Improved Gaussian process approximations for spatial and flow fields

Talay Masood Cheema

Supervisor: Prof Carl Edward Rasmussen

Advisor: Prof Richard E Turner

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Churchill College March 2024

Declaration

This thesis is the result of my own work and includes nothing which is the outcome of work done in collaboration except as declared in the preface and specified in the text. It is not substantially the same as any work that has already been submitted, or, is being concurrently submitted, for any degree, diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the preface and specified in the text. It does not exceed the prescribed word limit for the relevant Degree Committee.

Talay Masood Cheema
March 2024

Abstract

Many modelling problems in scientific and engineering applications require quantitative models of uncertainty, for example to appropriately weight different sources of information, or to make risk-aware decisions. A natural way to formalise this is as a probabilistic model with a Bayesian point of view. When the main object of interest is a function – such as a spatial or flow field – Gaussian processes (GPs) are suitable probabilistic models. These can incorporate expert knowledge (priors), and data can be used to learn the model parameters. The cost of learning is high, motivating approximations, most notably variational approximations. This thesis improves on variational GP approximations in different settings.

Firstly, I consider approximate learning of low dimensional spatial fields. In Chapter 2, I argue for *precomputable* variational approximations, which only access the training data once during learning. These lead to significant computational savings, but are limited to an excessively narrow class of priors. In Chapter 3, I develop a class of precomputable approximations which first replace the prior with a periodic approximation. These *approximate Fourier series* methods outperform existing methods, have compelling theoretical guarantees, and are applicable to a broad class of priors – stationary priors whose covariance functions have well-defined spectral densities. Since many spatial fields are highly non-stationary, in Chapter 4 I construct a new class of non-stationary priors based on multiresolution (discrete wavelet) approximations, with a compatible precomputable approximation.

In Chapter 5 I turn to modelling flow fields for system identification problems. Here I propose a new approach – approximating the state posterior with Kalman smoothing – leading to faster and less biased learning, while also applicable to either discrete or continuous time.

Overall, this thesis presents improved GP variational approximations to make learning cheaper, with applicability to a broader class of priors, supported by a mix of theoretical guarantees and empirical evaluation.

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Acknowledgements

Good work is rarely the product of one person alone, and there are many people who enabled me to produce this thesis. What follows is, due to weakness of memory or in the interest of brevity, only a partial list of acknowledgements.

I am grateful to those in the lab who, despite adverse circumstances during the Covid-19 pandemic, fostered an intellectually stimulating, friendly, and productive environment. I appreciated my supervisor, Carl Edward Rasmussen, for his guidance on the right way to go about research, and for so often having the right question to orient me in scientifically worthwhile direction.

Lots of other students contributed to the environment in the lab, but I want to mention a few in particular. I enjoyed reading with Elre Oldewage, Austin Tripp, Marine Schimel, Kris Jensen, Stratis Markou, Adrian Goldwaser, and Matt Ashman. Each brought their own perspectives, often packed with both valuable insights and good humour; I learnt a lot from and with them, and the regular online meetings were a highlight during the extensive period we were barred from meeting in-person.

I enjoyed collaborating with Vidhi Lalchand and Kenza Tazi, and benefitted from advice from other students of Carl, including Vidhi, Alessandro Ialongo, Adrià Garriga-Alonso, and David Burt.

I also want to thank a handful of other lab members with whom I had fun or interesting interactions on many, many occasions – Siddharth Swaroop, Marton Havasi, Erik Daxberger, Greg Flamich, Miguel García-Ortegón, and James Allingham.

I am grateful to Siddharth, Kenza, and Will Tebutt for valuable feedback on the thesis, and to my examiners Carl Henrik Ek and Roderick Murray-Smith for an enjoyable viva, and feedback which helped me improve the thesis.

Outside of the lab, I'm very thankful to the college, which sourced my funding via the Bill Brown fund, supported me through various administrative hurdles, and let

us flex the accommodation rules a little at a trying time. My friends – especially those living in Cambridge – were a regular and fun escape.

I am eternally grateful for the support of my family. My parents sacrificed much so that I could have all the opportunities which paved my way here. My mum told me of a recent dream in which she ran down the street holding onto and announcing to all her 'mother's award'. She can soon live it out on King's Parade when I hand her my degree.

I am most grateful of all to my partner Cat, who found many typos for me, and without whom I doubt I would have had either the perseverance or the inner peace to see this through.

Table of contents

List of figures

List of tables

List of theorems

List of symbols

Gaussian process regression

A	Covariance matrix of y, $\mathsf{K_{ff}} + \sigma^2\mathsf{I}$
A'	Variational approximation to A, $\mathsf{K}_{uf}^*\mathsf{K}_{uu}\mathsf{K}_{uf} + \sigma^2\mathsf{I}$
B	The $M \times M$ matrix $\mathsf{K}_{uu} + \sigma^{-2}\mathsf{K}_{uf}\mathsf{K}_{uf}^*$
B'	The matrix to precompute, $\mathsf{K}_{uf}\mathsf{K}_{uf^*}$
c	Inter-domain covariance $\mathrm{Cov}[u, f]$
$\hat{c}$	Approximating inter-domain covariance
D	Dimension of $\mathcal{X}$, of x
f	Function with a GP prior
f	Function evaluated at the training inputs; elements $f(x_n)$
$\mathcal{GP}$	Gaussian process
$\mathcal{H}$	Space of f
$\mathcal{H}_k$	RKHS of k
ι	(For multiresolution covariance functions) the canonical scale
k	Covariance function
$\bar{k}$	Inducing domain covariance $\mathrm{Cov}[u, u]$
$\hat{\bar{k}}$	Approximating inducing domain covariance
$\mathcal{K}$	Covariance operator
$\hat{k}$	Approximating covariance
k_P	Periodised covariance function
$\hat{\mathsf{K}}_{\bullet\star}$	As K but for approximating covariances
$\mathsf{K}_{\bullet\star}$	Matrix of covariances between the variables $\bullet$ and $\star$
k_T	Truncated covariance function
k_{TP}	Periodised truncated covariance function
$\mathcal{L}$	Log marginal likelihood
ℓ	(Usually) covariance function lengthscales
M	Number of variational inducing values
N	Number of training points

ω	The observation functional
φ	Inducing features, $u = \langle f, \varphi \rangle$
ψ	(Usually) Mercer basis functions
ρ	Noise random variable
s	Spectral measure, or spectral density
$\widehat{s}$	Approximating spectrum
σ	Noise standard deviation
σ_f	GP marginal variance
t	The trace $\mathrm{tr}(\mathsf{K}_{\mathsf{ff}} - \mathsf{K}_{u\mathsf{f}}\mathsf{K}_{uu}\mathsf{K}_{u\mathsf{f}}^*)$
τ	The covariance function error trace $\mathrm{tr}\left(\mathsf{K}_{\mathsf{ff}} - \widehat{\mathsf{K}}_{\mathsf{ff}}\right)$
$\widehat{t}$	The approximating version of t
θ	Parameters of the model
u	Inducing values
v	Scaling function, scale j and shift t
$\mathcal{L}'$	Variational objective
$\mathcal{L}''$	Approximate variational objective using $k, \widehat{c}, \widehat{\bar{k}}$
W	Approximation width
w_{jt}	(Usually) wavelet function, scale j and shift t
$\mathcal{W}$	Gaussian white noise process
W_x	Data width
x	Training inputs
$\mathcal{X}$	Domain of f; space of x
x_*	Test inputs
y	Training outputs
y_*	Test outputs
z	Inducing inputs or frequencies
ζ	(Usually) GP mean function

State-space models

A	State transition or velocity matrix
$\widehat{\mathsf{A}}$	Kalman linearisation matrix
$\widehat{\mathsf{A}}'$	Approximate posterior transition matrix
β	Brownian motion
b	State transition or velocity bias vector
$\widehat{\mathsf{b}}$	Kalman linearisation bias
$\widehat{\mathsf{b}}'$	Approximate posterior transition bias
C	Measurement matrix

D	Dimension of x
Δ	Dimension of y
f	Dynamics function
g	Measurement function
κ	Process noise
L	Process noise covariance matrix factor, $\mathsf{Q} = \mathsf{L}\mathsf{L}^\top$
μ	Approximating marginal state mean
P	Measurement noise covariance matrix factor, $\mathsf{R} = \mathsf{P}\mathsf{P}^\top$
Q	Process noise covariance matrix
$\hat{\mathsf{Q}}$	Kalman linearisation transition variance
$\hat{\mathsf{Q}}'$	Approximate posterior transition variance
ρ	Measurement noise
R	Measurement noise covariance matrix
Σ	Approximating marginal state covariance
T	The number of measurements
t_j	The time of the j^{th} measurement
x	Latent state
y	Observations

Other symbols

$\langle \bullet, \star \rangle$	Inner product, conjugate linear in $\star$ *or* linear functional evaluation
$\otimes$	Kronecker product (matrices)
$*$	Khatri-Rao operator
$\star$	(as an operator) convolution
$\mathbb{C}$	Complex numbers
$\nabla\cdot$	Divergence operator
∇	Gradient operator
∇^2	Laplacian or Laplace-Beltrami operator, $\nabla \cdot \nabla$
δ	Dirac delta *or* Kronecker delta *or* evaluation functional
D_{KL}	KL divergence
e	Euler's number, $\exp(1)$
$\mathbb{E}$	Expectation
$\mathcal{F}$	Fourier transform operator
Γ	Gamma function *or* generalised Gamma function *or* Gamma distribution
I	Identity matrix
i	(Usually) the imaginary unit, $\sqrt{-1}$

$\mathcal{N}$	Gaussian distribution
p	Non-approximating probability distribution
$\mathbb{P}$	Probability measure
$\mathbb{R}$	Real numbers
sgn	Sign of a real valued argument
Ш_a	Dirac comb with spacing a
tr	Matrix trace
$\mathbb{Z}$	Integers

Acronyms

AFS	Approximate Fourier Series
BP	Belief propagation
CT	Continuous time
DT	Discrete time
EKF	Extended Kalman filter
EKS	Extended Kalman smoother
EM	Expectation maximisation
EP	Expectation propagation
FS	Fourier Series
GP	Gaussian process
iff	if and only if
IID	Independently and identically distributed
LBFGS	Limited memory BFGS
RKHS	Reproducing kernel Hilbert space
SE	Squared exponential
SLF	Statistically linearised filter
SLS	Statistically linearised smoother
SSM	State-space model
RKHS FS	RKHS Fourier Series (alias for VFF)
VFF	Variational Fourier Features (alias for RKHS FS)
VI	Variational inference

Chapter 1

Introduction

1.0.1 Problems in science, engineering, and beyond involve making decisions in the presence of uncertainty. This uncertainty could arise from many different sources, such as the limited accuracy of measurements, the finite precision of computations, or a lack of knowledge of the governing physical, biological, or social dynamics.

1.0.2 In these uncertain contexts, mathematical models endeavour to make quantitative predictions to inform decision making in a principled way. Historically, many mathematical models were constructed by combining deep expert knowledge, which impose strong constraints (for example, from well-founded physical laws) with empirical observations. More recently, there has been an increasing emphasis on the use of *data driven* models, where the behaviour of models is determined much more by gathered training data than by expert knowledge. The semi-automated process of learning models from data is machine learning.

1.0.3 Regardless of how models are created, it is useful to have a quantitative notion of uncertainty. This can be used to combine different sources of information (like measurements and expert knowledge), weighting each by their uncertainty, and to make effective, risk-aware, decisions.

