of 1000 randomly generated polynomials with measurement noise, U_x, distributed uniformly between $-x$ and x; with $x = 5\sqrt{3}$ in Table 5.4 and $x = 7\sqrt{3}$ in Table 5.5 corresponding to noise cases of standard deviations 5 and 7 respectively. In all cases GPs can be seen to outperform LS regression.

5.3 Generalizations

The current formulations apply to more than just polynomial regression. This work extends in the obvious way to allow for fast and efficient GP regression for any function which is expressible as a finite linear combination of basis functions;

$$F(\mathbf{Z}) = \sum_{i=0}^{N} a_i f_i(\mathbf{Z}),$$

where the a_i are unknown coefficients and the f_i are given basis functions (clearly this corresponds to polynomial functions when $f_i(x) = x^i$). Thus, the outlined GP approach can be applied in a range of possible scenarios including multiple linear regression, multivariate polynomial regression and even fourier type analysis if the f_i are chosen to be sinusoids.

5.4 Batched Approaches to Polynomial Gaussian Processes

The proposed approaches to Batched GP regression of Sections 4.6.2 and 4.6.3 are now applied in the polynomial GP regression case.

5.4.1 Combined-Statistics Iteration Using Polynomial GP Structure

The technique of Combined-Statistics for propagating GP information will be considered here for polynomial GPs. Initially, it might seem that this approach will not be applicable in the polynomial case. This is due to the need to invert posterior polynomial GP covariance matrices (Equations 4.17 and 4.18), which have been shown to necessarily be singular, as discussed in Remark 2. However, it turns out that these singularity issues can be avoided by taking advantage of the inbuilt structure present in polynomial covariance matrices.

Let Λ_1 and Λ_2 be our covariance matrices from the posterior polynomial GP for a vector of output values, $\mathbf{F}_{\mathbf{Z}^*}$. These are of the form (see for example the prediction equations in Appendix B),

$$\Lambda_1 = VAV^T$$

$$\Lambda_2 = VBV^T.$$

A and B are both $(d+1)\times(d+1)$ matrices, for a polynomial of degree d, and generally do not have any ill-conditioning issues. Hence, A and B are invertible. Now, $(\Lambda_1 + \Lambda_2)$ is non-invertible, therefore for all $\epsilon > 0$ the term $(\Lambda_1 + \Lambda_2 + \epsilon I)$ will be invertible.

Observe that,

$$\begin{aligned}
\lim_{\epsilon\to 0} \Lambda_1(\Lambda_1+\Lambda_2+\epsilon I)^{-1}\Lambda_2 &= \lim_{\epsilon\to 0} VAV^T(VAV^T+VBV^T+\epsilon I)^{-1}VBV^T \\
\text{(By Lemma 3)} &= \lim_{\epsilon\to 0} VA(V^TVA+V^TVB+\epsilon I)^{-1}(V^TV)BV^T \\
&= \lim_{\epsilon\to 0} VA(V^TV(A+B)+\epsilon I)^{-1}(V^TV)BV^T \\
&= VA(A+B)^{-1}(V^TV)^{-1}(V^TV)BV^T \\
&= VA(A+B)^{-1}BV^T.
\end{aligned}$$

Similarly,

$$\begin{aligned}
\lim_{\epsilon\to 0} \Lambda_2(\Lambda_1+\Lambda_2+\epsilon I)^{-1} &= VB(A+B)^{-1}(V^TV)^{-1}V^T \\
\lim_{\epsilon\to 0} \Lambda_1(\Lambda_1+\Lambda_2+\epsilon I)^{-1} &= VA(A+B)^{-1}(V^TV)^{-1}V^T.
\end{aligned}$$

For an invertible matrix H, $\lim_{\epsilon\to 0}(H+\epsilon I)^{-1} = H^{-1}$. Therefore, we have managed to derive expressions for $\Lambda_1(\Lambda_1+\Lambda_2)^{-1}\Lambda_2$ and related equations which circumvent the singularity issues. It is not the case that we have managed to invert non-invertible matrices, the above process should rather be thought of as solving for X in the equation system:

$$\Lambda_1 = X(\Lambda_1+\Lambda_2),$$

with the expression $\Lambda_1(\Lambda_1+\Lambda_2)^{-1}\Lambda_2$ then being equal to $X\Lambda_2$. The combined predictive distribution, $N(\boldsymbol{\mu}_+, \Lambda_+)$, can therefore be determined, in the polynomial case, using the above expressions to evaluate Equations 4.17 and 4.18.

This approach in the case of polynomial regression does at first appear to work, with combined function predictions which look sensible. However, further testing reveals that this approach suffers from prediction anomalies, namely, we can end up with combined predictions which are nonsensical since their mean lies outside those of both the GPs we are combining. Figure 5.6 shows an example of this.

Therefore, it appears that in the polynomial case *combined-posteriors* approach to iterative GPs will not work, even in the simplified case where we are only interested

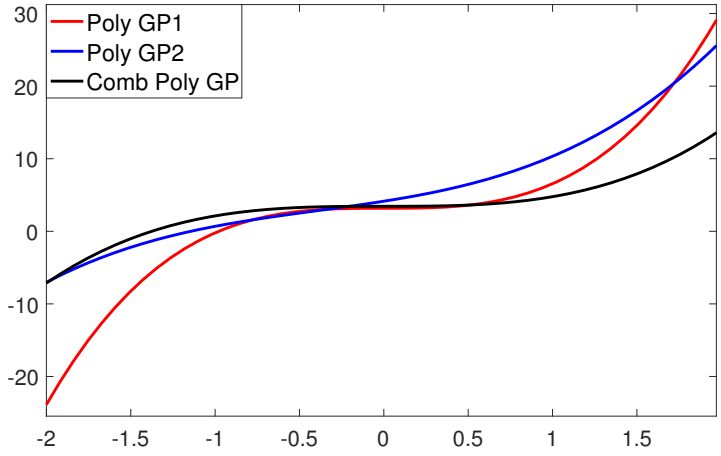

Figure 5.6: An example of the issues which can occur when performing combined-statistics iteration in the case of polynomial regression.

in the predictive statistics at a fixed set of points. For completeness, the combined-posteriors approach was also tested on general non-linear GP regression examples. The results in the more general case mirror those seen here for polynomials, more specifically, in the general case anomalous predictions are also obtained for which predicted mean values lie outside of the mean values for either of the GP models being combined. The details of these experiments can be found in Appendix C.

5.4.2 Posterior-to-Prior Approach to Iterative Polynomial GPs

In the case of polynomial regression, assume we have a prior (potentially non-zero mean) GP, $\mathcal{GP}_{\text{pr}} = (\{f_t\}_{t\in\mathbf{T}}, \mu_{\text{pr}}, k_{\text{pr}})$. As discussed, our prior means are of the form $\boldsymbol{\mu}_x^{\text{pr}} = V_x\mathbf{q}_{\text{pr}}$ and covariance matrices are all of the form $\Lambda_{xy}^{\text{pr}} = V_x\Delta_{\text{pr}}V_y^T$ where Δ_{pr} is symmetric and positive definite. It can be seen that this corresponds to a prior GP over polynomial coefficients with mean vector[1] $\mathbf{q}_{\text{pr}}$ and covariance matrix Δ_{pr}. Denoting new measured data corresponding to the input vector $\mathbf{Z}$ by $\mathbf{Y}$ as before, and again letting $\widetilde{\mathbf{Y}}$ be measurements adjusted by the prior mean; applying Lemma 3 to the polynomial GP prediction equations gives us a posterior distribution over polynomial coefficients, $N(\mathbf{q}_{\text{po}}, \Delta_{\text{po}})$, where:

[1]This prior mean is either zero if we started from a zero-mean prior, or it is the coefficients obtained from previous polynomial GP regressions if a number of iterations have already been performed.

$$\begin{aligned}\mathbf{q}_{\text{po}} &= \mathbf{q}_{\text{pr}} + \Delta_{\text{pr}}(V_{\mathbf{z}}^T V_{\mathbf{z}} \Delta_{\text{pr}} + \xi I)^{-1} V_{\mathbf{z}}^T \widetilde{\mathbf{Y}} \\ \Delta_{\text{po}} &= \Delta_{\text{pr}} + \Delta_{\text{pr}}(V_{\mathbf{z}}^T V_{\mathbf{z}} \Delta_{\text{pr}} + \xi I)^{-1} V_{\mathbf{z}}^T V_{\mathbf{z}} \Delta_{\text{pr}}.\end{aligned}$$

The above formulas show that in order to iterate from one probabilistic description of the polynomial coefficients to the next, all that is required from previous iterates is the most recent mean and covariance matrix for the coefficients. Furthermore, for any vector of input points, $\mathbf{Z}^*$, the distribution over these points in the GP posterior from the above regression has the following mean vector and covariance matrix,

$$\begin{aligned}\boldsymbol{\mu}_{\mathbf{Z}^*}^{\text{po}} &= V_{\mathbf{z}^*} \mathbf{q}_{\text{po}}, \\ \Lambda_{\mathbf{Z}^*\mathbf{Z}^*}^{\text{po}} &= V_{\mathbf{z}^*} \Delta_{\text{po}} V_{\mathbf{z}^*}^T,\end{aligned}$$

respectively. These expressions uniquely determine the mean and covariance functions in the posterior and hence the posterior GP, $\mathcal{GP}_{\text{po}}$, from regression on $\mathbf{Y}$ with prior $\mathcal{GP}_{\text{pr}}$, is completely determined by $\mathbf{q}_{\text{po}}$ and Δ_{po}. It follows that iteration of GPs in this case is equivalent to the iteration of the multivariate Gaussian distributions over the polynomial coefficients.

Thus we have shown that performing polynomial regression on batched data is possible using a posterior-to-prior iterative approach which propagates coefficient distributions from one stage to the next. This will be prove to be a useful insight when we move to the general case, although more work is required there to develop a practical iterative scheme as compared to the polynomial case considered here.

5.5 Discussion

5.5.1 Gaussian processes vs Least Squares

Significant results have been obtained with respect to understanding the relationship between GPs and LS for linear regression modelling (recall that, while polynomial regression has been the focus of this chapter, the formulations and theoretical results

apply equally to linear regression over arbitrary basis functions). First it was shown that efficient formulations of GP polynomial regression allow for it to be applied to any number of datapoints, and fast enough to make direct comparisons with LS meaningful. It was further proved theoretically that GPs have lower expected errors compared to LS, and improved asymptotic behaviour as the noise level increases. These theoretical observations were verified experimentally in various numerical experiments comparing the two methods. These findings are important since they indicate that GPs might in fact be a better candidate for the 'standard' linear regression approach, a position held by LS for around 200 years.

5.5.2 Propagation: GP Models vs GP Predictive Statistics

The considered approaches to GP iteration illustrate two overarching philosophies for propagating information using a GP. One via propagation of an underlying GP model; the second via propagation of predictive statistics. The combined-posteriors approach of the previous sections was initially seeking to be of the first type, propagating a GP model, but on finding that the GP models could not be combined consistently using this approach, we considered propagating the GP's predictive statistics instead.

A GP is a probability distribution over a class of functions and contains information about the values taken by functions in the given class at, and between, every point in the domain through its mean function μ and covariance function k. It was hoped that by propagating the GP's predictive statistics, the essential parts of the GP model would be preserved. However, by doing this we are actually abandoning our GP model and simply combining two multivariate Gaussian distributions, distributions which do not contain anywhere near the volume of information of the original GP model and which also lack the fundamental structure which makes GPs such a powerful tool. It seems to be because we have abandoned the underlying GP model that we see results (as in Figure 5.6) which are not meaningful in terms of combining GP predictions for our function. The resulting 'non-meaningful' predictions were foreshadowed by the fact that we couldn't use combined-posteriors to produce a refined full GP model due to ill-definedness of the mean and covariance functions; implying that the result of

which could not be meaningful in a functional sense. This was then borne out by the experiments with combining predictive statistics for polynomial GPs.

All together this implies that while predictive statistics are often the goal of applying a GP, any technique which allows for *meaningful* iterative GP regression will need to be one which consistently propagates a full GP model. Furthermore, given that it has now been shown that,

1. Posterior-to-prior GP iteration is equivalent to standard GP regression on all of the data

2. In the case of polynomial GPs, posterior-to-prior iteration can be performed efficiently and without increasing memory requirements,

it seems that a principled approach to iterative GP methods should be posterior-to-prior in form, while attempting to leverage experience of the polynomial case to avoid increasing computational costs and memory requirements with each iteration.

We therefore abandon the combined-statistics approach to iterative GPs, which has been shown to result in anomalous predictions, and concentrate on generalising posterior-to-prior iteration.

5.5.3 Expected Behaviour and Relationship to Standard GPs

The analysis of Section 5.4 demonstrates another important consideration for when developing an iterative approach to GPs. That being the expected behaviour of a given method and its relationship to standard GP regression. Combined-Posteriors iteration has been shown to be prone to anomalous behaviour. This was picked up on during testing, however, since a majority of results look sensible these issues conceivably could have gone unnoticed. It would therefore be reasonable to ask whether the posterior-to-prior approach might also display similar behaviour given enough test cases? This question can be safely answered in the negative, however, thanks to Theorem 1 which shows that the iterative scheme is in fact equivalent to standard GP polynomial regression. The expected behaviour from this method is therefore identical to that of a

standard GP model.

The knowledge of how a given method relates back to standard GP regression therefore allows us to have confidence in its expected behaviour. In the general case the relationship will not be an equality, as it is for polynomials, but one would hope that an understanding as to the degree of similarity could still be established. This point will be revisited in the following chapter. Thus, expected behaviour gives a further reason for developing iterative GPs based on the posterior-to-prior approach, since the relationship to standard GPs is known.

Chapter 6

Sufficient-Subset Gaussian Process Iteration

This chapter documents a new approach to iterative Gaussian processes (as per Definition 2), the development of which is guided by the possibility for dimensionality reduction seen previously for polynomial GPs. In order to learn from the polynomial case, links must be established which relate GPs for polynomials to more general GP function classes. As will be shown, the key to establishing such a link lies in the structure of GP covariance matrices. This link is established in Sections 6.1 and 6.2. Section 6.3 then considers how the lessons learned thus far in the context of polynomial GPs can be extended to the general case in light of covariance matrix structural similarities. Sufficient-Subset Gaussian process iteration is then developed from Section 6.4 onwards.

6.1 Degenerate Gaussian Distributions

In the definition of a Gaussian process (Definition 1) the case of a degenerate Gaussian, i.e. one with a singular covariance matrix, is explicitly included. This is necessary for two reasons, first because polynomial (and more generally, linear) regression with GPs results in singular covariance matrices, and polynomial regression turns out to be an important case in which to understand iterative GPs in order to inform in the

general case; secondly because while in most general cases the covariance matrices in the GP posteriors from functional regression are theoretically non-singular, finite computational precision renders them effectively singular for practical purposes (this will also be discussed further) and so a thorough understanding of iterative approaches to GPs requires the singular case to be investigated.

We start by considering the multivariate Gaussian probability density function; if $\mathbf{Y} \sim N(\boldsymbol{\mu}, \Lambda)$, with Λ non-singular and $\mathbf{Y}$ having dimension n, then;

$$p(\mathbf{Y}) = \frac{1}{\sqrt{(2\pi)^n |\Lambda|}} \exp\left(-\frac{1}{2}(\mathbf{Y}-\boldsymbol{\mu})^T \Lambda^{-1} (\mathbf{Y}-\boldsymbol{\mu})\right).$$

We now re-express this density function following the formulations of [55]. Since Λ is real, symmetric and positive-definite, its inverse shares these properties, and so admits the spectral expansion,

$$\Lambda^{-1} = \sum_{i=1}^{n} \lambda_i^{-1} \mathbf{U}_i \mathbf{U}_i{}^T \quad ,$$

where the λ_i are the eigenvalues of the matrix Λ and the $\mathbf{U}_i$ their corresponding orthonormal eigenvectors. Hence, we can re-write the density function as,

$$P(\mathbf{Y}) = \prod_{i=1}^{n} (2\pi\lambda_i)^{-1/2} \exp\left(-\frac{\lambda_i^{-1}}{2}\left((\mathbf{Y}-\boldsymbol{\mu})^T \mathbf{U}_i\right)^2\right).$$

Let the first r eigenvalues be fixed and non-zero, while the others tend to zero in order to transition to the degenerate case of rank $r < n$. These eigenvalues tending to zero results in each multiplier with indices $r+1, \ldots, n$ to tend towards a Dirac delta function as per its definition, obtaining;

$$\begin{aligned} P(\mathbf{Y}) &= 2\pi^{-r/2} \prod_{i=1}^{r} (\lambda_i)^{-1/2} \exp\left(-\frac{\lambda_i^{-1}}{2}\left((\mathbf{Y}-\boldsymbol{\mu})^T \mathbf{U}_i\right)^2\right) \prod_{i=r+1}^{n} \delta\left((\mathbf{Y}-\boldsymbol{\mu})^T \mathbf{U}_i\right) \\ &= 2\pi^{-r/2} \left(\prod_{i=1}^{r} \lambda_i\right)^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{Y}-\boldsymbol{\mu})^T \Lambda^{+} (\mathbf{Y}-\boldsymbol{\mu})\right) \prod_{i=r+1}^{n} \delta\left((\mathbf{Y}-\boldsymbol{\mu})^T \mathbf{U}_i\right), \end{aligned}$$

where $\Lambda^{+} := \sum_{i=1}^{r} \lambda_i^{-1} \mathbf{U}_i \mathbf{U}_i{}^T$ is the Moore-Penrose pseudoinverse of Λ. Care must be taken when interpreting the above expression since the Dirac delta belongs to the class

of generalised distributions and therefore is only properly defined under the integral sign. However, this does not present a problem here since we can simply infer the effect of these delta functions by considering their impact on the values of $\mathbf{Y}$ obtained from the above density function. It is apparent that the delta functions indicate linear dependences between the values in $\mathbf{Y}$, specifically that $(\mathbf{Y} - \boldsymbol{\mu})^T \mathbf{U}_i = 0$ for each i from $r + 1$ to n.

6.2 Degeneracy in GP Posterior Covariance Matrices

GP posterior covariance matrices tend to be ill-conditioned and close to singular. In the work of [29] it was shown that large eigenvalues of the GP covariance matrix and their corresponding eigenvectors are associated with the large scale smooth structure of the predicted function, whereas small eigenvalues/vectors are associated with the small scale and more noise like variations along the function. Therefore, since predicted functions from GP regression are smooth we should expect the smaller eigenvalues to quickly tend towards zero, rendering the covariance matrix itself effectively singular in terms of finite computational accuracy. Another way of interpreting the near singularity of the covariance matrices in GP posteriors comes from the delta functions discussed in the previous section. It was shown that delta functions in the pdf effectively set up linear dependencies between the possible values taken by the function. This corresponds to the idea that a singular covariance matrix occurs if knowing some of our function values fixes the values taken by the function at other points; for a smooth function prediction, if we pick a cluster of nearby points and assume we know the values taken there, then the value taken at some other point internal to this cluster becomes very heavily constrained. This indicates that there will be *delta function like* terms in the pdf and so we would expect covariance matrices for smooth functions to be near singular. The presence of near singular and singular covariance matrices further motivates the study of the polynomial case of GP regression carried out in Chapter 5; this is because polynomial covariance matrices are always singular, as discussed in Section 5.1.1.

6.3 Learning from the Polynomial Case

Posterior-to-prior GP iteration is shown in Theorem 1 to offer a consistent way of propagating a GP model while refining it through conditioning on new batches of data, and was even shown to be equivalent to performing a single GP regression on the whole set of data. In the general case this garnered no great advantage due to increasing computational times and memory demands at each stage of iteration, but for polynomial regression it was shown that these issues are not present. Polynomial regression proves to be a special case due to the fact that a polynomial function is completely determined by the values of its coefficients. This means that all of the information we have about the values taken by a polynomial function across our domain can be condensed into information about the polynomial coefficients, and condensing everything into this form doesn't result in any loss of information. Recalling the discussion of Section 5.1.1 we see that this possibility for dimensionality reduction is indicated by the fact that polynomial covariance matrices are singular whenever the number of datapoints increases above $d+1$, where d is the degree of the polynomial. This is in line with the fact that a polynomial of degree d has $d+1$ coefficients and knowing the value of the function at $d+1$ distinct points determines the whole polynomial function.

We now consider the case of GPs for general non-parametric functions. In Section 6.2 we discussed the fact that posterior covariance matrices tend to be ill-conditioned or 'close to singular' as the number of considered points increases, this suggests that it is worth considering whether this near singularity is offering us any kind of dimensionality reduction as was possible in the polynomial case. This question is explored in the following analysis: consider a zero-mean GP model of a function, f, and let $\mathbf{P}$ be an N dimensional subset of the functional domain such that the covariance matrix $\Lambda_{\mathbf{PP}}$ is non-singular, but the covariance matrix $\Lambda_{\mathbf{P}^\dagger\mathbf{P}^\dagger}$ is singular for all z where,

$$\mathbf{P}^\dagger = \begin{bmatrix} \mathbf{P}^T & z \end{bmatrix}^T.$$

It follows that,

$$\Lambda_{\mathbf{P}^\dagger\mathbf{P}^\dagger} = UDU^T = \begin{bmatrix} U_{\mathbf{P}} & \mathbf{v}_z \\ \mathbf{w}_z^T & u_z \end{bmatrix} \begin{bmatrix} D_{\mathbf{P}} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} \begin{bmatrix} U_{\mathbf{P}}^T & \mathbf{w}_z \\ \mathbf{v}_z^T & u_z \end{bmatrix},$$

where U is an orthogonal matrix and $D_{\mathbf{P}}$ is diagonal and positive definite. The column vectors of $U_{\mathbf{P}}$ span $\mathbb{R}^m$, where $z \in \mathbb{R}^m$. Note that u_z cannot be zero, since, if $u_z = 0$ the orthogonality of U implies that $\mathbf{v}_z$ is orthogonal to all the columns vectors of $U_{\mathbf{P}}$. However, the column vectors of $U_{\mathbf{P}}$ span $\mathbb{R}^N$ and so this means $\mathbf{v}_z$ itself must be zero, contradicting the orthogonality of U.

Thus, $\begin{bmatrix} \mathbf{v}_z^T & u_z \end{bmatrix}^T$ is an eigenvector of $\Lambda_{\mathbf{P}^\dagger\mathbf{P}^\dagger}$ with eigenvalue 0. As seen in Section 6.1 this corresponds to the pdf of the joint distribution for f, at points in $\mathbf{P} \cup \{z\}$, including a delta function of the form,

$$\delta(\mathbf{F}_{\mathbf{P}}^T\mathbf{v}_z + f_z u_z).$$

Hence, for any z,

$$f_z = -\frac{\mathbf{F}_{\mathbf{P}}^T\mathbf{v}_z}{u_z} = -\frac{\mathbf{v}_z^T\mathbf{F}_{\mathbf{P}}}{u_z}.$$

It immediately follows that for all z and z^*,

$$\Lambda_{zz^*} = \left(\mathbf{v}_z^T/u_z\right) \mathbb{E}(\mathbf{F}_{\mathbf{P}}\mathbf{F}_{\mathbf{P}}^T) \left(\mathbf{v}_{z^*}^T/u_{z^*}\right) = \left(\mathbf{v}_z^T/u_z\right) \Lambda_{\mathbf{PP}} \left(\mathbf{v}_{z^*}^T/u_{z^*}\right),$$

and more generally, for vectors of points $\mathbf{Z}$ and $\mathbf{Z}^*$, we have,

$$\Lambda_{\mathbf{ZZ}^*} = V_{\mathbf{Z}}\Lambda_{\mathbf{PP}}V_{\mathbf{Z}^*}^T,$$

for some $V_{\mathbf{Z}}$ and $V_{\mathbf{Z}^*}$. We have therefore demonstrated that any covariance structure for which the addition of new points eventually yields a singular covariance matrix has the same underlying structure as the polynomial covariance matrices which have been studied in the current work. Thus, in any case where this holds, knowing the values taken by the function at the points in $\mathbf{P}$ is enough to determine its values taken everywhere; and so, dimensionality reduction can theoretically be achieved if

all information about the function could be re-expressed in terms of function values at points in $\mathbf{P}$. Furthermore, in cases where the addition of new points in $\mathbf{P}$ would theoretically never result in a truly-singular covariance matrix, each addition of a new point will cause the smallest eigenvalue to decrease in size; eventually this smallest eigenvalues will become close enough to zero to approximate the above situation, and so dimensionality reduction should be possible which allows for function values across the domain to be determined from information re-expressed at points in $\mathbf{P}$ up to some desired accuracy. The following section discusses how this may be done in such a way as to yield a posterior-to-prior iterative method of GP regression via dimensionality reduction of the type discussed here.

6.4 Sufficient-Subset Gaussian Process Regression

In Section 5.4.2 it was shown that iterative GP regression could be performed in the case of polynomials without having to carry forward all of the previously measured data. In fact, the amount of data which needs to be propagated is shown to be very modest. This is due to the dimensionality reduction afforded by the fact that the only information needed for the next iterate was that learned about the finite and small number (relative to possible amounts of measured data) of coefficients. It was further discussed in Section 6.3 that the possibility for dimensionality reduction in this case is indicated by the fact that polynomial covariance matrices are singular when the number of considered points is larger than $d + 1$ where d is the order of the polynomial. In Section 6.1 it was seen that singularity results in delta functions appearing in the multivariate distributions for the GP and these in turn indicate redundancies in the amount of information required to know which values the function takes at given points; since linear dependencies arise between the function values, knowing a certain number of them fixes the possible function values elsewhere. Finally, at the end of the previous section it was shown that, in principle, there is the potential for dimensionality reduction in GP regression for general non-linear functions via the repackaging of information to reflect what is happening only on some finite subset of points, chosen such that the values at other input point are then either known exactly or can be determined up to some chosen

level of accuracy.

Covariance matrices in GP models are notorious for being ill-conditioned and close to singular, if it was not for the presence of noise in measurements then GP regression would not generally be viable and even when no noise is present it is common to add a noise like term along the diagonal, called 'jitter' [56], in order to improve numerical stability. With infinite precision the covariance matrices for general non-linear functions would not actually be singular, however, with finite computational precision they become effectively singular very quickly as new points are added. Until now this has been seen as an inconvenience and an unfortunate obstacle. In other areas where matrix numerics are employed this is indeed the case. However, since in the context of GPs we are dealing with probability distributions, this ill-conditioning is in fact an indication that there is redundancy of information.

SSGP iteration seeks to exploit the redundancy of information, indicated by ill-conditioned covariance matrices, by re-packaging large amounts of measured data into a small packet of pseudo-data which still contains all of the relevant information about the underlying function, up to a precision we choose. We seek to propagate and update this pseudo-data in an iterative scheme. This will allow both for GP updating when new data becomes available, but also for large datasets to be partitioned and processed in batches in order to overcome the usual cubic complexity of GP regression implementations. For the sake of clarity, the posterior-to-prior approach of Section 4.6.3 will be referred to as 'naive-Bayesian iteration' throughout the following sections.

6.4.1 Developing the SSGP Theory

The various ideas which have been discussed so far are now used to develop an iterative GP method. It has been shown that singularity in GP covariance matrices, for a given set of points $\mathbf{P}$, indicates that information at those points defines the function everywhere (either exactly or up to some level of precision). We therefore now consider how information from across the domain might be re-expresssed as information at the points in this chosen set, $\mathbf{P}$. More formally: given a finite subset, $\mathbf{P}$, of points in our functional domain, a covariance function, k, and noisy measurements, $\mathbf{Y_D}$, corresponding to the

input points $\mathbf{D}$, we ask the following question:

"*Can a new dataset, $\mathbf{Y_P}$, be generated consisting of one measurement at each point in $\mathbf{P}$ and from which GP regression would result in the same predictions and covariance structure across points in $\mathbf{P}$ as when using the original dataset $\mathbf{D}$?*"

Assuming the existence of such a set, observe that the mean predictions at $\mathbf{P}$ from regression on these two datasets are;

$$\begin{aligned} \text{from } \mathbf{Y_D} \quad &- \quad \Lambda_{\mathbf{PD}}(\Lambda_{\mathbf{DD}} + \Omega_{\mathbf{D}})^{-1}\mathbf{Y_D} \\ \text{from } \mathbf{Y_P} \quad &- \quad \Lambda_{\mathbf{PP}}(\Lambda_{\mathbf{PP}} + \Omega_{\mathbf{P}})^{-1}\mathbf{Y_P}, \end{aligned}$$

with covariances,

$$\begin{aligned} \text{from } \mathbf{Y_D} \quad &- \quad \Lambda_{\mathbf{PP}} - \Lambda_{\mathbf{PD}}(\Lambda_{\mathbf{DD}} + \Omega_{\mathbf{D}})^{-1}\Lambda_{\mathbf{DP}} \\ \text{from } \mathbf{Y_P} \quad &- \quad \Lambda_{\mathbf{PP}} - \Lambda_{\mathbf{PP}}(\Lambda_{\mathbf{PP}} + \Omega_{\mathbf{P}})^{-1}\Lambda_{\mathbf{PP}}. \end{aligned}$$

The above terms equate pairwise if we let:

$$(\Lambda_{\mathbf{PP}} + \Omega_{\mathbf{P}})^{-1}\mathbf{Y_P} = M(\Lambda_{\mathbf{DD}} + \Omega_{\mathbf{D}})^{-1}\mathbf{Y_D} \tag{6.1}$$

$$(\Lambda_{\mathbf{PP}} + \Omega_{\mathbf{P}})^{-1} = M(\Lambda_{\mathbf{DD}} + \Omega_{\mathbf{D}})^{-1}M^T, \tag{6.2}$$

where M satisfies the equation,

$$\Lambda_{\mathbf{PD}} = \Lambda_{\mathbf{PP}}M. \tag{6.3}$$

It therefore follows that rather than explicitly generating $\mathbf{Y_P}$, which we will refer to as the 'pseudo-data at $\mathbf{P}$', we can instead deal with the 'pseudo-terms',

$$\begin{aligned} \Gamma &:= (\Lambda_{\mathbf{PP}} + \Omega_{\mathbf{P}})^{-1}\mathbf{Y_P} \\ \Psi &:= (\Lambda_{\mathbf{PP}} + \Omega_{\mathbf{P}})^{-1}, \end{aligned}$$

defined by Equations 6.1 and 6.2. The GP predictions at $\mathbf{P}$ therefore have mean and

covariance matrix,

$$\begin{aligned} \text{mean} &= \Lambda_{\mathbf{PP}}\Gamma \\ \text{covariance matrix} &= \Lambda_{\mathbf{PP}} - \Lambda_{\mathbf{PP}}\Psi\Lambda_{\mathbf{PP}}, \end{aligned}$$

and by construction these terms agree with those obtained from the standard GP formulas.

Assume that we have followed the above process for measured data at points $\mathbf{D}_1$, giving us pseudo-terms Γ_1 and Ψ_1 which encode the predictive structure of the data at $\mathbf{D}_1$. Further, assume that we have access to the individual terms $\Lambda^1_{\mathbf{PP}}$, $\Omega^1_{\mathbf{P}}$ and $\mathbf{Y}^1_{\mathbf{P}}$ of Γ_1 and Ψ_1. If new measurements $\mathbf{Y}_{\mathbf{D}_2}$ at points $\mathbf{D}_2$ become available, the above formulations can be applied to the compounded data,

$$\mathbf{Y}_{\mathbf{D}} = \begin{bmatrix} \mathbf{Y}^1_{\mathbf{P}} \\ \mathbf{Y}_{\mathbf{D}_2} \end{bmatrix} \quad \text{with} \quad \mathbf{D} = \begin{bmatrix} \mathbf{P} \\ \mathbf{D}_2 \end{bmatrix}.$$

In order to form the pseudo-terms, which will be denoted $\Gamma_{1\&2}$ and $\Psi_{1\&2}$, for this $\mathbf{Y}_{\mathbf{D}}$ we must solve for M in,

$$\Lambda_{\mathbf{PD}} = \Lambda_{\mathbf{PP}}M.$$

With the current $\mathbf{D}$, $\Lambda_{\mathbf{PD}} = [\ \Lambda_{\mathbf{PP}} \quad \Lambda_{\mathbf{PD}_2}\ ]$ and so,

$$M = [\ I \quad M_2\ ],$$

where M_2 satisfies,

$$\Lambda_{\mathbf{PD}_2} = \Lambda_{\mathbf{PP}}M_2.$$

Forming $\Gamma_{1\&2}$ and $\Psi_{1\&2}$ also requires the inverse term $(\Lambda_{\mathbf{DD}} + \Omega_{\mathbf{D}})^{-1}$, obtained using block matrix inversion as follows:

$$\Lambda_{\mathbf{DD}} + \Omega_{\mathbf{D}} = \begin{bmatrix} \Lambda^1_{\mathbf{PP}} + \Omega^1_{\mathbf{P}} & \Lambda_{\mathbf{PD}_2} \\ \Lambda_{\mathbf{D}_2\mathbf{P}} & \Lambda_{\mathbf{D}_2\mathbf{D}_2} + \Omega_{\mathbf{D}_2} \end{bmatrix},$$

if,

$$\Lambda_{\mathbf{DD}} + \Omega_{\mathbf{D}} = \begin{bmatrix} Q & R \\ R^T & S \end{bmatrix},$$

then,

$$(\Lambda_{\mathbf{DD}} + \Omega_{\mathbf{D}})^{-1} = \begin{bmatrix} Q^{-1} + Q^{-1}RXR^TQ^{-1} & -Q^{-1}RX \\ -XR^TQ^{-1} & X \end{bmatrix}, \tag{6.4}$$

where $X = (S - R^TQ^{-1}R)^{-1}$. Therefore,

$$\begin{aligned}
\Gamma_{1\&2} &= M(\Lambda_{\mathbf{DD}} + \Omega_{\mathbf{D}})^{-1}\mathbf{Y_D} \\
&= \begin{bmatrix} I & M_2 \end{bmatrix} \begin{bmatrix} Q^{-1} + Q^{-1}RXR^TQ^{-1} & -Q^{-1}RX \\ -XR^TQ^{-1} & X \end{bmatrix} \begin{bmatrix} \mathbf{Y^1_P} \\ \mathbf{Y_{D_2}} \end{bmatrix} \qquad (6.5) \\
&= Q^{-1}\mathbf{Y^1_P} + \left[Q^{-1}R - M_2\right] X \left[R^TQ^{-1}\mathbf{Y^1_P} - \mathbf{Y_{D_2}}\right] \\
&= \Gamma_1 + \left[M_2 - \Psi_1\Lambda_{\mathbf{PD_2}}\right] \left[\Lambda_{\mathbf{D_2D_2}} - \Lambda_{\mathbf{D_2P}}\Psi_1\Lambda_{\mathbf{PD_2}} + \Omega_{\mathbf{D_2}}\right]^{-1} \left[\mathbf{Y_{D_2}} - \Lambda_{\mathbf{D_2P}}\Gamma_1\right] \qquad (6.6)
\end{aligned}$$

$$\begin{aligned}
\Psi_{1\&2} &= M(\Lambda_{\mathbf{DD}} + \Omega_{\mathbf{D}})^{-1}M^T \\
&= \begin{bmatrix} I & M_2 \end{bmatrix} \begin{bmatrix} Q^{-1} + Q^{-1}RXR^TQ^{-1} & -Q^{-1}RX \\ -XR^TQ^{-1} & X \end{bmatrix} \begin{bmatrix} I \\ M_2^T \end{bmatrix} \\
&= \Psi_1 + \left[M_2 - \Psi_1\Lambda_{\mathbf{PD_2}}\right] \left[\Lambda_{\mathbf{D_2D_2}} - \Lambda_{\mathbf{D_2P}}\Psi_1\Lambda_{\mathbf{PD_2}} + \Omega_{\mathbf{D_2}}\right]^{-1} \left[M_2 - \Psi_1\Lambda_{\mathbf{PD_2}}\right]^T \qquad (6.7)
\end{aligned}$$

And as above, the mean and covariance matrix at points in $\mathbf{P}$ are correspondingly;

$$\begin{aligned}
\text{mean} &= \Lambda_{\mathbf{PP}}\Gamma_{1\&2} \\
\text{covariance matrix} &= \Lambda_{\mathbf{PP}} - \Lambda_{\mathbf{PP}}\Psi_{1\&2}\Lambda_{\mathbf{PP}},
\end{aligned}$$

where this time, the equality between these terms and those obtained with standard GP iteration is not a given, as will be discussed. The above theory suggests an iterative scheme for performing GP regression and making predictions at any points in the domain, not just those in $\mathbf{P}$. This iterative scheme will be referred to as *Sufficient-Subset*

GP Iteration. We first formalise this iterative scheme before exploring its properties in relation to standard GP and the posterior-to-prior GP iteration (naive-Bayesian Iteration) discussed previously.

6.4.2 Sufficient-Subset GP (SSGP) Iteration

Definition 3. *An **SSGP tuple** is a GP structure of the form $\mathcal{GP} = (k, \mathbf{P}, \Gamma, \Psi)$ consisting of a covariance function, k, a subset of domain points, $\mathbf{P}$, the pseudo-precision matrix Ψ; and Γ which corresponds to Ψ multiplied by pseudo-data $\mathbf{Y_P}$. The mean at point z for this GP is,*

$$\boldsymbol{\mu}_z = \Lambda_{z\mathbf{P}}\Gamma,$$

and the covariance between points z_1 and z_2 is,

$$\Phi_{z_1 z_2} = \Lambda_{z_1 z_2} - \Lambda_{z_1\mathbf{P}}\Psi\Lambda_{\mathbf{P}z_2}.$$

Note that all $\Lambda_{(\cdot)(\cdot)}$ matrices are constructed using the covariance function k.