1.0.4 Moreover, real problems usually require composing multiple models. To propagate the uncertainty in the basic inputs, such as measurements, to the final result, each component model needs to be well designed to quantitatively handle uncertainty. This motivates taking a probabilistic approach, where the estimates which

mathematical models deal with are not single values, but probability distributions over plausible values.

1.0.5 To give an idea of flavour of problems which motivate the work in this thesis, I provide two idealised examples.

Example 1.1 (Spatiotemporal modelling for flooding.) A government team is responsible for deciding how much to invest in flood defences in a particular drainage basin. They use several models in their decision making process.

- There is a hydrological model for the river basin, which tries to predict the spatiotemperal distribution of flood levels given the spatiotemporal distribution of rainfall. This depends on measured inputs, such as the terrain and soil surveys, as well as expert-derived standard models of the flow of water on and through the surface.

- The national weather forecaster provides long term forecasts of precipitation levels, including average levels of rainfall and statistical estimates of unusual events or extremes, such as consecutive days of heavy rainfall following dry days. These are based on localising long term, coarse scale models of the climate, based on a mix of coarse-scaled simulations of simplified versions of the governing partial differential equations, data-driven adjustments to account for the changing climate, and data-driven adjustments to scale the resolution of the simulation down to the local river basin.

- Economists provide a cost model for both the possible interventions and the damage caused by flooding, which is determined using a mixture of expert estimates and insurance data.

The team decide to make the interventions which minimise the expected cost, taking into account both the cost of intervention and the cost of damage. This approach works well, since each model takes into account the uncertainty estimates produced by others. For example, the hydrological model takes into account many different plausible climate change scenarios via the precipitation model, weighing each according to its plausibility.

Nonetheless, the decision is only as good as its basic assumptions. For example, a climate model that significantly underestimates future greenhouse gas emissions will be a poor predictor of future extreme precipitation due to storms.

The probabilistic parts of the spatial or spatiotemporal models are fit to large amounts of simulated or measured data – perhaps ranging from the order of 10^5 to 10^9 measurement locations. This might require the use of additional approximations. As with the simplifying assumptions used in the simulators, it is important that the approximations do not badly bias the conclusions.

Example 1.2 (Experiment design for scientific enquiry.) A research group is investigating the function of a single-celled organism. The true biochemical dynamics which govern the cell's systems are too complex for the group to simulate, but they have some ideas, based on the related literature, about simplified models which might explain the observed phenomena fairly well.

They suspect that the phenomena can be described by a differential equation model with four states, but they are not sure exactly what the dynamics function (or velocity field) should be. They are not interested in accurately identifying the velocity field for any possible state of the system, but want to learn about useful features of the dynamics, such as equilibria and their basins of attraction or repulsion.

They could evaluate possible models using experimental data, but the experiments are very expensive. They decide to take an *active learning* approach, where they choose experiments which maximise the information gain about the features they are interested in. This approach will require them to trade off between running experiments which explore parts of the state space which they have not explored much before – where uncertainty about the behaviour is high – and narrowing down regions of the state space which are more interesting – like the boundaries between basins of attraction and repulsion.

In this example, in addition to determining the model structure, the experts will need to subjectively quantify what it takes for behaviour to be interesting. Uncertainty is key to evaluating the value of exploration, and learning the parameters well (with low bias) is key to estimating the value of exploitation.

1.0.6 This thesis's focus is on models which estimate a spatial field by regression (as in components of Example 1.1) or the dynamics of a system (Example 1.2). There are many candidate models one could use to solve these problems, sharing many of the same underpinning principles. I focus on Gaussian process models since they serve as relatively simple but very flexible probabilistic models in both cases.

1.0.7 My emphasis in the case of spatial fields is in improving the computational scalability, so that the learning process can be fast even when the amount of training data is very large. My emphasis in the system dynamics case is in improving the quality of what is learnt, reducing biases in the learnt parameters without severe additional costs.

1.0.8 This chapter provides the basic background material – on probabilistic inference (Section 1.1), machine learning (Section 1.2), Gaussian processes (Section 1.3), and models of dynamical systems (Section 1.4) – needed to make the above sketch more precise. I close the chapter with an on overview of the thesis (Section 1.5). The expert reader, who is already steeped in knowledge of probabilistic machine learning, is advised to skip to Section 1.5 and return selectively to the intervening sections for clarifications or notational conventions.

1.1 Bayesian inference in principle

1.1.1 The Bayesian point of view is that probability is a mathematical language for quantifying plausibility (of outcomes, models, physical laws, and so on). This is well supported in principle by a number of classic arguments, most notably Cox's theorem, that systems to quantify beliefs subject to reasonable restrictions must be essentially equivalent to probability (Jaynes, 2003, chapter 2).

1.1.2 Let $\mathbb{P}$ measure probability, and let D and H be two measurable events. Then Bayes' rule says

$$\mathbb{P}[H|D] = \mathbb{P}[D|H]\frac{\mathbb{P}[H]}{\mathbb{P}[D]} \propto \mathbb{P}[D|H]\mathbb{P}[H] \tag{1.1}$$

where $\mathbb{P}[\bullet|E]$ is the conditional probability given, or assuming, E, for any measurable event E. We can ignore the denominator for interpretation, as its role is to normalise the conditional probabilities so that they sum to 1. Then if we think of H as being some hypothesis, like a candidate model or law for explaining data, and if D is the collected, measured, data, Bayes' rule says that after looking at the data, the *prior* plausibility of the hypothesis $\mathbb{P}[H]$ should be scaled up (or down) by how plausible the data is given the hypothesis (the *likelihood* of the hypothesis $\mathbb{P}[D|H]$).

1.1.3 An alternative perspective is to quantify how surprising an event is by mapping through negative log, which is nonnegative and monotonically decreasing on the interval $[0, 1]$. Then Bayes' rule reads

$$-\log \mathbb{P}[H|D] = -\log \mathbb{P}[H] - \log \mathbb{P}[D|H] + \text{ constant} \tag{1.2}$$

where the constant is with respect to the hypothesis H. Relative to other hypotheses, H gets more surprising by adding how surprising the data is given the hypothesis (adding $-\log \mathbb{P}[D|H]$).

1.1.4 Consider the general modelling setting where the model is labelled f, measurements labelled y are taken at inputs x, and the model has some adjustable underlying assumptions θ. Here and elsewhere, I will use the notation p for probability distributions and densities, relying on the argument of the function for disambiguation; see Appendix B.1 on the relationship between distributions, densities, and abuses of notation. Then the Bayesian distribution over models is

$$p(f|y, x, \theta) = \frac{p(y|f, x, \theta)p(f|x, \theta)}{p(y|x, \theta)}. \tag{1.3}$$

1.1.5 A non-Bayesian way of making predictions is to make some estimate of the model f, which I label $\hat{f}$. Then given some new input x_*, the predictions would be given by evaluating the model at the estimate. This could still produce a probabilistic prediction as motivated in Paragraph 1.0.3. The predictive distribution would be approximated as

$$p(y_*|x_*, x, y, \theta) \approx p(y_*|x_*, \hat{f}, \theta). \tag{1.4}$$

1.1.6 The Bayesian approach uses the residual uncertainty over the model calculated in Equation (1.3): predict using every possible model, weighted by their plausibility.

$$p(y_*|x, y, \theta) = \int p(y_*|x_*, f, x, y, \theta)p(f|x, y, \theta)\, df = \mathbb{E}_f[p(y_*|x_*, f, x, y, \theta)|x, y] \tag{1.5}$$

1.1.7 Here and elsewhere, $\mathbb{E}$ is the expectation operator. Comparing the two approaches, they coincide if the distribution over models has zero uncertainty at $\hat{f}$ after seeing the data. Averaging over multiple models is very important when different models lead to very different outcomes, as in different climate scenarios in Example 1.1.

Suppose that in that example the cost of deciding a when the outcome is y_* is given by a function $C(a, y_*)$, then the expected cost minimisation is

$$\min_a \mathbb{E}_{y_*}[C(a, y_*)|x_*, y, x, \theta]. \tag{1.6}$$

where the distribution of y_* is calculated by averaging over models (Equation (1.5)).

1.1.8 The Bayesian approach extends up the hierarchy of underlying assumptions. After averaging out f, we can average out the underlying assumptions θ in the same way: calculate its distribution given the data and its prior $p(\theta|x)$. Of course, for any of this to work, we must have produced the prior $p(\theta|x)$: this will have its own underlying assumptions.

1.1.9 Throughout, I assumed x was something fixed or known exactly, like some global input to the decision making system. This is reasonable in Example 1.1, where it represents a location in space. In some cases, the inputs to one model are the uncertain outputs of another.

1.1.10 The Bayesian approach to decision making is philosophically appealing, and there are clear advantages over decision making methods which do not appropriately propagate uncertainty, such as being able to weight different scenarios appropriately. But this is all very abstract: we need ways to come up with the probabilistic models for each component, including prior distributions on assumptions. And, given these models, we need to be able to calculate and store the posterior distributions. These are often going to be unfeasible tasks, so a pragmatic approach is needed. The next section reviews these practicalities in more detail.

1.2 Inference and learning in practice

1.2.0.1 The inferential approach of Section 1.1 contains two key operations: calculating posteriors (Equation (1.3)) and averaging over them (Equations (1.5) and (1.6)). For some combinations of probability distributions, both of these operations will produce a closed form result. But if we want to retain this property as we move further up the hierarchy of assumptions (calculating the posterior over θ, and so on), we would have to choose each prior very carefully to yield closed form solutions when

combined with the corresponding likelihood. That likelihood is the marginal likelihood of the assumptions, which is, at the first level,

$$p(y|x,\theta) = \int p(y|x,f,\theta)p(f|x,y,\theta)\,df. \tag{1.7}$$

1.2.0.2 In principle, we should choose our priors for f, θ and beyond to encode our beliefs before we see any data, yet for an arbitrary choice of priors, this will not lead to being able to calculate all the quantities we want. We will need to compromise in two ways:

1. only do inference so far up the hierarchy, and
2. choose probability distributions which compromise between what we actually believe and what is computationally convenient.

1.2.0.3 A generic strategy to minimise the impact of the second compromise is the use of sample-based inference, which avoids the need for explicit posterior distributions entirely. Sampling-based methods are also referred to as Monte Carlo methods (Murphy, 2023, Chapters 11-13). These usually rely on sampling from a candidate, or proposal, distribution which is easy to sample from, and (possibly stochastically) discarding samples that are not considered to be of sufficient quality, either because they are unlikely to be sampled under the posterior or because they are not sufficiently independent of the previous samples.

1.2.0.4 These approximate posterior samples can be used to calculate posterior expectations such as Equations (1.5) and (1.6). Explicitly, if the predictive samples are $y_*^{(1:S)}$, a simple Monte Carlo approximation to the expectation of some function g is

$$\mathbb{E}_{y_*}[g(y_*)|x,y] \approx S^{-1}\sum_{s=1}^{S} g(y_*^{(s)}). \tag{1.8}$$

1.2.0.5 Sampling-based methods are useful, since they impose fewer constraints on the form of the prior or likelihood functions; even systems without explicit likelihood functions can be handled (Lintusaari et al, 2018). However, the computational cost incurred in producing enough samples, sufficiently independent, can be prohibitive in many settings. The number of posterior samples which need to be stored to reuse the results of inference repeatedly may be very large, which compares unfavourably

to analytical methods, which produce a closed-form posterior summarised by a relatively small number of parameters.

1.2.0.6 Often, the amount of time or computational resources which a modeller is willing to dedicate to solving a problem are limited. Faster analytical approximations can be very impactful in these settings, though we may wish to be convinced that the compromises made along the way (for example, in the choice of prior) do not lead to unreasonable conclusions.

1.2.0.7 Typically, I am focused on the setting where we retain a Bayesian approach for the basic model f, which could be a spatial field from Example 1.1 or the dynamics function of Example 1.2, but where we merely make an estimate of the value of any other tunable parameters θ. This estimate can then be used as in Equation (1.4). Finding a good estimate of θ from the data is the *learning* problem.

1.2.1 Maximum likelihood and posterior

1.2.1.1 One way to select a parameter θ is as the most probable; that is

$$\hat{\theta} = \arg\max_{\theta} p(\theta|x, y) = \arg\max_{\theta} p(y|\theta, x)p(\theta) = \arg\max_{\theta} \log p(y|\theta, x) + \log p(\theta). \quad (1.9)$$

1.2.1.2 The last step uses the monotonicity of log. Unlike in Paragraph 1.1.3, these could be probability densities, so are not in the range $[0, 1]$, and the log value does not have the same interpretation.

1.2.1.3 We can treat this parameter estimation problem as a decision problem in the style of Equation (1.6) by minimising some cost $C(\hat{\theta}, \theta)$ on average over $p(\theta|x, y)$. If θ is discrete, maximising the posterior is equivalent to minimising the probability of getting θ wrong (setting $C(\hat{\theta}, \theta) = 1$ if $\theta = \hat{\theta}$, and 0 otherwise).[1]

1.2.1.4 If we are broadly uncertain about the value of θ, we could assume $p(\theta)$ is invariant and just maximise the marginal likelihood.

[1] Or, for optimists, maximising the probability of getting θ right.

1.2.1.5 Compared to maximising the likelihood, maximising the posterior (or *maximum a posteriori*, MAP) penalises the parameter settings which are thought to be less plausible, which should lead to more sensible solutions. A classic example is fitting a polynomial to data. If the number of coefficients is larger than the number of data points, then interpolating the data will yield a model under which the data has relatively high probability; it will maximise the likelihood. But we suspect that the underlying pattern is not some very wiggly polynomial, but one which is more regular (less wiggly). The irregular maximum likelihood is *overfitting* the data; it is probably explaining measurement noise as part of the underlying pattern, which will lead it to generalise very poorly. But by placing a prior on the coefficients with fast decay with respect to the polynomial order, the MAP solution will tend to be more regular.

1.2.1.6 But regardless of which of these procedures is used, the problems with point estimations persist. If there are multiple plausible explanations for the data which lead to very different outcomes, we will not be taking them into account (Paragraphs 1.1.5 to 1.1.7).

1.2.1.7 One option is to simply add a little bit of uncertainty around the estimate. Laplace's approximation makes this uncertainty Gaussian, with covariance equal to the local curvature of the objective, the log posterior (Murphy, 2023, Chapter 7). But if there are multiple plausible explanations which lead to very different outcomes, it may be because the log posterior has multiple modes. One approximate way to deal with this is to average θ over multiple posterior modes, each weighted by the posterior, or perhaps a local Laplace's approximation to each, though it would be challenging to determine the correct number of modes.

1.2.2 Elements of optimisation

1.2.2.1 So far, I have not dealt with the question of how to find a mode in the first place. If the log posterior, or whatever objective function is used for estimation, happens to be concave, then there are fast, well-engineered optimisation methods to converge to a global optimum, which must be the only mode (Boyd & Vandenberghe, 2004). These are centred on gradient-based optimisers. These can be used even when the objective is not concave, with the understanding that they will find a local optimum.

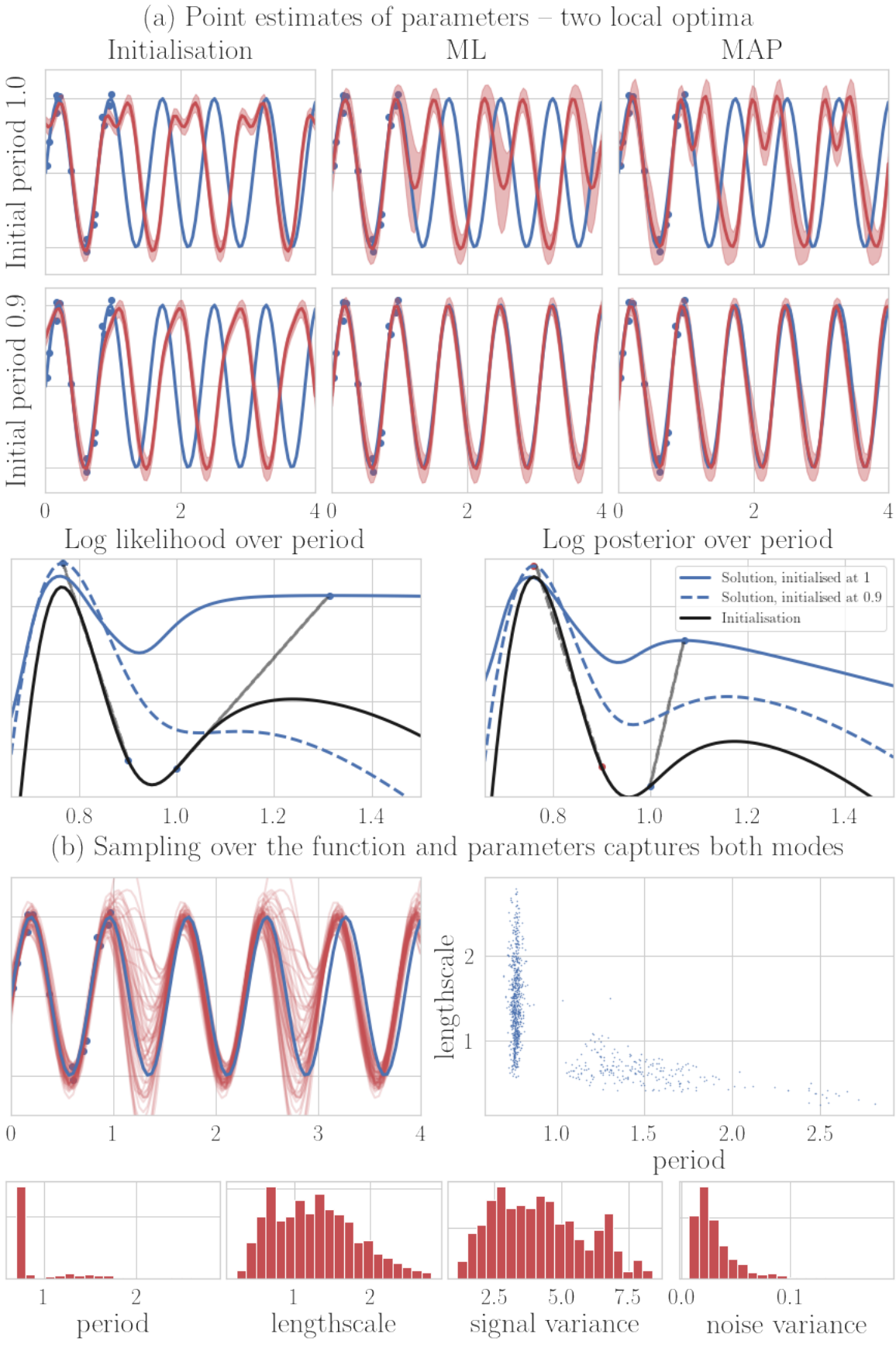

(a) Point estimates of parameters – two local optima
Initialisation
ML
MAP
Initial period 1.0
Initial period 0.9
Log likelihood over period
Log posterior over period
Solution, initialised at 1
Solution, initialised at 0.9
Initialisation
(b) Sampling over the function and parameters captures both modes
lengthscale
period
period
lengthscale
signal variance
noise variance

Figure 1.1 (On facing page.) Inference and learning with a periodic prior.

The underlying function (blue) is a sinusoid with period ≈ 0.769 ($y_n = 3 + \sin 2\pi 1.3 x_n + \rho_n$); a small number of measurements are sampled with each x_n sampled IID uniformly on $[0, 1]$ and the measurement corrupted by noise of standard deviation 0.1 (training data shown as blue dots).

(a) Parameter estimation. The model is a GP with a periodic covariance function (MacKay, 1998, Equation 47). It is initialised with either a period of 1 (top row) or 0.9 (second row), in either case with signal variance and lengthscale 1, and noise variance 0.01. The initial predictive posterior is plotted in the first column in red (here and elsewhere, the solid line is the mean, and the shaded region is two standard deviations either side). The parameters are estimated by ML (middle column) and MAP (right column). Two different modes are found, depending on the initialisation. For MAP, the priors are $\Gamma(1, 1)$ for every parameter except the period, which has the faster decaying $\Gamma(1, 25)$, all using the shape and rate parameterisation of the Gamma distribution.

The third row shows the value of the training objective for ML (left) and MAP (right) as a function of the period, with all other parameters fixed at the initialisation (black), the top row's solution (blue solid), and the second row's solution (blue dashed). The initial and final values are marked and joined by grey lines. The secondary mode is suppressed by the prior, but in any case is less favourable than the primary mode.

(b) Parameter inference. Now the period prior is relaxed to $\Gamma(1, 1)$ also, and the inference is performed using Hamiltonian Monte Carlo (HMC). This captures both modes, and appropriately shows that the primary mode dominates. The parameter samples are shown, pairwise for the period and lengthscale (upper right) and marginally as histograms (bottom row).

The two explanations have clear interpretations: either the period is short enough to be identified by the data, which is the more probable explanation; alternatively, the period is greater than 1, in which case it is not well identified by the data – but then the lengthscale cannot be too long in order for the data to oscillate so much over the interval $[0, 1]$. Because the posterior over the parameters includes both modes, the posterior predictive is less confident, but robust to initialisation issues.

1.2.2.2 The pitfalls of optimisation in practice are sketched in Figure 1.1. The prior asserts that the data is periodic, which is useful to generalisation if the period is learnt well. But there are two modes: either the underlying process has the period exposed by the data, or it has a period somewhat larger than the width of the data and therefore hard to determine from the data.

1.2.2.3 If we really believe the data is sufficient to determine the period, we should use a prior which does not support larger periods. If we express a more mild prior, as in the figure, the second mode is only slightly suppressed.

1.2.2.4 These two modes are both philosophically reasonable, but the point-estimate approach can lead to finding the secondary, much less plausible, mode only, for example if the optimiser is initialised carelessly with the implementation's default period of 1. This is a pathology which is not shared by sample-based inference in this case. However, it is worth noting that modes separated by regions of very low probability are a challenge for sample-based inference algorithms in general.

1.2.2.5 **Commonly used optimisers** Iterative optimisation algorithms follow a two step process.

1. Choose a direction in parameter space.
2. Choose a distance to travel in that direction.

1.2.2.6 Let the objective function be $\mathcal{L}(\theta)$. The basic first order gradient descent uses the observation that the gradient $\nabla\mathcal{L}$ points in the steepest direction. Although line search algorithms for the second step are well-studied, typically, the distance to travel is chosen heuristically as the gradient multiplied by a learning rate. One widespread variant is the Adam optimiser (Kingma & Ba, 2015), which is designed for the stochastic setting where $\mathcal{L}$ is estimated at each step by subsampling the data. Nonetheless, it is widely used elsewhere, including in this thesis. Adam adapts the scale of the learning rate dynamically to the parameters, and uses "momentum" (that is, the descent direction is a combination of the previous descent direction and the current gradient).