Assume that we have the prior GP tuple, $\mathcal{GP}_{pr} = (k, \mathbf{P}, \Gamma_{pr}, \Psi_{pr})$, with mean $\boldsymbol{\mu}^{pr}_{\mathbf{Z}}$ and covariance matrix $\Phi^{pr}_{\mathbf{ZZ}}$ at the points in $\mathbf{Z}$. Given new measured data $\mathbf{Y_D}$, with input points $\mathbf{D}$, a refinement of the prior can be performed (by applying equations 6.6 and 6.7) which yields the posterior GP tuple defined as follows:

Definition 4. *Let $\mathcal{GP}_{pr} = (k, \mathbf{P}, \Gamma_{pr}, \Psi_{pr})$ be a prior GP tuple which describes our beliefs about some function, f. Given new measured data $\mathbf{Y_D}$, at input points $\mathbf{D}$, we define the refined **posterior SSGP tuple** to be $\mathcal{GP}_{po} = (k, \mathbf{P}, \Gamma_{po}, \Psi_{po})$, where;*

$$\Gamma_{po} = \Gamma_{pr} + [M - \Psi_{pr}\Lambda_{\mathbf{PD}}][\Lambda_{\mathbf{DD}} - \Lambda_{\mathbf{DP}}\Psi_{pr}\Lambda_{\mathbf{PD}} + \Omega_{\mathbf{D}}]^{-1}[\mathbf{Y_D} - \Lambda_{\mathbf{DP}}\Gamma_{pr}] \tag{6.8}$$

$$\Psi_{po} = \Psi_{pr} + [M - \Psi_{pr}\Lambda_{\mathbf{PD}}][\Lambda_{\mathbf{DD}} - \Lambda_{\mathbf{DP}}\Psi_{pr}\Lambda_{\mathbf{PD}} + \Omega_{\mathbf{D}}]^{-1}[M - \Psi_{pr}\Lambda_{\mathbf{PD}}]^{T}. \tag{6.9}$$

M is determined by solving the equation system,

$$\Lambda_{\mathbf{PD}} = \Lambda_{\mathbf{PP}}M. \tag{6.10}$$

This generation of a refined posterior GP tuple from new data can clearly be repeated as new data becomes available, with the current tuple becoming the prior for the next refinement step:

$$\mathcal{GP}^i_{pr} = (k, \mathbf{P}, \Gamma^i_{pr}, \Psi^i_{pr}) \xrightarrow[\text{on } \mathbf{D}]{\text{regression}} [\mathcal{GP}^i_{po} = (k, \mathbf{P}, \Gamma^i_{po}, \Psi^i_{po})] \equiv [\mathcal{GP}^{i+1}_{pr} = (k, \mathbf{P}, \Gamma^{i+1}_{pr}, \Psi^{i+1}_{pr})] \tag{6.11}$$

Hence, SSGP iteration is a posterior-to-prior iterative approach to GP regression. Furthermore, the initial generation of the Γ and Ψ terms (Equations 6.1 and 6.2) is in fact equivalent to an initial prior tuple of $\mathcal{GP}_{pr} = (k, \mathbf{P}, \mathbf{0}, \mathbf{0})$. SSGP iteration avoids the need to invoke the assumption that the new dataset, $\mathbf{Y_P}$, and its associated covariance matrix exist. Only the assumption, that M exists, remains. However, the relationship to standard GP regression remains to be established.

The predicted mean and covariance matrix at some arbitrary points $\mathbf{Z}$, are now considered. Observe the following:

1. $\mathbf{Y_D} - \Lambda_{\mathbf{DP}}\Gamma_{pr} = \widetilde{\mathbf{Y}}_{\mathbf{D}}$, the new measurements adjusted by the prior mean
2. $[\Lambda_{\mathbf{DD}} - \Lambda_{\mathbf{DP}}\Psi_{pr}\Lambda_{\mathbf{PD}} + \Omega_{\mathbf{D}}]^{-1} = [\Phi^{pr}_{\mathbf{DD}} + \Omega_{\mathbf{D}}]^{-1}$
3. $\Lambda_{\mathbf{ZP}}[M - \Psi_{pr}\Lambda_{\mathbf{PD}}] = \Lambda_{\mathbf{ZP}}M - \Lambda_{\mathbf{ZP}}\Psi_{pr}\Lambda_{\mathbf{PD}} = \Phi^{pr}_{\mathbf{ZD}} + (\Lambda_{\mathbf{ZP}}M - \Lambda_{\mathbf{ZD}})$.

Hence, from the above identities we obtain,

$$\begin{aligned}
\boldsymbol{\mu}^{po}_{\mathbf{Z}} &= \Lambda_{\mathbf{ZP}}\Gamma_{po} \\
&= \boldsymbol{\mu}^{pr}_{\mathbf{Z}} + (\Phi^{pr}_{\mathbf{ZD}} + (\Lambda_{\mathbf{ZP}}M - \Lambda_{\mathbf{ZD}}))(\Phi^{pr}_{\mathbf{DD}} + \Omega_{\mathbf{D}})^{-1}\widetilde{\mathbf{Y}}_{\mathbf{D}} \\
&= \boldsymbol{\mu}^{pr}_{\mathbf{Z}} + \tilde{\Phi}^{pr}_{\mathbf{ZD}}(\Phi^{pr}_{\mathbf{DD}} + \Omega_{\mathbf{D}})^{-1}\widetilde{\mathbf{Y}}_{\mathbf{D}}
\end{aligned} \tag{6.12}$$

$$\begin{aligned}
\Phi^{po}_{\mathbf{ZZ}} &= \Lambda_{\mathbf{ZZ}} - \Lambda_{\mathbf{ZP}}\Psi_{po}\Lambda_{\mathbf{PZ}} \\
&= \Lambda_{\mathbf{ZZ}} - \Lambda_{\mathbf{ZP}}\Psi_{pr}\Lambda_{\mathbf{PZ}} \qquad (6.13) \\
&\quad -(\Phi^{pr}_{\mathbf{ZD}} + (\Lambda_{\mathbf{ZP}}M - \Lambda_{\mathbf{ZD}}))(\Phi^{pr}_{\mathbf{DD}} + \Omega_{\mathbf{D}})^{-1}(\Phi^{pr}_{\mathbf{DZ}} + (M^T\Lambda_{\mathbf{PZ}} - \Lambda_{\mathbf{DZ}})) \\
&= \Phi^{pr}_{\mathbf{ZZ}} - \tilde{\Phi}^{pr}_{\mathbf{ZD}}[\Phi^{pr}_{\mathbf{DD}} + \Omega_{\mathbf{D}}]^{-1}\tilde{\Phi}^{pr}_{\mathbf{DZ}}, \qquad (6.14)
\end{aligned}$$

where $\tilde{\Phi}^{pr}_{\mathbf{ZD}} = \Phi^{pr}_{\mathbf{ZD}} + (\Lambda_{\mathbf{ZP}}M - \Lambda_{\mathbf{ZD}})$. These formulas are of the same form as those we

obtained when discussing iterative GP regression in Section 4.6.3 (Equations 4.19 and 4.20). In fact, under certain conditions these two approaches to iterative GP regression can be show to be equivalent.

Theorem 3. *Sufficient-Subset GP iteration and naive-Bayesian GP iteration (and so by Theorem 1 also standard GP regression on all of the data simultaneously) are equivalent if both are started from the same initial prior and if the prior covariance function, k, results in covariance matrices such that for the matrices M defined by,*

$$\Lambda_{\mathbf{PD}} = \Lambda_{\mathbf{PP}} M,$$

at each stage of iteration, we have $\Lambda_{\mathbf{ZP}} M = \Lambda_{\mathbf{ZD}}$ for all $\mathbf{Z}$.

Proof. We will prove this via induction. Since any GP with a non-zero prior mean can be made into a zero prior mean GP by subtracting the mean from measured values; without loss of generality, we assume to have a zero initial prior mean. Hence we have the initial prior of mean zero and covariance function, k, for naive-Bayesian iteration and an initial prior tuple, $(k, \mathbf{P}, \mathbf{0}, \mathbf{0})$, for SSGP iteration. This forms our base case since the mean and covariance matrix between any points can easily be seen to coincide for these two types of GP. Now, assume that a number of iterations have been performed on identical datasets for the two methods and that they still result in equivalent GPs (same mean and covariance functions). The next SS iteration on a new dataset is as shown in Equations 6.12 and 6.14; since for all $\mathbf{Z}$ we have $\Lambda_{\mathbf{ZP}} M = \Lambda_{\mathbf{ZD}}$ it follows that $\tilde{\Phi}^{pr} = \Phi^{pr}$ and having assumed that the two GP methods match in the prior, Equations 6.12 and 6.14 for SS iteration are equivalent to those for naive-Bayesian iteration, Equations 4.19 and 4.20, resulting in equivalent posterior GPs. It follows by induction that this will be the case for any finite number of iterations. □

Corollary 1. *Naive-Bayesian Iteration and Sufficient Subset Iteration for GP regression are equivalent for the case of polynomial functions, and more generally for linear regression.*

Proof. As has been discussed in Section 5.1.1, polynomial functions lead to covariance

matrices of the form $\Lambda_{xy} = V_x \Delta V_y^T$. Hence, when solving for M in,

$$\Lambda_{\mathbf{PD}} = \Lambda_{\mathbf{PP}} M,$$

we have the solution $M = V_{\mathbf{P}}(V_{\mathbf{P}}^T V_{\mathbf{P}})^{-1} V_{\mathbf{D}}^T$ (not necessarily a unique solution). For M of this form it follows that for all $\mathbf{Z}$, $\Lambda_{\mathbf{ZP}} M = \Lambda_{\mathbf{ZD}}$ and so by the above theorem the equivalence has been shown for polynomials. In the case of general linear regression the covariance matrices have the same form, with the V matrices redefined for the correct set of basis functions, hence, the result follows similarly. □

Assuming $\Lambda_{\mathbf{PP}}$ is invertible[1] (for the sake of simplicity) the term $\Lambda_{\mathbf{ZP}} M$, with M defined as in Theorem 3, can be re-expressed as follows;

$$\begin{aligned} \Lambda_{\mathbf{ZP}} M &= \Lambda_{\mathbf{ZP}} (\Lambda_{\mathbf{PP}}^{-1} \Lambda_{\mathbf{PP}}) M \\ &= \Lambda_{\mathbf{ZP}} \Lambda_{\mathbf{PP}}^{-1} (\Lambda_{\mathbf{PP}} M) \\ &= \Lambda_{\mathbf{ZP}} \Lambda_{\mathbf{PP}}^{-1} \Lambda_{\mathbf{PD}}. \end{aligned}$$

Therefore, when investigating the degree to which the terms $\Lambda_{\mathbf{ZP}} M$ and $\Lambda_{\mathbf{ZD}}$ agree, rather than solving explicitly for M we can instead use the the term $\Lambda_{\mathbf{ZP}} \Lambda_{\mathbf{PP}}^{-1} \Lambda_{\mathbf{PD}}$ in place of $\Lambda_{\mathbf{ZP}} M$.

Lemma 4. *If either* $\mathbf{Z} \subset \mathbf{P}$ *or* $\mathbf{D} \subset \mathbf{P}$ *(or both), then* $\Lambda_{\mathbf{ZP}} \Lambda_{\mathbf{PP}}^{-1} \Lambda_{\mathbf{PD}} = \Lambda_{\mathbf{ZD}}$.

Proof. We show the result in the case $\mathbf{Z} \subset \mathbf{P}$, the other case follows via a symmetrical argument.

If $\mathbf{Z} \subset \mathbf{P}$ then $\Lambda_{\mathbf{ZP}} = J \Lambda_{\mathbf{PP}}$, where J is the matrix containing 1's and 0's which extracts the relevant rows in the correct order from $\Lambda_{\mathbf{PP}}$. Now,

$$\begin{aligned} \Lambda_{\mathbf{ZP}} \Lambda_{\mathbf{PP}}^{-1} \Lambda_{\mathbf{PD}} &= J \Lambda_{\mathbf{PP}} \Lambda_{\mathbf{PP}}^{-1} \Lambda_{\mathbf{PD}} \\ &= J \Lambda_{\mathbf{PD}} \\ &= \Lambda_{\mathbf{ZD}}. \end{aligned}$$

[1] This does not have to be the case, e.g. when considering a polynomial GP as in Corollary 1.

□

Lemma 4 therefore tells us that at points in $\mathbf{P}$ the relation required for SSGP to be equivalent to naive-Bayesian iterative GP regression (Theorem 3) holds. From this, along with the implicit smoothness assumptions for function predictions with a GP, we would also expect the approximations of $\Lambda_{\mathbf{ZD}}$ by $\Lambda_{\mathbf{ZP}}M$ to be good at points z which are 'close' to points in $\mathbf{P}$. All together, if points in $\mathbf{P}$ are equidistributed, this implies that we should come closer to satisfying the requirements of Theorem 3 as $\#\mathbf{P}$ increases. These claims are further explored in the following sections.

6.4.3 Determining the 'Sufficient-Subset'

The 'Sufficient-Subset' in SSGP is of course referring to our set of points $\mathbf{P}$. The word 'sufficient' is used loosely to imply that $\mathbf{P}$ has been chosen to be sufficient for the task at hand. The following discussion should help illuminate this choice of wording[2].

Assume that we are performing GP regression on some function, $f : W \to \mathbb{R}$, and are interested in its values taken on $U \subset W$. Given a covariance function, k, we have already discussed the fact that for a set of points $\mathbf{X} \subset U$ the covariance matrix generated by k, $\Lambda_{\mathbf{XX}}$, becomes close to singular as $\#\mathbf{X}$ increases. Assuming infinite precision we would ideally choose our sufficient subset, $\mathbf{P}$, to be a set $\mathbf{X}^\dagger$ such that $\Lambda_{\mathbf{X}^\dagger\mathbf{X}^\dagger}$ is non-singular but if $\mathbf{A} = [\ (\mathbf{X}^\dagger)^T \ \ z\ ]^T$ then $\Lambda_{\mathbf{AA}}$ is singular for any $z \in U$. In this case $\mathbf{P}$ is 'sufficient' in the sense that information and predictions at $\mathbf{P}$ determine exactly what is happening at all other points in U; as indicated by the fact that the matrix becoming singular means delta function are appearing in the pdf, as discussed in Sections 6.1 and 6.3.

When applying GPs, we have only finite computational precision, plus only in certain cases will we reach a stage where the addition of any one new point renders the covariance matrix singular. Therefore, rather than choosing $\mathbf{P}$ to be such that adding any other point gives us a singular matrix, we instead choose it such that the

[2]In general cases of sparse approximations the set $\mathbf{P}$ is the set of 'inducing points' discussed in Section 4.1. In the current work the name 'Sufficient-Subset' has been chosen in order to reflect the fact that matrix conditions numbers have indicated dimensionality reduction as being possible and appropriate, hence we wish this philosophy to be reflected in the nomenclature.

condition number of the resulting covariance matrix, $\Lambda_{\mathbf{PP}}$, is above some specified limit. We are thus choosing $\mathbf{P}$ to be sufficiently large that the covariance matrix is close to singular and hence such that information at the points $\mathbf{P}$ very closely determines what is happening at all other function values. Therefore, in this case the word 'sufficient' is referring to the fact that our choice of $\mathbf{P}$ allows for the information across the function to be determined from information at $\mathbf{P}$, up to some accuracy determined by the limit we choose for the condition number of $\Lambda_{\mathbf{PP}}$. For the sake of simplicity the points in $\mathbf{P}$ are always chosen to be equidistant from each other across our domain of interest, hence, it is only the number of points and not their location that we choose. This especially makes sense in the case of covariance functions with a constant length-scale.

It transpires that this choosing of $\mathbf{P}$ based on matrix condition number corresponds to how well the matrices $\Lambda_{\mathbf{ZP}}M(=\Lambda_{\mathbf{ZP}}\Lambda_{\mathbf{PP}}^{-1}\Lambda_{\mathbf{PD}})$ and $\Lambda_{\mathbf{ZD}}$ (see Theorem 3) agree, thus reconciling these two aspects of the SSGP theory. This is illustrated by way of examples. We consider the most commonly used covariance function, the 'squared exponential', for one dimensional inputs z,

$$k(z,z') = a\exp\left(-\frac{1}{2}\frac{(z-z')^2}{l}\right),$$

with lengthscale hyperparameter l (we use the lengthscale, l, here rather than the inverse-lengthscale as presented in Section 3.5, this makes the following discussion more intuitive). From the above formula it should be clear that a larger lengthscale results in covariances between input points decaying more slowly as the distance between the points increases. For a covariance function of this form with $(a,l) = (6.7, 6.0)$, we choose a domain of interest, $z \in [2,8]$, and for a given choice of $\#\mathbf{P}$ the points in $\mathbf{P}$ will be placed equidistantly throughout our domain. The similarities between the two matrices $\Lambda_{\mathbf{ZP}}\Lambda_{\mathbf{PP}}^{-1}\Lambda_{\mathbf{PD}}$ and $\Lambda_{\mathbf{ZD}}$ are considered by plotting contour maps of the 'matrix approximation error',

$$E(\mathbf{Z},\mathbf{D}) = \Lambda_{\mathbf{ZP}}\Lambda_{\mathbf{PP}}^{-1}\Lambda_{\mathbf{PD}} - \Lambda_{\mathbf{ZD}},$$

for the case that both $\mathbf{Z}$ and $\mathbf{D}$ contain 50 equidistributed points across the domain. Figures 6.1-6.4 show contour plots of these error matrices for different $\#\mathbf{P}$.

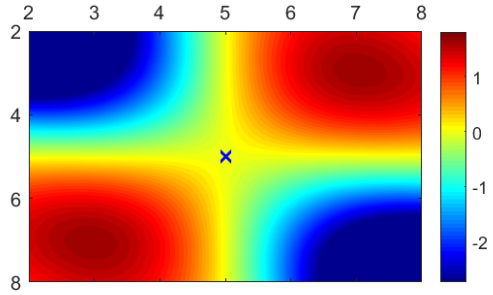

Figure 6.1: Error matrix contour plot with $\#\mathbf{P} = 1$ for points in the domain $[2, 8]$. Points in $\mathbf{P}$ are indicated by blue crosses.

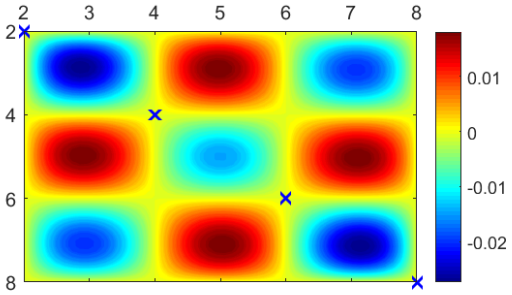

Figure 6.2: Error matrix contour plot with $\#\mathbf{P} = 4$ for points in the domain $[2, 8]$. Points in $\mathbf{P}$ are indicated by blue crosses.

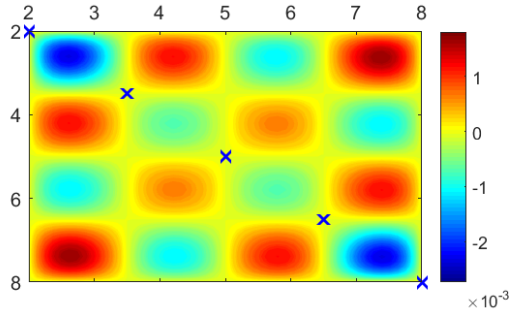

Figure 6.3: Error matrix contour plot with $\#\mathbf{P} = 5$ for points in the domain $[2, 8]$. Points in $\mathbf{P}$ are indicated by blue crosses. Note the scaling of 10^{-3}.

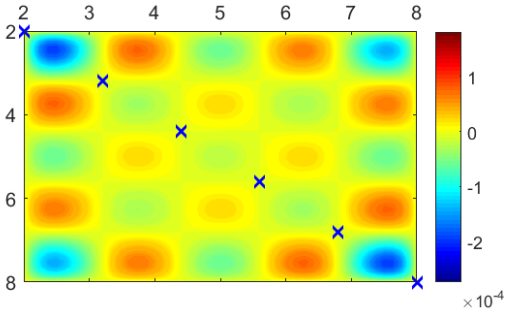

Figure 6.4: Error matrix contour plot with $\#\mathbf{P} = 6$ for points in the domain $[2, 8]$. Points in $\mathbf{P}$ are indicated by blue crosses. Note the scaling of 10^{-4}.

These error matrix plots demonstrate several key points. First, by noting the dramatic changes in scaling for each plot we see that the addition of new points in $\mathbf{P}$ very quickly reduces the magnitude of the deviations of between $\Lambda_{\mathbf{ZP}}M$ and $\Lambda_{\mathbf{ZD}}$. Second, we see that the error in our approximation goes to zero whenever considering covariances involving at least one point in $\mathbf{P}$, as was predicted by the analysis in Section 6.4.2. Finally, these plots indicate a very specific structure to these deviations, in the form of repeating patterns of over and under-estimated blocks. The presence of these patterns can be seen to stem from the re-expression of information in terms of its relation to points in $\mathbf{P}$, and is a direct consequence of having chosen a squared-exponential covariance function. We illustrate this with a one dimensional example: for $\mathbf{Z} = z$,

$\mathbf{D} = d$ and $\mathbf{P} = p$, all scalars, then;

$$\begin{aligned}\Lambda_{zp}\Lambda_{pp}^{-1}\Lambda_{pd} &= k(z,p)k(p,p)^{-1}k(p,d) \\ &= a\exp\left(-\frac{1}{2}\frac{(z-p)^2+(p-d)^2}{l}\right),\end{aligned}$$

and,

$$\begin{aligned}\Lambda_{zd} &= k(z,d) \\ &= a\exp\left(-\frac{1}{2}\frac{(z-d)^2}{l}\right) \\ &= a\exp\left(-\frac{1}{2}\frac{((z-p)+(p-d))^2}{l}\right) \\ &= a\exp\left(-\frac{1}{2}\frac{(z-p)^2+(p-d)^2+2(z-p)(p-d)}{l}\right)\end{aligned}$$

therefore, in the current case we have;

$$\begin{aligned}&\Lambda_{zp}\Lambda_{pp}^{-1}\Lambda_{pd} = \Lambda_{zd} && \text{if either } z = p,\ d = p \text{ or both} \\ &\Lambda_{zp}\Lambda_{pp}^{-1}\Lambda_{pd} \leq \Lambda_{zd} && \text{if } z > p \text{ and } d > p \textbf{ or if } z < p \text{ and } d < p \\ &\Lambda_{zp}\Lambda_{pp}^{-1}\Lambda_{pd} \geq \Lambda_{zd} && \text{if } z > p \text{ and } d < p \textbf{ or if } z < p \text{ and } d > p,\end{aligned}$$

and this is exactly what we see in Figure 6.1 with $\#\mathbf{P} = 1$. The other contour plots then have this same pattern repeating about the points in $\mathbf{P}$. It is also worth discussing the role played here by the length-scale hyperparameter. Both intuitively and from looking at the definition of the squared exponential covariance function, an increase in length-scale should decrease the errors in our covariance approximation for a given $\#\mathbf{P}$; in practice this is indeed seen to be the case as shown in Figure 6.5 which contains the contour plot for $\#\mathbf{P} = 1$ after the length-scale has been increased by a factor of 4; in contrast to Figure 6.1 the errors here can be seen to be significantly smaller.

We therefore draw the following two conclusions:

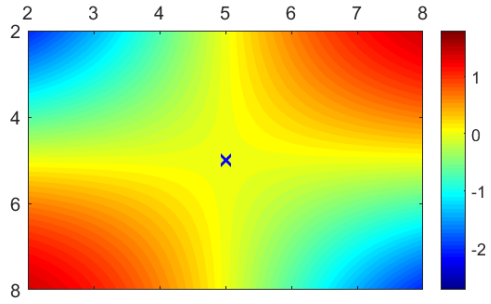

Figure 6.5: A longer lengthscale example with $\#\mathbf{P} = 1$.

1. As the number of points in $\mathbf{P}$ increases, the 'matrix approximation error' decreases

2. The magnitude of the 'matrix approximation error' for a given $\#\mathbf{P}$ is dependent on the length-scale of the covariance function (for the squared-exponential case).

It will therefore not be possible to choose a single $\#\mathbf{P}$ which is optimal for all cases. However, the problem of choosing $\#\mathbf{P}$ can be made very much simpler due to the fact that the condition number of $\Lambda_{\mathbf{PP}}$ can be used as a proxy for the matrix approximation error. As previously discussed, and due to the probabilistic context in which we are working, the ability of our SSGP to re-package information about the whole function (into data only at points in $\mathbf{P}$) while still allowing for accurate predictions across the whole domain of interest is intrinsically linked to the condition number of $\Lambda_{\mathbf{PP}}$. In any given case, as $\#\mathbf{P}$ increases, so will the condition number of $\Lambda_{\mathbf{PP}}$. Each term in the covariance matrix has the amplitude hyperparameter, a, multiplying it and therefore a will not be a factor when considering the 'matrix approximation error' or condition numbers[3]. Thus, it will only be the length-scale hyperparameter which must be considered here. It is shown above that for covariance functions with different length-scales, the resulting matrix approximation errors are of similar size when for each covariance function, $\#\mathbf{P}$ is chosen such that both covariance matrices have the same (or similar) condition numbers. This is illustrated in the following example: a new covariance function is used with the same a value as for the above figures, but the length-scale hyperparameter was changed to $l = 3.2$. We shall refer to the original covariance function as k_1 and the new one as k_2. In Figure 6.3 (with k_1) $\#\mathbf{P} = 5$ and the

[3]This was seen in our one dimensional example where two a terms cancel, leaving the third.

covariance matrix in this case, $\Lambda^1_{\mathbf{PP}}$, has condition number $\text{cond}(\Lambda^1_{\mathbf{PP}}) = 665$; Figure 6.6 (with k_2) shows the error contour plot with $\#\mathbf{P} = 6$ and the covariance matrix in this case, $\Lambda^2_{\mathbf{PP}}$, has condition number $\text{cond}(\Lambda^2_{\mathbf{PP}}) = 663$. These two examples therefore have different number of points in $\mathbf{P}$, but the condition numbers are very close and it can be seen that this results in very similar 'matrix approximation errors'.

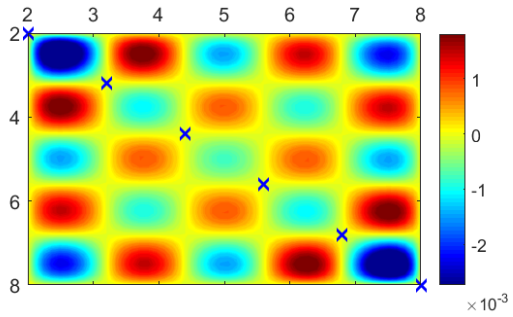

Figure 6.6: Error matrix contour plot with $\#\mathbf{P} = 6$ for points in the domain $[2, 8]$. The altered covariance function k_2 was used here. Note the scaling of 10^{-3} and the fact that this contour plot is of the same magnitude as those in Figure 6.3, despite having a different $\#\mathbf{P}$.

We will therefore choose $\#\mathbf{P}$ in any given case by simply setting a magnitude of condition number, and increasing the number of points in $\mathbf{P}$ until we find a number of points which passes this limit. Currently, the condition number limits used are around 10^7 to 10^{10}, depending on required accuracy. Limits in this range have been found to produce high accuracy results while also being low enough so as to allow the M matrices to be solved for without numerical accuracy complications. When a condition number limit of 10^7 is applied with the covariance function k_1, we end up with $\#\mathbf{P} = 8$ and $\text{cond}(\Lambda_{PP}) = 1.1 \times 10^7$, the resulting matrix-approximation-errors are of order of magnitude 10^{-6} or smaller.

Higher Dimensional Sufficient-Subsets

The results of the current section also hold when considering higher dimensional inputs, the exception being that the points in $\mathbf{P}$ become a grid of points in 2-dimensions and n-dimensional cubic lattices in n-dimensions. However, we must also account for the possibility that each dimension has a different inverse-lengthscale. Consider the case

of n-dimensions and assume we have our Sufficient-Subset lattice, $\mathbf{P}$, and have chosen some member of $\mathbf{P}$ which we denote $\mathbf{P_0}$. We then denote by $\mathbf{P_1}, \ldots, \mathbf{P_n}$ closest points to $\mathbf{P}_0$ in $\mathbf{P}$ along each input dimension and let L_i be the distance between $\mathbf{P_0}$ and $\mathbf{P_i}$. A 3-dimensional example is shown in Figure 6.7.

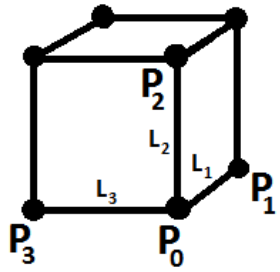

Figure 6.7: 3-dimensional Sufficient-Subset lattice example.

Letting k be a squared exponential covariance function in n-dimensions (see Equation 3.3) with inverse-lengthscales b_i, for $i = 1 \ldots n$, it should be clear that we wish to choose the distances L_i such that the covariance between adjacent members of $\mathbf{P}$ in the lattice have the same covariance along each input dimension. This will be true if we can satisfy:

$$k(\mathbf{P_0}, \mathbf{P_1}) = k(\mathbf{P_0}, \mathbf{P_j}) \qquad \text{for all } j = 2 \ldots n$$

$$\Leftrightarrow \qquad e^{b_1 L_1^2} = e^{b_j L_j^2} \qquad \text{for all } j = 2 \ldots n,$$

and hence,

$$L_j = \left(\sqrt{\frac{b_1}{b_j}} \right) L_1 \qquad \text{for all } j = 2 \ldots n.$$

Therefore, it is possible to express all lattice edge-lengths in terms of that along a single input dimension. When determining $\mathbf{P}$ in multiple dimensions it follows that one need only choose the spacing (or equivalently number of points) in $\mathbf{P}$ along one dimension, with the others being determined by the above proportionality expression.

6.4.4 Implementation and Speed of SSGP Iteration

Determining an Initial Prior

In the formulations for SSGP iteration of previous sections we have always assumed that we know the prior covariance function, k, of the SSGP tuple. In practice, k, will need to be determined and how this is done will depend on the specifics of the problem at hand; in general GP regression will not be applied to a single data-set 'blind', where by blind we mean without any access to existing data, similar cases or computer models which can generate data which approximates what we expect to obtain from measurements. This existing information should be utilised in order to determine k and doing so properly takes advantage of this prior information. This will most commonly come down to determining hyperparameters from existing data as happens in standard GP regression. If only some of the data is available due to temporal constraints then the currently available data can be used, or if a computational model is available then the prior can be determined using data generated by this model. Similarly, if different batches of data have different noise levels then a single batch can be used etc. In the case of a true one-shot regression on blind data then either all, or a subset of the data can be used to determine k; from which SSGP iteration allows for GP regression which accounts for the whole data set and can be further updated if more data becomes available.

In the context of the wind turbine dynamics identification problem, as discussed in Section 2.7, there is a lot of prior information from which the covariance structure k can be pre-determined and, as will be shown in the next chapter, it is in fact possible to determine a generic covariance structure which is suitable for application on any given wind turbine. This is especially important for this particular application since expensive optimisation procedures must be avoided at all costs. One optimisation which is unavoidable is that of the noise variance hyperparameter ξ for each batch, since noise values are likely to change in time. Having pre-determined k however, determining ξ via maximum likelihood can be done using a very simple root finding procedure as shown in the following section.

Although not required for the application considered in this thesis, it is possible

to formulate SSGP in such a way as to allow for on-line learning and updating of the covariance function k. To avoid disrupting the overall narrative, the outline of this more general procedure is deferred to Section 6.6.

Training for Noise Variance

In general the noise variance for each batch of data in SSGP iteration is not known, or changes. In these cases, noise hyperparameter training can easily be accommodated into the SSGP framework at each iterate. This is done via the standard maximum likelihood approach to find the most likely noise variance for each batch of data before it is processed through the SSGP algorithm, the other hyperparameters being those defined for the covariance function k in the SSGP tuple. Importantly, this means that SSGP iteration can perform regression on data which contains multiple or changing noise variances, simply by splitting the data up into batches, each containing a given noise level. This is of course assuming that it is possible to track when noise variance is likely to change, but this will be possible in a large number of practical cases, e.g. if measurement equipment noise depends on ambient temperature then temperature records could be used to batch the data into similar noise groupings. In other cases the noise might change across the domain of the input vector, in low dimensional cases this allows us to identify regions of different noise variances with respect to the input values and batch the data accordingly.

When noise training for each batch of data in SSGP regression, it is possible to reduce the maximum likelihood optimisation to a simple root-finding problem, this can be done as follows: Consider the covariance matrix for a batch of noisy data at input points $\mathbf{D}$,

$$\Lambda_{\mathbf{DD}} + \Omega_{\mathbf{D}}.$$

We pull out both the amplitude and noise variance hyperparameters, $a = e^{\phi_a}$ and $\xi = e^{\phi_\xi}$ from $\Lambda_{\mathbf{DD}}$ and $\Omega_{\mathbf{D}}$ giving,

$$\Lambda_{\mathbf{DD}} + \Omega_{\mathbf{D}} = e^{\phi_a}\bar{\Lambda}_{\mathbf{DD}} + e^{\phi_\xi}\bar{\Omega}_{\mathbf{D}},$$

where both $\bar{\Lambda}_{\mathbf{DD}}$ and $\bar{\Omega}_{\mathbf{D}}$ are defined in the obvious way, note that $\bar{\Omega}_{\mathbf{D}}$ will most commonly be the identity matrix, however, this analysis is performed in this more general setting which allows for more complicated, scaled noise structures. We now define a new log-hyperparameter β such that,

$$e^{\phi_\xi + \beta} = a, \tag{6.15}$$

and so we can express our covariance matrix as,

$$\Lambda_{\mathbf{DD}} + \Omega_{\mathbf{D}} = e^{\phi_\xi}(e^{\beta}\bar{\Lambda}_{\mathbf{DD}} + \bar{\Omega}_{\mathbf{D}}).$$

For the above covariance matrix (we'll call it $\hat{\Lambda}_{\mathbf{DD}}$), and with the given log-hyperparameters, the partial derivative of L^* (Equation 5.3) with respect to ϕ_ξ is,

$$\begin{aligned} \frac{\partial L^*}{\partial \phi_\xi} &= \text{trace}\left[\hat{\Lambda}_{\mathbf{DD}}^{-1}\hat{\Lambda}_{\mathbf{DD}}\right] - \mathbf{Y}_{\mathbf{D}}^T\hat{\Lambda}_{\mathbf{DD}}^{-1}\hat{\Lambda}_{\mathbf{DD}}\hat{\Lambda}_{\mathbf{DD}}^{-1}\mathbf{Y}_{\mathbf{D}} \\ &= d - \mathbf{Y}_{\mathbf{D}}^T\hat{\Lambda}_{\mathbf{DD}}^{-1}\mathbf{Y}_{\mathbf{D}}, \end{aligned}$$

where $d = \#\mathbf{D}$. In order to find a minima of L^* we wish to solve $\frac{\partial L^*}{\partial \phi_\xi} = 0$, and since we know the value of a from k in the prior SSGP tuple we also want Equation 6.15 to hold; hence we have the following pair of simultaneous equations:

1. $\mathbf{Y}_{\mathbf{D}}^T e^{-\phi_\xi}(e^{\beta}\bar{\Lambda}_{\mathbf{DD}} + \bar{\Omega}_{\mathbf{D}})^{-1}\mathbf{Y}_{\mathbf{D}} - d = 0$
2. $e^{\phi_\xi + \beta} = a,$

combining these we obtain a single equation,

$$\mathbf{Y}_{\mathbf{D}}^T(\Lambda_{\mathbf{DD}} + e^{\phi_\xi}\bar{\Omega}_{\mathbf{D}})^{-1}\mathbf{Y}_{\mathbf{D}} - d = 0 \tag{6.16}$$

in which ϕ_ξ is the only unknown. This can then be solved via root-finding for the function of ϕ_ξ on the left hand side. While in the above equation the noise variance, ξ, itself could be solved for; testing on various examples it has been found best to solve for ϕ_ξ, as above, since numerically this form behaves better in root finding algorithms.

In general, the noise will be assumed to be independent and identically distributed, giving $\bar{\Omega}_{\mathbf{D}} = I$. In this common case, further gains can be made in terms of computational speeds. Consider the function in Equation 6.16 for which we are root-finding, it will be denoted by $h(\phi_\xi)$, for the case $\bar{\Omega}_{\mathbf{D}} = I$,

$$h(\phi_\xi) = \mathbf{Y}_{\mathbf{D}}^T(\Lambda_{\mathbf{DD}} + e^{\phi_\xi} I)^{-1}\mathbf{Y}_{\mathbf{D}} - d.$$

The need to invert the interior matrix during each iteration when root-finding can be avoided as follows: Let USU^T be the singular-value decomposition (SVD) of $\Lambda_{\mathbf{DD}}$ then the function h and its derivative $\frac{dh}{d\phi_\xi}$ can be re-expressed as,

$$h(\phi_\xi) = \widehat{\mathbf{Y}}_{\mathbf{D}}^T \begin{bmatrix} \frac{1}{s_1+e^{\phi_\xi}} & 0 & \dots & 0 \\ 0 & \frac{1}{s_2+e^{\phi_\xi}} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{1}{s_d+e^{\phi_\xi}} \end{bmatrix} \widehat{\mathbf{Y}}_{\mathbf{D}} - d, \tag{6.17}$$

$$\frac{dh(\phi_\xi)}{d\phi_\xi} = -\widehat{\mathbf{Y}}_{\mathbf{D}}^T \begin{bmatrix} \frac{e^{\phi_\xi}}{(s_1+e^{\phi_\xi})^2} & 0 & \dots & 0 \\ 0 & \frac{e^{\phi_\xi}}{(s_2+e^{\phi_\xi})^2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{e^{\phi_\xi}}{(s_d+e^{\phi_\xi})^2} \end{bmatrix} \widehat{\mathbf{Y}}_{\mathbf{D}}, \tag{6.18}$$

where the s_i are the singular-values appearing along the diagonal of S and we define,

$$\widehat{\mathbf{Y}}_{\mathbf{D}} := U^T\mathbf{Y}_{\mathbf{D}}.$$

The above formulations allow h and $\frac{dh}{d\phi_\xi}$ to be evaluated during each iteration (of any given root-finding procedure) without the need for matrix inversion, instead only requiring a single SVD to be performed prior to iteration.