1.2.2.7 Second order methods are based on Newton's method, which chooses the descent direction which minimises a local quadratic approximation to $\mathcal{L}$, which therefore requires calculating and inverting the second derivatives $\nabla\nabla^{\mathrm{T}}\mathcal{L}$, also known as the Hessian matrix. The BFGS algorithm is a widely used computationally cheaper approximation to Newton's method which avoids inversion and estimates the influence of the curvature using only first derivatives (Broyden, 1970; Byrd et al, 1995; Fletcher, 1970; Goldfarb, 1970; Shanno, 1970).

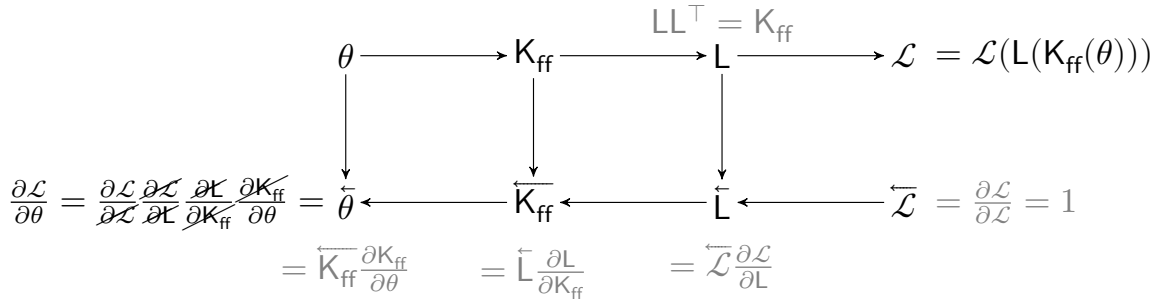


Figure 1.2 Example computation graph for automatic differentiation in reverse-mode, in the context of GP regression (Section 1.3). The parameters θ (top left) are used to calculate the covariance matrix K_{ff}, from which the Cholesky root L is calculated, which in turn allows for efficient calculation of the objective function $\mathcal{L}$. Each step in the operation has a defined adjoint operation, which defines how to post-multiply by the derivative matrix. This might be defined directly for an operation or algorithm, or the computation graph may be broken down further into elementary operations.

1.2.2.8 In either case, derivatives are typically calculated automatically using the following procedure, illustrated in Figure 1.2. The objective function $\mathcal{L}(\theta)$ is calculated using a sequence of operations. This is a forward pass along a computation graph, and the intermediate values are stored. A backward pass is then performed, which for each operation applies the adjoint operation, which is canonically post-multiplication by the matrix of partial derivatives (often known as the Jacobian). This typically requires the intermediate values. The completed backward pass produces the derivative.

1.2.2.9 Usually, this means that derivatives do not need explicit consideration, except if the objective is not differentiable everywhere, or if the operations do not have efficiently implemented adjoints, as in Chapter 4.

1.2.3 Analytical approximate inference

1.2.3.1 So far, I have principally discussed the learning process for the parameters θ, and inference over f has been assumed. Often, f will either not have a closed form posterior or the posterior will be prohibitively expensive to calculate. If sample-based methods are also not feasible, we want analytical methods which lead to good posterior approximations in closed form.

1.2.3.2 Variational inference (VI) (Murphy, 2023, Chapter 7) uses an objective which is a lower bound on the log marginal likelihood. It proposes a candidate posterior $q(f)$ from within a family of tractable distributions. The gap between the lower bound and the log marginal likelihood is the KL divergence D_{KL} from $q(f)$ to the posterior.

$$D_{KL}(q(f)\,||\,p(f|x,y,\theta)) = \int q(f)\log\frac{q(f)}{p(f|x,y,\theta)}\,df \tag{1.10}$$

1.2.3.3 This is a well known measure of the discrepancy between distributions, which is nonnegative and zero iff the two distributions are equivalent. Distributions which are closer in KL will generally have more similar properties.

1.2.3.4 The form for the variational objective is as follows. References to x – which has so far been conditioned on everywhere – are suppressed to lighten the notation.

$$\begin{aligned}
\mathcal{L}'(\theta,q) &= \int q(f)\log\frac{p(f,y|\theta)}{q(f)}\,df && (1.11)\\
&= \int q(f)\log\frac{p(f|y,\theta)p(y|\theta)}{q(f)}\,df = \log p(y|\theta) - D_{KL}(q(f)||p(f|y,\theta)) && (1.12)\\
&= \int q(f)\log\frac{p(y|f,\theta)p(f|\theta)}{q(f)}\,df && (1.13)\\
&= \int q(f)\log p(y|f,\theta)p(f|\theta)\,df - \int q(f)\log q(f)\,df && (1.14)
\end{aligned}$$

1.2.3.5 Equation (1.12) confirms the lower bound, and shows that the maximisation with respect to $q(f)$ is KL minimisation. Equation (1.14) gives a practical formula since it depends on distributions that are known (the likelihood and the prior).

1.2.3.6 Expectation maximisation (EM) is an iterative method that alternates between maximising the objective with respect to q (E-step; KL minimisation from Equation (1.12)) and with respect to the parameters θ (M-step; maximisation of the approximate expected log joint probability from Equation (1.14)). Where the KL can be minimised exactly, the first step always leaves $\mathcal{L}'$ equal to the marginal likelihood, so this iterative scheme maximises the marginal likelihood (although it may find a local optimum).

1.2.3.7 When exact posterior matching is not possible, maximising $\mathcal{L}'$ no longer maximises $\mathcal{L}$, and in particular the E-step no longer leave $\mathcal{L}'$ equal to $\mathcal{L}$. Then whether we

iterate E and M steps, or directly maximise $\mathcal{L}'$ with a gradient-based optimiser, we cannot discriminate between higher $\mathcal{L}'$ due to smaller KL and higher $\mathcal{L}'$ due to larger marginal likelihood. If the family of candidate posteriors is not flexible enough to express the true posterior, this can lead to severe biases. For example, a unimodal family approximating a multimodal posterior will tend to fit one mode. On the other hand, if the posterior family is sufficiently flexible, the KL may be small for every parameter setting, in which case there is unlikely to be significant bias. This is illustrated in the case of GP regression in Figure 1.3.

1.2.3.8 Regardless of which optimisation method is used, the approximations used and the priors assumed could affect the quality of either inference or learning. In particular, the impact of the choice of prior model and of approximations are not independent, but interact in ways which are often hard to predict.

1.2.3.9 We have two recourses for reassurance that the approach we use in practice is sensible. The first is theoretical. We can try to prove that, at least under some idealised conditions, the approximations or assumptions do not significantly negatively impact the learning or inference procedures. This could be, for example, by showing that the variational objective gets very close to the marginal likelihood, or showing that the conditions for a sample-based method to converge are met.

1.2.3.10 The second is practical. Even if there are theoretical guarantees in some idealised setting, this does not guarantee that a particular learnt model is good. Validation methods withhold some data to evaluate the predictive performance of a model.

1.2.3.11 These reassurances can be used in the design stage of an approximation, learning algorithm, or model; the results in this thesis are of these forms. They can also be used in production, to select the best of many different possible choices of model and approximations.

1.3 Gaussian process regression

1.3.0.1 GP regression has a long history in probabilistic modelling, particularly in the spatiotemporal setting (Rasmussen & Williams, 2006). It offers a first choice for expressive, probabilistic regression which has been used for a wide range of modelling

tasks, such as Bayesian optimisation (Garnett, 2023), control (Deisenroth & Rasmussen, 2011), and probabilistic versions of commonplace numerical routines (Hennig et al, 2022).

1.3.0.2 For a task which can more immediately be viewed as regression, consider statistical downscaling (Pielke Sr & Wilby, 2012) – predicting spatial fields at high resolution from low resolution simulations and measurements. A probabilistic approach is necessary to produce insights about extreme values in between non-extreme measurements (Bürger et al, 2012), and there is interest in better applying GPs to this task (Ekanayaka et al, 2022).

1.3.0.3 To introduce GP regression, consider the simple regression problem.

$$\left.\begin{aligned} y_n &= f(x_n) + \rho_n \\ \rho_n &\overset{\text{IID}}{\sim} p_{\text{noise}} \end{aligned}\right\} \text{ for } n \in \{1{:}N\} \tag{1.15}$$

1.3.0.4 There are already some big assumptions in this model. The measurements all have the same measurement noise (homoscedasticity), which is independent at each measurement. This might be reasonable in many settings, for instance where the noise is supposed to absorb unresolvable, subscale processes. But in other cases, it might be preferable to generalise this slightly. For example, if the dataset is generated not by empirical measurement but by simulation, or the outputs of some other probabilistic model, correlated or spatially variable noise might be more appropriate; see Paragraph 1.3.2.6 for suitable adjustments.

1.3.0.5 Assume that $y_n \in \mathbb{R}$ and $x_n \in \mathbb{R}^D$ for some $D \geq 1$. A convenient model for calculations is the conjugate Gaussian, linear parametric model. That is,

$$f(x_n) = \langle x_n, w \rangle \tag{1.16}$$

$$p_{\text{noise}}(\rho_n) = \mathcal{N}(\rho_n | 0, \sigma^2) \tag{1.17}$$

$$w \sim \mathcal{N}(\mu_w, \Sigma_w) \tag{1.18}$$

where $\langle x, w\rangle$ is the standard inner product on $\mathbb{R}^D$, elsewhere often denoted $w^\top x$ or $w \cdot x$, and $\mathcal{N}(x|\mu, \Sigma)$ is the Gaussian or Normal distribution, whose density is

$$\mathcal{N}(x|\mu, \Sigma) = \frac{1}{\sqrt{|2\pi\Sigma|}} e^{-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)}. \tag{1.19}$$

1.3.0.6 In the above equation and elsewhere, the notation $|\mathsf{A}|$ for a matrix A is understood to mean the determinant. Since the matrix Σ is symmetric and positive semi-definite, denoted $\Sigma \geq 0$, its eigenvectors are orthogonal, and the eigenvalues are nonnegative. Then it follows that the density's level sets are ellipsoids whose principal axes are given by the eigenvectors of Σ, with the relative lengths of the principle axes given by the corresponding eigenvalues.

1.3.0.7 The maximum likelihood model draws a straight line through the origin such that the squared distance from the measurements to the line is minimised. Equivalently, it finds the subspace for which the orthogonal projection of the data onto the subspace has minimal error.

1.3.0.8 The model can be straightforwardly generalised to produce a broader class of functions by mapping x through some real-valued basis functions $\varphi_{1:L}$, which can be collected into an L dimensional function φ. The only change in the form of the model is that now $w \in \mathbb{R}^L$ and

$$f(x_n) = \langle \varphi(x_n), w\rangle. \tag{1.20}$$

1.3.0.9 The maximum likelihood function is now the closest subspace to the data mapped into a feature space, rather than the raw data, and if the feature space is big enough the estimated model may interpolate the data. This yields the polynomial of best fit example when $D = 1$ and each $\varphi_\ell(x) = x^{\ell-1}$. In this setting, the prior becomes more important: it can weight the posterior over f away from irregular solutions.

1.3.0.10 For what follows, further details on calculations are available in Appendix A.2.

1.3.0.11 Collecting the observations into the vector $y \stackrel{\text{def}}{=} y_{1:N}$, and the feature function evaluations $\{\varphi(x_n)\}_{n\in\{1:N\}}$ into the $L \times N$ matrix Φ, the maximum likelihood

estimate has the classic closed form solution

$$\hat{w}_{\text{ML}} = (\Phi\Phi^\top)^{-1}\Phi y. \tag{1.21}$$

1.3.0.12 The posterior mode is at

$$\hat{w}_{\text{MAP}} = (\sigma^{-2}\Phi\Phi^\top + \Sigma_w^{-1})^{-1}(\Sigma_w^{-1}\mu_w + \sigma^{-2}\Phi y). \tag{1.22}$$

1.3.0.13 But the convenient feature of the Gaussian model is that the the posterior is also Gaussian (the prior is the likelihood's conjugate).

$$p(w|x, y, \mu_w, \Sigma_w) = \mathcal{N}(w|\hat{w}_{\text{MAP}}, (\sigma^{-2}\Phi\Phi^\top + \Sigma_w^{-1})^{-1}) \tag{1.23}$$

1.3.0.14 The covariance matrix has the rearrangement

$$\Sigma_w - \Sigma_w(\Sigma_w + \sigma^2(\Phi\Phi^\top)^{-1})\Sigma_w. \tag{1.24}$$

1.3.0.15 The marginal likelihood can be used to select the model or optimise the prior parameters. It is

$$p(y|x) = \mathcal{N}(y|\Phi^\top\mu_w, \Phi^\top\Sigma_w\Phi + \sigma^2\mathsf{I}). \tag{1.25}$$

1.3.0.16 If the basis functions are numerous or sophisticated, it is challenging to reason in terms of their relative weights. It is more helpful to frame the distributions in terms of the regression function evaluations. Let $\mathsf{f} = \mathsf{f}_{1:N}$ with each $\mathsf{f}_n = f(x_n)$. Then $\mathsf{f} = \Phi^\top w$, so

$$p(\mathsf{f}|\theta) = \mathcal{N}(\mathsf{f}|\Phi^\top\mu_w, \Phi^\top\Sigma_w\Phi). \tag{1.26}$$

1.3.0.17 The posterior can be similarly transformed.

1.3.0.18 Instead of parameterising the function via some specified feature functions and prior parameters μ_w, Σ_w, we could directly specify a Gaussian distribution on functions by designing a function to calculate the elements of $\Phi^\mathrm{T}\mu_w$ and $\Phi^\top\Sigma_w\Phi$ as a function of the locations $x_n, x_{n'}$. Then we extend this to arbitrary locations: let $\mathsf{f}_j = f(x_j)$ for any $x_j \in \mathbb{R}^D$. Then for any finite J, let $\mathsf{f} = \mathsf{f}_{1:J}$ be Gaussian distributed (Rasmussen & Williams, 2006)

$$\mathsf{f} \sim \mathcal{N}(\mathsf{f}|\zeta_\mathsf{f}, \mathsf{K}_{\mathsf{ff}}) \tag{1.27}$$

$$\text{where} \qquad [\zeta_{\mathsf{f}}]_j = \zeta(x_j) \stackrel{\text{def}}{=} \mathbb{E}[f(x_j)] \tag{1.28}$$

$$\text{and} \qquad [\mathsf{K}_{\mathsf{ff}}]_{jj'} = k(x_j, x_{j'}) \stackrel{\text{def}}{=} \mathrm{Cov}[f(x_j), f(x_{j'})]. \tag{1.29}$$

1.3.0.19 The learning objective and the posterior over f are available in closed form, and this is entirely comparable to the parametric setting above. Indeed, the parametric setting is a special case where

$$\zeta(x) = \sum_{\ell=0}^{L} [\mu_w]_\ell \varphi_\ell(x) \tag{1.30}$$

$$k(x, x') = \sum_{\ell=1}^{L} \sum_{\ell'=1}^{L} [\Sigma_w]_{\ell\ell'} \varphi_\ell(x) \varphi_{\ell'}. \tag{1.31}$$

1.3.0.20 The mean function ζ can be chosen more or less freely, but the covariance function k must yield a positive semi-definite matrix for any set of points. Such a bivariate function is called positive definite. We write the distribution as

$$f \sim \mathcal{GP}(\zeta, k) \tag{1.32}$$

and for convenience, I will now assume that $\zeta = 0$ to simplify the presentation; to apply the subsequent results for non-zero ζ, just shift the data to $y - \zeta_{\mathsf{f}}$.

1.3.0.21 As in the simpler, parametric, case, the posterior is also a Gaussian process. Explicitly, the posterior process has the predictive distribution

$$p(f(x_*)|x, y, \theta) = \mathcal{N}(f(x_*) \mid \mathsf{K}_{*\mathsf{f}}(\mathsf{K}_{\mathsf{ff}} + \sigma^2\mathsf{I})^{-1}y,\ \mathsf{K}_{**} - \mathsf{K}_{*\mathsf{f}}(\mathsf{K}_{\mathsf{ff}} + \sigma^2\mathsf{I})^{-1}\mathsf{K}_{\mathsf{f}*}) \tag{1.33}$$

where I is an identity matrix, x_* are the evaluation points, $f(x_*)$ is the vector of function evaluations at the evaluation points, K_{**} is its covariance matrix, and $\mathsf{K}_{*\mathsf{f}}$ is the covariance between f evaluated at the data points and the evaluation points. Note that $\mathsf{K}_{\mathsf{f}*} = \mathsf{K}_{*\mathsf{f}}^\top$.

1.3.0.22 The marginal likelihood is

$$p(y|x, \theta) = \mathcal{N}(y|0, \mathsf{K}_{\mathsf{ff}} + \sigma^2\mathsf{I}) \tag{1.34}$$

and so the log likelihood is

$$\mathcal{L}(\theta) \stackrel{\text{def}}{=} \log p(y|x,\theta) = -\frac{1}{2}\log|2\pi(\mathsf{K_{ff}} + \sigma^2\mathsf{I})| - \frac{1}{2}y^\top(\mathsf{K_{ff}} + \sigma^2\mathsf{I})^{-1}y \tag{1.35}$$

to which a log prior term can be added to get the log posterior. The model parameters θ are the noise standard deviation σ and any parameters the covariance function has. If a non-zero mean function is used, then its parameters can also be used. To distinguish these from first level parameters such as w, these are referred to as hyperparameters. But since there are no such first level parameters here (the model is *non-parametric*), I refer to θ as parameters without ambiguity.

1.3.0.23 As outlined in Section 1.2, learning the parameters θ generally requires additional approximations since their posteriors will not typically be available in closed form, so typically maximum likelihood is used.

1.3.0.24 The covariance function is a key determinant of the form of the posterior. From the posterior mean expression, looking at a single point x_*,

$$\mathbb{E}[f(x_*)|x,y] = \sum_n w_n k(x_*, x_n) \tag{1.36}$$

where $w_n = [(\mathsf{K_{ff}} + \sigma^2\mathsf{I})^{-1}y]_n$. In other words, compared to the parametric case, the basis functions are $k(\bullet, x_n)$; the nature of the basis functions is determined by the covariance function, and there are as many of them as data points. This is often seen as a desirable property, since the complexity of functions the model can produce scales up as it sees more data, as long as k has infinite rank (that is, $\{k(\bullet, x_j)\}_{j\in\{1:J\}}$ are linearly independent for any J as long as there are no repeats in x_j). In contrast, a parametric model has some fixed L-dimensional feature space: it will be hard to determine w when the size of the dataset is small, and the ability of the model to fit the data will tend to saturate when N becomes large.

1.3.0.25 A first example of a covariance function is the Gaussian or squared exponential covariance function,

$$k_{\text{SE}}(x,x') = \sigma_f^2 e^{-\frac{1}{2}(x-x')^\top\Lambda^{-1}(x-x')}. \tag{1.37}$$

This yields a prior on smooth (infinitely differentiable) functions. The parameter Λ should be positive definite, and σ_f should be positive. It is worth noting that this

covariance function can be derived by considering the parametric model with a continuum of Gaussian basis functions.

1.3.0.26 A rougher alternative is the Matérn-ν covariance function, which has the form

$$k_{\text{Matérn-}\nu}(x, x') = \sigma_f^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} s^2\right)^{\nu} K_\nu \left(\sqrt{2\nu} s^2\right) \tag{1.38}$$

$$s^2 = (x - x')^\top \Lambda^{-1} (x - x') \tag{1.39}$$

where Γ is the gamma function and K_ν is the modified Bessel function of the second kind. This yields a prior for functions which are $\lceil \nu - 1 \rceil$ times differentiable, and the $\nu \to \infty$ limit recovers the squared exponential case.

1.3.0.27 Both of these covariance functions are monotonically decaying functions of the separation $x - x'$, and they decay faster than any polynomial. Usually, the matrix Λ is chosen to be diagonal. Then an equivalent interpretation of the covariance function is as a standard, spherically symmetric version ($\Lambda = \mathsf{I}$) applied to data which has been linearly scaled down by dimension-dependent *lengthscale*, which is the square root of the diagonal elements of Λ.

1.3.0.28 These covariance functions are widely used, though many other choices exist, encoding different assumptions, such as for near-periodic functions (Wilson & Adams, 2013), multivariate divergence-free or irrotational fields (Álvarez et al, 2012; Berlinghieri et al, 2023), and fields with other desirable properties (Holderrieth et al, 2021). In general, covariance functions are closed under addition and multiplication, which provides a recipe for constructing new, more complex covariance functions out of familiar ones (Rasmussen & Williams, 2006, Chapter 4).

1.3.0.29 We can arrive at a modal approximation of the same form by replacing the inner product of the feature functions in parametric regression with the covariance function.

$$\langle \varphi(x), \varphi(x') \rangle = k(x, x') \tag{1.40}$$

1.3.0.30 For historical reasons, this is often referred to as kernel ridge regression. But the dominant cost of inference is inverting $(\mathsf{K}_{\mathsf{ff}} + \sigma^2 \mathsf{I})$, whose cost is in $\mathcal{O}(N^3)$. Indeed, if

we already have $\mathsf{K}_{*\mathsf{f}}(\mathsf{K}_{\mathsf{ff}} + \sigma^2\mathsf{I})^{-1}$ for the posterior mean, then the posterior covariance comes with almost no extra cost.

1.3.0.31 This is an important point: GP regression adds an uncertainty estimate to non-parametric regression almost for free. It provides a first choice to fit functions with uncertainty estimates, at least if Gaussian likelihoods are a reasonable assumption, so that everything comes out in closed form.

1.3.1 Approximations

1.3.1.1 Turning to the matter of computational cost, $\mathcal{O}(N^3)$ quickly becomes prohibitively large. There are several strategies to deal with this.

1.3.1.2 In this thesis, I focus on variational approximations, because they generally yield easy to calculate and computationally efficient approximations to the posterior and marginal likelihood. If posterior samples are needed in downstream components, it is possible to use variational approximations to efficiently generate high quality posterior function samples, wherever efficient routines for sampling from the prior exist (Wilson et al, 2020).

1.3.1.3 In their simplest form, GP variational approximations involve a kind of sparse approximation, where the covariance between data points is approximated using a set of inducing points.