Reformulating the maximum likelihood procedure into a root finding problem improves algorithmic speed, but perhaps more importantly, it also massively reduces the processing power and computational machinery required to perform regression. A full optimisation procedure with derivative and Hessian information requires fairly com-

plicated software and not inconsiderable processing power, however, the above root finding problem can be solved using very lightweight approaches, such as Newton's method, which can be performed on much less powerful devices, such as PL controllers. This opens up the possibility, for example, for SSGP regression to be embedded within existing real-time systems without the need to introduce additional processing power.

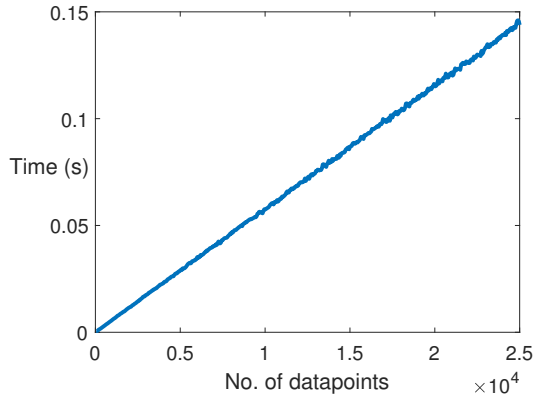

Figure 6.8: Execution times for SSGP implementation with no noise training for up to 25,000 data points. Note the linear scaling.

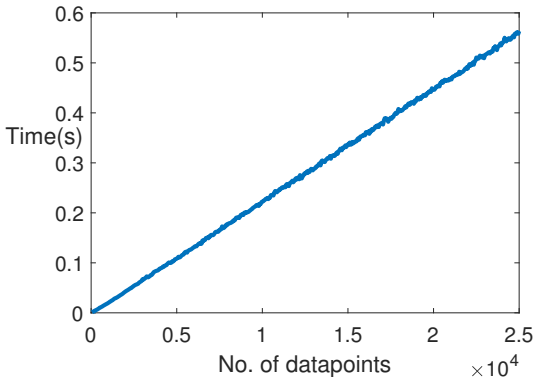

Figure 6.9: Execution times for SSGP implementation with noise training for up to 25,000 data points. Note the linear scaling.

Optimal Batch Sizes

Consider the case of SSGP regression for an underlying function, f. Given noisy measurements of f, $\mathbf{Y_D}$, at input points $\mathbf{D}$ we wish to determine how best to implement the above formulations. Since SSGP regression can be implemented iteratively we must decide how much of the data $\mathbf{Y_D}$ to process at each stage. Consider the numerical complexity of SSGP updates: from Equations 6.8 and 6.9 it follows that the SSGP formulas scale as $O(m^3)$ where m is the number of data points being used to update the current SSGP iteration (we will refer to this as the *batch number*). For $n = \#D$ there will need to be $\frac{n}{m}$ iterations in order to have processed all of the data $\mathbf{Y_D}$. Thus, the complexity for SSGP iteration is $O(nm^2)$. Since m is a constant which will be chosen and then fixed (this is discussed further below) it is in fact the case that the numerical complexity of SSGP regression is $O(n)$, i.e. it is linear in the total number of measured data points.

Figures 6.8 and 6.9 show explicitly this linear scaling for SSGP regression in the case where noise variance is known and when it is trained for during each iterate (via the root finding implementation of maximum likelihood discussed previously) respectively; these examples had batch sizes of $m = 100$. This linear scaling is in stark contrast to the usual cubic scaling of GP algorithms.

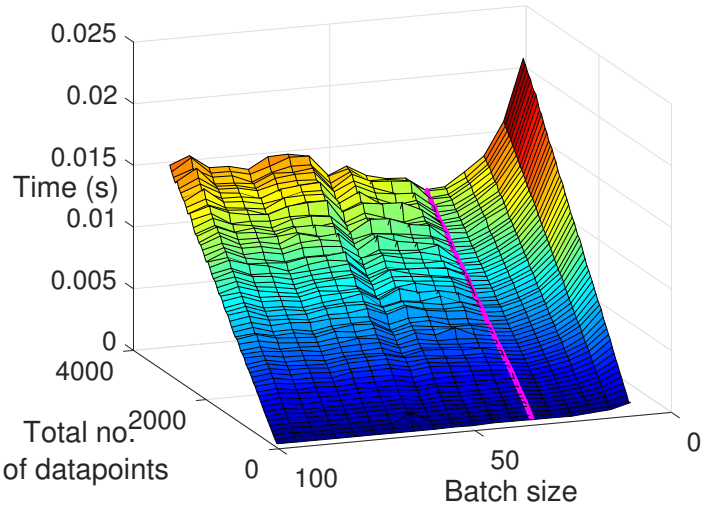

Figure 6.10: SSGP training times for different batch sizes and numbers of datapoints. In this case the noise level is known *a priori*.

In order to determine the optimal value of m it is instructive to consider the $O(nm^2)$ scaling present before we have fixed a value for m. This indicates that minimising complexity equates to minimising the value of m, however, we must be wary of the assumptions which were implicitly made when determining these complexities; one of these assumptions being that $m > \#\mathbf{P}$. Therefore, we should minimise m but be careful once it reaches smaller values. Numerical tests around small values of m easily determine the point after which decreasing m further starts to increase computational times again. The results of such tests for the current implementation of SSGP iteration are shown in Figures 6.10 and 6.11 for $\#\mathbf{P} = 10$; these results indicate that batch sizes of $m = 35$ (when noise variance is known) and $m = 40$ (when noise variance is trained for at each iteration) are about optimal.

The optimal batch size, from the point of view of computational times, will change

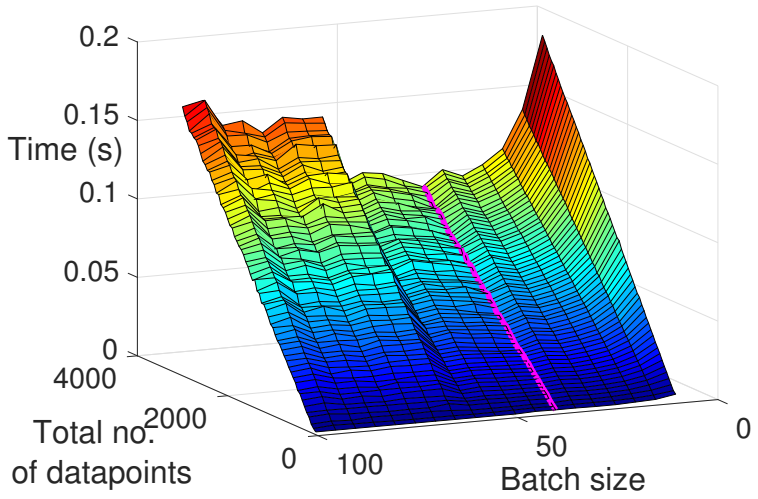

Figure 6.11: SSGP training times for different batch sizes and numbers of datapoints. In this case the noise level is trained for on each batch.

with $\#\mathbf{P}$. The following gives a useful rule of thumb for estimating optimal m: in the SSGP formulae the complexity for each update is $\mathcal{O}(m^3)$ assuming that $m > p$ as previosuly discussed, where $p = \#\mathbf{P}$. The next most costly operations in SSGP have a complexity $4m^2p$ and the assumptions used when determining complexity break down when $4m^2p = m^3$, *i.e.* when $m = 4p$. We then look to find an optimal batch size by minimising m up until the point where our assumptions break down, and so the above analysis suggests setting $m = 4\#\mathbf{P}$; in the previous example this suggests an optimal value of $m = 40$, which from the results in Figures 6.10 and 6.11 can be seen be a good estimate.

While the current section has focussed on computational speed, it is worth noting that in the case of noise training one must also choose batch sizes which are large enough so as to allow for effective determination of the noise variance. This will very much depend on the specifics of each given regression problem but generally one would have more points than is suggested by optimising for speed alone. The analysis of this section does however give a useful lower limit in these cases.

6.4.5 SSGP Regression Example

Examples of SSGP iterative regression will now be given for a sinusoidal function, again using a squared exponential covariance function. $\#\mathbf{P} = 10$ and the points in $\mathbf{P}$ are equidistributed across the domain. First, amplitude and lengthscale hyperparameters are determined using 200 data points, this is simulating the determination of hyperparameters from currently available data. Next, SSGP iteration is performed 500 times with 30 noisy datapoints generated at each iterate for a total of 15,000 datapoints; the noise standard deviation for each 30 point dataset being drawn uniformly from between 0.2 and 0.8. Figure 6.12 shows the true function along with a batch of noisy data and function predictions from the first SSGP iterate, as well as the function predictions from the final SSGP iterate. Figure 6.13 shows the first SSGP function predictions and confidence intervals as well as the final SSGP predictions and confidence intervals. Figures 6.14 and 6.15 show the root-mean-squared-error (RMSE) across the function at each iterate and a histogram of the error standard deviations from the 500 datasets. The condition number of $\Lambda_{\mathbf{PP}}$ with $\#\mathbf{P} = 10$ is around 7×10^7. These results demonstrate that SSGP iteration can refine GP predictions effectively when data is batched and with varying noise levels. The SSGP implementation is also fast, in this example the entire iterative process (after determining the initial prior) on 15,000 data points (and including the time taken to generate all these datapoints) takes less than 1s on a desktop computer.

6.4.6 Comparative Performance of SSGP and Standard GP regression

SSGPs were originally developed for iterative GP regression in cases where a standard GP becomes impractical, such as when new data becomes available at given time intervals or when different batches of data have different noise levels. However, as was seen in the previous section SSGP iteration is also very fast; this opens up the possibility of using SSGP iteration as a fast alternative to standard GP regression on large datasets by splitting the data into batches and then performing SSGP iteration on the batched data. Therefore, to validate SSGP regression as a method for iterative regression, both in the case of genuinely batched data and as a fast alternative for large datasets, it is

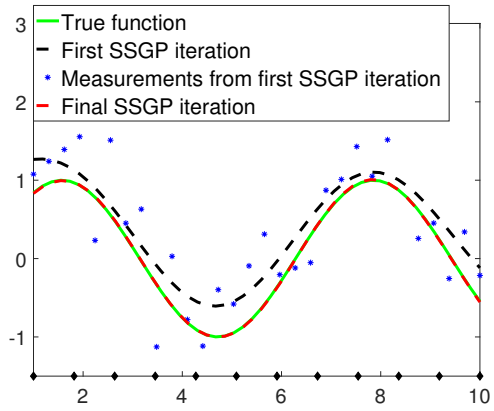

Figure 6.12: SSGP regression example showing the true function, first iterate and last iterate and the first batch of data. Points in $\mathbf{P}$ are shown on the x-axis.

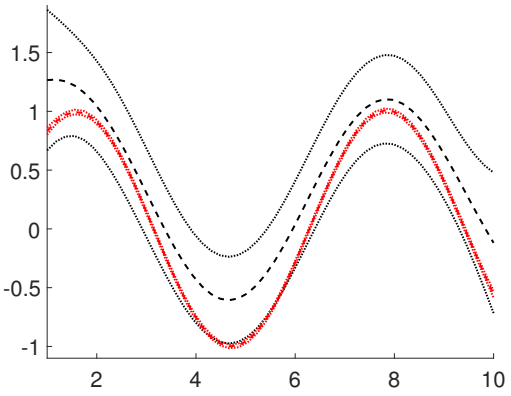

Figure 6.13: First (black) and last (red) SSGP regression predictions and confidence intervals.

necessary to compare the performance of SSGP with standard GP regression in cases where they can both be used.

SSGP regression has been shown to be able to handle much larger quantities of data than the standard GP implementations, for these direct comparisons however we shall restrict the quantities of data to that which both approaches can process. In each case the covariance function is a squared exponential. The standard GP approach used optimised code developed in [29] with hyperparameter optimisation which includes derivative information.

There are two test cases, both using squared-exponential covariance functions. Case 1 will consider the function,

$$f(x) = \sin(x) + \frac{x}{2},$$

and Case 2 the two dimensional function,

$$g(x, y) = \sin(x) + \frac{x}{2} + \sin(1.5y).$$

Figure 6.16 shows f and a sample of noisy data, $\#\mathbf{P} = 11$. Figure 6.17 show the same for g along with the Sufficient-Subset; note that the density of points in $\mathbf{P}$ is different in the x and y directions due to the different lengthscales, as discussed in Section 6.4.3,

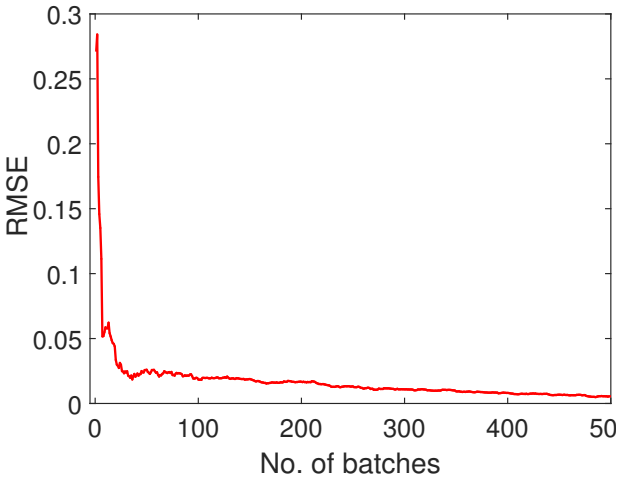

Figure 6.14: RMSE of SSGP regression at each iterate.

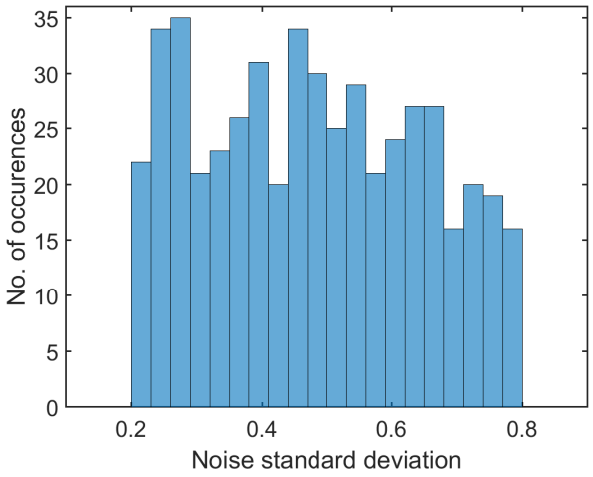

Figure 6.15: Histogram of the noise standard deviations present across the 500 datasets used in SSGP example.

$\#\mathbf{P} = 40$. Gaussian noise of standard deviation 0.2 is used throughout. The two test cases look to compare standard GP and SSGP in terms of both speed and accuracy when they have the same covariance structure. Prior information in order to generate a covariance structure k is simulated by optimisation on 500 noisy datapoints. k is then used for both standard GP and SSGP regression, with the noise hyperparameter being determined by root-finding in each case. RMSEs for both methods across the functional domains, as well as execution times, were recorded for input data ranging from 200 to 2000 datapoints. The results from this are shown in Figures 6.18-6.21 for the two cases. In both cases there are no significant differences in terms of RMSE, but SSGP can be seen to be much faster and with superior scaling.

6.5 Super Sufficient-Subsets

In the development of SSGPs so far it has generally been assumed that the Sufficient-Subset $\mathbf{P}$ is chosen such that the covariance matrix $\Lambda_{\mathbf{PP}}$ is invertible. This is also assumed to be the case in the other existing methods for batched GPs discussed in Chapter 4. When determining $\mathbf{P}$ it was shown, in Section 6.4.3, that one would ideally choose $\mathbf{P}$ such that $\Lambda_{\mathbf{PP}}$ is invertible, while the addition of any one extra points

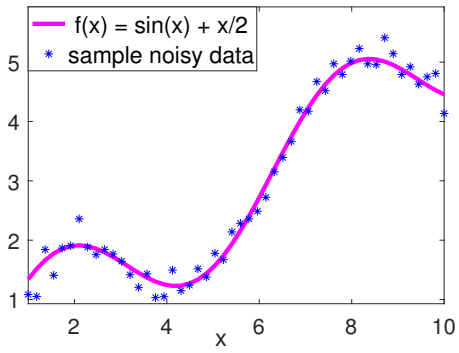

Figure 6.16: The function used in test Case 1 along with a sample of noisy data.

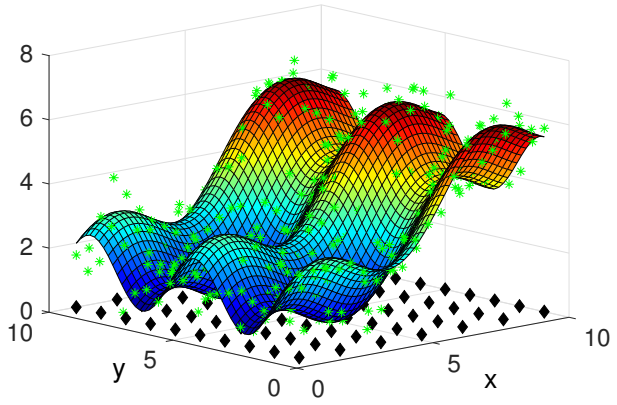

Figure 6.17: The function used in test Case 2 along with a sample of noisy data (green) and Sufficient-Subset (black).

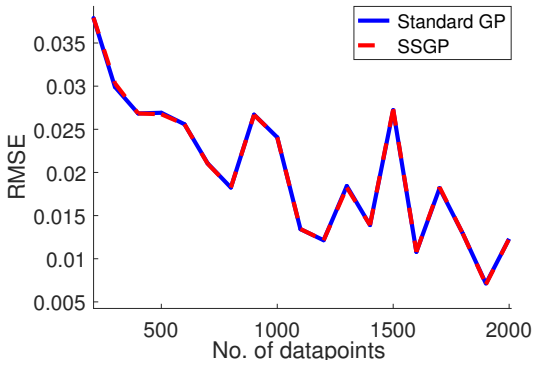

Figure 6.18: RMSEs for both SSGP and standard GP regression across different sizes of dataset in test Case 1.

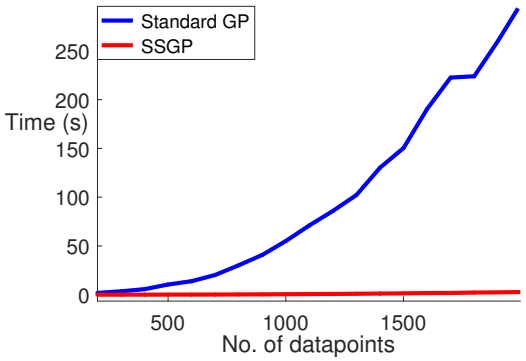

Figure 6.19: Execution times for both SSGP and standard GP regression across different sizes of dataset in test Case 1.

renders it singular. This being the point after which all of the information about the underlying GP model can be recreated from information at points in $\mathbf{P}$. Hence, the GP covariance structure determines the number of points necessary to allow for an accurate representation of the functions generated by this GP. When performing SSGP iteration it is therefore implicitly assumed that the measured data has been generated by the GP prior, with its given covariance structure. However, in practice when applying SSGPs to measured data this is no longer the case; a covariance structure is determined which is hopefully appropriate for the underlying function, but locally the

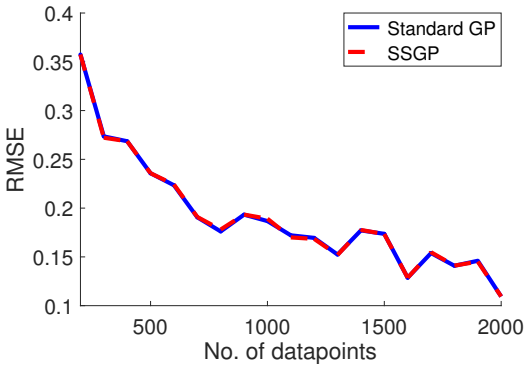

Figure 6.20: RMSEs for both SSGP and standard GP regression across different sizes of dataset in test Case 2.

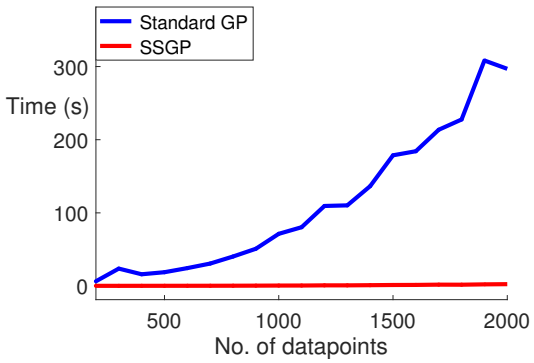

Figure 6.21: Execution times for both SSGP and standard GP regression across different sizes of dataset in test Case 2.

function may have features which vary faster than indicated by the global covariance structure. There is therefore the possibility that a given set, $\mathbf{P}$, results in a condition number of the required magnitude, while still being too sparse on some regions of the particular function[4]. Ideally one would therefore like to add more points to $\mathbf{P}$ whenever this was the case, however, this will render the matrix $\Lambda_{\mathbf{PP}}$ singular (either actually or numerically) and hence the matrix M can no longer be determined by matrix inversion. The following analysis tackles this problem, allowing additional points to be added to $\mathbf{P}$ past the point where $\Lambda_{\mathbf{PP}}$ becomes singular. We start with some definitions:

Definition 5. *A Sufficient-Subset is* **minimal** *if its covariance matrix is non-singular but the addition of any other point renders it singular.* $\mathbf{P}_0$ *will be used to denote a minimal Sufficient-Subset.*

Definition 6. *A set of points* $\mathbf{P}$ *is a* **Super Sufficient-Subset** *if it strictly contains a minimal Sufficient-Subset, i.e. if* $\exists \mathbf{P}_0$ *s.t.* $\mathbf{P}_0 \subsetneq \mathbf{P}$.

It was shown in Section 6.3 that for a minimal Sufficient-Subset, $\mathbf{P}_0$, and for sets of points $\mathbf{Z}$ and $\mathbf{Z}^*$, all covariance matrices are of the form,

$$\Lambda_{\mathbf{ZZ}^*} = V_{\mathbf{Z}} \Lambda_{\mathbf{P}_0\mathbf{P}_0} V_{\mathbf{Z}^*}^T.$$

[4]This scenario is relevant to the current work since it has been found to occur in real data and, in particular, for the wind turbine aerodynamic functions considered in this thesis

This followed by showing that once $\mathbf{P}_0$ has been found, the function values at additional points, z say, can be rewritten as,

$$f_z = -\frac{\mathbf{v}_z^T}{u_z}\mathbf{F}_{\mathbf{P}_0},$$

for some $\mathbf{v}_z$ and u_z. Covariance matrix entries were then determined by taking the expectation of products, $\mathbb{E}(f_z f_{z^*})$ (technically the current formulae are for zero prior mean cases, however, non-zero priors follow as above by subtracting the mean from measured data.).

We now consider a Super Sufficient-Subset $\mathbf{P}$ (so $\mathbf{P}_0 \subsetneq \mathbf{P}$).

Lemma 5. For any set of points $\mathbf{Z}^*$,

$$\Lambda_{\mathbf{PZ}^*} = \Lambda_{\mathbf{PP}}\widetilde{V}_{\mathbf{Z}^*}^T,$$

for some matrix $\widetilde{V}_{\mathbf{Z}^*}$.

Proof.

$$\begin{aligned}
\Lambda_{\mathbf{P}z^*} &= \mathbb{E}(\mathbf{F}_{\mathbf{P}} f_{z^*}) \\
&= \mathbb{E}(\mathbf{F}_{\mathbf{P}} \left(\mathbf{F}_{\mathbf{P}_0}^T \frac{\mathbf{v}_z}{-u_z} \right)) \\
&= \Lambda_{\mathbf{PP}_0} \frac{\mathbf{v}_z}{-u_z} \\
&= \Lambda_{\mathbf{PP}} \left(J \frac{\mathbf{v}_z}{-u_z} \right),
\end{aligned}$$

where J is the matrix of 1's and 0's which extracts the relevant columns from $\Lambda_{\mathbf{PP}}$, since $\mathbf{P}_0 \subset \mathbf{P}$. Thus the result follows by applying the above to each element of $\mathbf{Z}^*$. □

This result will allow for the matrix M to be determined during each iterate. Recall that for new data $\mathbf{D}$, M is required such that;

$$\Lambda_{\mathbf{PP}} M = \Lambda_{\mathbf{PD}}.$$

Since $\mathbf{P}$ is Super Sufficient, $\Lambda_{\mathbf{PP}}$ is singular and, being symmetric, therefore decomposes

into [57],

$$\begin{aligned}\Lambda_{\mathbf{PP}} &= UDU^T \\ &= \begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \end{bmatrix} \begin{bmatrix} D_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_{11}^T & u_{21}^T \\ u_{12}^T & u_{22}^T \end{bmatrix},\end{aligned}$$

with U orthogonal and D_1 diagonal and positive definite. Partitioning $\Lambda_{\mathbf{PD}}$ similarly, $\Lambda_{\mathbf{PD}} = \begin{bmatrix} (\Lambda_{\mathbf{PD}})_1 \\ (\Lambda_{\mathbf{PD}})_2 \end{bmatrix}$, it follows that;

$$\begin{aligned} \Lambda_{\mathbf{PP}} M &= \Lambda_{\mathbf{PD}} \\ \iff UDU^T M &= \Lambda_{\mathbf{PD}} \\ \iff DU^T M &= U^T \Lambda_{\mathbf{PD}} \\ \iff \begin{bmatrix} D_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_{11}^T & u_{21}^T \\ u_{12}^T & u_{22}^T \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \end{bmatrix} &= \begin{bmatrix} u_{11}^T & u_{21}^T \\ u_{12}^T & u_{22}^T \end{bmatrix} \begin{bmatrix} (\Lambda_{\mathbf{PD}})_1 \\ (\Lambda_{\mathbf{PD}})_2 \end{bmatrix} \\ \iff \begin{bmatrix} D_1 u_{11}^T M_1 + D_1 u_{21}^T M_2 \\ 0 \end{bmatrix} &= \begin{bmatrix} u_{11}^T (\Lambda_{\mathbf{PD}})_1 + u_{21}^T (\Lambda_{\mathbf{PD}})_2 \\ u_{12}^T (\Lambda_{\mathbf{PD}})_1 + u_{22}^T (\Lambda_{\mathbf{PD}})_2 \end{bmatrix}. \end{aligned}$$

Hence, for a solution to exist we require that $u_{12}^T (\Lambda_{\mathbf{PD}})_1 + u_{22}^T (\Lambda_{\mathbf{PD}})_2 = 0$. This is indeed the case since,

$$\begin{aligned} u_{12}^T (\Lambda_{\mathbf{PD}})_1 + u_{22}^T (\Lambda_{\mathbf{PD}})_2 &= \begin{bmatrix} u_{12} \\ u_{22} \end{bmatrix}^T \Lambda_{\mathbf{PD}} \\ \text{(from Lemma 5)} \quad &= \begin{bmatrix} u_{12} \\ u_{22} \end{bmatrix}^T \Lambda_{\mathbf{PP}} \widetilde{V}_{\mathbf{D}}^T \\ &= \left(\Lambda_{\mathbf{PP}} \begin{bmatrix} u_{12} \\ u_{22} \end{bmatrix} \right)^T \widetilde{V}_{\mathbf{D}}^T \\ &= 0. \end{aligned}$$

The final equality holds since the columns of $\begin{bmatrix} u_{12} \\ u_{22} \end{bmatrix}$ constitute a basis for the null-space of $\Lambda_{\mathbf{PP}}$. Having shown that the necessary condition of the above expression being zero

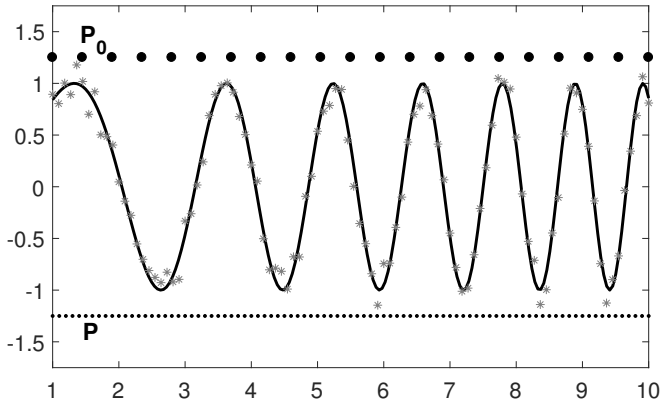

Figure 6.22: The function $\sin(x^{1.6})$ along with a sample of noisy data and Sufficient-Subsets.

is met, determining an appropriate matrix M now only requires a solution to;

$$\begin{aligned} D_1 u_{11}^T M_1 + D_1 u_{21}^T M_2 &= u_{11}^T (\Lambda_{\mathbf{PD}})_1 + u_{21}^T (\Lambda_{\mathbf{PD}})_2 \\ \iff D_1 \begin{bmatrix} u_{11} \\ u_{21} \end{bmatrix}^T M &= \begin{bmatrix} u_{11} \\ u_{21} \end{bmatrix}^T \Lambda_{\mathbf{PD}}, \end{aligned}$$

the most straightforward solution to which is,

$$M = \begin{bmatrix} u_{11} \\ u_{21} \end{bmatrix} D_1^{-1} \begin{bmatrix} u_{11} \\ u_{21} \end{bmatrix}^T \Lambda_{\mathbf{PD}}. \tag{6.19}$$

Note, the only inversion required here is of D_1 which will be chosen such that it is invertible, hence this solution will always be well defined. Equation 6.19 therefore allows us to use a Super Sufficient-Subset without singularity causing an issue. Furthermore, in practise it is not necessary to exactly determine $\mathbf{P}_0$, instead when a need for more points in $\mathbf{P}$ is observed one simply increases the number of points as required while keeping the same limit on the condition number of D_1 one would apply to $\Lambda_{\mathbf{P}_0\mathbf{P}_0}$.

The need for Super Sufficient-Subsets in the turbine aerodynamics case will be discussed further in Chapter 7. For now, the error reduction potential of using a Super Sufficient-Subset is demonstrated for the function $\sin(x^{1.6})$, shown in Figure 6.22. This function was chosen since it would not be generated by a GP prior due to the lengthscale

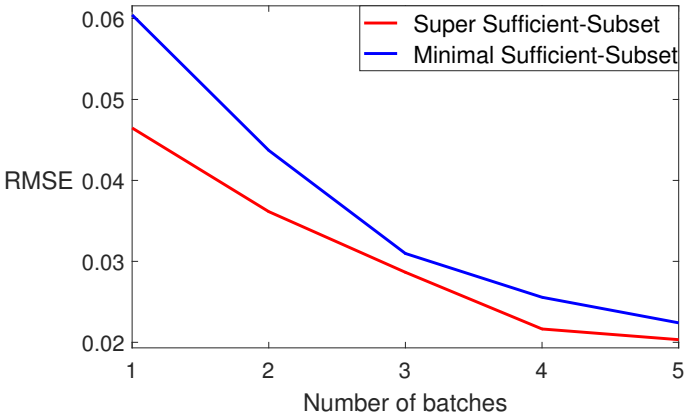

Figure 6.23: RMSE error from SSGP predictions using a minimal or Super Sufficient-Subset.

being non-constant across the function, as can be seen in the figure. Since, even in this case, applying a Sufficient-Subset with maximum condition number still results in good predictions, we choose a lower limit in order to demonstrate on this toy example what has been seen for real data. Our 'minimal' Sufficient-Subset, $\mathbf{P_0}$, for this example was therefore chosen with condition number 1.8×10^3 and is shown in Figure 6.22. A Super Sufficient-Subset, $\mathbf{P}$, was also generated which contains 4.5 times the number of points in $\mathbf{P_0}$, again shown in Figure 6.22. A condition number limit of 1.8×10^3 is placed on the D_1 matrix of M in this case. Figure 6.23 shows the RMSE results from performing SSGP iteration on batches of 100 noisy data points when using $\mathbf{P_0}$ and $\mathbf{P}$ respectively. These results clearly show that, although the same limit is applied to the condition number of the inverted term in M, the use of a Super Sufficient-Subset results in lower prediction errors for this example.

6.6 Extension of SSGPs to Include Online Hyperparameter Learning

While the application for which SSGPs have been developed requires that online optimisations of hyperparameter values is avoided, it is in fact still possible within the SSGP framework. Since this is technically out of scope for this thesis, an outline only

is given here as to how this may be done. In future work these more general SSGP implementations will be investigated fully.

Consider an SSGP model $(k, \mathbf{P}, \Gamma, \Psi)$. Recall from Section 6.4.1 that Γ and Ψ are the propagated forms of the 'normalised' pseudo-data and precision matrix on $\mathbf{P}$, $(\Lambda_{\mathbf{PP}}+\Omega_{\mathbf{P}})^{-1}\mathbf{Y_P}$ and $(\Lambda_{\mathbf{PP}}+\Omega_{\mathbf{P}})^{-1}$ respectively. We start by assuming we have direct access to the covariance matrix, $(\Lambda_{\mathbf{PP}} + \Omega_{\mathbf{P}})$, and pseudo data, $\mathbf{Y_P}$. Given new data, $\mathbf{Y_D}$, we denote the combined input and measurement vectors by,

$$\mathbf{Z} = \begin{bmatrix} \mathbf{P} \\ \mathbf{D} \end{bmatrix},$$

and,

$$\mathbf{Y_Z} = \begin{bmatrix} \mathbf{Y_P} \\ \mathbf{Y_D} \end{bmatrix},$$

respectively. The covariance matrix for $\mathbf{Y_Z}$ is,

$$\Lambda_{\mathbf{ZZ}} + \Omega_{\mathbf{Z}} = \begin{bmatrix} \Lambda_{\mathbf{PP}} + \Omega_{\mathbf{P}} & \Lambda_{\mathbf{PD}} \\ \Lambda_{\mathbf{DP}} & \Lambda_{\mathbf{DD}} + \Omega_{\mathbf{D}} \end{bmatrix},$$

and so the negative log-likelihood function (with constants and scaling removed) for $\mathbf{Y_Z}$ is;

$$nLL^* = \mathbf{Y_Z}^T(\Lambda_{\mathbf{ZZ}} + \Omega_{\mathbf{Z}})^{-1}\mathbf{Y_Z} + \log\det(\Lambda_{\mathbf{ZZ}} + \Omega_{\mathbf{Z}}). \tag{6.20}$$

From an application of the block matrix inversion identity (Equation 6.4) it follows that evaluation nLL^* only requires access to Γ and Ψ from the SSGP model (except for a constant term which can be ignored for the purposes of minimisation), as shown below for the relevant terms:

$$(\Lambda_{\mathbf{ZZ}} + \Omega_{\mathbf{Z}})^{-1} = \begin{bmatrix} \Psi - \Psi\Lambda_{\mathbf{PD}}\eta\Lambda_{\mathbf{DP}}\Psi & -\Psi\Lambda_{\mathbf{PD}}\eta \\ -\eta\Lambda_{\mathbf{DP}}\Psi & \eta \end{bmatrix},$$

with $\eta = (\Lambda_{\mathbf{DD}} - \Lambda_{\mathbf{DP}}\Psi\Lambda_{\mathbf{PD}} + \Omega_{\mathbf{D}})^{-1}$. Hence,

$$\begin{aligned}\mathbf{Y_Z}^T(\Lambda_{\mathbf{ZZ}} + \Omega_{\mathbf{Z}})^{-1}\mathbf{Y_Z} &= \mathbf{Y_P}^T\Gamma - \Gamma^T\Lambda_{\mathbf{PD}}\eta\Lambda_{\mathbf{DP}}\Gamma - \mathbf{Y_D}^T\eta\Lambda_{\mathbf{DP}}\Gamma - \Gamma^T\Lambda_{\mathbf{PD}}\eta\mathbf{Y_D} + \mathbf{Y_D}^T\eta\mathbf{Y_D} \\ &= \mathbf{Y_P}^T\Gamma - (\mathbf{Y_D} + \Lambda_{\mathbf{DP}}\Gamma)^T\eta\Lambda_{\mathbf{DP}}\Gamma + (\mathbf{Y_D} - \Lambda_{\mathbf{DP}}\Gamma)^T\eta\mathbf{Y_D}.\end{aligned}$$

Similarly for the determinant,

$$\begin{aligned}\log\det(\Lambda_{\mathbf{ZZ}} + \Omega_{\mathbf{Z}}) &= \log(\det(\Lambda_{\mathbf{PP}} + \Omega_{\mathbf{P}})\det(\eta^{-1})) \\ &= \log\left(\frac{1}{\det\Psi}\det(\eta^{-1})\right) \\ &= -\log\det(\Psi) + \log\det(\Lambda_{\mathbf{DD}} - \Lambda_{\mathbf{DP}}\Psi\Lambda_{\mathbf{PD}} + \Omega_{\mathbf{D}}).\end{aligned}$$

Note that the terms $\mathbf{Y_P}\Gamma$ and $-\log\det(\Psi)$ are fixed by the current SSGP model and so are constants which can be dropped from any minimisation. It is therefore possible to allow for the covariance functions which define the $\Lambda_{(\cdot,\cdot)}$ and $\Omega_{\mathbf{D}}$ matrices (these being k and a noise term) to vary with respect to their set of hyperparameters, θ, and then to perform likelihood maximisation in order to update these hyperparameters in light of new data. These updated hyperparameters can then be used when refining the SSGP model, essentially adjusting the SSGP covariance function, k. Since the Sufficient-Subset, $\mathbf{P}$, for an SSGP model is determined by matrix condition numbers, which in turn are determined by hyperparameter values, we briefly discuss how this aspect of SSGP iteration might be adjusted in order to accommodate the updating of hyperparameters.