1.3.1.4 Let the inducing points for $m \in \{1{:}M\}$ be z_m, and let $u = f(z_m)$. Then the conditional distribution of $f(x_*)$ given $u \stackrel{\text{def}}{=} u_{1:M}$ is Gaussian.

$$\left.\begin{bmatrix} f(x_*) \\ u \end{bmatrix}\right| z, x_* \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \mathsf{K}_{**} & \mathsf{K}_{*u} \\ \mathsf{K}_{u*} & \mathsf{K}_{uu} \end{bmatrix}\right) \tag{1.41}$$

$$\Rightarrow f(x_*)|u \sim \mathcal{N}(\mathsf{K}_{*u}\mathsf{K}_{uu}^{-1}u, \mathsf{K}_{**} - \mathsf{K}_{*u}\mathsf{K}_{uu}^{-1}\mathsf{K}_{u*}) \tag{1.42}$$

The second line follows using the standard formulas for Gaussian conditioning. The posterior approximation is given by the marginalisation

$$q(f) = \int p(f|u)q(u)\,du. \tag{1.43}$$

Compared to the true posterior, which conditions on y, the approximate posterior conditions on u, which should therefore act as a good summary for y. This is illustrated in Figure 1.3. If $M = N$ and $z_n = x_n$ for each $n \in \{1{:}N\}$, then $\mathsf{K}_{uf} = \mathsf{K}_{\mathsf{ff}} = \mathsf{K}_{uu}$, and so the approximate posterior is equal to the exact posterior. But the posteriors can become very close even with M much smaller than N (compare the upper and lower halves of Figure 1.3).

1.3.1.5 The optimal $q(u)$ (that is, the KL minimising one) is Gaussian, and has the following form (Titsias, 2009).

$$q(u) \sim \mathcal{N}(\mu_u, \Sigma_u) \quad \text{with} \quad \begin{cases} \Sigma_u^{-1} = \mathsf{K}_{uu}^{-1}(\mathsf{K}_{uu} + \sigma^{-2}\mathsf{K}_{uf}\mathsf{K}_{uf}^\top)K_{uu}^{-1}, \\ \mu_u = \sigma^{-2}\Sigma_u \mathsf{K}_{uu}^{-1}\mathsf{K}_{uf} y \end{cases} \tag{1.44}$$

1.3.1.6 This yields the following closed form variational objective, and its rearrangement using standard linear algebra manipulations (see Appendix A.2).

$$\mathcal{L}'(\theta, q(u)) = \log \mathcal{N}(y|0,\, \mathsf{K}_{uf}^\top \mathsf{K}_{uu}^{-1}\mathsf{K}_{uf} + \sigma^2 \mathsf{I}) - \frac{1}{2}\sigma^{-2}\operatorname{tr}(\mathsf{K}_{\mathsf{ff}} - \mathsf{K}_{uf}^\top \mathsf{K}_{uu}^{-1}\mathsf{K}_{uf}) \tag{1.45}$$

$$\begin{aligned} &= -\frac{1}{2}\left[N \log 2\pi\sigma^2 + \log \frac{|\mathsf{K}_{uu} + \sigma^{-2}\mathsf{K}_{uf}\mathsf{K}_{uf}^\top|}{|\mathsf{K}_{uu}|} + \sigma^{-2}||y||_2^2 \right. \\ &\quad \left. - \sigma^{-4} y^\top \mathsf{K}_{uf}^\top (\mathsf{K}_{uu} + \sigma^{-2}\mathsf{K}_{uf}\mathsf{K}_{uf}^\top)^{-1}\mathsf{K}_{uf} y + \sigma^{-2}\operatorname{tr}\left(\mathsf{K}_{\mathsf{ff}} - \mathsf{K}_{uf}^\top \mathsf{K}_{uu}^{-1}\mathsf{K}_{uf}\right) \right] \end{aligned} \tag{1.46}$$

1.3.1.7 The posterior predictive at new points x_* is calculated as

$$\begin{aligned} q(f(x_*)|x_*, x, y) &= \int p(f(x_*)|x, z, u) q(u)\, du \\ &= \mathcal{N}(f(x_*)|\mathsf{K}_{u*}^\top \mathsf{K}_{uu}^{-1}\mu_u,\, \mathsf{K}_{**} - \mathsf{K}_{u*}^\top \mathsf{K}_{uu}^{-1}\mathsf{K}_{u*} + \mathsf{K}_{u*}^\top \mathsf{K}_{uu}^{-1}\Sigma_u \mathsf{K}_{uu}^{-1}\mathsf{K}_{u*}). \end{aligned} \tag{1.47}$$

1.3.1.8 The difficult, cubic cost, operations have been isolated to K_{uu} and the matrix

$$\mathsf{K}_{uu} + \sigma^2 \mathsf{K}_{uf}\mathsf{K}_{uf}^\top. \tag{1.48}$$

1.3.1.9 But the cost is not totally reduced from $\mathcal{O}(N^3)$ to $\mathcal{O}(M^3)$. In practice, the $\mathcal{O}(NM^2)$ cost of computing the second term is usually the dominant cost.

1.3.1.10 So far, the choice of z has not been addressed in detail. It should act as a summary for x, which could be generated by K-means clustering or approximately sampling

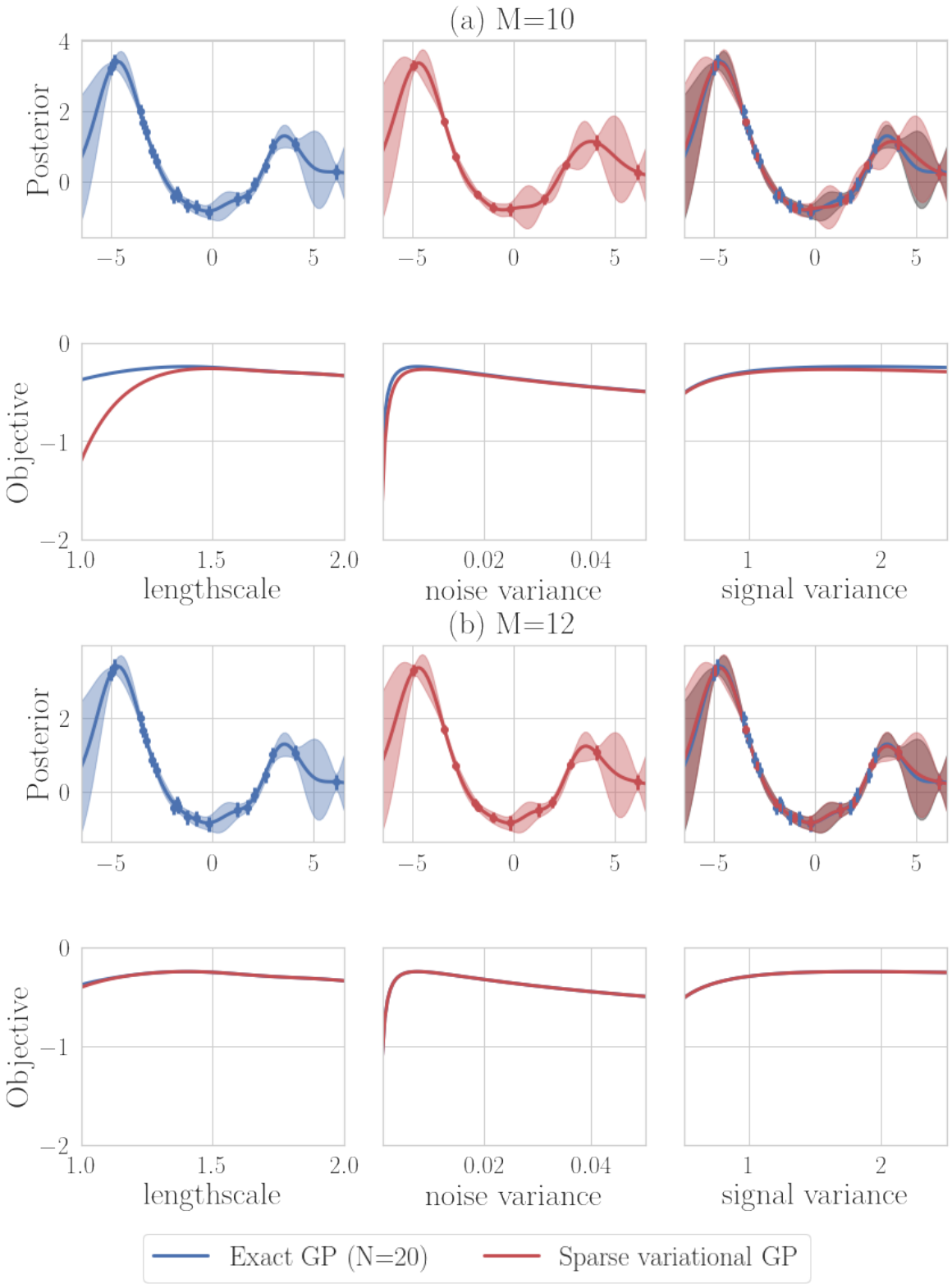

(a) M=10
Posterior
4
2
0
−5
0
5
Objective
0
−1
−2
1.0
1.5
2.0
lengthscale
0.02
0.04
noise variance
1
2
signal variance
(b) M=12
Posterior
Objective
Exact GP (N=20)
Sparse variational GP

Figure 1.3 (On facing page.) Conjugate GP regression using the exact method (blue) and the a sparse variational method (red). The training data ($N = 20$) is plotted in blue with error bars showing two standard deviations of the true observation noise. The data is sampled from a GP with a squared exponential covariance function with unit variance and lengthscale, and the noise variance is 0.01. The first and third rows show the posterior predictive distributions at the generating parameters, using 10 and 12 inducing points respectively, with the marginal inducing value distributions indicated in red. The right panel shows the two posteriors overlayed.

The second and fourth rows show the corresponding training objective as a function of each parameter, keeping the other parameters fixed at the maximum likelihood setting. Fewer inducing points leads to larger bias at low lengthscales, but the gap rapidly closes with a small increase in the number of points.

The maximum likelihood lengthscale, noise variance and signal variance are respectively approximately $(1.40, 0.0075, 1.85)$; the maximum variational objective parameters are almost identical when $M = 12$, but for $M = 10$ we get $(1.89, 0.013, 2.28)$.

from a determinantal point process (DPP). In the second case, using the covariance function as the kernel of the DPP is effective. To reduce the cost, leaving z fixed, or alternating the optimisation of parameters and z in an EM approach, can work well (Burt et al, 2020b).

1.3.1.11 A popular way of scaling machine learning methods to larger datasets is to stochastically estimate the loss function using minibatches. Then each optimisation step is less informative, but quicker. This is used in combination with variational approximations, for example. In this setting, the inducing mean and covariance are not calculated in closed form, but optimised numerically (Hensman et al, 2013, 2015).

1.3.1.12 In Chapters 2 to 4, I argue that a better strategy is to construct better variational posteriors, which require the data to be processed only once in a precomputation stage. Other approaches to tackle the scalability problem are discussed in Section 1.A.

1.3.2 Extensions and generalisations

1.3.2.1 I now briefly address variations on the initial setting of Equation (1.15). Firstly, consider the multi-output setting where each $y_n \in \mathbb{R}^\Delta$. The noise model could be a general multivariate Gaussian, with covariance $\mathsf{R} \geq 0$.

$$\rho_n \overset{\text{IID}}{\sim} \mathcal{N}(0, \mathsf{R}) \tag{1.49}$$

1.3.2.2 The covariance function now needs to be a function

$$k : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}^{\Delta \times \Delta} \geq 0. \tag{1.50}$$

In other words, one which produces a positive definite covariance function for any pair (x, x'), but any of the marginal covariance functions $[k(x, x')]_\delta$ still need to be positive definite.

1.3.2.3 One representation of the measurements is to stack them into a large length DN vector, and arrange the covariance matrices conformally. Then this is nothing but a special case where the covariance function has a special structure, other than the change of the noise covariance from $\sigma^2 \mathsf{I}_N$ to

$$\mathsf{R} \otimes \mathsf{I}_N \tag{1.51}$$

where the subscript is used to clarify the dimensionality of the identity matrix, and $\otimes$ denotes the Kronecker product.