The simplest case to consider is if we are using a Super-Sufficient Subset. In this case $\mathbf{P}$ has already been chosen to be a set of points which contains more input locations than is necessary to render $\Lambda_{\mathbf{PP}}$ singular, where here $\Lambda_{\mathbf{PP}}$ is that determined by k, rather than the SSGP model. While updating of hyperparameters will alter the covariance structure, it seems unlikely that the initial estimate (from computer models or the first batch of data) would be so different as to render $\Lambda_{\mathbf{PP}}$ no longer singular. Assuming then that $\Lambda_{\mathbf{PP}}$ remains singular, the new hyperparameter values can be used without altering $\mathbf{P}$.

If, on the other hand, $\mathbf{P}$ is required to be minimal, then a change in hyperparameter

values might result in the condition number of $\Lambda_{\mathbf{PP}}$ falling below its specified limit. In this case a new Sufficient-Subset, $\mathbf{P}_{\text{new}}$, can be defined (following the standard procedure) which regains this required limit for the new hyperparameter values. The most recent data, along with the current SSGP model, can then be used to define a new SSGP model on $\mathbf{P}_{\text{new}}$. This can be done by using the same procedure that generates the initial SSGP model in Section 6.4.1. Note that the identities used here for evaluating nLL^* are also required in order to determine the new SSGP model on $\mathbf{P}_{\text{new}}$.

6.7 Relationship to Recursive Gaussian Processes

As was mentioned in Section 4.4, another iterative approach to GPs is given in [47] under the title of Recursive Gaussian processes. In [47] two approaches to iterative GPs are described. The first assuming that hyperparameters are known and the second updating hyperparameters at each iterate via a sigma point method. Note that in [47] it is always assumed that the set of inducing points, $\mathbf{m}$ in Section 4.4 or $\mathbf{P}$ in SSGP notation, is known and remains fixed throughout for both methods, although there is no discussion as to how or why any particular set should be chosen. As Huber himself points out, the latter method requires additional assumptions and since we have developed the current SSGP method from a standpoint of known (or well approximated) hyperparameters, we will consider comparisons between SSGP and this first method of Huber's.

The following theorem shows that in terms of the final equation sets, Huber's method and SSGP iteration agree;

Theorem 4. *In the case of known hyperparameters, Huber's final Recursive GP updating equation set agrees with those of SSGP iteration.*

Proof. Note that between our matrix M and Huber's Matrix J holds the relationship $M^T = J$. When we use the matrix M a subscript will be given here to indicate which set of points are being used to define it. Without loss of generality, we assume an initial prior mean of zero. For the sake of clarity means and covariance matrices will be denoted by ${}^S\boldsymbol{\mu}$ and ${}^S\Phi$ for SSGP equations and ${}^H\boldsymbol{\mu}$ and ${}^H\Phi$ for Huber's equations when

converted into the current notation. As in general Λ will denote initial prior covariance matrices in both cases.

We start by expressing the SSGP Equations in terms of priors and posteriors:

$$\begin{aligned} {}^{S}\boldsymbol{\mu}_{z}^{po} &= \Lambda_{z\mathbf{P}}\Gamma_{po} \\ &= {}^{S}\boldsymbol{\mu}_{z}^{pr} + [\Lambda_{z\mathbf{P}}M_{\mathbf{D}} + \Lambda_{z\mathbf{P}}\Psi_{pr}\Lambda_{\mathbf{PD}}][{}^{S}\Phi_{\mathbf{DD}}^{pr} + \Omega_{\mathbf{D}}]^{-1}[\mathbf{Y}_{\mathbf{D}} - ({}^{S}\boldsymbol{\mu}_{\mathbf{D}}^{pr})] \\ &= {}^{S}\boldsymbol{\mu}_{z}^{pr} + [{}^{S}\Phi_{z\mathbf{D}}^{pr} + (\Lambda_{z\mathbf{P}}M_{\mathbf{D}} - \Lambda_{z\mathbf{D}})][{}^{S}\Phi_{\mathbf{DD}}^{pr} + \Omega_{\mathbf{D}}]^{-1}[\mathbf{Y}_{\mathbf{D}} - ({}^{S}\boldsymbol{\mu}_{\mathbf{D}}^{pr})] \end{aligned} \quad (6.21)$$

$$\begin{aligned} {}^{S}\Phi_{zz}^{po} &= \Lambda_{zz} - \Lambda_{z\mathbf{P}}\Psi_{po}\Lambda_{\mathbf{P}z} \\ &= {}^{S}\Phi_{zz}^{pr} - [\Lambda_{z\mathbf{P}}M_{\mathbf{D}} + \Lambda_{z\mathbf{P}}\Psi_{pr}\Lambda_{\mathbf{PD}}][{}^{S}\Phi_{\mathbf{DD}}^{pr} + \Omega_{\mathbf{D}}]^{-1}[M_{\mathbf{D}}^{T}\Lambda_{\mathbf{P}z} + \Lambda_{\mathbf{DP}}\Psi_{pr}\Lambda_{\mathbf{P}z}] \\ &= {}^{S}\Phi_{zz}^{pr} - [{}^{S}\Phi_{z\mathbf{D}}^{pr} + (\Lambda_{z\mathbf{P}}M_{\mathbf{D}} - \Lambda_{z\mathbf{D}})][{}^{S}\Phi_{\mathbf{DD}}^{pr} + \Omega_{\mathbf{D}}]^{-1}[{}^{S}\Phi_{z\mathbf{D}}^{pr} + (\Lambda_{z\mathbf{P}}M_{\mathbf{D}} - \Lambda_{z\mathbf{D}})]^{T} \end{aligned} \quad (6.22)$$

We now consider the equations from [47]. Huber's equations require for the posterior equations at $\mathbf{P}$ to be determined first. In our SSGP notation this is:

$$\begin{aligned} {}^{H}\boldsymbol{\mu}_{\mathbf{P}}^{po} &= {}^{H}\boldsymbol{\mu}_{\mathbf{P}}^{pr} + {}^{H}\Phi_{\mathbf{PP}}^{pr}M_{\mathbf{D}}[\Lambda_{\mathbf{DD}} - M_{\mathbf{D}}^{T}\Lambda_{\mathbf{PD}} + M_{\mathbf{D}}^{T}\,{}^{H}\Phi_{\mathbf{PP}}^{pr}M_{\mathbf{D}} + \Omega_{\mathbf{D}}]^{-1}[\mathbf{Y}_{\mathbf{D}} - M_{\mathbf{D}}^{T}({}^{H}\boldsymbol{\mu}_{\mathbf{P}}^{pr})] \\ {}^{H}\Phi_{\mathbf{PP}}^{po} &= {}^{H}\Phi_{\mathbf{PP}}^{pr} - {}^{H}\Phi_{\mathbf{PP}}^{pr}M_{\mathbf{D}}[\Lambda_{\mathbf{DD}} - M_{\mathbf{D}}^{T}\Lambda_{\mathbf{PD}} + M_{\mathbf{D}}^{T}\,{}^{H}\Phi_{\mathbf{PP}}^{pr}M_{\mathbf{D}} + \Omega_{\mathbf{D}}]^{-1}M_{\mathbf{D}}^{T}\,{}^{H}\Phi_{\mathbf{PP}}^{pr} \end{aligned}$$

Across the whole domain, at some point z say, predictions are then made as follows (Equation 9 in [47], in his notation the expressions are those for μ_t^p and C_t^p):

$$\begin{aligned} {}^{H}\boldsymbol{\mu}_{z}^{po} &= M_{z}^{T}\,{}^{H}\boldsymbol{\mu}_{\mathbf{P}}^{po} \\ {}^{H}\Phi_{zz}^{po} &= \Lambda_{zz} - M_{z}^{T}\Lambda_{\mathbf{P}z} + M_{z}^{T}\,{}^{H}\Phi_{\mathbf{PP}}^{po}M_{z} \end{aligned}$$

Substituting in for the posteriors at $\mathbf{P}$, from the above it follows that:

$$\begin{aligned} {}^{H}\boldsymbol{\mu}_{z}^{po} &= {}^{H}\boldsymbol{\mu}_{z}^{pr} + M_{z}^{T}\,{}^{H}\Phi_{\mathbf{PP}}^{pr}M_{\mathbf{D}}[{}^{H}\Phi_{\mathbf{DD}}^{pr} + \Omega_{\mathbf{D}}]^{-1}[\mathbf{Y}_{\mathbf{D}} - ({}^{H}\boldsymbol{\mu}_{\mathbf{D}}^{pr})] \\ &= {}^{H}\boldsymbol{\mu}_{z}^{pr} + [\,{}^{H}\Phi_{z\mathbf{D}}^{pr} + (M_{z}^{T}\Lambda_{\mathbf{PD}} - \Lambda_{z\mathbf{D}})][\,{}^{H}\Phi_{\mathbf{DD}}^{pr} + \Omega_{\mathbf{D}}]^{-1}[\mathbf{Y}_{\mathbf{D}} - ({}^{H}\boldsymbol{\mu}_{\mathbf{D}}^{pr})] \end{aligned} \quad (6.23)$$

$$
\begin{aligned}
{}^{H}\Phi^{po}_{zz} &= \Lambda_{zz} - M_z^T \Lambda_{\mathbf{P}z} + M_z^T \, {}^{H}\Phi^{pr}_{\mathbf{PP}} M_z + M_z^T \, {}^{H}\Phi^{pr}_{\mathbf{PP}} M_{\mathbf{D}} [\, {}^{H}\Phi^{pr}_{\mathbf{DD}} + \Omega_{\mathbf{D}}]^{-1} M_{\mathbf{D}}^T \, {}^{H}\Phi^{pr}_{\mathbf{PP}} M_z \\
&= {}^{H}\Phi^{pr}_{zz} + [\, {}^{H}\Phi^{pr}_{z\mathbf{D}} + (M_z^T \Lambda_{\mathbf{PD}} - \Lambda_{z\mathbf{D}})][\, {}^{H}\Phi^{pr}_{\mathbf{DD}} + \Omega_{\mathbf{D}}]^{-1} [{}^{H}\Phi^{pr}_{z\mathbf{D}} + (M_z^T \Lambda_{\mathbf{PD}} - \Lambda_{z\mathbf{D}})]^T \quad (6.24)
\end{aligned}
$$

These final expressions for both $\boldsymbol{\mu}_z^{po}$ and Φ_{zz}^{po}, in Equations 6.21-6.22 and 6.23-6.24 respectively, are then identical to each other given the same initial mean vector and covariance matrix; this follows since it is clear from the definition of M that,

$$\Lambda_{z\mathbf{P}} M_{\mathbf{D}} = M_z^T \Lambda_{\mathbf{PD}},$$

in all cases. □

Hence, in terms of the predictive equation set, SSGP regression agrees with RGP regression. Note that while the final equations sets turn out to be equal, the theoretical approaches by which these equations are derived are very different, with RGPs using Kalman-filtering theory rather than GP regression theory. Furthermore, the theory underpinning the SSGP approach has gone well beyond that considered for RGP. This point will be made more generally in the following section, when SSGPs and the other batched GP approaches are discussed, but briefly, RGP theory gives no mention of how or why a given set of inducing points should be chosen, doesn't consider the possibility of singular covariance matrices and offers a technique for hyperparameter learning which requires additional assumptions. SSGP theory on the other hand deals with all of these points and, as has been shown in Section 6.6, also leads to a method for hyperparameter updating that remains grounded in GP theory and requires no additional assumptions. Furthermore, for SSGPs it has been explicitly shown how the method relates back to standard GP regression, and also when SSGP iteration and standard GPs are equivalent.

6.8 Relationship to Other Batched GP Approaches

The other existing approaches to batched GPs, these being FITC, VFE and the other associated methods obtained by varying the α divergence parameter in Equation 4.16, turn out not to be identical to SSGPs. In order to begin comparing these methods we

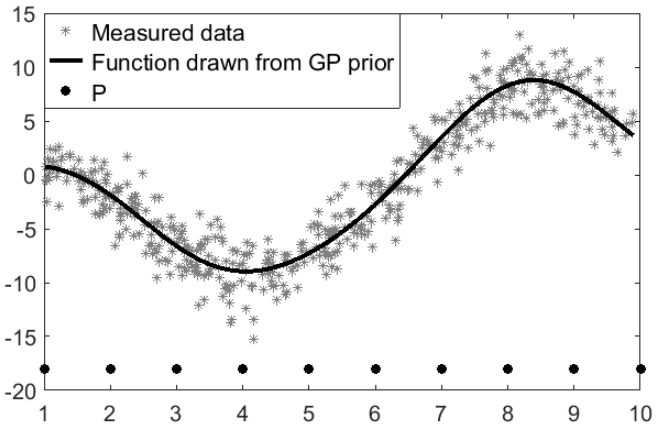

Figure 6.24: Example function and noisy data drawn from the GP prior.

first consider the accuracy of the predictive equations for the case of known hyperparameters and fixed induction points (i.e. $\mathbf{m} = \mathbf{P}$, all fixed).

Standard GP, SSGP, FITC and VFE predictive distributions on $\mathbf{P}$, from conditioning on noisy data, were compared for 1000 functions drawn from a GP prior with zero mean and a squared exponential covariance function. The amplitude and inverse-lengthscale hyperparameters for the covariance function were both set to 1. Noise was added to measurements with a standard deviation of 0.1. An example function drawn from the GP prior is shown in Figure 6.24 along with a sample of noisy data. The set $\mathbf{P}$ is also shown in the figure. 500 noisy measurements are taken from each function and used for regression with each method. Standard GP processed the entire dataset at once, whereas the batched approaches received the data in a sequence of 50 point subsets, a total of 20 batches. The predicted mean values and standard deviations were then calculated across $\mathbf{P}$ for each method. Table 6.1 shows the differences between each of the batched GP methods and standard GP for the mean and standard deviation values. These results clearly indicate that there are no significant differences between the predictive performances of any of the methods with respect to standard GP regression in this case. Similar results are obtained if the hyperparameters which define the GP prior are adjusted. Hence, we have demonstrated that in terms of accuracy, the performance of the SSGP predictive equations is in-line with existing state of the art methods.

Regression Method	Mean Err. on **P**	Mean % Std. Err. on **P**
GP	8.63×10^{-3}	-
SSGP	8.49×10^{-3}	0.2783
FITC	8.53×10^{-3}	0.2787
VFE	8.51×10^{-3}	0.2779

Table 6.1: Error statistics for standard and batched GP methods for a cohort of 1000 functions drawn from a GP prior. The second column gives the mean error of function predictions on **P** and the third gives the mean percentage difference between the standard deviations resulting from standard GP regression and each of the batched approaches.

A comparison of SSGPs with other batched GP methods therefore comes down to considerations of hyperparameter and inducing point selection and the completeness of existing theoretical knowledge. With regards to this last point we now revisit the questions posed in Section 4.5, for which existing theory did not provide an answer, and discuss whether SSGP theory has provided insight.

Q1. *Does the underlying structure of a GP model suggest any particular approach to performing regression on batched data, and what relationship would such a method have with standard GP regression on all of the data at once?*

SSGP theory is effectively a direct answer to **Q1** since it was explicitly constructed (see Section 6.4.1) from GP regression equations in order to agree with standard GP regression at points in **P**, with the choice of Sufficient-Subset (based on matrix condition number) ensuring that from **P** the SSGP can extrapolate predictions with minimal error. The relationship to standard GP regression was explicitly shown in Theorem 3 and, effectively, comes down to the extent to which covariance expressions can be transformed through **P** by the M matrices defined by Equation 6.10.

Q2. *Given a GP prior, but as yet no measured data, does the GP itself indicate any information about the number and location of inducing points necessary to accurately model this GP?*

This question was addressed during the development of SSGP regression through a study of singular covariance matrices and the subsequent extension to the general case in Sections 6.1 and 6.3. Having shown that singularity in covariance matrices indicates

linear dependencies between function values, the concept of a Sufficient-Subset was developed in order to be the minimal set of points for which the GP is effectively determined by information on $\mathbf{P}$, hence the use of the word *sufficient*. Note that the Sufficient-Subset is determined entirely from the GP prior, without requiring measured data. In terms of the location of the inducing points, assuming only information about the GP prior there is no reason to choose anything other than equal spacing and hence it is the number, rather than the exact position which is determined. With respect to functions not strictly drawn from a GP prior, it has been shown that it is possible to use a Super Sufficient-Subset in order to avoid the need to determine the best location for each inducing point, avoiding the requirement for optimisation procedures.

Super Sufficient-Subsets were found to be the natural result of investigating the limit of SSGP behaviour as the covariance matrix becomes singular, as more points are added to $\mathbf{P}$. As discussed in Section 4.5, existing methods all assume the covariance matrix $\Lambda_{\mathbf{PP}}$ to be invertible, with equations which are ill-defined if this is not the case. Since additional points in $\mathbf{P}$ are effectively adding information, it seems contradictory that the regression equations should break down as more points are added. It transpired that for SSGPs this is not the case, singular covariance matrices were shown to be entirely valid, with the correct form in which they should be handled determined by the analysis of Section 6.5. Furthermore, in the case where the covariance matrix is exactly singular (rather than singular due to finite computational precision), it was shown that SSGP iteration is exactly equal to standard GP regression. As with the rest of SSGP theory, these results follow from a principled consideration of the underlying structure of GP covariance matrices.

Q3. *Can such techniques be applied without the need for demanding optimisation and, if so, when might this be possible?*

SSGP investigations have shown that the answer to this last question is that optimisation procedures may be avoided whenever a good prior can be determined for the problem on which regression is to be performed. Once this prior has been found, the choice of inducing points follows from Sufficient-Subset theory and SSGP regression can proceed without the need for updating of hyperparameters or the locations of points in

P. In the following chapter the process by which a suitable prior may be determined is demonstrated for the wind turbine dynamics identification problem. It will also be shown that accuracy in the turbine dynamics case is improved through the application of a Super Sufficient-Subset.

SSGP iteration theory therefore allows for GP regression on batched data while thoroughly addressing existing gaps in the knowledge base of this area. It shows how a GP model itself contains the relevant information for determining how it is best approximated and its relationship to standard GP regression has been explicitly shown. Furthermore, as will be outlined in the following chapter, none of the existing batched GP techniques are suitable for application in the turbine dynamics problem. This is not only due to the need to avoid expensive optimisation, but also because the theory of Super Sufficient-Subsets will be shown to play a key role in the development of a general GP prior, suitable for application in any wind turbine.

6.8.1 The Future Development of SSGPs with Respect to Existing Methods

While the development of SSGPs, up to the point reached in this work, will be shown to result in a viable method for the identification of wind turbine dynamics, there is a lot of scope for the further development of SSGPs in order to make them applicable in more general settings, for example in cases where large quantities of prior information is not available. This was touched upon is Section 6.6, where it was shown that hyperparameter learning can be accommodated within the SSGP framework, along with the potential for reselecting the points in the Sufficient-Subset, or applying a Super Sufficient-Subset. It is therefore possible to develop a more 'black box' version of SSGPs which can be applied to data without pre-determining the prior. It is this version of SSGPs which can then be compared thoroughly with existing methods, since it is in this setting that they have been principally developed. Key questions to be answered there will relate to the differences in inducing point location determined by SSGP theory and the optimisation procedures of VFE, FITC etc, along with differences in accuracy and speed. A pertinent question will also be whether Sufficient and

Super-Sufficient Subset theories render the need to optimise inducing point locations unnecessary. Finally, existing methods will need to be re-examined in the context of the theoretical insights contributed by SSGP theory, some of which apply generally, such as the GP covariance matrix structure discussion of Section 6.3.

With respect to the aims of this thesis, the further development of SSGPs to include hyperparameter optimisation falls out of scope[5], and hence the extension of SSGPs (and the subsequent further comparisons with existing techniques) outlined here necessarily falls to being future work.

6.9 Discussion

This chapter has documented the development of a new GP regression approach, SSGP iteration, which can handle batched data and changing noise levels, while scaling linearly with respect to the number of measurements. The method itself, and its associated theory has been shown to provide answers to some fundamental questions not addressed by existing methods. SSGPs were developed by generalising results seen for GP polynomial regression in Chapter 5. The key factor which provided a link between the polynomial and general nonlinear case being similarities in covariance matrix structure. SSGP iteration was developed from these initial observations into a full, practical regression technique. The relationship between SSGPs and standard GP regression was explicitly shown and experiments verified that SSGPs can perform with the same level of accuracy as a standard GP on all of the data at once.

The same covariance structure considerations which led to SSGP regression were then shown to lead to the possibility of using a number of inducing points which renders the covariance matrix singular, a Super Sufficient-Subset. It was demonstrated that the application of a Super Sufficient-Subset can reduce regression errors when dealing with functions not strictly drawn from a GP prior. In the following chapter this will turn out to be a crucial development in terms of defining a suitable SSGP prior in the wind turbine dynamics identification problem.

[5]Since in the wind turbine context such optimisation procedures must be avoided for reasons outlined in Section 2.6.3.

Finally, it was demonstrated that there is scope for the SSGP regression model to be extended to include online learning of hyperparameters, and the updating of inducing point locations (or the application of a Super Sufficient-Subset to render this unnecessary). There is therefore a lot of potential for SSGP regression to play an important role in the future development of GP regression techniques, both theoretically and practically. Future work will look to fully realise this potential.

With respect to the aims of this thesis, we have now developed the necessary tools with which to begin looking at the learning of turbine dynamics from measured data. The following chapter therefore considers the practical application of the GP techniques developed in Chapter 5 and 6 to the turbine dynamics identification problem.

Chapter 7

Wind Turbine Dynamics Identification

In this chapter we consider the application of both polynomial and SSGP regression to the identification of wind turbine dynamics in below and above rated conditions respectively. Since the proposed approaches will be tested on real data, we start by considering the processing and conditioning of this data in Section 7.1. Such pre-processing is necessary to remove anomalous signals in the data which are unrelated to the sought dynamics. Appropriate sampling times are discussed in Section 7.2. In Section 7.3 the efficient polynomial regression approach developed in Chapter 5 is applied to the maximum efficiency tracking region regression problem for both simulated and real turbine data. Sections 7.4-7.7 consider the application of SSGP theory in the above rated regression case. This involves using prior information, in the form of known aerodynamics tables, to develop a general SSGP prior and Sufficient-Subset for aerodynamic tables. For practical reasons it is shown that the regression formulation derived in Section 2.6.1 will in fact be unsuitable for dynamics learning given the narrow trajectory of an operating wind turbine. An alternative formulation is therefore presented in Section 7.8, followed by the development of an appropriate SSGP prior and Super Sufficient-Subset. This new approach is then tested on simulated and real data in Section 7.10.

7.1 Data Conditioning

When the regression approaches developed in this thesis are applied to real data it is necessary to perform certain pre-processing tasks in order to 'clean' the data and remove signal contributions which are unrelated to the sought dynamics. This section documents these various procedures.

7.1.1 Measurement Equipment Faults and Non-Standard Operation

A simple but important task when using real data is that of removing measurements which have resulted from equipment errors which are unrelated to simple measurement noise. This might include one or more transducers having failed, and so not generating data, as well as any measurements relating to a period of time where the turbine is not being operated as normal (e.g. data from when the turbine is shut down for routine maintenance, even though the wind speed might be high enough for power generation under normal circumstances).

7.1.2 Filtering nP Peaks and Higher Frequency Harmonics

The rotation of the blades on a wind turbine rotor results in harmonics being propagated through the system at frequencies which are multiples of the turbine's rotational speed. This is mainly due to deterministic and stochastic sampling of the wind field, as well as imbalance in the rotor caused buy non-identical blade manufacturing [24]. For a 3 bladed turbine the most prominent harmonics appear at 1P (rotor imbalance), 3P and 6P, where P is denoting rotational speed (we use this notation since it is the most common when referring to these harmonics, this should not be mistaken for the Sufficient-Subsets of the previous chapter). Other harmonics such as the tower mode might also appear in the frequency spectrum. Since these harmonics are independent of the aerodynamics of the wind turbine, we wish to filter them out of measured data in order to avoid their interfering with the regression process. Figures 7.1 and 7.2 show power spectral density (PSD) plots from real turbine data before and after filtering generator torque and rotor speed respectively.

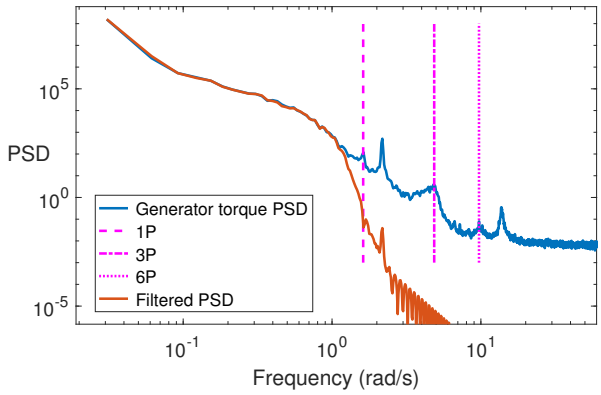

Figure 7.1: Power spectral density plots for generator torque measurements before and after filtering.

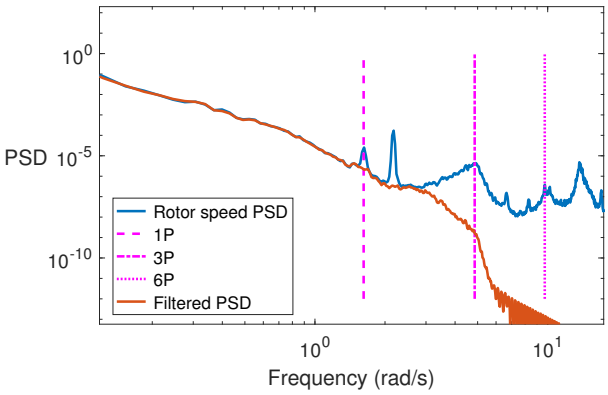

Figure 7.2: Power spectral density plots for rotor speed measurements before and after filtering.

In the unfiltered PSDs of both figures 1P, 3P and 6P can be seen clearly, along with what is a probable tower mode at just over 2 rad/s. Since the dynamics important to the regression problem are all at frequencies lower than these peaks, for all signals except for rotor speed, they are processed using a fifth order Butterworth filter [58] with cutoff frequency 1.2 rad/s. As can be seen for the generator torque measurements in Figure 7.1 this low-pass filter preserves the low-frequency components of the signal, while attenuating the high frequency contributions, and in particular the harmonic peaks.

The rotor speed measurements require a slightly different filtering approach. This is because changes in rotor speed over just a few seconds will be important for the regression problem due to the inertial component of Equation 2.6, furthermore, obtaining rotor acceleration measurements requires differentiating the rotor speed time signal (by adding a derivative term to the filter) and this will exaggerate any peaks which haven't been heavily attenuated. It is therefore necessary use a Butterworth filter with a higher cutoff frequency of 3 rad/s (to capture faster changes in signal values), and to explicitly remove the first two harmonic peaks with notch filters. The PSD of the resulting signal can be seen in Figure 7.2.

These various filtering techniques result in time-shifted output signals and so it is necessary to correct for this using autocorrelation to determine the time-shift on each

filtered signal and then correct the output times as appropriate.

7.1.3 Spatial Filtering of Wind Speed Measurements

The concept of rotor effective wind speed was discussed briefly in Section 2.3. We now consider this concept again. With respect to frequency components of signals, the effective wind speed can be defined as the wind speed averaged over the rotor area such that the spectrum of aerodynamic torque remains unchanged [59]. In [60] a transfer function between point and rotor-effective wind speed is derived and shown to be:

$$V_e = \frac{\frac{1}{\sigma}\left(s + \frac{\sqrt{2}}{\sigma}\right)}{\left(s + \frac{\sqrt{2}}{\sigma\sqrt{a'}}\right)\left(s + \frac{\sqrt{a'}}{\sigma}\right)} V_p,$$

with

$$\sigma = \frac{\gamma R}{\bar{v}}.$$

R is the rotor radius, $\bar{v}$ is the point wind speed averaged by a first order filter of the form;

$$\bar{V} = \frac{k}{s+b} V_p,$$

$a' = 0.55$ and the turbulent decay factor, γ, has a standard value of 1.3 [59, 60].

The above transfer function for v_e was derived for a constant value of σ. In the current case σ will vary with time and so the following approach is required to implement this spatial filter.

First note that the above expression for V_e can be re-expressed as,

$$\begin{aligned} V_e &= \frac{s\sigma + \sqrt{2}}{\left(s\sigma\sqrt{a'} + \sqrt{2}\right)\left(\frac{s\sigma}{\sqrt{a'}} + 1\right)} V_p \\ &= \frac{s\sigma + \sqrt{2}}{\sigma^2 s^2 + \left(\sqrt{a'} + \frac{\sqrt{2}}{\sqrt{a'}}\right)\sigma s + \sqrt{2}} V_p \\ &= \frac{s\sigma + \sqrt{2}}{\sigma^2 s^2 + \left(\frac{a'+\sqrt{2}}{\sqrt{a'}}\right)\sigma s + \sqrt{2}} V_p. \end{aligned}$$

Reverting to the time domain, this transfer function corresponds to the differential equation:

$$\sigma^2 \ddot{v}_e - \sigma \dot{v}_p = -\left(\frac{a' + \sqrt{2}}{\sqrt{a'}}\right) \sigma \dot{v}_e + \sqrt{2} v_p - \sqrt{2} v_e.$$

Defining the differential operator $\mathcal{D}_\sigma = \sigma \frac{d}{dt}$ this becomes,

$$\mathcal{D}_\sigma \left(\mathcal{D}_\sigma v_e - v_p\right) = -\left(\frac{a' + \sqrt{2}}{\sqrt{a'}}\right) \left(\mathcal{D}_\sigma v_e - v_p\right) + \left(\sqrt{2} - \frac{a' + \sqrt{2}}{\sqrt{a'}}\right) v_p - \sqrt{2} v_e.$$

The above second order equation can then be formed into a system of two first order differential equations. Let $y_1 = v_e$ and $y_2 = \mathcal{D}_\sigma v_e - v_p$, this gives rise to the system,

$$\begin{aligned} \mathcal{D}_\sigma y_1 &= y_2 + v_p \\ \mathcal{D}_\sigma y_2 &= -\left(\frac{a' + \sqrt{2}}{\sqrt{a'}}\right) y_2 + \left(\sqrt{2} - \frac{a' + \sqrt{2}}{\sqrt{a'}}\right) v_p - \sqrt{2} y_1, \end{aligned}$$

dividing through by σ and integrating gives,

$$\begin{aligned} y_1 &= \int \frac{y_2 + v_p}{\sigma} dt \\ y_2 &= \int \frac{-\left(\frac{a' + \sqrt{2}}{\sqrt{a'}}\right) y_2 + \left(\sqrt{2} - \frac{a' + \sqrt{2}}{\sqrt{a'}}\right) v_p - \sqrt{2} y_1}{\sigma} dt. \end{aligned}$$

This system of integral equations can then be solved in Simulink software using the block diagram shown in Figure 7.3, resulting in an effective wind speed time history. Figure 7.4 shows example time histories of measured and effective wind speeds from an anemometer, determined using this spatial filtering method.

7.2 Measurement Correlations and Sampling Times

The data available to a wind turbine controller is very high frequency, i.e. in the range of 20-100Hz (20Hz being the frequency of real data available for this work). In terms of extracting information on which to perform regression, we require that measurements are approximately independent of each other, since otherwise we are effectively double counting and this will impact the quality of predictions. For regression purposes the

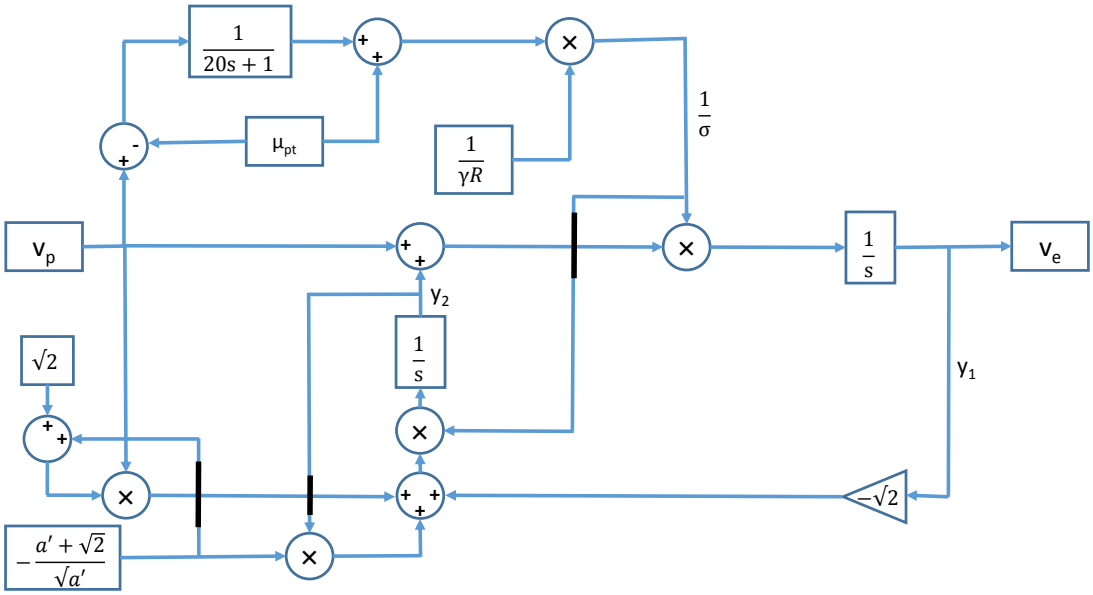

Figure 7.3: Effective wind-speed block diagram. μ_{pt} is the average point wind speed across the whole time history. The thick black lines indicate crossing points.

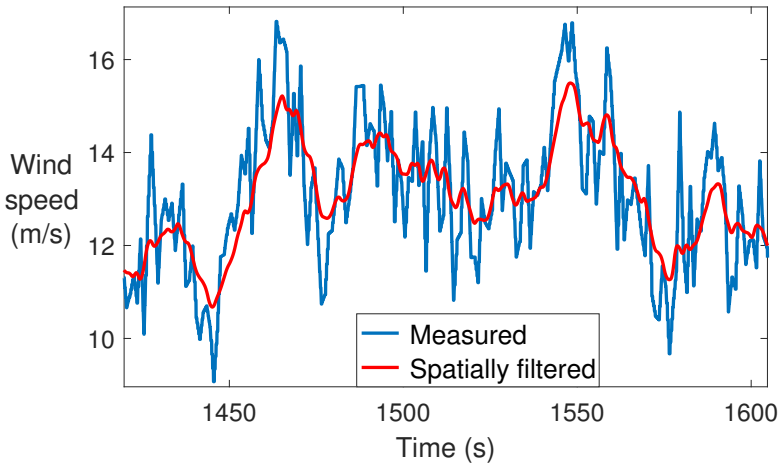

Figure 7.4: Measured and spatially filtered (i.e. effective) wind speed time histories.

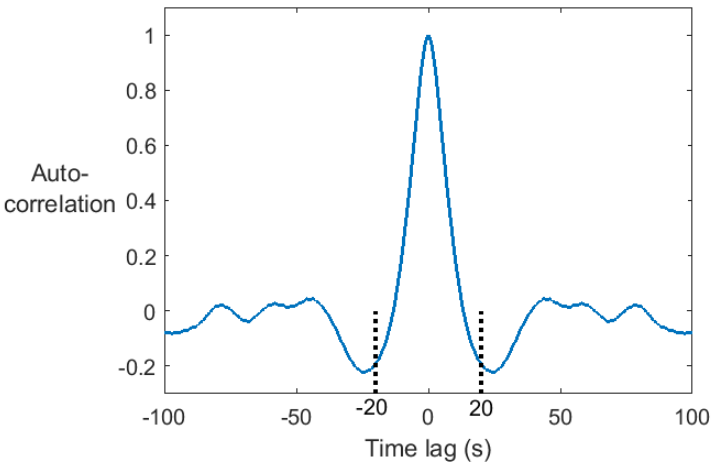

Figure 7.5: Autocorrelation plot for a wind turbine rotor speed time history.

frequency of available data is much higher than is required and, therefore, measurements must be sampled with a time interval which avoids these data correlations. In order to determine a suitable sampling interval, autocorrelation analysis was performed for various wind turbine data time-series. An example of this is shown in Figure 7.5 for rotor speed. The various results all show 20s to be a time lag for which the autocorrelation values have made their initial crossing of the zero line. Hence, 20s was chosen as the sampling interval for regression data.

7.3 Maximum Efficiency Region Dynamics Identification

As discussed in Section 2.6.2, a regression equation can be formed for during maximum efficiency tracking of the wind turbine in below rated conditions which has the form,

$$\hat{G} = \Theta_2 + \mathcal{L}^*(\hat{\omega}_r^{-1}) + \delta.$$

Where the measurements $\hat{G}$ are formed from data available to the turbine controller and, assuming a linear torque loss function $\mathcal{L}$, the above equation is a quadratic polynomial in $\hat{\omega}_r^{-1}$ with additive noise.

In this section both GP and LS regression will be applied to this problem using data both from simulations and from a real wind turbine. The fast polynomial GP

implementations developed in Chapter 5 are used throughout.

7.3.1 Learning from Simulated Data

Simulated data for these investigations was generated using the Supergen Exemplar 5MW wind turbine model. As outlined in Section 2.8, this is a high quality Simulink based turbine simulation model developed by the Supergen Wind Hub to allow for the development of advanced control techniques [26]. Both the power curve and torque-speed diagrams seen in early chapters, along with the sample trajectory, in Figures 2.3 and 2.7 were produced using this model. Simulations were run over a range of wind conditions with mean wind speeds between 5 and 8 m/s and turbulence intensities of between 5 and 20%. The relevant data, i.e. that required to determine $\hat{G}$, was extracted from the model.

Figures 7.6 and 7.8 show the values of C_{Pmax} and the drivetrain loss functions[1] respectively predicted by both LS and GP regression techniques. Each prediction is from a dataset containing 500 points, corresponding to roughly 3 hours of realtime operation.

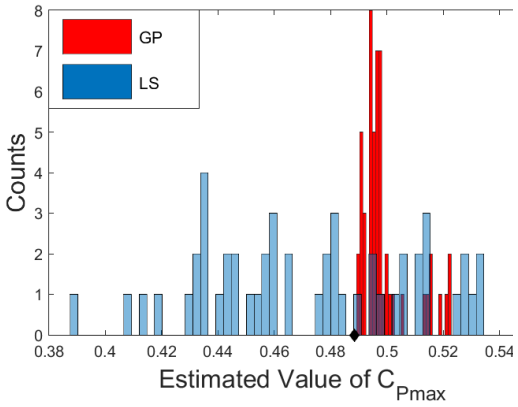

Figure 7.6: $C_{\mathrm{p,max}}$ estimates from both GP and LS. The true value is given by the black diamond.

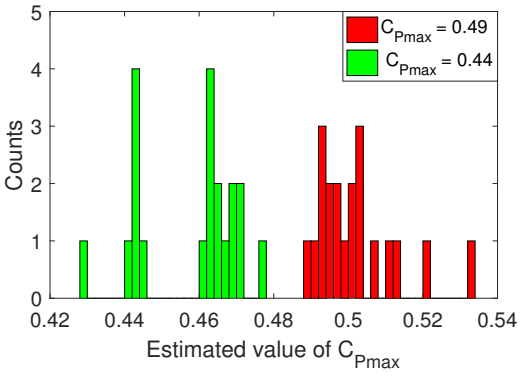

Figure 7.7: $C_{\mathrm{p,max}}$ estimates from datasets with two different values of maximum efficiency.

In Figure 7.6 it is clear that the GP approach is superior to using LS, since, while the

[1]Note that the loss contours are indicating the torque losses that would result in a power loss of the given percentage at each rotational speed.

GP predictions are clustered close to the true value of C_{Pmax}, with some amount of positive bias (potential sources of which will be discussed), the LS results are spread almost uniformly across the range of probable values. These results are typical of all test runs carried out with the simulated data. Based on the GP clustering, one would expect a shift in C_{Pmax} by some small amount to be detectable. In order to test this hypothesis, regression was performed on a dataset generated with all C_P values reduced by 0.05 and compared to regression on the original data. C_{Pmax} estimates from these two cases are shown in Figure 7.7 where the sets of prediction clusters are clearly separate, and hence the GP predictions are indeed able to detect this shift in aerodynamic efficiency.

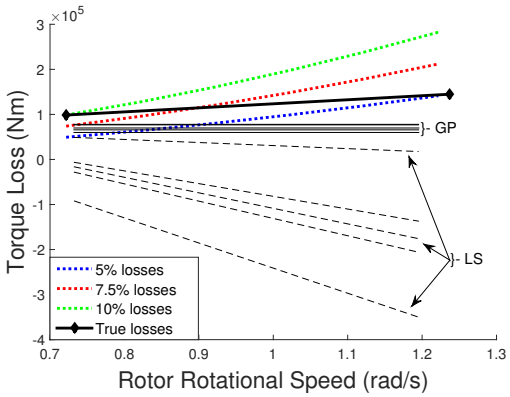

Figure 7.8: Drivetrain loss predictions along with loss contours (percentages in terms of design power values at each rotational speed).

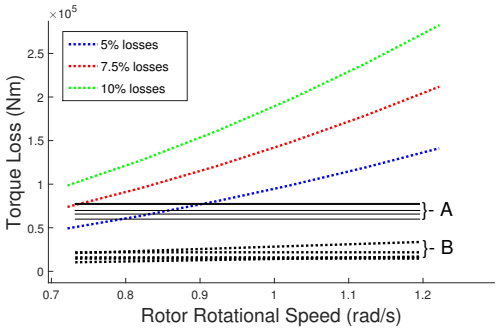

Figure 7.9: Drivetrain loss predictions along with loss contours for two different loss function cases.

Similar results are then observed for the drivetrain loss predictions in Figure 7.8 where the true losses in the model are shown along with various GP and LS regression predictions. Again, the GP predictions are much more tightly clustered and, while they do show a bias, GP regression here can be seen to give both more accurate and more consistent predictions of the losses in the drivetrain.

The GP method was then further tested with respect to sensitivity to drivetrain losses by adjusting the losses function in the model. Figure 7.9 shows the GP predictions from Figure 7.8, using the depicted true losses function. This set of predictions has been labelled cluster A in Figure 7.8. Regression was then performed for data generated

with an altered loss function obtained by shifting the left hand value of the original loss function down by 2.5%. This second set of predictions is labelled cluster B. It can be seen that these two clusters are clearly separate and so this shift in losses is detectable when using the GP regression approach, indicating that the GP predictions are accurate enough to detect small changes in loss values.

7.3.2 Possible Bias Correction

In the above results for both C_{Pmax} and drivetrain losses there is clearly the presence of some bias in the predictions. We now considered the possible sources of this bias and whether anything can be done to correct for it. Since the variation present which results in 'noise' here is due to the control system attempting to track maximum efficiency, it is quite possible that it is the underlying controller dynamics and turbine operation which is to blame here. However, there is at least one source of bias which stems from the assumptions used to develop the regression equation, this being the Taylor expansion of the term $\lambda^{-3}C_p(\lambda)$. As can be seen in Figure 2.2 the C_p curve is very flat around $\lambda = \lambda_{max}$ and hence a Taylor expansion should not results in any bias. However, the term λ^{-3} is both steep and asymmetrical at this point. Even if the variation in $\lambda^{-1} = \frac{v}{\omega R}$ is approximately Gaussian, the resulting variations about λ_{max}^{-3} will certainly not be. This issue is depicted in Figure 7.10. The following analysis attempts to define a correction term for this source of bias.

Since the turbine is attempting to maintain a tip-speed ratio of λ_{max} and wind speed fluctuations are the main cause of disturbances in the system; for a given rotor speed ω we define v_{max} such that,

$$\lambda_{max} = \frac{\omega R}{v_{max}},$$

and then the measured λ's are of the form,

$$\lambda = \frac{\omega R}{v_{max} + \zeta},$$

for some 'error' term ζ which represents the deviation of the true wind speed, v, from

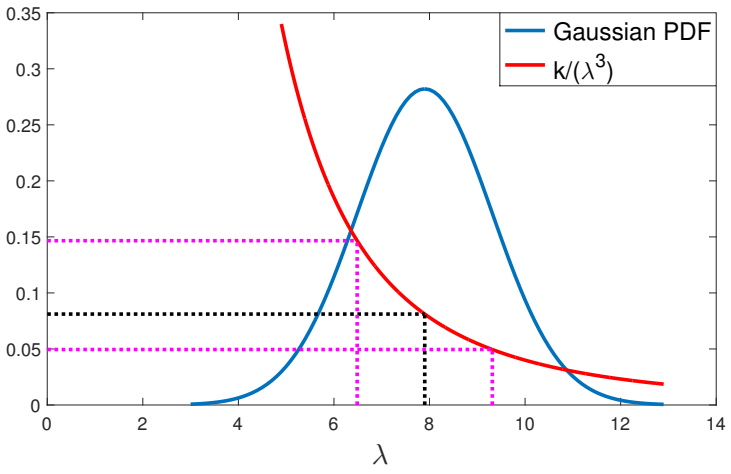

Figure 7.10: Asymmetric outputs example. Transforming the mean and standard deviation values of the input variable, λ, through the given function, it is clear that the output values are not distributed symmetrically about the transformed mean.

v_{max}. Assuming ζ is Gaussian the expected value of λ^{-3} can then be expressed as,

$$\begin{aligned}
\mathbb{E}[\lambda^{-3}] &= \mathbb{E}\left[\left(\frac{v_{max}+\zeta}{\omega R}\right)^3\right] \\
&= \frac{1}{(\omega R)^3}\mathbb{E}[v_{max}^3 + 3v_{max}^2\zeta + 3\zeta^2 v_{max} + \zeta^3] \\
&= \frac{1}{(\omega R)^3}(v_{max}^3 + 3\tau v_{max}) \qquad \text{where } \tau = \mathbb{E}(\zeta^2) \\
&= \lambda_{max}^{-3}\left(1 + \frac{3\tau\lambda_{max}^2}{(\omega R)^2}\right).
\end{aligned}$$

The above expression indicates that we should expect the predictions of the term $\lambda_{max}^{-3}C_{Pmax}$, and hence C_{Pmax} itself, to be over-predicted by a factor of,

$$\left(1 + \frac{3\tau\lambda_{max}^2}{(\omega R)^2}\right).$$

It also follows that correcting this bias can be achieved by dividing the C_{Pmax} predictions by this same term.

For simulated data the ζ values can be calculated explicitly as,

$$\zeta = \omega R(\lambda^{-1} - \lambda_{max}^{-1}),$$

and hence also $\tau = \mathrm{Var}(\zeta)$. Figure 7.11 shows the result of applying this bias correction factor to polynomial GP predictions of C_{Pmax} using simulated data. While the bias correction does slightly improve the predictions, moving them closer to the true value of C_{Pmax}, it is only by a very small amount and in no way removes all of the positive bias present.

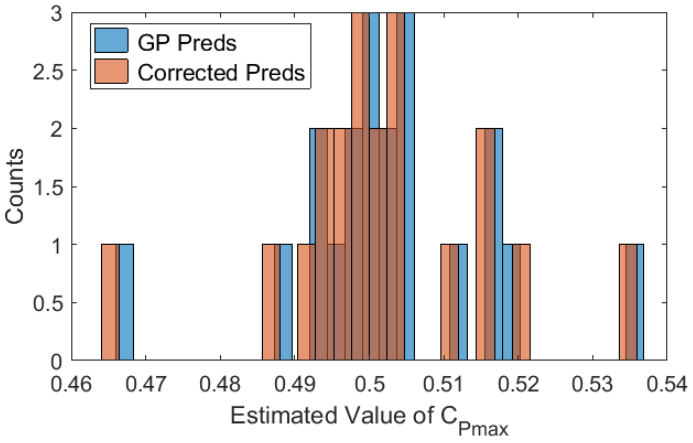

Figure 7.11: C_{Pmax} predictions with and without bias correction.

It therefore seems that the bias present is due to some factor other than our formulation of the regression equation, probably related to the specifics of the wind turbine's operation. This implies that it will not be possible to use this technique for accurate predictions of a wind turbine's C_{Pmax} value and drivetrain losses.

However, the approach developed here might still prove useful for wind turbine health monitoring and operation and maintenance (O&M) applications as discussed in Section 7.3.4.

Closed Loop System Identification

In control theory it is well documented that, in a closed loop setting, plant model identification can suffer from biases resulting from the feedback loop being present [61, 62]. In the current work we are not performing system identification in the same sense, however, since the 'noise' present in our below rated regression formulation results from operational deviations away from the design trajectory, the closed loop control present while C_{Pmax} tracking could conceivably be biasing predictions in a similar way

to that observed in control theoretic settings. It is therefore recommended that this possible source of bias is investigated as part of future work in this area.

7.3.3 Learning Below Rated Dynamics with Real Data

GP polynomial regression was also applied to real data from an operating wind turbine. Of the available data, roughly 27 hours (5000 datapoints when taken at 20s intervals) lies in the maximum efficiency tracking region. This data is shown in Figure 7.12. Note that the trajectory of operating points here is rather different from that seen in the design operating strategy shown in Figure 2.5. In practice, operating curves will differ from the ideal case due to practical considerations which are unique to each turbine, for example each generator will have its own operational start-up requirements depending on topology, technology type, grid connection and similar design considerations [6]. In the current case it can be seen that C_{Pmax} tracking is only initiated about one third of the way along the turbine's range of rotational speeds. Furthermore, due to the higher speed regions of C_{Pmax} tracking potentially including data from transitory or constant speed operation (i.e. the transition between operating regions 3 and 4 in Figure 2.5), it is necessary to apply a high speed cut-off to ensure we only consider data which is definitely still within the maximum efficiency region.

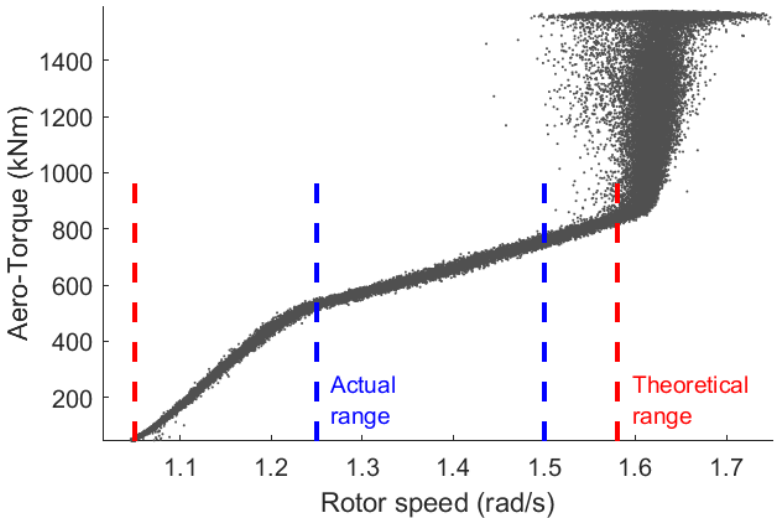

Figure 7.12: Measured torque vs rotor speed for measured wind turbine data.

It is from within these limits that data is used for the maximum efficiency dynamics identification problem. Figure 7.12 shows this difference between the theoretical and actual range of rotational speeds from which data can be used. Note that the actual range of available rotor speeds is really quite restrictive. It is from this actual range of data that the available 5000 datapoints are obtained.

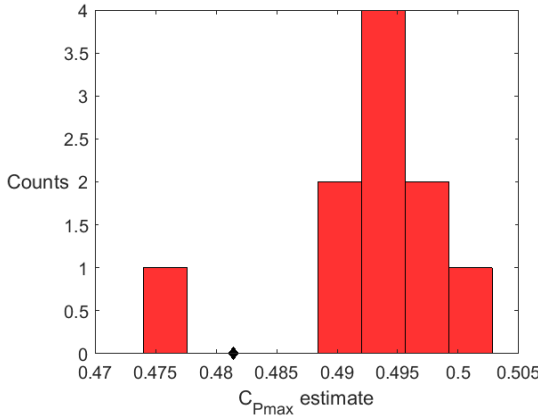

Figure 7.13: GP C_{Pmax} predictions for datasets containing 500 points each from real wind turbine data. The true value is given by the black diamond.

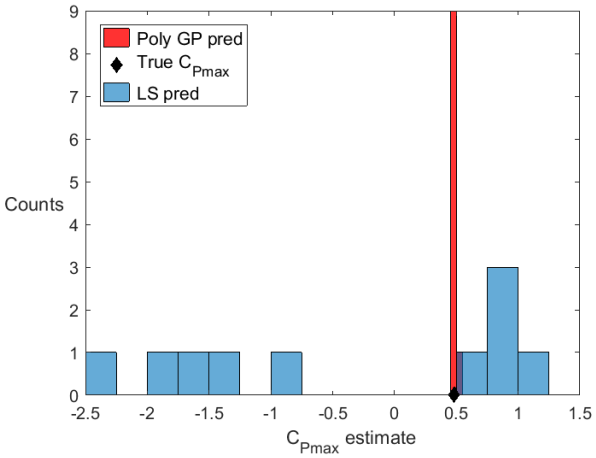

Figure 7.14: GP and LS C_{Pmax} predictions from real data.

As in the simulated case, the available data is split into sets of 500 datapoints and polynomial GP regression is applied to each set, resulting in 10 predictions for the value of C_{Pmax} and the drivetrain loss function. Figure 7.13 shows the predicted C_{Pmax} values, along with the real value for this turbine. These results mimic those seen in the simulated data case of the previous section, with the predictions tightly clustered and showing a small amount of positive bias. Furthermore, the results from LS polynomial regression are scattered almost uniformly across a large range of values, as shown in Figure 7.14. These results therefore validate the findings seen thus far for C_{Pmax} predictions.

The losses prediction results, on the other hand, are less encouraging. For all 10 of the real datasets, the GP predictions for losses have coefficients which are effectively zero (to within machine precision). This indicates that the GP is unable to detect any changes in the losses as rotor speed changes, and therefore sets the associated

coefficients to zero. It seems likely that this is due to the restricted range of rotor speeds from which data is available as part of the C_{Pmax} tracking region (as shown in Figure 7.12). The loss values represent only a very small contribution to the overall measured values (being only a few percent of measurements)[2], one would therefore expect the losses to be difficult to detect to begin with, it seems feasible that this followed by a restriction of the rotational speeds for which losses are seen could lead to them being undetectable. Figure 7.15 shows the regression data, $\hat{G}$, with and without losses present for the simulated data from the Supergen Exemplar turbine model. All data in the figure has been centred about zero (by subtracting mean values) to allow for easy comparisons of the impact of losses to the data (avoiding offsets between the two datasets which would make this more difficult). Note the large signal-to-noise ratio with respect to the loss function and measured data, indicating that we should expect losses to be difficult to detect. This demonstrates why a restricted range of rotational speeds might reasonably lead to the GP being unable to detect losses in the data. In future work it will therefore be necessary to obtain data from a wind turbine with a greater range of rotational speeds while C_{Pmax} tracking in order to test whether losses can be detected for a larger speed range.

7.3.4 Applications in Wind Turbine Health Monitoring

The results of the previous sections show that there is the potential for wind turbine C_{Pmax} values and drivetrain losses identification using polynomial GPs. Although, in the case of drivetrain losses more work is required to verify whether they can be detected in real, rather than just simulated, wind turbine data and future work will look to obtain data from a wind turbine with a larger range of rotational speeds for this purpose.

While promising results were obtained on simulated data (and for the maximum efficiency coefficient also on real data), the predictions were seen to exhibit offsets from their true values. This section considers the potential uses of such predictions which,

[2]Note that while the size of losses is very small, we would still ideally like to be able to identify them in order to detect if they are changing; this is because changes in the losses could indicate that a drivetrain fault is present and so contribute to the maintenance scheduling of the wind turbine.

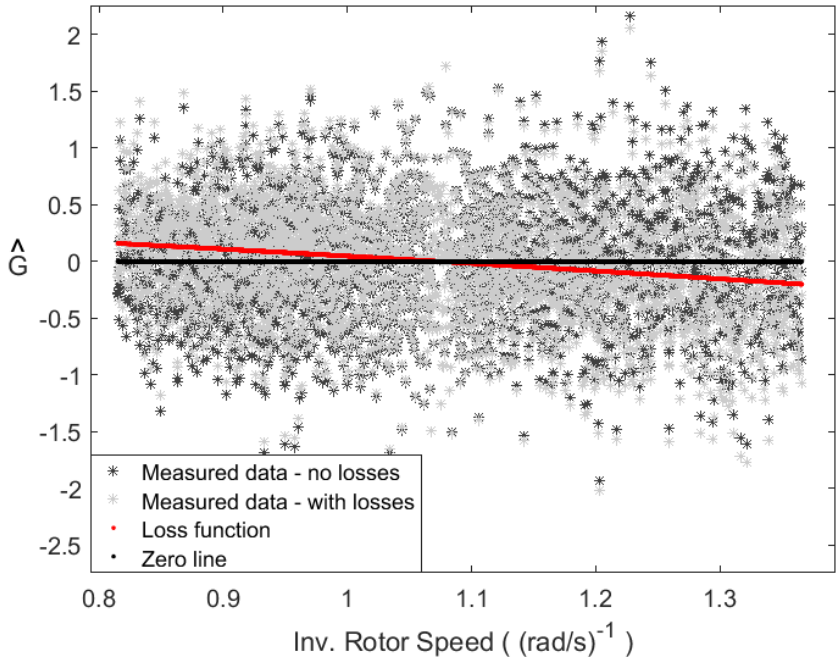

Figure 7.15: Below rated regression measurements, $\widehat{G}$, with and without losses. The zero (no losses) line and loss function are also shown. Mean values have been subtracted to centre each set of data about zero to allow for easy visual comparisons.

while they cannot be used to determine the exact values of the identified quantities, might still be applied usefully for turbine monitoring.

As wind turbine assets grow in size and move further offshore, where access becomes more problematic and expensive, the need for increased reliability becomes even more pronounced. Therefore, there is a need for a diverse range of monitoring techniques with which faults and behavioural changes in wind turbines can be quickly detected and appropriate steps taken. Due to this, there have been a huge number of new measurement techniques developed in recent years, all with their associated costs.

The ability to cheaply detect when aerodynamic efficiency or drivetrain losses have changed could help indicate faults and the scheduling of turbine inspection and maintenance. Changes in aerodynamic efficiency can either happen rapidly, for example via icing of the blades [63], or slowly, for example via blade degradation [64]. Changes in the mechanical losses in the drivetrain can be indicative of damage or an imminent failure. In all cases, prior warning of any changes, and the tracking of such changes over

time, is desireable to inform the O&M of wind turbines. For these types of applications it is the ability to detect *changes* in these values, rather than the exact values themselves, which is key to forewarning of potential issues. Hence, for turbine monitoring applications these offsets do not necessarily pose a significant problem.

Automated detection of turbine faults using operational data has been previously explored in the literature, for example, in [65] 10-minute SCADA data is used to detect abnormal turbine behavior using neural network models. It is interesting to note that in both this and related work techniques are generally based around 10 minute averaged data. Since the GP formulations of the current section use high frequency data from the controller, it seems relevant to consider whether a technique using higher frequency data might see improved performance relative to those using 10 minute averages. Since this PhD project is focussed on GP machine learning techniques and their applications to control, the further development of an O&M technique, and comparisons with other such techniques, is out of scope and so this particular direction will not be pursued further in the current work.

7.4 SSGPs Applied to Wind Turbine Aerodynamic Tables

The above rated and second constant speed region regression problem is now considered. Specifically, we look to apply SSGP theory for the identification of $C_Q(\lambda, \beta)$ tables, from which the sensitivity function $\frac{\partial Q}{\partial \beta}$ can be determined. The development of a successful approach turns out to require some further theoretical work which is documented in the following sections. Having developed a workable method, its performance is then tested on simulated and real turbine data.

7.4.1 The Proper Use of Prior Information

When machine learning techniques are developed, it is usually in as much generality as possible. This is in order to maximise usefulness and applicability, while also reflecting the usual gap between researchers and final end users, who are spread across a large range of different fields. One aspect in which there is a lot of variance between different applications is in the availability and form of prior information. In the previous chapter,

the potential gains from leveraging prior information was shown to be the possibility for the SSGP prior covariance function, k, and Sufficient-Subset, $\mathbf{P}$, to be pre-determined and then fixed, removing the need for computationally expensive updating of these parameters during batching. In order for our desired approach to wind turbine dynamics identification to be successfully implemented within turbine controllers, removing the need for optimisation based parameter updating is crucial.

As was mentioned in Section 2.7, prior information for this particular wind turbine application comes in the form of existing wind turbine design C_P/C_Q tables. Importantly, these tables are non-dimensional. As can be seen in Equation 2.7 the C_Q values give the proportion of total possible torque through the rotor which the turbine extracts, rather than an absolute value. As such, C_Q (or equivalently C_P) tables are similar across wind turbines of various sizes. Since tip-speed ratio values also remain approximately the same, even as turbines grow in size[3] [66], the regression equation terms $\lambda^{-2}C_Q(\lambda,\beta)$ will also be similar for different turbines. Therefore, the following sections looks to harness this prior information in order to generate a prior SSGP covariance function and Sufficient-Subset which is applicable to general wind turbine aerodynamic tables.

7.4.2 Initial Attempts at Developing an SSGP Prior and Sufficient-Subset

This section is concerned with the construction of a Sufficient-Subset for application in the above rated and second constant speed region regression problem derived in Section 2.6.1.

Recall that in the formulation of the above rated regression problem, the term which related to the wind turbine aerodynamics was $C_P(\lambda,\beta)/\lambda^3$, which we shall denote C_s (the s denoting 'scaled') for the sake of brevity. With λ denoting tip speed ratio and β the blade pitch angle. Figure 7.16 shows a typical C_s table for a wind turbine, along with its standard operating points. Below rated operation corresponds to $\beta = 0$, and above rated is when $\beta \neq 0$. While the wind turbine control system will keep the

[3]This is due to the fact that larger turbines will generally rotate more slowly in order to maintain an optimal tip-speed ratio.

turbine operating close to the points shown in this figure, turbulence in the wind field will cause the operational trajectory to deviate about that seen here. We must also consider the fact that in order to apply this method to a wind turbine whose C_s tables and operating trajectory we do not know, generous allowances must be made for the operating trajectory to lie above or below the one seen here. Having said that, there are certainly areas of the C_s tables which for all wind turbines can be safely ignored, such as regions which correspond to negative power coefficients. Taking this all together, we shall concentrate on SSGP regression for a generous neighbourhood around the operating points, assuming that points which lie outside this neighbourhood are not going to be visited by the operating wind turbine. This neighbourhood is shown in Figure 7.16.

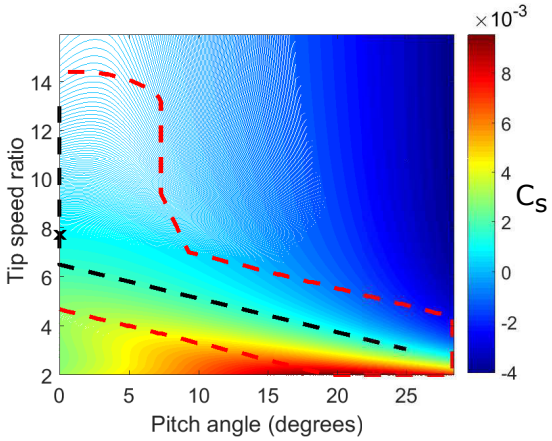

Figure 7.16: Wind turbine C_s table with standard operating points (black dashed line), the operating point corresponding to C_{Pmax} (black x) and the chosen neighbourhood of the operating points on which regression will be focused (red dashed line).

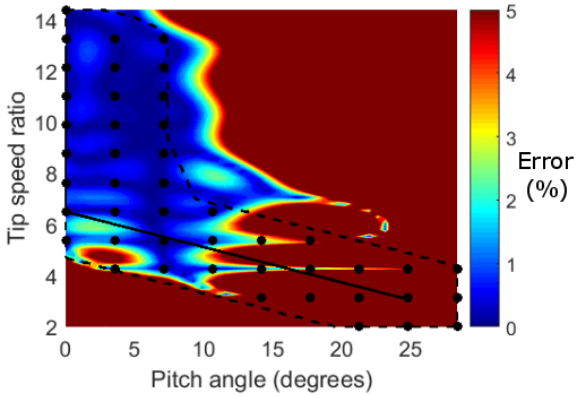

Figure 7.17: SSGP percentage prediction errors across the C_s table. The chosen sufficient subset, $\mathbf{P}$, is shown as black dots.

SSGP regression requires for us to determine the prior covariance function, k, and then choose our Sufficient-Subset of points, $\mathbf{P}$, so as to have a covariance matrix $\Lambda_{\mathbf{PP}}$ which is invertible, while still having a sufficiently high condition number (we are currently not using a Super Sufficient-Subset to see how far we can get without one). The ability of the SSGP to determine function values solely from information at points in $\mathbf{P}$ can then be tested by conditioning on the true values taken by the C_s table at

points in **P**, without any noise (this amounts to using the standard SSGP formulae with the noise variance set to zero in order to interpolate between points). This gives the SSGP exact information about the function values on **P**, and so its predicted values at other points tells us how effectively it can reconstruct the rest of the function. Since no noise is present, a good choice of Sufficient-Subset should be able to reconstruct the function values between points in **P** with very high accuracy.

Figure 7.17 shows the absolute percentage errors (as a percentage of the C_s value at C_{Pmax}) after having determined a prior, k, (via maximum likelihood) across the whole of the chosen neighbourhood. With the lengthscales determined by log-likelihood maximisation, a sufficient subset **P** was chosen which is distributed uniformly throughout the neighbourhood and resulted in a $\Lambda_{\mathbf{PP}}$ condition number of 6×10^8. In this figure it can be seen that the SSGP suffers from very large errors throughout most of the above rated operating region. This turns out to be due to a rapidly changing lengthscale as we move through the neighbourhood. This changing lengthscale is evident if one looks again at the C_s table in Figure 7.16, where we move from a relatively flat and slowly changing plateau in the higher λ regions, along to much steeper and rapidly changing areas towards the far end of the neighbourhood.

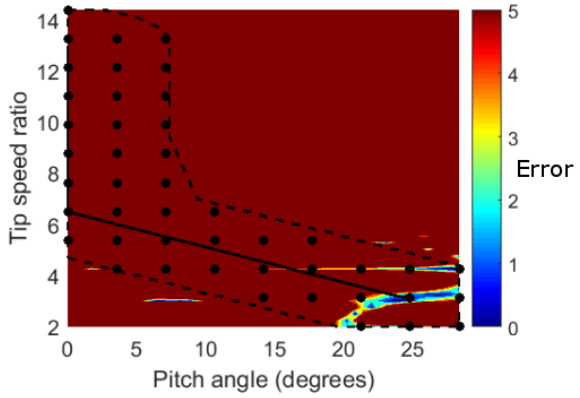

Figure 7.18: Absolute difference between the SSGP expected percentage errors, as given by confidence intervals, and the true SSGP prediction errors. The sufficient subset, **P**, is shown as black dots.

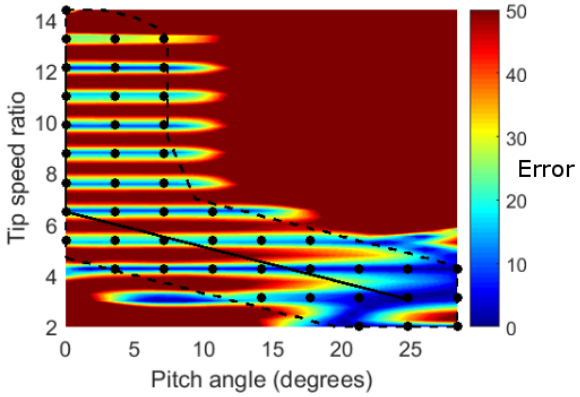

Figure 7.19: Absolute difference between the SSGP expected errors, as given by confidence intervals, and the true SSGP prediction errors. The scale has been increased by a factor of 10 relative to Figure 7.18. The sufficient subset, **P**, is shown as black dots.

While the function value predictions themselves are important, one of the attractions of GPs is that they can also provide confidence intervals, *i.e.* a measure of uncertainty, about their predictions. It turns out that this changing lengthscale, while causing large errors in SSGP function predictions, has an even more catastrophic effect on the GPs ability to give error estimates. Treating one standard deviation of the GPs confidence intervals as the GPs 'expected error' at each point, Figure 7.18 plots the absolute difference between the GPs expected errors and the actual errors for the SSGP predictions (again expressed as a percentage of the C_s value at C_{Pmax}). Figure 7.19 shows the same plot but with the scale increased by a factor of 10.

These figures all clearly show that the ability of the SSGP to make accurate predictions and uncertainty estimates has been completely undermined by the changing lengthscale and that in its current form SSGP (or indeed standard GP regression) is useless for making predictions on the C_s table. This is not very surprising when one considers the fitting of hyperparameters to a function which has a changing lengthscale. Inevitably if one end requires a long lengthscale and the other a short lengthscale, then one will end up with a lengthscale somewhere between the two. Thus at the long lengthscale end information is not carried far enough, increasing uncertainty estimates unnecessarily, whereas at the short lengthscale end information is carried too far, leading to prediction errors due to what essentially amounts to overly strong 'smoothing'. These are exactly the effects seen here. It is therefore necessary to digress briefly in order to address this problem.

7.5 GP Regression on Functions with Varying Lengthscale

When considering GP theory for functional regression, it is always implicitly assumed that the function one is attempting to identify is in fact drawn from the applied GP prior, with its associated covariance structure given by the choice of covariance function. Therefore, when GPs are applied to general functions, not drawn from a GP prior, we are attempting to find a prior covariance function which best captures the structure of

the function underlying the measured data. It is for this reason that a specific covariance function is chosen and its hyperparameters determined via maximum likelihood.

In many practical cases, when a GP model is applied, the standard choices of covariance function will fail to capture the full structure of the underlying function, even when hyperparameters are determined via maximum likelihood. In these cases additional work is required in order to produce good regression results. This section considers a specific example of this: when the function has a changing lengthscale across the functional domain. Various methods which deal with non-stationary lengthscales were outlined in Section 3.9, however, these techniques have been developed to be as general as possible and, as such, introduce large numbers of additional parameters and computationally expensive implementations (MCMC for example). As discussed in Section 2.6.3 the wind turbine dynamics regression task requires efficient implementation and the avoidance of additional processing requirements. Therefore, a refinement of one of the documented techniques, that of transforming the input variable values [32], has been developed. This is an intuitive and computationally lightweight implementation of this idea, suitable for application in the wind turbine regression problem. This simplified approach is documented below.

As in previous sections we focus here on the squared-exponential covariance function (Equation 3.3); the presented results can be applied to other covariance functions with little alteration. A toy regression problem, constructed to exhibit the required traits, is used in order to illustrate the new implementation of the input transformation technique.

Consider the function,

$$g(z) = \frac{\sin\left(z(1+z^2)\right)}{1+z}. \tag{7.1}$$

This function is shown in Figure 7.20 where it can be seen that the lengthscale changes dramatically moving from left to right across the input domain. The function is slowly varying on the left, with the lengthscale steadily decreasing as z increases.

Because of this, when performing GP regression on g there will not be a single inverse-lengthscale d which is appropriate everywhere. When the 'best' value of d is determined, via maximum likelihood, we would therefore expect the final value to

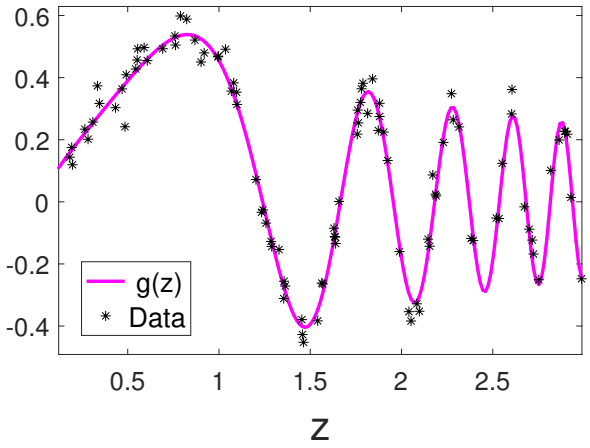

Figure 7.20: An example of a function, $g(z)$, with a lengthscale which changes across its domain.

lie somewhere between the lengthscales which are appropriate for the two extremes. Intuitively, one would expect this to result in overfitting to data in the slowly varying left side of the domain and over-smoothing in the more quickly varying right hand side. To test this hypothesis a standard GP, with squared exponential covariance function, was fitted to noisy measurements of $g(z)$ (shown in Figure 7.20). The resulting GP predictions do in fact display the expected problematic behaviour at the two extremes, as shown in Figures 7.21 and 7.22.

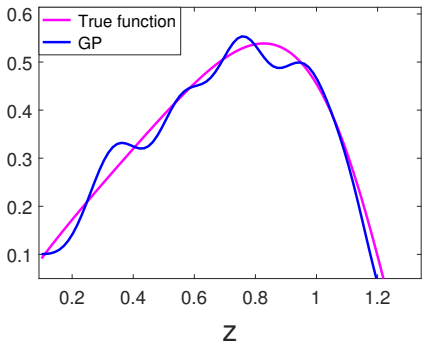

Figure 7.21: Example of overfitting from standard GP regression on part of a function with varying lengthscale.

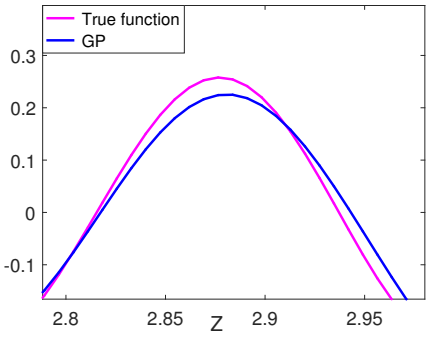

Figure 7.22: Example of over-smoothing from standard GP regression on part of a function with varying lengthscale.

The following sections develop techniques for dealing with these issues when varying lengthscales are present.