1.3.2.4 For sparse regression the calculations proceed as in the single-output case, but now the key matrix for which the inverse and log determinant are needed is

$$\mathsf{K}_{uu} + \mathsf{K}_{u\mathsf{f}}(\mathsf{R} \otimes I_N)^{-1}\mathsf{K}_{u\mathsf{f}}^\top \tag{1.52}$$

since the noise covariance can no longer be pulled out of the centre of the second term.

1.3.2.5 If the output dimensions are modelled as independent, then $k(x, x')$ is always a diagonal matrix. Several methods for constructing covariance functions for the

multi-output setting involve linearly mixing together independent GPs, for example by matrix multiplication or convolution.

1.3.2.6 Note that if the noise variance is not IID (the heteroscedastic case), it can be handled by replacing $\sigma^2\mathsf{I}$ with the general noise covariance, as $R \otimes I_N$ in the multi-output setting.

1.4 Modelling dynamical systems

1.4.0.1 Now I turn to the problem of modelling systems such as in Example 1.2. The model proposed there is a first order differential equation. Alternatively, for practical purposes, if the measurements are taken at regular intervals, we can work in discrete time. Then the model is a first order difference equation. In either case, the problem involves *system identification* – learning the governing dynamics of the system from data.

1.4.0.2 For discrete time, the model is

$$\begin{aligned} x_t &= f(x_{t-1}) + \kappa_t \\ y_t &= g(x_t) + \rho_. \end{aligned} \tag{1.53}$$

1.4.0.3 Here κ and ρ are both white noise noise processes; κ is referred to as *process* noise, and ρ as observation or measurement noise. Each $x_t \in \mathbb{R}^D$ and each $y_t \in \mathbb{R}^\Delta$. The system identification problem is learning f, g. If the class of functions considered is sufficiently broad, then there are some non-identifiability concerns, as complexity in the dynamics could be traded off for complexity in the observer. On the other hand, if the class of functions is too narrow, then the model does not have the capacity for complex behaviour.

1.4.0.4 To understand what kind of model has the right capacity, it is important to understand the qualitative behaviour of the observations y which are generated by different models. In the case that the functions are both linear and time invariant,

and neglecting the noise processes for now

$$\begin{aligned} x_t &= \mathsf{A}x_{t-1} \\ y_t &= \mathsf{C}x_t, \end{aligned} \tag{1.54}$$

then the solutions are exponential in the eigenvalues of A. This includes solutions that blow up exponentially (eigenvalues with positive real part), converge exponentially (eigenvalues with negative real part) to a fixed point, or oscillate (eigenvalues with nonzero imaginary part). Inference and learning will also be straightforward in this case (Paragraph 1.4.1.10). Nonlinear models contain richer behaviour, such as multistability and chaos, but inference and learning become hard (Hirsch et al, 2004, Chapter 15; Särkkä et al, 2013).

1.4.0.5 The state transition distribution is given by

$$p(x_t|x_{t-1}, f) = p(\kappa_t = f(x_{t-1}) - x_{t-1}) \tag{1.55}$$

whereas in continuous time, the transition is governed by a stochastic differential equation, or an ordinary differential equation with uncertain initial conditions, depending on the noise model. We can interpret f as the flow map for a particular discretisation of the continuous time system, though not every f will correspond to a continuous time system. This is discussed further in Chapter 5.

1.4.0.6 **Why not use regression?** The *solutions* – the trajectory of a state over time – are multi-output functions of one variable, which is canonically time. Explicitly, the data is some IID sequences $y = y_{1:T} = y_{1:T_n}^{(1:N)}$, where each $y_j^{(n)}$ is measured at time $t_j^{(n)}$.

1.4.0.7 Fitting g is regression with unobserved inputs, and fitting f is regression with both the inputs and outputs not directly observed, so we expect this task to be much more difficult than standard regression.

1.4.0.8 But GP regression offers a simple way to fit functions, so we could use it to fit the solutions. Compared to the previous section, the input is now time (one dimensional) and we are in the multi-output setting. However, this has a number of limitations.

1.4.0.9 Firstly, if the generating system really is a dynamical system, then the trajectory does not depend only on time in the absolute sense but also on the initial conditions

(y_1, t_1). If the initial conditions at training and test time are the same, or this is easy to correct for by shifting time, then this is not an issue.

1.4.0.10 Secondly, regression is not suited to forecasting, whether that is predicting the extension of a trajectory forward in time, or predicting a new trajectory over a longer time period than the training data covers. The long term behaviour of a regression model will depend heavily on the prior or parametric form of the regression function. For example, in Figure 1.1, the prior was chosen to be periodic which led to good generalisation. For more complex trajectories, it could be challenging to determine suitable priors. But a dynamical system would typically evolve on a relatively limited part of its state space, making regression in this space more useful.

1.4.0.11 **Structured alternatives** Consider the two following possible classes of dynamical models. Autoregressive models (AR; Figure 1.4) model the evolution of the measurements as a function of preceding measurements. The *order* or lag of the model is the number of previous measurement on which the next measurement directly depends. State space models (SSMs; Figure 1.5) model the observation of conditionally independent measurements of an unobserved, or latent, first order AR model. Figures 1.4 and 1.5 show how the joint distribution of the variables factorises. Circular nodes are variable nodes, and square nodes are factor nodes, whose neighbours are conditionally dependent on one another. Unshaded variable nodes are unobserved during training.

1.4.0.12 Both of these are more structured models than in the basic regression setting. In the first part of this section, I briefly motivate using SSMs over AR models. Then, in Section 1.4.1 I give a brief overview of efficient inference algorithms for this structured setting, focusing on methods which are used in Chapter 5.

1.4.0.13 The general joint distribution of the observations is (assuming $T_n = T$ for all n for simplicity)

$$p(y) = p(y_1) \prod_{t=2}^{T} p(y_t | y_{1:t-1}) = \prod_{n=1}^{N} p(y_1^{(n)}) \prod_{t=2}^{T} p(y_t^{(n)} | y_{1:t-1}^{(n)}) \tag{1.56}$$

which allows for the modelling of possible long term dependencies between the observations, but inference will be challenging. The order L AR model has the joint

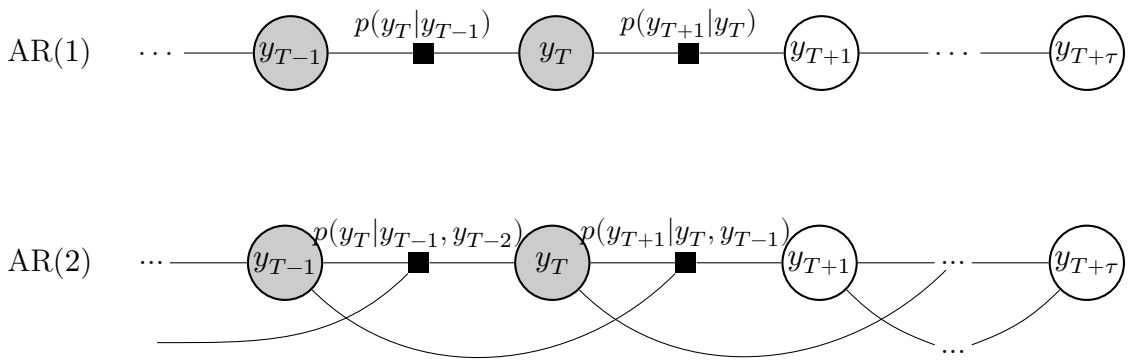


Figure 1.4 Factor graph segments for simple autoregressive models where we have observations up to T (Equation (1.57)), and a length τ prediction horizon. Top: first order AR; each y_t is directly dependent only on its neighbours. Bottom: second order AR; the dependency travels one step further.

distribution

$$p(y) = p(y_{1:L}) \prod_{t=L}^{T} p(y_t|y_{t-L:t-1}). \tag{1.57}$$

1.4.0.14 The SSM introduces an unobserved state x_t. The joint distribution is

$$p(y, x) = \left[\prod_{t=1}^{T} p(y_t|x_t)\right] p(x_1) \prod_{t=2}^{T} p(x_t|x_{t-1}) \tag{1.58}$$

so that the state x_t stores sufficient information to predict the observation y_t and the next state x_{t+1}. The first order AR model is a special case of the SSM where $y_t = x_t$, but in general the SSM allows the AR hidden state to be of higher dimension than the observation, so that long term dependencies in the measurements can be captured. Indeed, if we marginalise out the states, we find that the structure is equivalent to an infinite order AR model.

1.4.0.15 The SSM structure is convenient for modelling since models of *local* behaviour give rise to global phenomena. This is the same principle that motivates differential equation models (Hirsch et al, 2004).

1.4.0.16 The ability to generalise confidently over time in an SSM is a feature of this prior structure. The kinds of phenomena this prior encompasses are quite varied: they can include equilibrium systems, multistability, oscillations, and chaos. Contrast this

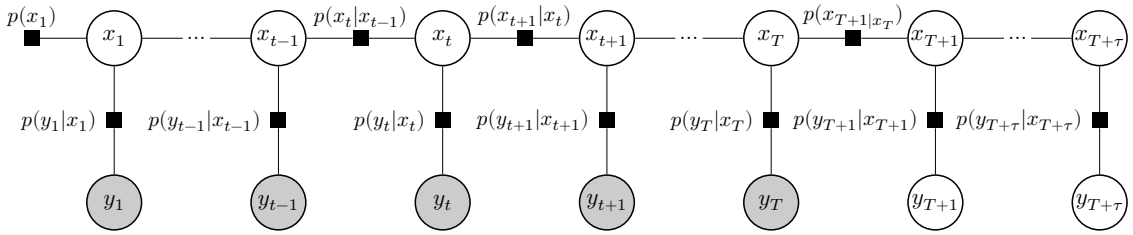


Figure 1.5 The factor graph of a state-space model (Equation (1.58)), assuming T steps of observations, and a length τ prediction horizon. Whilst there is a conditional independence assumption along the latent backbone, if we marginalise out the latents, the graph of the observations no longer factors.

with the regression approach, where commonly used priors would usually only include one of these.

1.4.0.17 GPs have been used as priors for system dynamics, in the earliest instances by Deisenroth & Mohamed (2012) and Frigola-Alcalde (2014). In Chapter 5, I extend existing work on variational GPSSMs. The remainder of this section provides much of the key background material on inference and learning in the more general state space context.

1.4.1 Inference and learning

1.4.1.1 It is worthwhile to first consider the general inference problem in models with cycle-free (or equivalently, tree-structured) factor graphs, including the SSM.

1.4.1.2 Reintroducing the parameters θ, the posterior marginals are given by

$$p(x_t|y_{1:T},\theta) = \int p(x_{1:T},\theta|y_{1:T})\,dx_{\setminus t} \propto \int p(y|x,\theta)p(x|\theta)\,dx_{\setminus t} \tag{1.59}$$

where $x_{\setminus t}$ is shorthand for $x_{\{1:T\}\setminus t}$. This appears to involve integrating over all the other variables. But if the factor graph showing the conditional dependencies is cycle-free, then the marginalisation admits a convenient recursive formulation. Example 1.3 gives the basic idea using a simple graph which includes some local parameters in addition to the states.