7.5.1 Local Covariance and Input Transformations

Assume we have a set of zero-mean random variables, $\{f_z\}$, which are being used to model the values taken by a function f over the domain of a one dimensional input, z. Let k be a squared-exponential covariance function with an inverse-lengthscale, d, which varies across the functional domain. Taking input point z and another point local to z, $z^* = z(1+\epsilon)$ for some small value of ϵ, the inverse-lengthscale becomes a function of z and the *local-covariance* at z can be defined as:

$$k_{\text{loc}}(z) = k(z, z^*) = ae^{-\frac{1}{2}d(z)z^2\epsilon^2}.$$

Letting $h(z) = \epsilon^2 d(z)$, the local covariance structure is then of the form,

$$k_{\text{loc}}(z) = ae^{-\frac{1}{2}h(z)z^2}.$$

We now construct the variable $x = \frac{z}{\alpha(z)}$, by scaling z at each point by $\alpha(z)^{-1}$. Then $z = \alpha(z)x$ and the the local-covariance function expression in terms of x is,

$$ae^{-\frac{1}{2}h(z)\alpha(z)^2x^2},$$

with inverse-lengthscale function $h(z)\alpha(z)^2$. Choosing $\alpha(z)$ such that,

$$h(z)\alpha(z)^2 = b \text{ (constant) for all } z, \tag{7.2}$$

it follows that as a function of x, f has constant lengthscale across the domain.

Assuming we know $h(z)$, Equation 7.2 is satisfied by setting;

$$\alpha(z) = \sqrt{\frac{b}{h(z)}},$$

and this in turn allows us to determine the variable x as,

$$x = \frac{z}{\alpha(z)} = \left(\sqrt{\frac{h(z)}{b}} \right) z = c \left(\sqrt{d(z)} \right) z,$$

where $c = \epsilon/\sqrt{b}$ is an arbitrary constant (since b is arbitrary). In practise it has been found that once the local inverse-lengthscale function, $d(z)$, is known, the constant c should then be chosen in order to give the transformed varaible;

$$x = c \left(\sqrt{d(z)} \right) z, \tag{7.3}$$

a sensible order of magnitude (*i.e* around 1) across the domain of interest; this has been found to stabilise numerics and improve the speed of GP optimisation for the parametrisable case considered next.

7.5.2 Parameterisable Local Inverse-Lengthscale Functions

If the function $d(z)$ has a parametric form, then it is possible to include a determination of a 'best' input transform into the log-likelihood procedure for GP regression by treating the function's parameters as additional hyperparameters.

We give an example of this for the case $d(z) = az^b$. Applying Equation 7.3 results in a transform of,

$$z \to (c\sqrt{a}) z^{1+\frac{b}{2}}. \tag{7.4}$$

If the parameters a and b are known then the above equation gives the transform; if, however, we wish to determine an optimal a and b then this transform can be expressed in the same way as the original $d(z)$ function,

$$z \to \tilde{a} z^{\tilde{b}},$$

with $\tilde{a}$ and $\tilde{b}$ the parameters to be determined. Note that this is the case due to the specific functional form of $d(z)$ here, the same would not be true if, for example, $d(z)$ is a polynomial with at least two non-zero terms.

The value for $\tilde{a}$ will simply scale the GPs inverse-lengthscale hyperparameter and

so (as discussed for choosing c above) $\tilde{a}$ is chosen so as to keep the transformed variable at a reasonable order of magnitude. Therefore, only $\tilde{b}$ needs to be determined during the log-likelihood optimisation procedure and this requires the first and second order partial derivatives of the covariance function with respect to $\tilde{b}$. Consider a squared exponential covariance function for input vectors of length n, and with a transformed first input dimension;

$$\begin{aligned} k([x_1, z_2, \ldots, z_n], [x_1', z_2', \ldots, z_n']) &= a \exp\left(-\frac{1}{2} d_1 (x_1 - x_1')^2 - \frac{1}{2} \sum_{i=2}^{n} d_i (z_i - z_i')^2 \right) \\ &= a \exp\left(-\frac{1}{2} d_1 \left(\tilde{a}(z_1)^{\tilde{b}} - \tilde{a}(z_1')^{\tilde{b}} \right)^2 - \frac{1}{2} \sum_{i=2}^{n} d_i (z_i - z_i')^2 \right). \end{aligned}$$

Recall that all the d_j are determined by their log-hyperparameters ϕ_{d_j} (since they must be greater than or equal to zero). Since $\tilde{b}$ could be either positive or negative we do not use a log-hyperparameter and unconditionally optimise $\tilde{b}$ itself. The required partial derivatives for optimisation which are additional to those used in standard GP regression are then:

$$\begin{aligned} \frac{\partial k}{\partial \tilde{b}} &= -d_1 \left(x_1 - x_1' \right) \left(\ln(z_1) x_1 - \ln(z_1') x_1' \right) k \\ \frac{\partial^2 k}{\partial \phi_{d_1} \partial \tilde{b}} &= \frac{\partial k}{\partial \tilde{b}} - \left(\frac{1}{2} d_1 (x_1 - x_1')^2 \right) \frac{\partial k}{\partial \tilde{b}} \\ \frac{\partial^2 k}{\partial \phi_{d_i} \partial \tilde{b}} &= -\left(\frac{1}{2} d_i (z_i - z_i')^2 \right) \frac{\partial k}{\partial \tilde{b}} \quad \text{for } i > 1 \\ \frac{\partial^2 k}{\partial \tilde{b}^2} &= -d_1 \left(\ln(z_1) x_1 - \ln(z_1') x_1' \right)^2 k \\ &\quad -d_1 \left(x_1 - x_1' \right) \left(\ln(z_1)^2 x_1 - \ln(z_1')^2 x_1' \right) k \\ &\quad -d_1 \left(x_1 - x_1' \right) \left(\ln(z_1) x_1 - \ln(z_1') x_1' \right) \frac{\partial k}{\partial \tilde{b}}. \end{aligned}$$

More generally, if the required transform is of type,

$$z \to x = t(z, \{\beta_j\}),$$

for some function t with parameters $\{\beta_j\}$, the additional partial derivatives for maximum likelihood optimisation for each j are:

$$\frac{\partial k}{\partial \beta_j} = -d_1 \left(t(z_1) - t(z_1')\right) \left(\frac{\partial t}{\partial \beta_j}(z_1) - \frac{\partial t}{\partial \beta_j}(z_1')\right) k \tag{7.5}$$

$$\frac{\partial^2 k}{\partial \phi_{d_1} \partial \beta_j} = \frac{\partial k}{\partial \beta_j} - \left(\frac{1}{2} d_1 \left(t(z_1) - t(z_1')\right)^2\right) \frac{\partial k}{\partial \beta_j} \tag{7.6}$$

$$\frac{\partial^2 k}{\partial \phi_{d_i} \partial \beta_j} = -\left(\frac{1}{2} d_i \left(t(z_1) - t(z_1')\right)^2\right) \frac{\partial k}{\partial \beta_j} \quad \text{for } i > 1 \tag{7.7}$$

$$\begin{aligned} \frac{\partial^2 k}{\partial \beta_l \partial \beta_m} = & -d_1 \left(\frac{\partial t}{\partial \beta_l}(z_1) - \frac{\partial t}{\partial \beta_l}(z_1')\right) \left(\frac{\partial t}{\partial \beta_m}(z_1) - \frac{\partial t}{\partial \beta_m}(z_1')\right) k \\ & -d_1 \left(t(z_1) - t(z_1')\right) \left(\frac{\partial^2 t}{\partial \beta_l \partial \beta_m}(z_1) - \frac{\partial^2 t}{\partial \beta_l \partial \beta_m}(z_1')\right) k \\ & -d_1 \left(t(z_1) - t(z_1')\right) \left(\frac{\partial t}{\partial \beta_m}(z_1) - \frac{\partial t}{\partial \beta_m}(z_1')\right) \frac{\partial k}{\partial \beta_l} \end{aligned} \tag{7.8}$$

7.5.3 Input Transformation Example

We return to the function $g(z)$ (Equation 7.1) shown in Figure 7.20. In order to determine the form of $d(z)$, local lengthscales are found by partitioning the input domain and performing maximum-likelihood optimisation for the inverse-lengthscale parameter centered on each section. A partition into seven sections results in the local inverse-lengthscales shown in Figure 7.23, which can be seen to increase roughly exponentially. Therefore, if $d(z)$ is of the form,

$$d(z) = ae^{bz},$$

it follows that the required input transformation will be,

$$t(z) = \tilde{a}e^{\tilde{b}z}z,$$

with $\tilde{a}$ arbitrary and an optimal $\tilde{b}$ to be determined.

From the fit in Figure 7.23 it follows that an optimal $\tilde{b}$ will be around 2 and hence there is no danger of extreme orders of magnitude for the transformed coordinates. We therefore set $\tilde{a} = 1$ for the sake of simplicity. Using the same data as for the GP predictions in Figures 7.21 and 7.22; $\tilde{b}$ was determined, along with the other hyperparameters, as part of a maximum-likelihood optimisation using the theory of the previous

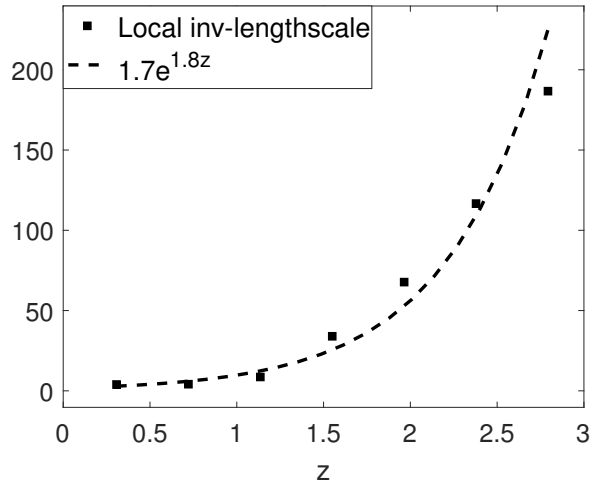

Figure 7.23: Local inverse-lengthscale fitting

section, in particular the partial derivatives of Equations 7.5-7.8. This procedure gave an optimal transformation parameter of $\tilde{b} = 0.5033$ and the resulting GP predictions are shown in Figures 7.24 and 7.25, along with the standard GP results seen previously. The Input Transformation approach has clearly improved the GP predictions in both the high and low variance regions of the test function, preventing over-fitting at one end and excessive smoothing at the other, as was desired.

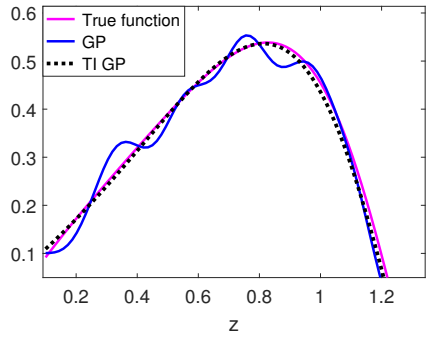

Figure 7.24: Example of overfitting from standard GP regression on part of a function with varying lengthscale.

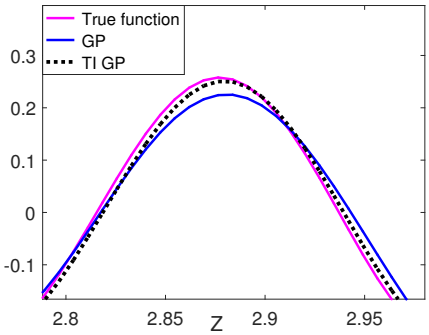

Figure 7.25: Example of over-smoothing from standard GP regression on part of a function with varying lengthscale.

It should be noted that while Input Transformations can be used to improve GP predictions in the presence of varying lengthscales, as demonstrated here; there are restrictions

on when this method can be applied. This is due to the fact that the transformation $t(z)$ must be monotonic in order for its application to be appropriate. This consideration will render these transformations unsuitable for some applications, however, the flexibility and ease of application offered by the above formulations makes this a powerful tool in the many cases where it can be applied. One of these cases being the identification of wind turbine dynamics.

7.6 Input Transformations for Wind Turbine Aerodynamic Tables

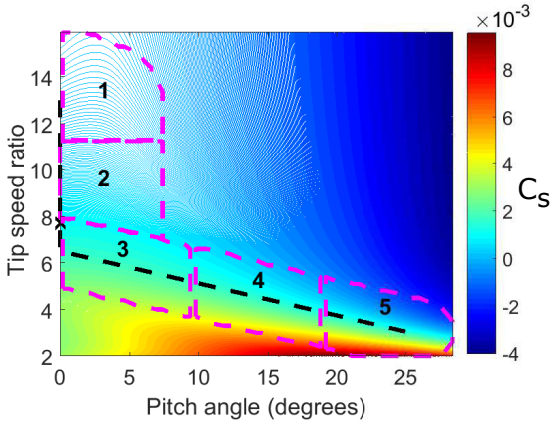

Figure 7.26: C_s table along with 5 local regions in which local inverse-hyperparameter values will be determined.

In order to avoid the issues presented by varying lengthscales, we apply the formulations of the previous section to the wind turbine aerodynamic tables problem. The first task being to look at determining a local inverse-lengthscale function.

We start by splitting the C_s neighbourhood into 5 local regions (chosen to be representative of the various local lengthscale regions through which the design operating curve passes) as shown in Figure 7.26. Log-likelihood optimisation is then performed for the points within each local region in order to give local lengthscales for both λ and β. This is done by choosing a random subset of points from throughout a given region, adding a known and small amount of noise to the measurements (this allows us to perform matrix inversion without singularity issues as we are dealing with a full

data set here, rather than a small subset as in SSGP) and then performing maximum likelihood optimisation with the noise variance fixed at its known value. Since the addition of noise represents a stochastic element here, the hyperparameters for a given region can vary slightly each time this process is run. We therefore ran the process 50 times in each region in order to determine means and standard deviations of the local inverse-lengthscale hyperparameter values. These are shown in Figures 7.28 and 7.27 for λ and β respectively. The hyperparameter values are all plotted against the central value of the relevant variable in each given region. Note that the x-axis for tip speed ratio has been reversed since this makes the local inverse-hyperparameters changing from left to right correspond to moving in the same direction through the local regions of the C_s table.

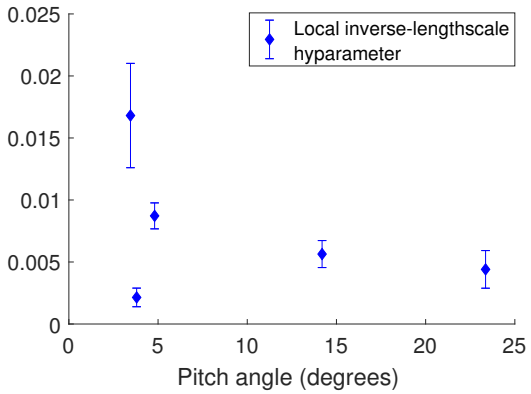

Figure 7.27: Means and standard deviations (from 50 runs each) of local inverse-lengthscale hyperparameters for β. Each point is plotted against the central β value in the given region.

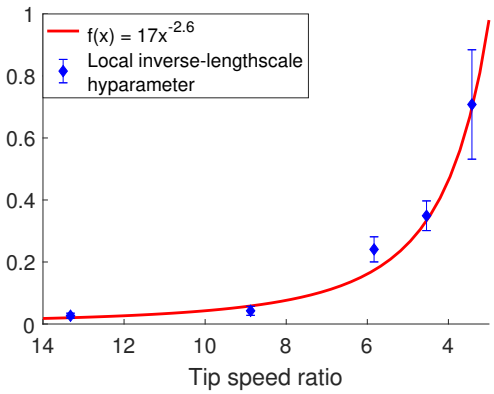

Figure 7.28: Means and standard deviations (from 50 runs each) of local inverse-lengthscale hyperparameters for λ. Each point is plotted against the central λ value in the given region.

In Figure 7.27 there is no clear functional form for the variations in lengthscale across the domain, however, it is also evident that while the local inverse-lengthscales for β are changing, they are not changing by a huge amount. This result is not surprising if one looks again at the C_s table (Figure 7.16), where the variations in function values are clearly much slower and more consistent in the β direction. It turns out that the small variations in local inverse-lengthscales with respect to β are not the cause of the issues found to effect SSGP regression for the wind turbine C_s table.

In Figure 7.28 we see that, for λ, the local inverse-lengthscale hyperparameter values vary a lot over the domain; with the largest mean values being as much as 26 times the smallest. Therefore, a transformation of the λ variable is appropriate. The mean values in Figure 7.28 can also be seen to be well fitted by the function $f(x) = 17x^{-2.6}$. From Equation 7.4, this suggest a transform of,

$$\lambda \to c\lambda^{1-\frac{2.6}{2}} \approx c\lambda^{-\frac{1}{3}}.$$

Since the regions used to determine these local hyparameters were chosen arbitrarily and also because there appears to be a parametric form to the inverse-lengthscale variations here; rather than simply using the above transformation, the theory of Section 7.5 was used to include a transform of λ, of the form:

$$\lambda \to c\lambda^{\tilde{b}},$$

in the log-likelihood procedure for determining the prior, thus allowing an optimal transform for λ to be determined. In order to choose a value for the constant, c, we note that if the exponent from function fitting is $-\frac{1}{3}$, then we would expect the final value of $\tilde{b}$ to be fairly close to this. We therefore set $c = 10$ in order to have transformed variables which lie somewhere between 0 and 10.

Running this optimisation procedure results in an exponent of $\tilde{b} = -0.665\ldots \approx -\frac{2}{3}$. Therefore, using the transform,

$$\lambda \to 10\lambda^{-\frac{2}{3}},$$

and the inverse-lengthscale hyperparameters determined at the same time; a Sufficient Subset, $\mathbf{P}$, is determined and the SSGP conditioned on noiseless data as before. Figure 7.29 show the prediction errors of the new, transformed-input SSGP (tSSGP) and Figure 7.30 shows the differences between its predicted errors and the true errors.

Figures 7.29 and 7.30 show a remarkable improvement compared to the same plots for the untransformed SSGP. We see that, having transformed the λ variable, the tSSGP is able to make very accurate predictions across almost the entire neighbourhood, while also generating very accurate confidence intervals. Note that the Sufficient Subset was

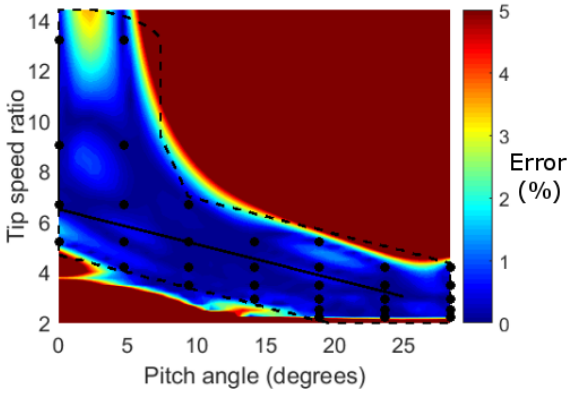

Figure 7.29: tSSGP percentage prediction errors across the C_s table. The chosen sufficient subset, $\mathbf{P}$, is shown as black dots.

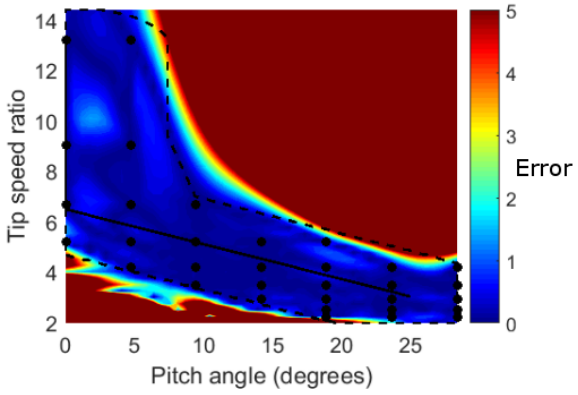

Figure 7.30: Absolute difference between the tSSGP percentage expected errors, as given by confidence intervals, and the true transformed input SSGP prediction errors. The sufficient subset, $\mathbf{P}$, is shown as black dots.

determined in the transformed coordinate system, since it is here where the lengthscales are constant; the resulting condition number of $\Lambda_{\mathbf{PP}}$ being 9×10^7. In the untransformed coordinates this can be seen to result in increased densities of points in areas where function values change more quickly. In Figure 7.29 the only significant predictive errors occur in the higher λ regions of the neighbourhood, and at worst the size of these errors is 2 or 3%. These errors could certainly be much reduced by the addition of a single extra point in $\mathbf{P}$ at around $(\lambda, \beta) = (14.5, 2.5)$. Figures 7.31 and 7.32 demonstrate how the additions of extra points can greatly improve both the point predictions and error predictions for the C_s tables.

7.6.1 Parameters for General C_s Tables

So far the current chapter has only considered the single C_s table example above. However, in order to ensure that the hyperparameters, including the transformation exponent, are suitable for as wide a range of wind turbines as possible the above procedure was carried out on a set of 6 different C_p tables from commercially available wind turbines. Full transformation optimisation was used in each case and the average of each parameter taken across the six C_s tables. These averaged values are what will

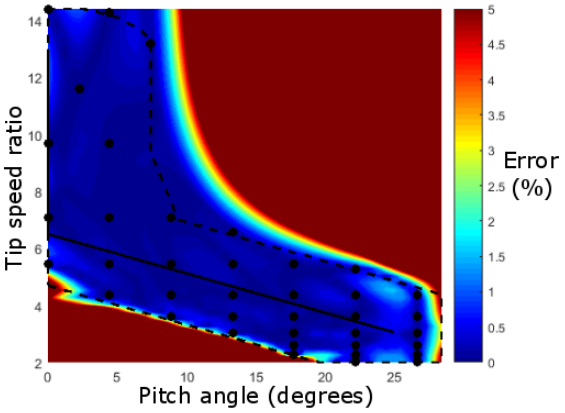

Figure 7.31: tSSGP percentage prediction errors across the C_s table. The chosen sufficient subset, $\mathbf{P}$, is shown as black dots.

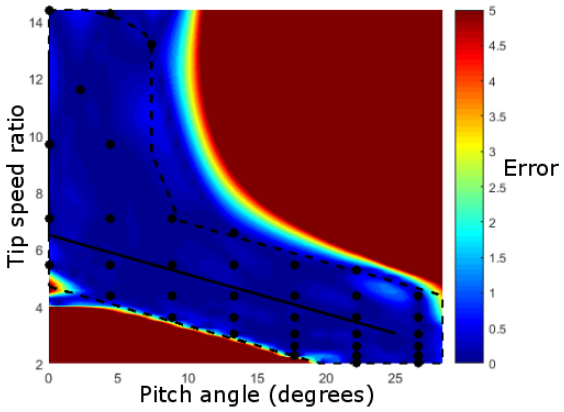

Figure 7.32: Absolute difference between the tSSGP percentage expected errors, as given by confidence intervals, and the true transformed input SSGP prediction errors. The sufficient subset, $\mathbf{P}$, is shown as black dots.

Hyperparameter	Value
amplitude	12.2723
λ inverse-lengthscale	2.5448
β inverse-lengthscale	0.0212
transformation exponent	-0.5092

Table 7.1: Average hyperparameter values across 6 tested C_p tables.

be used going forward, they are given in Table 7.1. The covariance structure resulting from these parameters, along with the mean function (obtained by averaging the 6 available C_P tables) therefore forms a general GP prior for wind turbine C_p tables.

7.6.2 The Need for a Super Sufficient-Subset

The presence of areas of increased percentage error, seen in Figure 7.29 even after choosing a set $\mathbf{P}$ which results in a suitably high condition number, indicates that while input transformations have greatly improved the GPs ability to recreate the underlying C_s table, areas of varying lengthscale still remain. It was shown above that this issue can be partially solved through the addition of some extra points in $\mathbf{P}$ where the errors occur. However, this is not a practical solution in general since the different C_s tables being used all require additional points in different places. This means that

a Super Sufficient-Subset, as developed in Section 6.5, is necessary in order to have a single Sufficient-Subset which can be used in the general case where the underlying table is not known. Since we are moving to a Super Sufficient-Subset the invertibility of $\Lambda_{\mathbf{PP}}$ is no longer a requirement and hence we can also move away from focusing on the boundary shown in the above figures. For the sake of simplicity it is now possible to define a set $\mathbf{P}$ which covers the entire C_s table. Note that in order to determine D_1 in the saturation formula, a condition number limit of 1×10^8 is used. Figures 7.33 and 7.34 show the chosen Super Sufficient-Subset in the transformed and untransformed coordinates respectively. This set $\mathbf{P}$ consists of 726 points with spacings determined by the lengthscale ratio formula of Section 6.4.3.

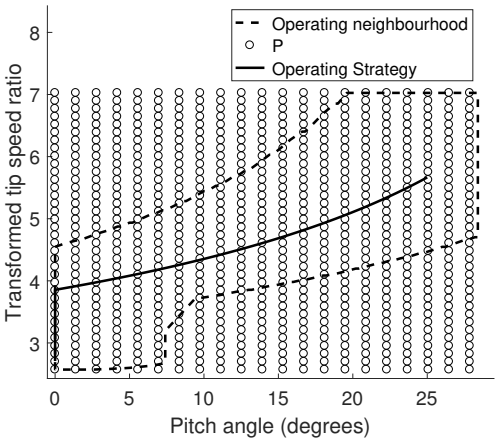

Figure 7.33: Super Sufficient-Subset in transformed coordinates for C_s table identification.

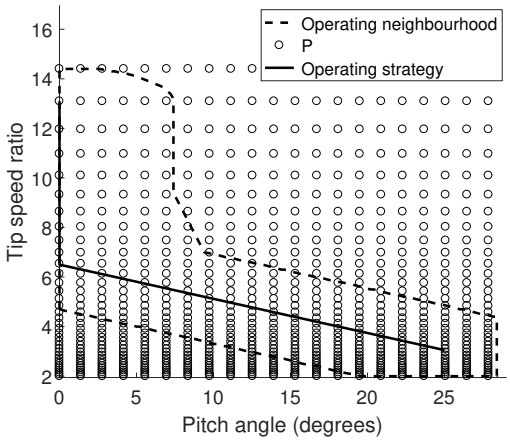

Figure 7.34: Super Sufficient-Subset in untransformed coordinates for C_s table identification.

Figures 7.35-7.38 show the interpolation errors for the chosen Super Sufficient-Subset $\mathbf{P}$ for 4 out of the 6 C_s tables. As can be seen in these figures, the SSGP with this choice of $\mathbf{P}$ is able to recreate all of the C_s tables accurately with the same Sufficient-Subset, hyperparameter values and input transformation. The errors can be seen to be almost always below 3%, and are generally below 1%. Thus demonstrating that the chosen Super Sufficient-Subset can accurately capture C_s table information, especially within our operating region of interest, indicated in the figures.

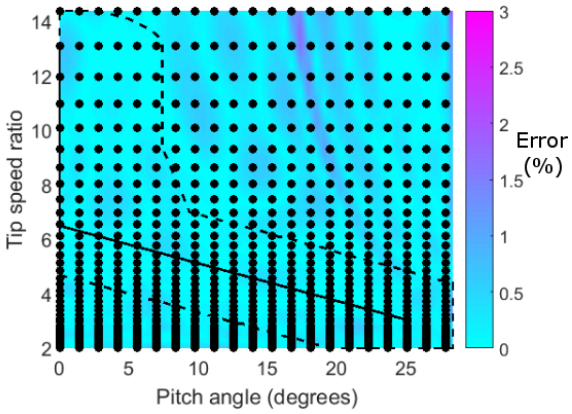

Figure 7.35: Absolute percentage errors of C_s interpolation for C_s table 1 using a Super Sufficient-Subset, shown by black dots.

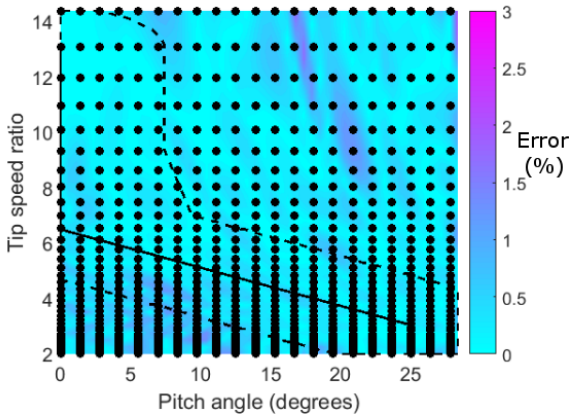

Figure 7.36: Absolute percentage errors of C_s interpolation for C_s table 2 using a Super Sufficient-Subset, shown by black dots.

7.7 Implementation Issues

Having determined a suitable prior covariance function and Super Sufficient-Subset. The ability of the SSGP to learn aerodynamic table information was tested using data from the Supergen Exemplar wind turbine model. At this point it became apparent that there was an issue with this particular approach which had not been foreseen. Recall that in the derivation of the regression equation in operating regions 4 and 5 (Equation 2.11) the main source of noise is due to the noisy anemometer wind speed measurements which, via separability theory, results in the measured data being interpreted as noisy measurements corresponding to the estimated, rather than actual, input locations. Figure 7.39 shows a one-dimensional example of this scenario on a toy problem. In this example, measurements of the true function are correct, but Gaussian noise has been added to the input values. As can be seen here, the measured data does indeed looking like noisy measurements of the true function. In general, this resulting noise will not necessarily be Gaussian, but in the wind turbine case, separability theory tells us that it will be Gaussian or well approximated as such.

This should in theory allow for regression to go ahead with no issues, however, the wind turbine is attempting to track a specific strategy in terms of pitch angle and tip speed ratio, as shown in Figure 7.16 and therefore, if the controller is doing its job, the

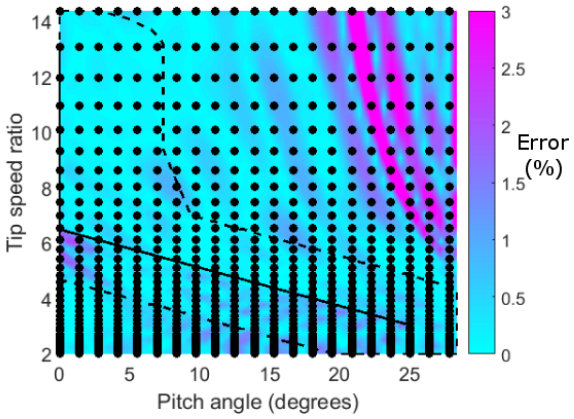

Figure 7.37: Absolute percentage errors of C_s interpolation for C_s table 3 using a Super Sufficient-Subset, shown by black dots.

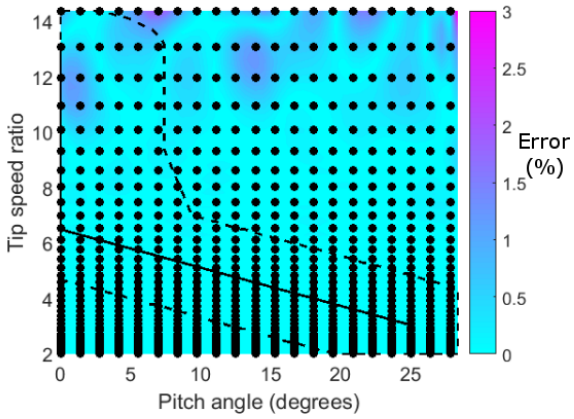

Figure 7.38: Absolute percentage errors of C_s interpolation for C_s table 4 using a Super Sufficient-Subset, shown by black dots.

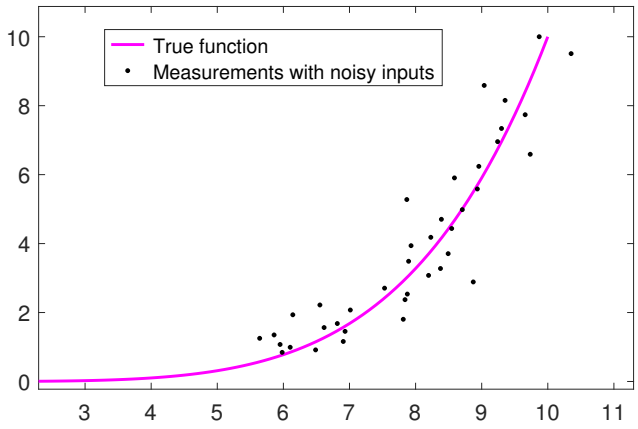

Figure 7.39: A one dimensional example of input noise resulting in noisy measurements of an underlying function.

range of input locations visited is ideally very narrow. In this more restrictive scenario the result of input noise can be catastrophic for regression. Figure 7.40 illustrates this second scenario using our one-dimensional example.

It is clear here that regression using the data in Figure 7.40 will never result in a good approximation of the underlying function, indeed, one would expect the resulting prediction to be approximately perpendicular to the true function. The problem here is practical, rather than theoretical, and comes down to the magnitude of the input noise in relation to the size of the interval over which measurements are taken. Unfortunately,

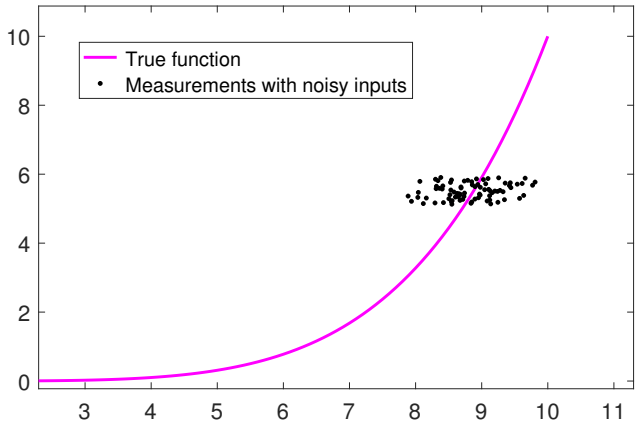

Figure 7.40: Example of resulting data when input measurement noise is present and true input values are restricted to within a narrow range. Here the true input locations are all between 8.75 and 9.

for wind turbine C_s table identification, it turns out we fall foul of the above scenario.

Figures 7.41 and 7.42 show C_s tables (note that k denotes the scaling of the tables to a sensible order of magnitude) along with measured data from simulation using the Supergen wind turbine model. In Figure 7.41 there is no noise on measured wind speed values, while in Figure 7.42 noise has been added with a standard deviation of 1 m/s.

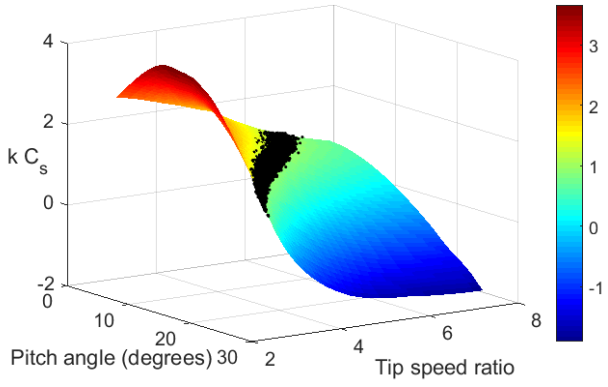

Figure 7.41: C_s table with measured data from simulation, note that the points visited constitute a narrow band across the table.

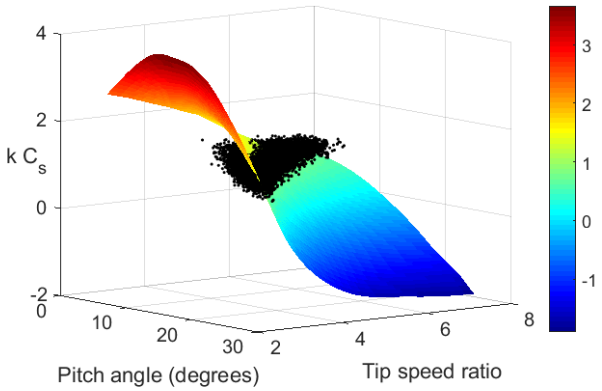

Figure 7.42: Effect of input noise on measured data for the wind turbine C_s table.

The same effect as was seen in the above one-dimensional example is clearly also

present in this wind turbine data. Measured values can be seen to splay out due to the narrow range of points visited, resulting in a dataset which is clearly unsuitable for regression. Figure 7.43 shows the result of attempting GP regression on such data; as expected, the resulting surface is effectively perpendicular to the true surface within the range of measured values.

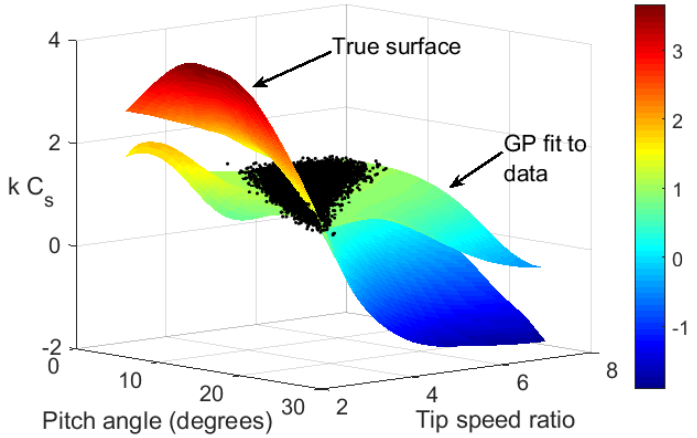

Figure 7.43: SSGP fit to noisy data when band of true input values is too narrow with respect to measurements noise.

7.8 Reformulation of the Regression Problem

As has been shown in the previous section, the restricted area of operation seen by a wind turbine in above rated conditions means that even small values of noise on wind speed measurements renders the above regression approach infeasible. Therefore, a new approach is necessary.

Since it has been found to be the noisy wind speed measurements which are the main source of noise, we now develop a formulation for which the wind speed is explicitly considered (rather than tip-speed ratio), along with pitch angle. Recall that the formula for aerodynamic torque is,

$$Q_a(v, \omega, \beta) = \frac{1}{2}\rho A R v^2 C_Q(\lambda, \beta). \tag{7.9}$$

From this equation, it follows that measurements taken at any given ω can be adjusted

such that $\omega = \omega_0$ (where ω_0 is rated rotor speed) while keeping λ fixed. This is done using the following transformations;

$$\omega \rightarrow \omega_0 \tag{7.10}$$

$$v \rightarrow \tilde{v} = \left(\frac{\omega_0}{\omega}\right) v \tag{7.11}$$

$$Qa \rightarrow \tilde{Q}_a = \left(\frac{\omega_0}{\omega}\right)^2 Qa. \tag{7.12}$$

Since ω is now held constant at ω_0, the transformed measurement set $\{(\tilde{v}, \beta, \tilde{Q}_a)\}$ contains data of the form,

$$\tilde{Q}_a = \frac{1}{2}\rho A R \tilde{v}^2 C_Q \left(\frac{\omega_0 R}{\tilde{v}}, \beta\right) = \frac{1}{2}\rho A R \tilde{v}^2 C_Q(\tilde{v}, \beta),$$

and so now $\tilde{Q}_a$ is a function of $\tilde{v}$ and β.

Note that while operating in sections 4 and 5 of the operating strategy shown in Figure 2.5, and away from the stall region, the relationship between λ and Q_a (and so also the relationship between $\tilde{v}$ and $\tilde{Q}_a$) is invertible for each fixed value of β. More formally we define,

$$f_\beta(\tilde{v}) = \tilde{Q}_a(\tilde{v}, \beta), \tag{7.13}$$

and then for each value of β, $f_\beta(\tilde{v})$ is invertible when avoiding the stall region. Invertibility is important here in terms of forming a solveable regression problem because it now allows us to reinterpret the measurements $\{(\tilde{v}, \beta, \tilde{Q}_a)\}$ as relating an output wind speed (which contains the most significant source of noise) to input values of torque and pitch angle. We are therefore considering regression for noisy measurements of the function;

$$\tilde{v}\left(\beta, \tilde{Q}_a\right) = f_\beta^{-1}(\tilde{Q}_a). \tag{7.14}$$

Regression in this form will be referred to as the *wind speed formulation*. This will allow for the large noise terms which will be present in wind speed measurements to be determined explicitly by the GP and also feature additively, rather than on the denominator, as was the case for λ. The transformation of measured wind speeds given by Equation 7.11 will also need to be accounted for in terms of the noise present,

however, additive noise will simply be transformed in the same way and hence a scaling of the noise covariance matrix is all that's required and can be easily incorporated into the SSGP procedure.

In order to develop a general prior and Sufficient-Subset which is suitable for any wind turbine's aerodynamic tables, it is necessary to work with a measure of aerodynamic torque which is not dependent on turbine size. We therefore choose our torque variable to be measurements of the scaled torque $\tilde{Q}_a/(\frac{1}{2}\rho AR) = \tilde{v}^2 C_Q$. The following section documents the development of a GP prior and Sufficient-Subset for this regression framework.

7.9 Developing a New GP Prior and Sufficient-Subset

We now look to develop a GP prior and Sufficient-Subset for the wind speed formulation of the wind turbine dynamics regression problem. Figure 7.44 shows an example contour plot of the function $v(\beta, v^2 C_Q)$ for a commercially available wind turbine.

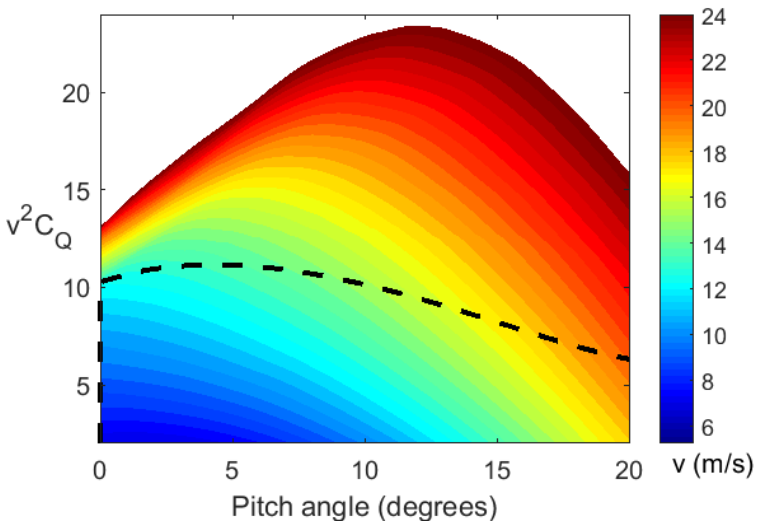

Figure 7.44: Contour plot of windspeed, v, interpreted as being a function of pitch angle and non-dimensional torque, $v^2 C_Q$. The design operating strategy is given by the dashed line.

First, a non-zero mean for the function is determined by averaging values across the six available sets of turbine dynamics. The process for determining hyperparameters is

Hyperparameter	Value
amplitude	0.4644
β inverse-lengthscale	0.0396
v^2C_Q inverse-lengthscale	0.0832

Table 7.2: Average hyperparameter values across 6 tested turbines for the wind speed formulation of the regression problem.

then identical to that used for the original formulation, except in each case our non-zero prior mean is first subtracted from the data before hyperparameters are learned. For each available set of aerodynamic tables the hyperparameter values were determined via maximum-likelihood. The mean of the results for each hyperparameter was then taken to be its fixed value going forward. Table 7.2 shows these final values.

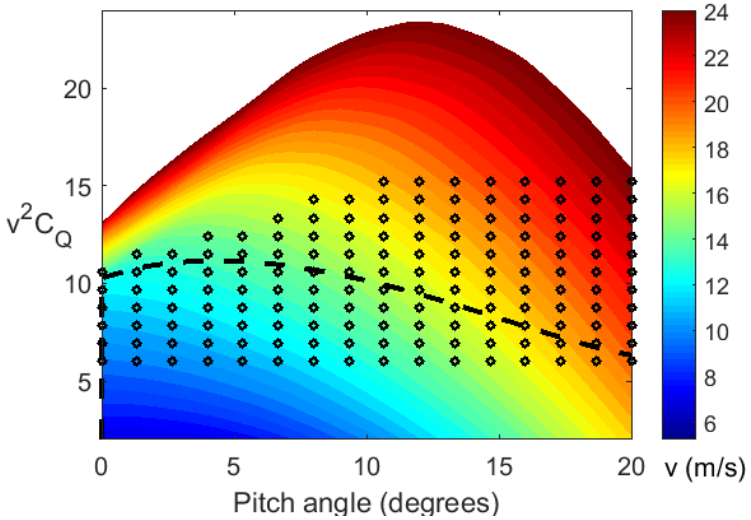

Figure 7.45: Contour plot of windspeed, v, interpreted as being a function of pitch angle and non-dimensional torque, v^2C_Q. The chosen Super Sufficient-Subset is also depicted by black circles.

A Super Sufficient-Subset was found to be necessary in order for the SSGP to be able to accurately reproduce the underlying function and, in this case, without the need for input transformations. The chosen Super Sufficient-Subset, $\mathbf{P}$, contains 153 points and is shown in Figure 7.45. The shape of $\mathbf{P}$ in the top left corner is to avoid stall, the region where the monotonic assumption for f_β breaks down. The condition number of the covariance matrix, $\Lambda_{\mathbf{PP}}$, for these points is around 10^{19} and so this choice of $\mathbf{P}$

would be impossible without the Super Sufficient-Subset theory developed in Section 6.5. With this choice of **P** the SSGP model can reproduce the function at intermediate points with incredibly high accuracy, as shown in Figure 7.46. These same levels of accuracy are seen across the available set of 6 aerodynamic tables.

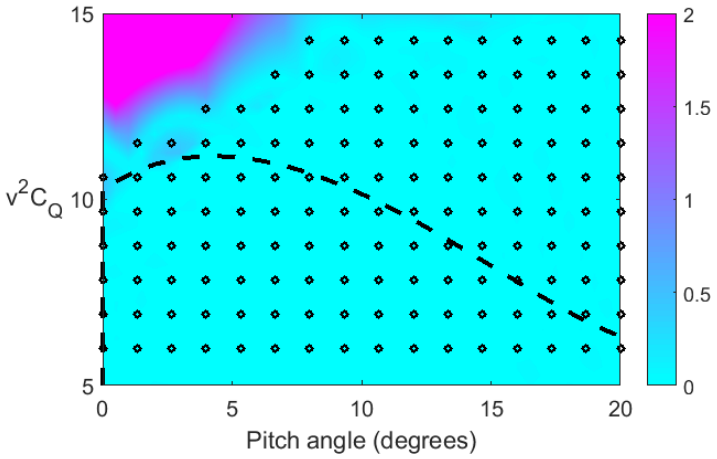

Figure 7.46: Absolute percentage errors for SSGP reproduction of the function $v(\beta, v^2 C_Q)$ using data on **P** only.

7.10 Rated Power and Second Constant Speed Region Dynamics Identification

7.10.1 Measured Data and Drivetrain Losses

When considering real turbine data, drivetrain losses will be present and hence the measured torque values are of the form,

$$N\hat{Q}_g + J\dot{\hat{\omega}}_r = Q_{aer} - \mathcal{L}(\hat{\omega}_r),$$

where the terms on the left hand side are the measurements available from a wind turbine. For above rated operation the controller is attempting to maintain the rotor speed at $\omega_r = \omega_0$. There will still be some variation present in the rotor speed values, making it necessary to apply the transformations of Section 7.8. In terms of the loss function, these small variations in rotor speed will not result in significant changes in

loss values and so the losses can be treated as a constant offset $\mathcal{L}$ on measured torques. As already discussed, a constant term will not effect derivative calculations and so from now on we shall ignore the loss offset in equations, simply noting that it is present in the data.

Measurements of $v^2 C_Q$ are then formed (including the transform to ω_0) as,

$$v^2 C_Q = \left(\frac{\omega_0}{\omega}\right)^2 \left(\frac{N\hat{Q}_g + J\dot{\hat{\omega}}_r}{\frac{1}{2}\rho A R}\right),$$

which follows from Equations 2.7 and 7.12. Wind speeds measurements are transformed according to Equation 7.11.

7.10.2 Estimating Wind Speed Measurement Noise

The effective wind speed of an operating wind turbine can be estimated if the C_Q tables are known. This involves reverse engineering the C_Q tables to determine the wind speed which would result in the measured torque value along with each measured rotational speed and pitch angle. While this work is considering cases where the tables are not known, this information is available for the turbine from which the real data was collected. Since this approach also doesn't account for losses, the resulting effective wind speeds can only be considered an estimate, however, the proposed analysis should at least indicate the levels of noise that can be expected between measured and effective wind speeds.

This analysis was performed for the available real turbine data, resulting in an estimated effective wind speed time history. The results of subtracting the effective wind speed values from anemometer measurements is shown in Figure 7.47. An offset of 0.85m/s is present between the two, with a noise standard deviation about this offset of 0.89m/s. The offset here does not pose any real problem since this 'adjusted' wind speed can simply become our measure of wind speed for this regression problem. In essence we are simply defining a new frame of reference for wind speed measurements. The key is to then ensure that this same reference frame is used consistently throughout the regression process. This therefore requires that the trajectory along which derivatives

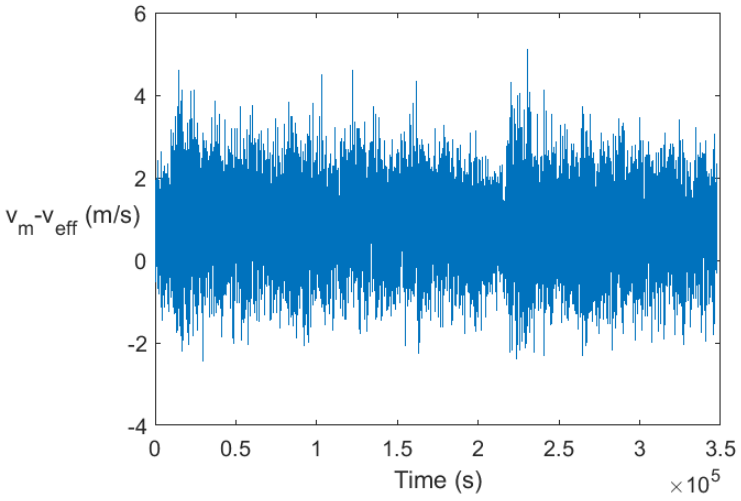

Figure 7.47: The difference between measured wind speed and estimated values of effective wind speed obtained via reverse engineering of known aerodynamic tables. This data has a mean of 0.85 and standard deviation of 0.89.

are taken to determine the aerodynamic sensitivity functions is also given with respect to this frame of reference. It turns out that this is not only possible, but also necessary, as discussed in the following section.

With respect to measured wind speeds the reverse engineering analysis indicates that noise levels on measurements should be expected to have standard deviations of around 0.89m/s. The extent to which the residuals, after removing the offset, can be considered Gaussian was also investigated. Figure 7.48 shows a histogram of these residuals which looks promisingly Gaussian. A Kolomogorov-Smirnov test was performed on the residuals which did not reject the null hypothesis that the data was Gaussian, returning a p-value of roughly 30%.

Therefore, when adding noise to wind speed measurements taken from simulated data we will consider Gaussian noise with standard deviations to 0.1, 0.5 and 1 m/s in order to have a range which includes the level estimated from real data.

7.10.3 Identifying the Operating Strategy

In above rated wind conditions, when pitch regulation is active, the turbine is attempting to maintain rated power by setting the pitch angle to the value which corresponds

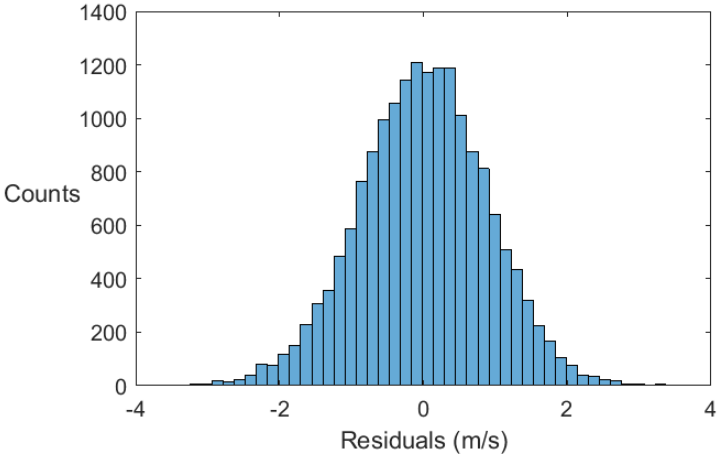

Figure 7.48: Residuals about offset between measured wind speeds and estimated values of effective wind speeds.

to rated power at each wind speed. An example curve for design values of pitch and wind speed is shown in Figure 7.49, from [67].

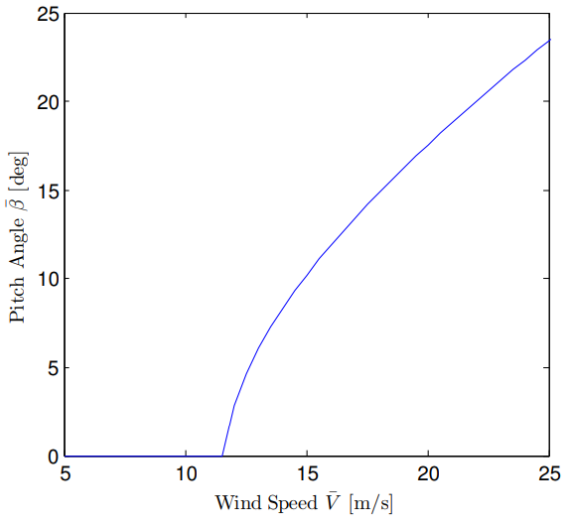

Figure 7.49: Example design curve for pitch vs wind speed.

The need for gain-scheduling was discussed in Section 2.4.2 and this requirement is clear when looking at this figure, since the slope of the curve is changing continuously. This curve defines a trajectory over the C_Q tables, and the aerodynamic sensitivity function used for gain-scheduling (see Figure 2.13) is calculated by taking finite differences on the C_Q table at points along this curve.

An important implication of this for the learning of a wind turbine's sensitivity function from measured data is that the operating curve in pitch angle and wind speed is also required. It is unlikely that a turbine manufacturer would disclose this information. Furthermore, even if they did, the design wind speed values are effective wind speeds, and hence are not consistent with the measured wind speed 'reference frame' as discussed in the previous section. It is therefore necessary to determine the operating curve from measured data. This will ensure consistency of wind speed values and remove the reliance on obtaining information from manufacturers. Fortunately this procedure can easily fit within the methodology developed so far for aerodynamic identification.

It is proposed that a second SSGP iteration procedure should be run which models the operating curve in pitch angle and wind speed. Since wind speed measurements are where most of the variation lies, wind speed will be modeled as a noisy function of pitch angle. Modelling this curve using an SSGP allows for it to be updated in batches as new data becomes available. Figure 7.50 shows an example fit of an operating curve to measurements of pitch angle and wind speed from the Supergen turbine model.

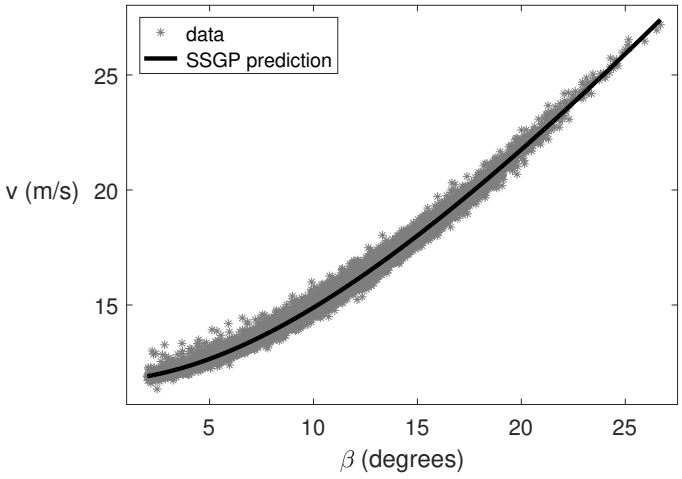

Figure 7.50: Determining the design operating curve in pitch angle and wind speed using an SSGP regression model.

This regression should be performed on wind speed values which result from the SSGP predictions of $v(\beta, v^2 C_Q)$, since these predicted wind speeds have had noise contribu-

tions removed. The apparent 'noise' present on the data in Figure 7.50 is therefore not from measurement noise, but rather is due to the fact that the turbine is deviating from the design operating strategy during operation; this will always happen to some extent, no matter how good the controller, due to the stochastic nature of the wind field.

7.10.4 Learning from Simulated Data

Simulations were performed using the Supergen Exemplar 5MW wind turbine model for wind speeds between 12 and 18 m/s and turbulence intensities of 10, 15 and 20% in order to generate a dataset from the equivalent of roughly 90 hours of above rated operation. As discussed in Section 7.2, this data was sampled at a frequency of 0.05Hz in order to avoid measurement correlations. The resulting dataset contains roughly 16,000 points. To allow for the ability of SSGPs to learn aerodynamic tables to be investigated, without having to account for vertical offsets, drivetrain losses were not included in the generation of this data. Furthermore, since our goal is to learn aerodynamic sensitivities, the offset caused by the losses will not factor when derivatives are calculated.

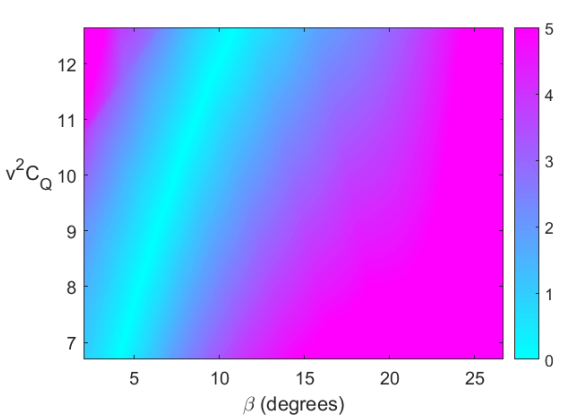

Figure 7.51: Contour plot of absolute percentage errors between the prior model and true values of $v(\beta, v^2 C_Q)$.

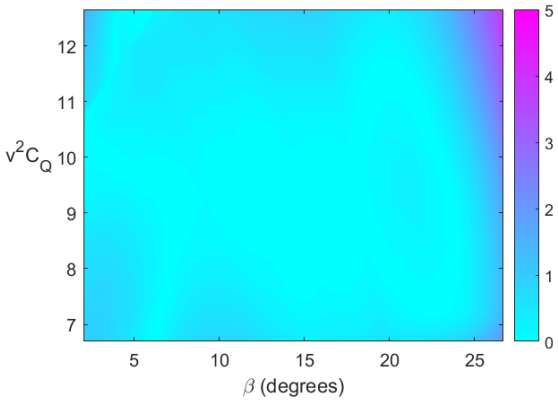

Figure 7.52: Contour plot of absolute percentage errors between the prior model and final model of $v(\beta, v^2 C_Q)$ after regression on 90 hours of operational data.

The SSGP prior and Super Sufficient-Subset developed in Section 7.9 were applied.

Data was processed in batches of 30 points, corresponding to 10 minutes of real time operation[4]. The operating strategy in pitch angle and wind speed was also determined using an SSGP model in order to allow for aerodynamic sensitivities to be calculated. Figure 7.51 shows the absolute percentage errors of the prior model for $v(\beta, v^2 C_Q)$ with respect to the true values. The 'slice' of low errors visible here is due to one surface passing through the other and so although the values become close, the derivatives are very different. Figure 7.52 then shows the same plot having performed SSGP regression on the generated dataset. The difference between effective and measured wind speed was modelled by adding Gaussian noise to the wind speed measurements with a standard deviation of 0.5m/s. As can be seen from these plots, regression on this dataset results in SSGP prediction errors which are almost entirely less than 1%.

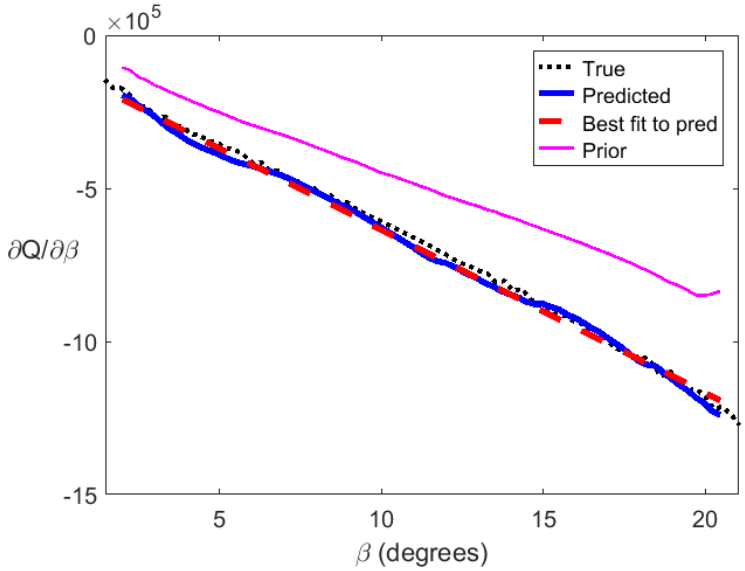

Figure 7.53: Predictions for the aerodynamic sensitivity function, $\frac{\partial Q}{\partial \beta}$, in the SSGP prior and after regression on 90 hours of data. The true values are also shown.

The predicted aerodynamic sensitivity function is considered next. The monotonic nature of the relationship between v and $v^2 C_Q$, for each β, allows for the swapping of input and output. Torque can then be predicted as a function of pitch angle and wind speed since, $Q = \frac{1}{2}\rho A R(v^2 C_Q)$, where now the term $v^2 C_Q$ is a function of β and v.

[4]This being the amount of time the wind field can be reasonably assumed to remain statistically stationary.

The aerodynamic sensitivity function $\frac{\partial Q}{\partial \beta}$ is then calculated by taking finite differences along the operating curve predicted alongside the dynamics. Figure 7.53 shows the aerodynamic sensitivity functions from the prior and final SSGP models, along with the true function. The line of best fit for the prediction is also shown, since it is this which is used as an input to controllers. The mean absolute percentage error between this line of best fit, and that obtained from the true sensitivity function is about 1.5%. This is a very promising result since it is well within our target accuracy of 3%.

Learning Rates

The rate at which a regression model will learn an underlying function is dependent on the noise present on measured data. Therefore, using the simulated data, learning rates were investigated by varying the magnitude of noise on the wind speed measurements and tracking the percentage error between the predicted sensitivity function and the true one. The results of doing this for noise standard deviations of 0.1, 0.5 and 1m/s are shown in Figures 7.54 - 7.56.

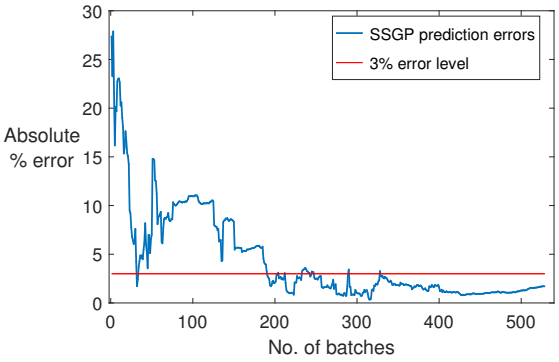

Figure 7.54: Learning rates for aerodynamic sensitivity identification with 0.1m/s wind speed measurement noise standard deviation.

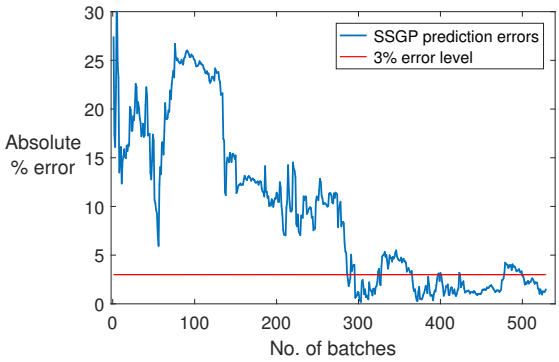

Figure 7.55: Learning rates for aerodynamic sensitivity identification with 0.5m/s wind speed measurement noise standard deviation.

For the smallest noise level, of standard deviation 0.1m/s, the predictions appear to converge after around 350 batches, with errors in the region of 1.5%. As expected, increasing the noise levels slows down the rate of convergence, but for both 0.5 and 1m/s noise standard deviations the errors can still be seen to reach the 3% level towards the

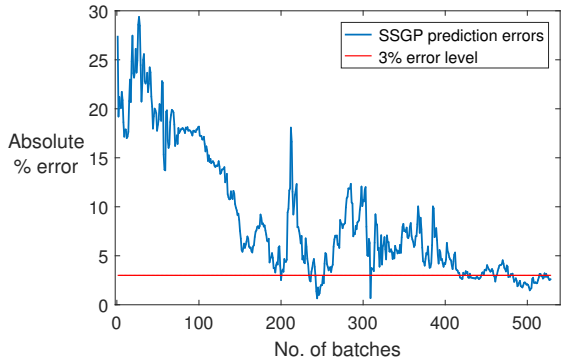

Figure 7.56: Learning rates for aerodynamic sensitivity identification with 1m/s wind speed measurement noise standard deviation.

end of processing the 90 hours of data, although neither has yet converged.

The above results give us our first answer to the key thesis question posed in Section 2.5. The aim of this work was to determine whether aerodynamic information could be learned from measured data to within an accuracy which is useful for control applications. More specifically we set out to learn aerodynamic sensitivities to within an accuracy of 3%. We therefore conclude that, using measured data from the Supergen turbine, aerodynamic sensitivities can be learned to within the required accuracy. The quantities of required data depend on the noise present, however, taking the above results altogether it seems clear that after 90 hours of above rated operation good results are achieved and we would certainly have expected the higher noise cases to have converged after another 90 hours of data, and probably quite a lot less.

7.10.5 Learning from Real Data

The learning of aerodynamic sensitivities was then tested on data from a real wind turbine. The data was processed as outlined in Sections 7.1 and sampled at 0.05HZ. The dataset contains 16,000 points roughly corresponding to 90 hours worth of operational data, the same quantity which was generated using simulations. As for the simulated data, when performing SSGP iteration on real data, batch sizes of 30 datapoints were used. Given the offsets present in both torque and wind speed, between the design and measured dynamics, direct comparisons of the predicted relationships are not mean-

ingful. However, as has been discussed, while offsets exist between the two versions of dynamic information, the derivatives should still agree if the learning procedure has been successful since the predictions and true values should only differ by an additive constant which disappears when derivatives are taken. It is therefore the design and predicted aerodynamic sensitivity functions which will be compared.

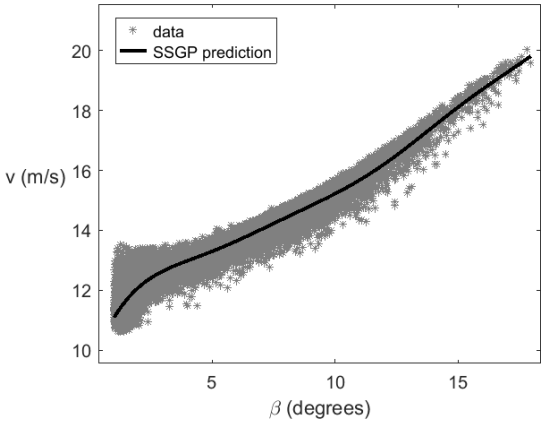

Figure 7.57: Operating curve learned from real wind turbine data. The shape of this curve indicates that a transform has been applied to the wind speed measurements.

Figure 7.57 shows the predicted operating curve in pitch angle and wind speed for the measured turbine data. Comparing this curve with the one in Figure 7.50, it is clear that a transformation has been applied to these wind speed measurements. This is commonly done in order to try and correct differences between the turbine nacelle wind speed and free-stream wind speed measured by the on-sight MET mast [2]. This ensures that the two wind speed measurements are well correlated, a factor which is important for energy yield warranty assessments of wind turbines. The difference between these two wind speed measurements occurs due to the slowing effect the turbine has on the flow field. This is caused by a region of high pressure in front of the turbine rotor which is balanced (energy wise) by a reduction in the flow speed [68]. Figure 7.58 shows the difference between anemometer and free stream wind speed determined in [2] by fitting a 5th order polynomial to measured data. Although the transform present in our wind speed measurements will not be identical to that of figure 7.58, it is clear that this type of transformation could easily lead to the small 'ripple' which is visible in figure 7.57.

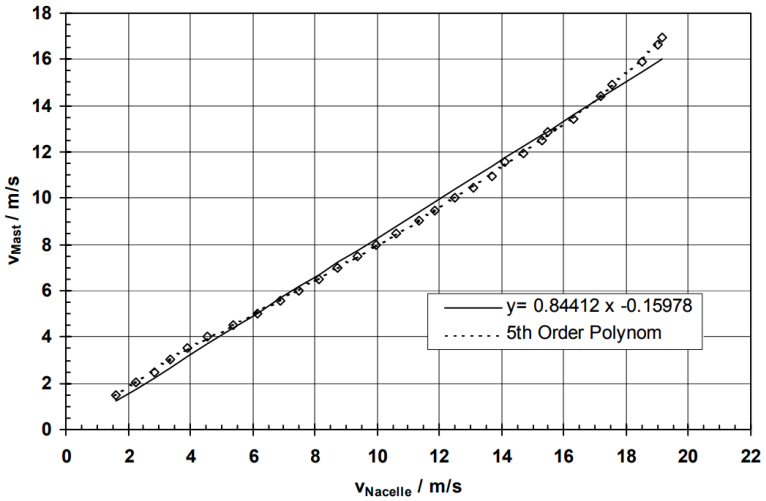

Figure 7.58: Difference in nacelle and MET mast wind speed measurements from measured data [2].

No information is available concerning the transformation applied to our measured data, and hence it cannot be removed from the currently available wind speed measurements. In the present case the regression formulation uses wind speed as the dependent variable and, therefore, the transformation will be introduced into the predicted aerodynamics. This in turn would be expected to impact the predicted derivatives, where small changes to the surface curvature will be magnified.

This is seen to be the case in the predicted sensitivity function, shown in Figure 7.59. In these results a ripple is clearly present in the derivatives which we hypothesise is caused the transformation which has been applied to the wind speed measurements. Note that even with this transformation present, the line of best fit to the predicted sensitivity function has an absolute percentage error of 2.7%, and hence still falls within our required 3% threshold.

The effect of a wind speed transformation on the predicted sensitivity function was tested using the simulated data from the Supergen model. In order to emulate the type of transformation present in Figure 7.57, a multiplicative sinusoidal transformation of the form,

$$v \to (1 + 0.025 \sin(k\beta))v,$$

was applied to the noisy wind speed measurements. The above transform was designed

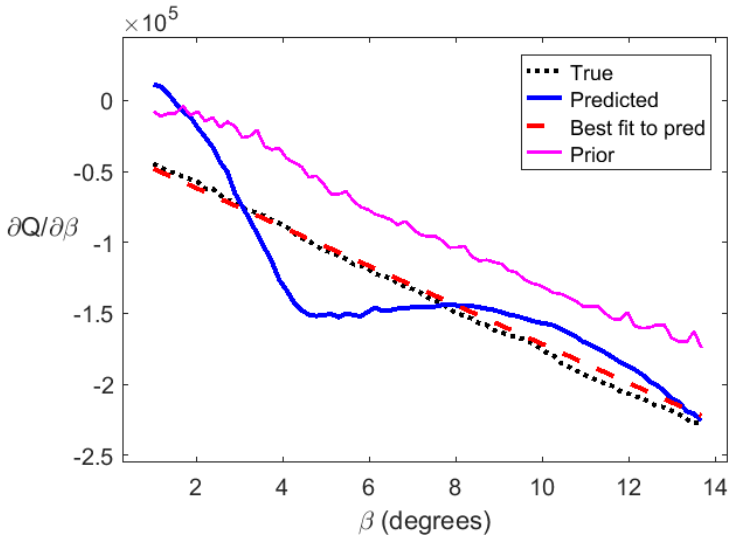

Figure 7.59: Predictions for the aerodynamic sensitivity function, $\frac{\partial Q}{\partial \beta}$, in the SSGP prior and after regression on 90 hours of real wind turbine data. Note that a transformation present on wind speed measurements has caused a ripple in the predicted derivatives.

in order to emulate the shape seen in Figure 7.58, since the coefficients of the fitted polynomial are not given in [2]. This results in the altered operating strategy are shown in Figure 7.60. While changes to the operating strategy resulting from the

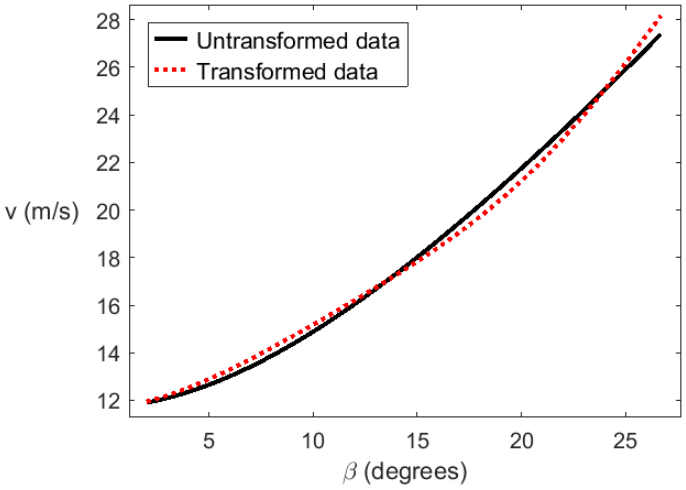

Figure 7.60: Learned operating strategy before and after the addition of a wind speed transformation.

data transformation appear to be only slight, the effect of these changes in wind speed

measurements on derivatives turns out to be significant, as shown in Figure 7.61. A distinct ripple is now present in the derivative predictions which, as can be seen in Figure 7.53, was not there before applying the transformation on wind speed. As in the real data case, although the derivative prediction has been severely distorted, the straight line best fit is still reasonable. However, one would not necessarily expect this to be the case in general.

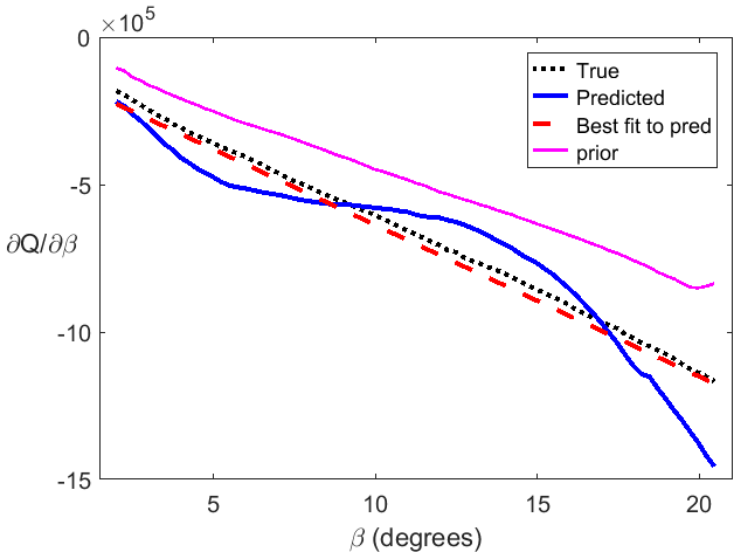

Figure 7.61: Predictions for the aerodynamic sensitivity function, $\frac{\partial Q}{\partial \beta}$, on simulated data with a transformation applied to wind speed measurements. The resultant ripple in the derivative predictions is similar to that seen for the real turbine data in Figure 7.59.

These results imply that the future development and application of this approach to dynamics identification will require a careful consideration of wind speed measurements and the processing of this data. Definitive results which fully validate these methods for real data will also require access to wind speed measurements which have not been transformed. In practise, untransformed wind speed measurements would be available to the SSGP regression procedure, this is because the wind speed transforms are applied as part of the processing procedures in the turbine nacelle. It would therefore be feasible to sequence the data processing so that a direct feed from the anemometer goes to the SSGP iteration procedure before any transform is applied.

Discussion

The results presented here demonstrate a strong similarity in the behaviour of the SSGP method when applied to simulated and real data in the presence of a wind speed transformation. There is therefore no reason, from the current results, to suspect that they should deviate if applied to untransformed data. This, combined with the exemplary results obtained when using simulated data lead us to conclude that the SSGP regression approach developed here can identify wind turbine dynamics to within the required accuracy of 3%, as long as untransformed wind speed measurements are available. The results of the present section also indicate that the quantity of data required to reach the specified error level is around 90 hours for above rated operation, a fairly modest amount which should be obtained within a month of operation or less, although for obvious reasons this will be weather dependent.

The most important implication of these results for above rated dynamics identification is that proper consideration will have to be given to the means by which wind speed is measured, and the processing of this data. As was already discussed, it should be possible to obtain untransformed wind speed measurements by tapping into the raw anemometer feed on the turbine nacelle. In the first instance of further developing this technique such raw wind speed data, along with the other required measurements, should be obtained in order to fully validate the results seen here for simulated data. However, given the high level of sensitivity to wind speed values, demonstrated by the results seen in Figures 7.59 and 7.61, the full range of available wind speed measurement techniques should be taken into consideration. While we would ideally like to make use of the nacelle cup-anemometer already present, a number of alternative devices exist including: sonic anemometry mounted on the turbine nose cone [69] and nacelle mounted LIDAR [70]. While the addition of such devices would have cost implications, they can also generate more sophisticated wind field data which can be employed in additional control tasks such as detecting turbine yaw misalignment [69, 70] and warning of large gusts [71]. Therefore, the overall cost might be justified if a given technique contributed to several processes simultaneously.

In Section 2.6.3 it was stated that wind speed noise would be assumed to be sta-

tionary and Gaussian in the current work to allow for the development and testing of the learning procedures. These assumptions were shown to be reasonable, since good regression results are obtained, however, in future work the structure of the noise on wind speed measurements should be further studied in order to determine whether a better noise model can be found which might improve on the results seen here. For example, there exist various so called 'colored' noise models which allow for short time correlations in noise values to be captured [72]. The sampling interval of 20 seconds was chosen to try and remove correlations from measured data, but, if some correlation is still present then it should be accounted for in the models to avoid degrading the performance of our learning procedure.

Chapter 8

Conclusions and Future Work

This thesis has developed techniques for the learning of wind turbine dynamic information from data available to a wind turbine controller. In the process of developing such techniques, fundamental contributions have been made in the area of sparse GP approximation theory and polynomial GPs.

This chapter summarises the major contributions of this thesis, and the future work arising in each case.

Polynomial Gaussian Processes Regression

Chapter 5 extensively studied the case of GP regression for polynomials. A polynomial GP implementation was presented which is very efficient and scales linearly in the number of measured datapoints. This included hyperparameter optimisation using derivative and Hessian information. The technique was then compared with Least-Squares regression both theoretically and experimentally and shown to be superior in all considered cases. Polynomial regression was then used to investigate the suitability of *posterior-to-prior* and *combined posteriors* approaches to GP iteration. The results of this work indicated that *combined posteriors* was unsuitable for pursuing further. However, for the polynomial case *posterior-to-prior* iteration was shown to be equivalent to standard GP regression on all of the data at once, with linear scaling and fixed memory requirements.

Sufficient-Subset Gaussian Process Iteration

An extension to the general case of the lessons learned from the iteration of polynomial

GPs was shown in Chapter 6 to lead to a new Sparse GP approximation method, SSGPs, which can handle batched data and changing noise levels. SSGP iteration was derived directly from GP regression theory, rather than from more general probability theories as is the case for the other batched GP approaches discussed in Chapter 4. The relationship between SSGPs and standard GP regression was explicitly shown. The concept of a Sufficient-Subset was developed which describes the minimal set of inducing points which recreate an underlying GP model up to a given limit of matrix condition number. This was extended to Super Sufficient-Subset theory, allowing for the minimal set to be populated beyond the point where the matrix becomes singular. It was shown that a Super Sufficient-Subset can reduce prediction errors when the underlying function is not strictly drawn from a GP prior. Super Sufficient-Subset theory turned out to be crucial to the turbine dynamics identification problem, allowing for a general GP prior to be defined for all turbines, as shown in Section 7.9. The development of SSGP theory was also shown to answer fundamental questions concerning sparse GP models not addressed in existing methods.

With respect to future work: it was shown in Chapter 6 that there is much scope for further development of SSGP iteration. For example, the relevant expressions were derived to allow for online hyperparameter learning and updating within the SSGP iterative scheme. Once this is possible it is then necessary to consider questions regarding the location and number of inducing points. This will require either the updating of the Sufficient-Subset, or a deliberate overpopulation to generate a Super Sufficient-Subset which can then remain fixed, even as the covariance structure changes. This extended SSGP should then be tested against existing methods which update hyperparemeter values and inducing point locations via optimisation of probabilistic 'distance' functions. The selected input locations from SSGP theory and these other optimisation procedures should be compared and the implications of SSGP theory for existing methods considered.

Wind Turbine Dynamics Identification

In chapter 7 the polynomial and SSGP regression approaches developed in this thesis were applied to simulated and real data. Polynomial GPs were applied to the 'maximum

efficiency' region dynamics identification problem where the sought quantities are the C_{Pmax} value and drivetrain losses. Results from simulated data showed that the GP approach outperforms Least Squares for this task, with Least Squares prediction being so scattered as to be effectively useless. Both approaches suffered from bias on the predicted values. GP results for C_{Pmax} identification from real data were seen to mirror those obtained for simulated data. However, the GP approach could not detect any changes in losses in the real data and so set the loss function coefficients to zero. As discussed, this could well be due to the restricted range of rotational speeds for which the turbine which generated the available data tracks maximum efficiency. Finally, while the presence of bias renders the approach unsuitable for learning the exact values of the sought dynamics, it was outlined how the same techniques could be used in order to monitor a wind turbine and warn of changes in behaviour which could indicate faults or damage to the turbine. Since such methods are only required to detect changes in the dynamics, the presence of bias on predictions would not necessarily be an issue in this setting.

In terms of future work: data should be sought from a turbine which tracks C_{Pmax} across a wider range of rotational speeds. This will allow for further testing to determine whether this regression approach can detect drivetrain losses given enough variation in rotational speed. There is also much potential work involved in developing these regression predictions into a fault detection technique. For example, the development of a process by which to track changes in predicted values and determine when the predictions have changed enough to warrant triggering an alarm to warn the turbine operator.

Next was considered the 'above rated' regression problem where the aerodynamic sensitivity function is sought. In order to generate an SSGP prior for this regression problem, as derived in Chapter 2, a theory of *local covariance* and a computationally efficient version of existing input transformation formulations were shown to allow for accurate GP regression on functions with a lengthscale which varies across the domain. This combined with Super Sufficient-Subset theory was shown to allow for a GP prior to be defined which is suitable for general turbine dynamics tables. It was then shown that

although an appropriate prior can be constructed, the turbine's trajectory through it's operating envelope renders this approach unfeasible for dynamics identification. The regression problem was therefore reformulated into the so called *wind speed formulation* and a new SSGP prior constructed. Tests were then performed using simulated and real data. On simulated data the results were exemplary, showing that the predicted sensitivity function came within the specified error threshold of 3% after roughly 90 hours of data. For real data it was apparent, due to an uncharacteristic 'ripple' being present in the pitch versus wind speed operating curve, that a transform had been applied to the wind speed measurements. The effect of this transform on predictions was shown to be the addition of a similar ripple in the sensitivity function predictions. In order to test the hypothesis that this had indeed been caused by a wind speed transformation, a transform of the same suspected type was applied in the simulated data case and regression repeated. Using the transformed data was shown to result in the same type of distortion in sensitivity function predictions. Taken together these results indicate that this regression approach is able to identify wind turbine sensitivity functions from measured data up to the required accuracy, however, in practice will require access to wind speed measurements before any transformation is applied.

In terms of future work: turbine data should be acquired for which the wind speed measurements have not been transformed, this will allow for a complete validation of the results presented here. Additionally, given the sensitivity of results to measurements of wind speed, the full range of available wind measurement devices should be investigated in terms of suitability for providing data to this regression problem. Finally, now that appropriate methods have been developed and tested, the implementation of these techniques in wind turbine controllers should be considered. This will involve a considerable amount of software engineering work since it requires for the entire regression formulation to be efficiently coded in C.

Appendix A

The Multivariate Gaussian Conditional Distribution

If $\mathbf{x}$ and $\mathbf{y}$ are jointly Gaussian random vectors,

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \sim N\left(\begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix}, \begin{bmatrix} A & C \\ C^T & B \end{bmatrix} \right)$$

then $\mathbf{x} \sim N(\boldsymbol{\mu}_x, A)$ and, crucially for regression, we can write down the conditional distribution of $\mathbf{x}$ given $\mathbf{y}$ [28];

$$\mathbf{x}|\mathbf{y} \sim N(\boldsymbol{\mu}_x + CB^{-1}(\mathbf{y} - \boldsymbol{\mu}_y), A - CB^{-1}C^T) \tag{A.1}$$

Note that this conditional distribution is again Gaussian.

Appendix B

Polynomial GPs - Efficient Implementation Formulae

In this appendix we present the equations for the implementation of Gaussian process polynomial regression which were not included in Section 5.1.2.

Derivative term for noise log-hyperparameter

$$\frac{\partial L^*}{\partial \phi_\xi} = \quad n - trace\left[(D^{-1}\Sigma)(V_{\mathbf{z}}^T V_{\mathbf{z}})\right] - \frac{1}{\xi}[\widehat{\mathbf{F}}^T\widehat{\mathbf{F}} - 2(\widehat{\mathbf{F}}^T V_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^T\widehat{\mathbf{F}}) + (\widehat{\mathbf{F}}^T V_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^T V_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^T\widehat{\mathbf{F}})]$$

Hessian terms

Case 1 (i,$j \neq \xi$).

$$\begin{aligned}\frac{\partial^2 L^*}{\partial\phi_i\partial\phi_j} = & \frac{\delta_{ij}\gamma_i}{\xi}\{V_i^TV_i - trace[(D^{-1}\Sigma)(V_{\mathbf{z}}^TV_i)(V_i^TV_{\mathbf{z}})]\}\\ & -\frac{\gamma_i\gamma_j}{\xi^2}\{(V_j^TV_i)^2 - trace[(D^{-1}\Sigma)(V_{\mathbf{z}}^TV_i)(V_i^TV_j)(V_j^TV_{\mathbf{z}})] - trace[(V_i^TV_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^TV_j)(V_j^TV_i)]\\ & +trace[(D^{-1}\Sigma)(V_{\mathbf{z}}^TV_i)(V_i^TV_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^TV_j)(V_j^TV_{\mathbf{z}})]\}\\ & -\frac{\delta_{ij}\gamma_i}{\xi^2}\{(\widehat{\mathbf{F}}^TV_i)^2 - (\widehat{\mathbf{F}}^TV_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^TV_i)(V_i^T\widehat{\mathbf{F}}) - (\widehat{\mathbf{F}}^TV_i)(V_i^TV_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^T\widehat{\mathbf{F}})\\ & +(\widehat{\mathbf{F}}^TV_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^TV_i)(V_i^TV_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^T\widehat{\mathbf{F}})\}\\ & +\frac{2\gamma_i\gamma_j}{\xi^3}\{(\widehat{\mathbf{F}}^TV_i)(V_i^TV_j)(V_j^T\widehat{\mathbf{F}}) - (\widehat{\mathbf{F}}^TV_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^TV_i)(V_i^TV_j)(V_j^T\widehat{\mathbf{F}})\\ & -(\widehat{\mathbf{F}}^TV_i)(V_i^TV_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^TV_j)(V_j^T\widehat{\mathbf{F}})\\ & +(\widehat{\mathbf{F}}^TV_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^TV_i)(V_i^TV_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^TV_j)(V_j^T\widehat{\mathbf{F}})\\ & -(\widehat{\mathbf{F}}^TV_i)(V_i^TV_j)(V_j^TV_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^T\widehat{\mathbf{F}})\\ & +(\widehat{\mathbf{F}}^TV_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^TV_i)(V_i^TV_j)(V_j^TV_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^T\widehat{\mathbf{F}})\\ & +(\widehat{\mathbf{F}}^TV_i)(V_i^TV_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^TV_j)(V_j^TV_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^T\widehat{\mathbf{F}})\\ & -(\widehat{\mathbf{F}}^TV_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^TV_i)(V_i^TV_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^TV_j)(V_j^TV_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^T\widehat{\mathbf{F}})\}\end{aligned}$$

Case 2 ($i \neq \xi$, $j = \xi$).

$$\begin{aligned}\frac{\partial^2 L^*}{\partial\phi_i\partial\phi_\xi} = & -\frac{\gamma_i}{\xi}\{(V_i^TV_i) - 2trace[(D^{-1}\Sigma)(V_{\mathbf{z}}^TV_i)(V_i^TV_{\mathbf{z}})] + trace[(D^{-1}\Sigma)(V_{\mathbf{z}}^TV_i)(V_i^TV_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^TV_{\mathbf{z}})]\}\\ & +\frac{2\gamma_i}{\xi^2}\{(\widehat{\mathbf{F}}^TV_i)^2 - (\widehat{\mathbf{F}}^TV_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^TV_i)(V_i^T\widehat{\mathbf{F}}) - 2(\widehat{\mathbf{F}}^TV_i)(V_i^TV_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^T\widehat{\mathbf{F}})\\ & +2(\widehat{\mathbf{F}}^TV_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^TV_i)(V_i^TV_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^T\widehat{\mathbf{F}})\\ & +(\widehat{\mathbf{F}}^TV_i)(V_i^TV_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^TV_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^T\widehat{\mathbf{F}})\\ & -(\widehat{\mathbf{F}}^TV_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^TV_i)(V_i^TV_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^TV_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^T\widehat{\mathbf{F}})\}\end{aligned}$$

Case 3 ($i = j = \xi$).

$$\frac{\partial^2 L^*}{\partial \phi_\xi^2} = trace[(D^{-1}\Sigma)(V_{\mathbf{z}}^T V_{\mathbf{z}})] - trace[(D^{-1}\Sigma)(V_{\mathbf{z}}^T V_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^T V_{\mathbf{z}})]$$
$$+\frac{1}{\xi}(\widehat{\mathbf{F}}^T\widehat{\mathbf{F}}) - \frac{4}{\xi}(\widehat{\mathbf{F}}^T V_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^T\widehat{\mathbf{F}}) + \frac{5}{\xi}(\widehat{\mathbf{F}}^T V_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^T V_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^T\widehat{\mathbf{F}})$$
$$-\frac{2}{\xi}(\widehat{\mathbf{F}}^T V_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^T V_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^T V_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^T\widehat{\mathbf{F}})$$

Prediction equations

For $\mathbf{z}$ a vector of measurement points and $\mathbf{z}^*$ a chosen vector of prediction points we can re-express the GP prediction formulas as follows (for the case of polynomial regression). These reformulations are required when using very large quantities of input data.

$$mean = V_{\mathbf{z}^*}\widetilde{\mathbf{q}}$$

where,

$$\widetilde{\mathbf{q}} = (\frac{1}{\xi}\Sigma)[V_{\mathbf{z}}^T\widehat{\mathbf{F}} - (V_{\mathbf{z}}^T V_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^T\widehat{\mathbf{F}})],$$

and contains the predicted coefficients.

$$covariance\ matrix = V_{\mathbf{z}^*}\widetilde{\Sigma}V_{\mathbf{z}^*}^T$$

where,

$$\widetilde{\Sigma} = \Sigma - \frac{1}{\xi}\Sigma[V_{\mathbf{z}}^T V_{\mathbf{z}} - (V_{\mathbf{z}}^T V_{\mathbf{z}})(D^{-1}\Sigma)(V_{\mathbf{z}}^T V_{\mathbf{z}})]\Sigma,$$

is non-diagonal as discussed in Remark 1.

Appendix C

Combined Posterior Experiments in the General Case

In this section, Combined-Posteriors iteration for GP models is explored in the general case. Note that the formulas for combined inverses (see below) necessarily requires inversion of posterior GP matrix terms. As was discussed in Section 6.2, GP posterior covariance matrices tend to be ill-conditioned. In the case of polynomial GPs an algebraic manipulation was found which allowed for this issue to be avoided. In the general case no similar work around exists and hence we will use pseudo-inversion in order to try and combine GP statistics.

Starting with two posterior GP multivariate distributions for our vector of function points $\mathbf{F}_{\mathbf{Z}^*}$ for a given function f,

$$\begin{aligned} P_1(\mathbf{F}_{\mathbf{Z}^*}) &= N(\boldsymbol{\mu}_1, \Lambda_1) \\ P_2(\mathbf{F}_{\mathbf{Z}^*}) &= N(\boldsymbol{\mu}_2, \Lambda_2). \end{aligned}$$

Appendix C. Combined Posterior Experiments in the General Case

Recall we are looking to form the combined distribution $P_+(\mathbf{F}_{\mathbf{Z}^*}) = N(\boldsymbol{\mu}_+, \Lambda_+)$ where,

$$\begin{aligned} \boldsymbol{\mu}_+ &= (\Lambda_1^{-1} + \Lambda_2^{-1})^{-1}(\Lambda_1^{-1}\boldsymbol{\mu}_1 + \Lambda_2^{-1}\boldsymbol{\mu}_2) \\ &= \Lambda_2(\Lambda_1 + \Lambda_2)^{-1}\boldsymbol{\mu}_1 + \Lambda_1(\Lambda_1 + \Lambda_2)^{-1}\boldsymbol{\mu}_2, \end{aligned} \tag{C.1}$$

$$\begin{aligned} \Lambda_+ &= (\Lambda_1^{-1} + \Lambda_2^{-1})^{-1} \\ &= \Lambda_1(\Lambda_1 + \Lambda_2)^{-1}\Lambda_2. \end{aligned} \tag{C.2}$$

The equations reformulated as above now only require a single inversion operation to be performed, that of $(\Lambda_1 + \Lambda_2)^{-1}$. This term is still singular, or close enough to being singular to render normal inversion useless, we instead form the Moore-Penrose pseudo-inverse as follows: first a singular value decomposition (SVD) is performed,

$$(\Lambda_1 + \Lambda_2) = UEU^T,$$

with E diagonal and containing the singular values of $(\Lambda_1 + \Lambda_2)$ in descending order,

$$E = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{bmatrix},$$

with $\lambda_1 > \lambda_2 > \ldots > \lambda_n$. A threshold value, τ, is then chosen below which to consider singular values to be effectively 0, 1×10^{-6} was used initially but a range of values have been tested. Having chosen this threshold value we define E^+, a diagonal matrix with entries:

$$[E^+]_{ii} := \begin{cases} \lambda_i^{-1} & \text{if } \lambda_i > \tau \\ 0 & \text{otherwise.} \end{cases}$$

The pseudo-inverse of $(\Lambda_1 + \Lambda_2)$, denoted $(\Lambda_1 + \Lambda_2)^+$, is then defined as;

$$(\Lambda_1 + \Lambda_2)^+ := UE^+U^T.$$

Note that for a non-singular matrix A and with $\tau = 0$, $A^{+} = A^{-1}$. The pseudo-inverse $(\Lambda_1 + \Lambda_2)^{+}$ is then used in place of $(\Lambda_1 + \Lambda_2)^{-1}$ for determining μ_{+} and Λ_{+}.

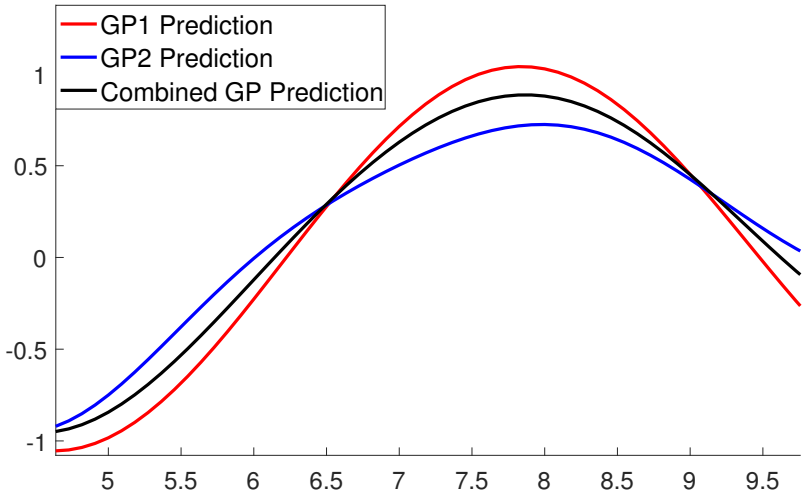

Figure C.1: An example of the resulting predictions from combined-statistics iteration using generalised inverses.

Figure C.1 shows an example of this method of combining the predictive statistics being applied to two posterior GPs, a squared exponential covariance function was used in both cases. The resultant combined prediction, obtained using generalised inverses with $\tau = 10^{-6}$, appears sensible; as do the confidence intervals of the combined prediction, shown in Figure C.2. However, in other cases anomalies have been found such as that seen in Figure C.3 where the combined predicted mean lies outside of both the GP means which are being combined. This type of anomaly has been found to occur across different values of τ and different numbers of points being combined. While the size of these anomalies is generally fairly small, with repeated iterations they could grow and increase the predictive errors, severely undermining both the combined predictions and the ability fo the confidence intervals to accurately reflect the true size of these errors. It therefore seems that the issues presented by ill-conditioned and singular covariance matrices cannot be avoided by the use of pseudo-inverses and the pseudo-inverse approach turns out to be unsuitable for forming refined regression estimates.

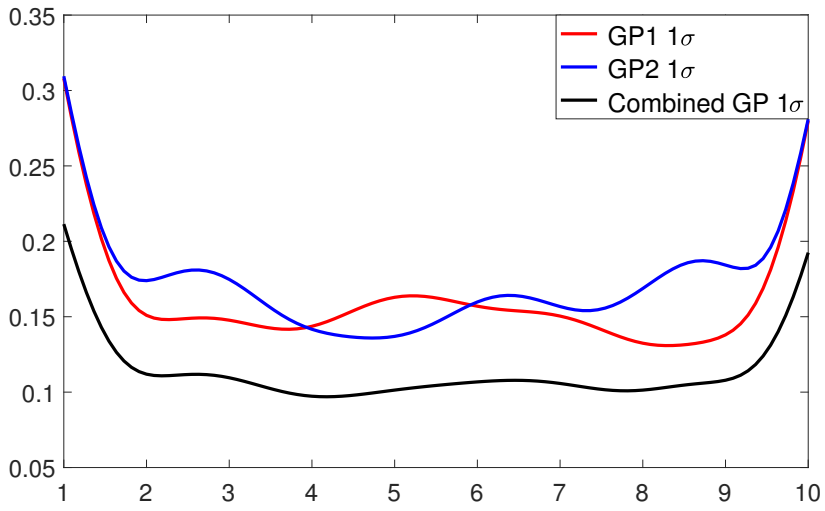

Figure C.2: An example of the resulting confidence intervals at one standard deviation from combined-statistics iteration using generalised inverses.

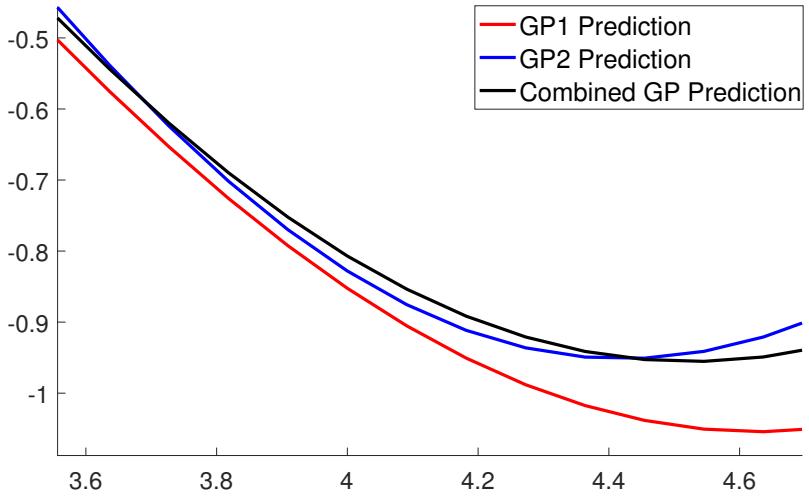

Figure C.3: An example of the anomalies which can occur when performing combined-statistics iteration using generalised inverses.

Bibliography

[1] W. E. Leithead and D. J. Leith, "On the Separability of Wind Turbine Rotor Aerodynamics," *Technical Report, University of Strathclyde, Glasgow*, 2001.

[2] R. Hunter, T. F. Pedersen, P. Dunbabin, I. Antoniou, S. Frandsen, H. Klug, A. Albers, and W. K. Lee, "European wind turbine testing procedure developments task 1: Measurement method to verify wind turbine performance characteristics," *Risø National Laboratory, Roskilde, Ris-R-1209(EN)*, 2001.

[3] "Wind Europe Statistics," https://windeurope.org/about-wind/statistics/, accessed: 2018-06-12.

[4] S. Pfaffel, S. Faulstich, and K. Rohrig, "Performance and Reliability of Wind Turbines: A Review," *Energies, 10, 1904*, 2017.

[5] P. D. Fleming and S. D. Probert, "The Evolution of Wind-Turbines: An Historical Review," *Applied Energy 18, 163–177*, 1984.

[6] T. Burton, D. Sharpe, N. Jenkins, and E. Bossanyi, *Wind Energy Handbook*. John Wiley & Sons, 2001.

[7] W. Leithead, D. Leith, F. Hardan, and H. Markou, "Global Gain-scheduling Control for Variable Speed Wind Turbines," *Proceedings of the European Wind Energy Conference*, 1999.

[8] M. D. Hansen, A. D. Hansen, T. J. Larsen, S. Oye, P. Sorensen, and P. Fuglsang, "Control Design for a Pitch-regulated, Variable Speed Wind Turbine," Risoe, Denmark, Tech. Rep. 1500, 2005.

[9] D. J. Leith and W. E. Leithead, "Implementation of Wind Turbine Controllers," *Technical Report, EEE, University of Strathclyde, Glasgow*, 2002.

[10] E. A. Bossanyi, "Wind Turbine Control for Load Reduction," *Wind Energ. 6:229–244*, 2003.

[11] W. E. Leithead and S. Dominguez, "Controller Design for the Cancellation of the Tower Fore-aft Mode in a Wind Turbine," *Proc. 44th IEEE Conf. Decis. Control*, pp. 1276 – 1281, 2005.

[12] National Grid Electricity Transmission plc, "The Grid Code," 2013.

[13] E. Muljadi, V. Gevorgian, M. Singh, and S. Santoso, "Understanding Inertial and Frequency Response of Wind Power Plants," *IEEE Power Electronics and Machines in Wind Applications*, pp. 1–8, 2012.

[14] F. D. Kanellos and N. D. Hatziargyriou, "Optimal Control of Variable Speed Wind Turbines in Islanded Mode of Operation," *IEEE Transactions on Energy Conversion*, vol. 25, no. 4, 2010.

[15] A. Stock, "Augmented Control for Flexible Operation of Wind Turbines," Ph.D. dissertation, University of Strathclyde, Dept. of Electronic and Electrical Engineering, 2015.

[16] W. E. Leithead and S. Dominguez, "Coordinated Control Design for Wind Turbine Control Systems," *Proc. European Wind Energy Conference*, 2006.

[17] E. A. Bossanyi, "Individual Blade Pitch Control for Load Reduction," *Wind Energy, p. 119-128*, vol. 6, no. 2, 2003.

[18] H. Yi and W. E. Leithead, "Alleviation of Extreme Blade Loads by Individual Blade Control During Normal Wind Turbine Operation," *Proc. European Wind Energy Association Conference*, pp. 90 – 94, 2012.

[19] D. Jena and S. Rajendran, "A Review of Estimation of Effective Wind Speed Based Control of Wind Turbines," *Renewable and Sustainable Energy Reviews*, vol. 43, pp. 1046–1062, 2015.

[20] M. H. Keegan, "Wind Turbine Blade Leading Edge Erosion: An investigation of rain droplet and hailstone impact induced damage mechanisms," Wind Energy Systems DTC, University of Strathclyde, Glasgow, UK, 2014.

[21] N. Gaudern, "A Practical Study of the Aerodynamic Impact of Wind Turbine Blade Leading Edge Erosion," *Journal of Physics: Conference Series 524 012–031*, 2014.

[22] B. R. Hohn, K. Michaelis, and M. Hinterstoiser, "Optimization of Gearbox Efficiency," *Goriva i Maziva (Fuels and Lubricants), 48, 4 : 441–480*, 2009.

[23] R. Martins, C. Fernandes, and J. Seabra, "Evaluation of Bearing, Gears and Gearboxes Performance with Different Wind Turbine Gear Oils," *Friction 3(4): 275–286*, 2015.

[24] F. D. Bianchi, R. J. Mantz, and H. D. Battista, *Wind Turbine Control Systems: Principles, Modelling and Gain Scheduling Design.* London: Springer London, 2007, pp. 7–28.

[25] I. Van der Hoven, "Power Spectrum of Horizontal Wind Speed in the Frequency Range from 0.0007 to 900 Cycles Per Hour," *Journal of Meteorology 14: 160-164*, 1957.

[26] "Supergen wind hub," https://www.supergen-wind.org.uk/.

[27] R. M. Neal, "Priors for Infinite Networks," *Technical Report CRG-TR-94-1, University of Toronto*, 1994.

[28] C. E. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning.* MIT Press, 2006.

[29] K. N. K. Seng, "Non-linear Dynamics Identification Using Gaussian Process Prior Models Within a Bayesian Context," Ph.D. dissertation, National University of Ireland, Hamilton Institute, 2008.

[30] D. Mackay, "Introduction to Gaussian Processes," *NATO ASI Series F Computer and Systems Sciences*, no. 168, pp. 133–166, 1998.

[31] A. M. Schmidt and A. O'Hagan, "Bayesian Inference for Non–Stationary Spatial Covariance Structure via Spatial Deformations," *JRSS, Series B (Statistical Methodology), Vol 65*, no. 3, pp. 743–758, 2003.

[32] M. N. Gibbs, "Bayesian Gaussian Processes for Regression and Classification," *PhD Thesis, Cambridge University*, 1997.

[33] E. Snelson, C. E. Rasmussen, and Z. Ghahramani, "Warped Gaussian Processes," *Advances in Neural Information Processing Systems (NIPS)*, 2003.

[34] S. Haykin, *Neural Networks - A Comprehensive Foundation.* Prentice Hall, 2005.

[35] Z. Jones and F. Linder, "Exploratory Data Analysis using Random Forests," *73rd annual MPSA conference*, 2015.

[36] N. Chen, Z. Qian, I. Nabney, and X. Meng, "Wind Power Forecasts Using Gaussian Processes and Numerical Weather Prediction," *IEEE Trans. Pow. Sys. 29–2*, 2014.

[37] E. Papatheou, N. Dervilis, E. Maguire, and K. Worden, "Wind Turbine Structural Health Monitoring: A Short Investigation Based on scada Data," *7th European Workshop on Structural Health Monitoring*, 2014.

[38] X. G. Larsen and A. Kruger, "Extreme Gust Wind Estimation Using Mesoscale Modeling," *Proceedings of EWEA*, 2014.

[39] J. Quiñonero-Candela and C. E. Rasmussen, "A Unifying View of Sparse Approximate Gaussian Process Regression," *Journal of Machine Learning Research*, vol. 6, pp. 1939–1959, 2005.

[40] M. K. Titsias, "Variational Learning of Inducing Variables in Sparse Gaussian Processes," *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics, JMLR Volume 5*, 2009.

[41] E. Snelson and Z. Ghahramani, "Sparse Gaussian Processes Using Pseudo-inputs," *Advances in Neural Information Processing Systems (NIPS)*, 2006.

[42] L. Csató, "Gaussian Processes - Iterative Sparse Approximations," Ph.D. dissertation, Aston University, Aston, 2002.

[43] T. D. Bui, C. V. Nguyen, and R. E. Turner, "Streaming Sparse Gaussian Process Approximations," *31st Conference on Neural Information Processing Systems, Long Beach, CA, USA*, 2017.

[44] T. Minka, "Power EP," *Microsoft Research Technical Report, MSR-TR-2004-149*, 2004.

[45] L. Csato and M. Opper, "Sparse Online Gaussian Processes," *Technical Report NCRG/2001/014, Aston University, Birmingham*, 2002.

[46] T. D. Buie, J. Yan, and R. E. Turner, "A Unifying Framework for Gaussian Process Pseudo-Point Approximations using Power Expectation Propagation," *Journal of Machine Learning Research*, 2017.

[47] M. F. Huber, "Recursive Gaussian Processes: online regression and learning," *Elsevier Pattern Recognition Letters*, vol. 45, pp. 85–91, 2014.

[48] M. Bauer, M. v. d. Wilk, and C. E. Rasmussen, "Understanding Probabilistic Sparse Gaussian Process Approximations," *30th Conference on Neural Information Processing Systems (NIPS)*, 2016.

[49] V. Tresp, "A Bayesian Committee Machine," *Neural Computation*, vol. 12, pp. 2719–2741, 2000.

[50] J. Wolberg, *Data Analysis Using the Method of Least Squares.* Springer, 2006.

[51] N. J. Higham, *Accuracy and Stability of Numerical Algorithms.* SIAM, 2002.

[52] E. L. Snelson, "Flexible and Efficient Gaussian Process Models for Machine Learning," Ph.D. dissertation, Gatsby Computational Neuroscience Unit, University College London, 2007.

[53] R. A. Horn and C. R. Johnson, *Matrix Analysis.* Cambridge University Press, 2012.

[54] T. Hastie, R. Tibshiran, and J. Friedman, *The Elements of Statistical Learning (2nd)*. SIAM, 2009.

[55] P. V. Mikheev, "Multidimensional Gaussian Probability Density and its Applications in the Degenerate Case," *Radiophysics and Quantum Electronics*, vol. 49, no. 7, 2006.

[56] R. M. Neal, "Monte Carlo Implementation of Gaussian Process Models for Bayesian Regression and Classification," University of Toronto, Tech. Rep. 9702, 1997.

[57] G. H. Golub and C. F. Van Loan, *Matrix Computations (3rd ed.)*. Johns Hopkins University Press, 1996.

[58] G. Bianchi and R. Sorrentino, *Electronic Filter Simulation and Design*. McGraw-Hill Professional, 2007.

[59] S. Hur and W. E. Leithead, "Model Predictive Control of a Variable-Speed Pitch-Regulated Wind Turbine," *UKACC 11th International Conference on Control*, 2016.

[60] V. Neilson, "Individual Blade Control for Fatigue Load Reduction of Large-scaled Wind Turbines: Theory and Modelling," PhD Thesis, Industrial Control Centre, Department of Electronic and Electrical Engineering, University of Strathclyde, 2010.

[61] A. Karimi and D. L. Landau, "Comparison of the Closed–Loop Identification Methods in Terms of the Bias Distribution," *Systems and Control Letters, 34 159–167*, 1998.

[62] U. Forssell and L. Ljung, "Issues in Closed–Loop Identification," Linkoping University, LiTH–ISY–R–1940, Tech. Rep., 1997.

[63] X. Wang, E. Bibeau, and N. G, "Experimental Investigation of Energy Losses Due to Icing of a Wind Turbine," *International Conference on Power Engineering, Hangzhou, China*, 2007.

[64] A. Sareen, S. C., and S. M., "Effects of Leading Edge Erosion on Wind Turbine Blade Performance," *Wind Energ. 17:1531-1542*, 2014.

[65] A. Zaher, S. McArthur, D. Infield, and Y. Patel, "Online Wind Turbine Fault Detection through Automated SCADA Data Analysis," *Wind Energy: An International Journal for Progress and Applications in Wind Power Conversion Technology*, vol. 12, no. 6, pp. 574–593, 2009.

[66] G. Sieros, P. Chaviaropoulos, J. D. Sorensen, B. H. Bulder, and P. Jamieson, "Upscaling Wind Turbines: theoretical and practical aspects and their impact on the cost of energy," *Wind Energ.; 15:3-17*, 2012.

[67] F. D. Adegas, R. Wisniewski, and L. F. S. Larsen, "Gain-Scheduled Model Predictive Control of Wind Turbines Using Laguerre Functions," *American Control Conference (ACC), Washington, USA*, 2013.

[68] J. N. Sorensen, "General Momentum Theory for Horizontal Axis Wind Turbines," *Springer International Publishing, Vol 4, Chapter 3*, 2016.

[69] T. F. Pedersen, G. Demurtas, A. Sommer, and J. Hojstrup, "Measurement of Rotor Centre Flow Direction and Turbulence in Wind Farm Environment," *Journal of Physics: Conference Series 524 012167*, 2014.

[70] P. A. Fleming, A. K. Scholbrok, A. Jehu, S. Davoust, E. Osler, A. D. Wright, and A. Clifton, "Field-test Results Using a Nacelle-mounted Lidar for Improving Wind Turbine Power Capture by Reducing Yaw Misalignment," *Journal of Physics: Conference Series 524 012002*, 2014.

[71] P. D. Towers and B. L. Jones, "Wind Turbine Gust Estimation Using Remote Sensing Data," *UKACC International Conference on Control, Loughborough, U.K.*, 2014.

[72] A. K. Boyat and B. K. Joshi, "A Review Paper: Noise Models in Digital Image Processing," *Signal & Image Processing : An International Journal (SIPIJ)*, vol. 6, no. 2, 2015.