Example 1.3 (Message passing.) Consider the joint distribution

$$p(x_{1:3},\theta_{1:2},y_{1:2}) = p_a(y_2|x_3)p_b(x_3|x_2)p_c(x_2|x_1)p_d(x_1)p_e(y_1|x_2,\theta_{1:2})p_f(\theta_1)p_g(\theta_2) \tag{1.60}$$

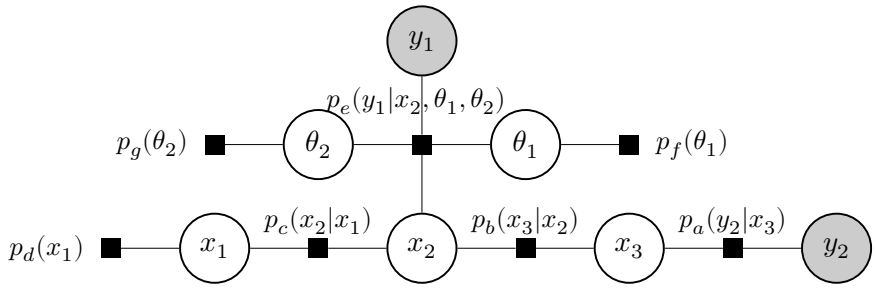


Figure 1.6 This is the factor graph for Example 1.3, where we aim to infer x_2 in the middle of a latent chain. Other, equivalent factorisations of the joint are clearly possible; for instance we could reverse the direction of the conditioning between x_1 and x_2 and move the prior node to x_2. However, the connections between variables do not change, except those of prior nodes.

where only y_1, y_2 are observed. This is represented as a factor graph in Figure 1.6. Suppose we wish to extract the posterior marginal $p(x_2|y)$ and the marginal likelihood $p(y)$. The joint distribution is

$$p(x_2, y) = \int p(x_{1:3}, \theta_{1:2}, y)\, dx_{\setminus 2}\, d\theta_{1:2} \tag{1.61}$$

$$= \int p_a(y_2|x_3) p_b(x_3|x_2) p_c(x_2|x_1) p_d(x_1) p_e(y_1|x_2, \theta_{1:2}) \tag{1.62}$$

$$\times\, p_f(\theta_1) p_g(\theta_2)\, d\theta_{1:2}$$

$$p(x_2|y)p(y) = \underbrace{\int p_c(x_2|x_1) p_d(x_1) dx_1}_{m_{c\to x_2}} \underbrace{\int p_e(y_1|x_2, \theta_{1:2}) p_f(\theta_1) p_g(\theta_2) d\theta_1 d\theta_2}_{m_{e\to x_2}} \tag{1.63}$$

$$\times \underbrace{\int p_b(x_3|x_2) p_a(y|x_3)\, dx_3}_{m_{b\to x_2}}$$

so the posterior is proportional to the three 'messages' coming in from the adjacent factor nodes. Now looking at the example message $m_{e\to x_2}$,

$$m_{e\to x_2} = \int p_e(y_1|x_2, \theta_{1:2}) \underbrace{p_f(\theta_1)}_{m_{\theta_1\to e}} \underbrace{p_g(\theta_2)}_{m_{\theta_2\to e}} d\theta_{1:2}. \tag{1.64}$$

Together, these furnish the general ideas that:

- factor nodes propagate the product of incoming messages from other variables, integrated over the factor;

- hidden variable nodes propagate incoming messages from other factors;
- observed variable nodes are ignored.

We could double check the second point by looking at the posterior of x_3, for instance.

1.4.1.3 For a general cycle-free factor graph with factors p_i and variables x_t, let $\nu(p_i)$ mean the set of variable nodes neighbouring p_i and similarly $\nu(x_t)$ for the factor nodes neighbouring x_t.

$$m_{i\to t}(x_t) = \int p_i(x_t, x_{\nu(p_i)\setminus t}, y) \prod_{\tau\in\nu(p_i)\setminus t} m_{\tau\to i}(x_t)\, dx_{\nu(p_i)\setminus t} \tag{1.65}$$

$$m_{\tau\to i}(x_t) = \prod_{j\in\nu(\tau)\setminus i} m_{j\to\tau}(x_t) \tag{1.66}$$

$$p(x_t|y,\theta_h) = \frac{1}{p(y|\theta_h)} \prod_{j\in\nu(x_t)} m_{j\to\tau}(x_t) = \frac{1}{p(y|\theta_h)} m_{t\to i}(x_t) m_{i\to t}(x_t) \tag{1.67}$$

$$\begin{aligned} p(y|\theta) &= \int p(x_t|y)p(y|\theta_h)\, dx_t = \int m_{t\to i}(x_t) m_{i\to t}(x_t)\, dx_t \\ &= \int p_i(x_{\nu(p_i)}) \prod_{\tau\in\nu(p_i)} m_{\tau\to i}(x_t) dx_{\nu(p_i)} \end{aligned} \tag{1.68}$$

1.4.1.4 In words, we propagate extrinsic messages (Equations (1.65) and (1.66)) starting from the leaves of the factor graph (consisting of observed variables and prior factors), and once we have gone through the whole graph, the intrinsic messages (Equations (1.67) and (1.68)) can be computed to get the posterior marginals and the marginal likelihood. This is the basic belief propagation (BP) algorithm.

1.4.1.5 There are two limitations to this. We might not be able to calculate the message passing integrals in closed form, and the graph might not be tree structured. Both of these occur in the identification of nonlinear systems. For the second limitation, the number of variables we have to integrate over for each message is the treewidth of the graph. However, we can ignore this issue and do loopy BP, in which we use the BP messages, and iterate until convergence (if it does converge). The fixed points of loopy BP (where it does converge) are local maxima of an approximation to a principled objective function (Murphy, 2023, Chapter 9).

1.4.1.6 Regardless of the graph structure, we are still left with the need to compute the integrals of the factor-to-variable type messages (Equations (1.65) and (1.66)).

1.4.1.7 **Expectation Propagation** One general method is expectation propagation (EP). One view of this (see Minka (2001) for more) is iterative refinement of the approximate posterior $q(x_t|y) = \prod_{j\in\nu(x_\tau)} m_{j\to\tau}(x_\tau)$ one factor at a time by minimising the KL divergence from the true update to the approximate. The arguments of the functions are suppressed for compactness.

$$\begin{aligned} m_{i\to t'}^{(\text{new})} &= \arg\min_{m_{i\to t'}} D_{KL}\left(\int p_i \prod_{t\in\nu(p_i)\setminus t'} m_{t'\to i}^{(\text{old})}\, dx_{\nu(p_i)\setminus t} \prod_{j\neq i} m_{j\to t'}^{(\text{old})} \middle\| Z_{t'}^{-1} \prod_{j\in\nu(x_{t'})} m_{j\to t'}^{(\text{old})}\right) \\ &= \arg\min_{m_{i\to t'}} D_{KL}\left(\underbrace{\int p_i \prod_{t\in\nu(p_i)\setminus t'} m_{t'\to i}^{(\text{old})}\, dx_{\nu(p_i)\setminus t}}_{\text{new factor from true } p_i} \prod_{j\neq i} m_{j\to t'}^{(\text{old})} \middle\| Z_{t'}^{-1} m_{i\to t'} \prod_{j\in\nu(x_{t'})\setminus i} m_{j\to t'}^{(\text{old})}\right) \end{aligned} \tag{1.69}$$

1.4.1.8 Here the Z values are just the appropriate normalising constants. If q is in an exponential family of distributions (such as Gaussian) then the KL is minimised with respect to q when the moments of p match the moments of q. If the moments are not available in closed form, then if the dimension of x_t is not too large, they can be estimated numerically.

1.4.1.9 Other objectives could be used in place of the KL minimisation. For example, in power EP (Minka, 2004), an α divergence is minimised, which requires only a small algorithmic variation, and recovers standard EP when $\alpha = 1$. The reverse KL $D_{KL}(q||p)$ could also be used; this is equivalent to using (global) variational inference (Paragraph 1.2.3.2) with a similarly structured approximate posterior. Like VI, EP yields an approximation for the log marginal likelihood for parameter optimisation, though it is no longer a strict lower bound.

1.4.1.10 **Message passing for the SSM** For the SSM's graph, it is convenient to collapse together the variable-to-factor and factor-to-variable messages. Then we can write down recursions for forward and backward messages along the latent chain m_{ft} and m_{bt} (from Equation (1.65)), which are combined with messages from the observations m_{ot} (see Figure 1.7 for illustration, and the Appendix A.2 for explicit expressions and calculation details).

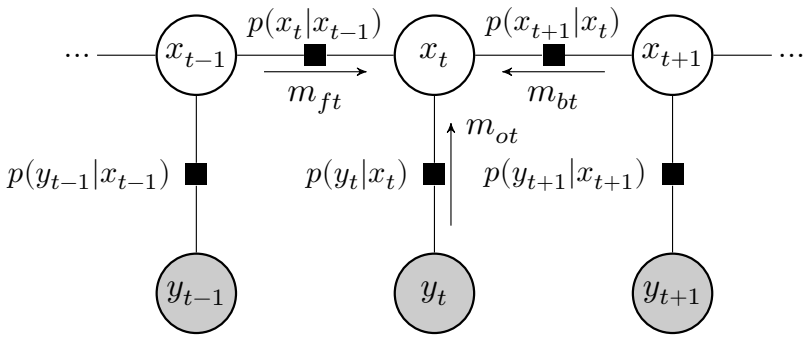


Figure 1.7 SSM with messages overlaid. The factor-to-variable and variable-to-factor messages are grouped together since there are only ever two variables connected to any one factor.

1.4.1.11 If the system is linear and the initial state and noise distributions are Gaussian, then the filtering and smoothing distributions are also Gaussian. Since linearity places severe limitations on the kinds of dynamics which can be expressed, we expect to have a nonlinear form for f (Paragraph 1.4.0.4). Then in general, inference will be intractable. But linearised or Gaussian approximations can be used more broadly (Paragraph 1.4.1.20).

1.4.1.12 The linear Gaussian SSM (LGSSM) is defined as follows.

$$\begin{aligned} x_t &= \mathsf{A}_{t-1}x_{t-1} + b_{t-1} + \kappa_{t-1} \\ y_t &= \mathsf{C}_t x_t + d_t + \rho_t \\ \kappa_t \sim \mathcal{N}(0, \mathsf{Q}_t), \rho_t \sim \mathcal{N}(0, \mathsf{R}_t), x_1 &\sim \mathcal{N}(\mu_1, \Sigma_1) \end{aligned} \tag{1.70}$$

1.4.1.13 This is allowed to vary over time, in order that it can be used for generalisations. The Kalman filtering and smoothing algorithms yield recursive updates for the marginal means and covariances, and are illustrated in Figure 1.8.

1.4.1.14 **Kalman filtering and smoothing** By constructing a recursion for $m_{ft}m_{ot}$ which is proportional to the filtering distribution $p(x_t|y_{1:t})$, we can produce a recursion for the filtering means and covariances $\vec{\mu}_t, \vec{\Sigma}_t$. The first step is to predict the next state (m_{ft}) (Figure 1.8, top row, red to red).

$$p(x_t|y_{1:t-1}) = \mathcal{N}(x_t|\mu_t^+, \Sigma_t^+) \text{ with } \begin{cases} \mu_t^+ = \mathsf{A}_{t-1}\vec{\mu}_{t-1} + b_{t-1} \\ \Sigma_t^+ = \mathsf{A}_{t-1}\Sigma_{t-1}\mathsf{A}_{t-1}^\top + \mathsf{Q}_{t-1} \end{cases} \tag{1.71}$$

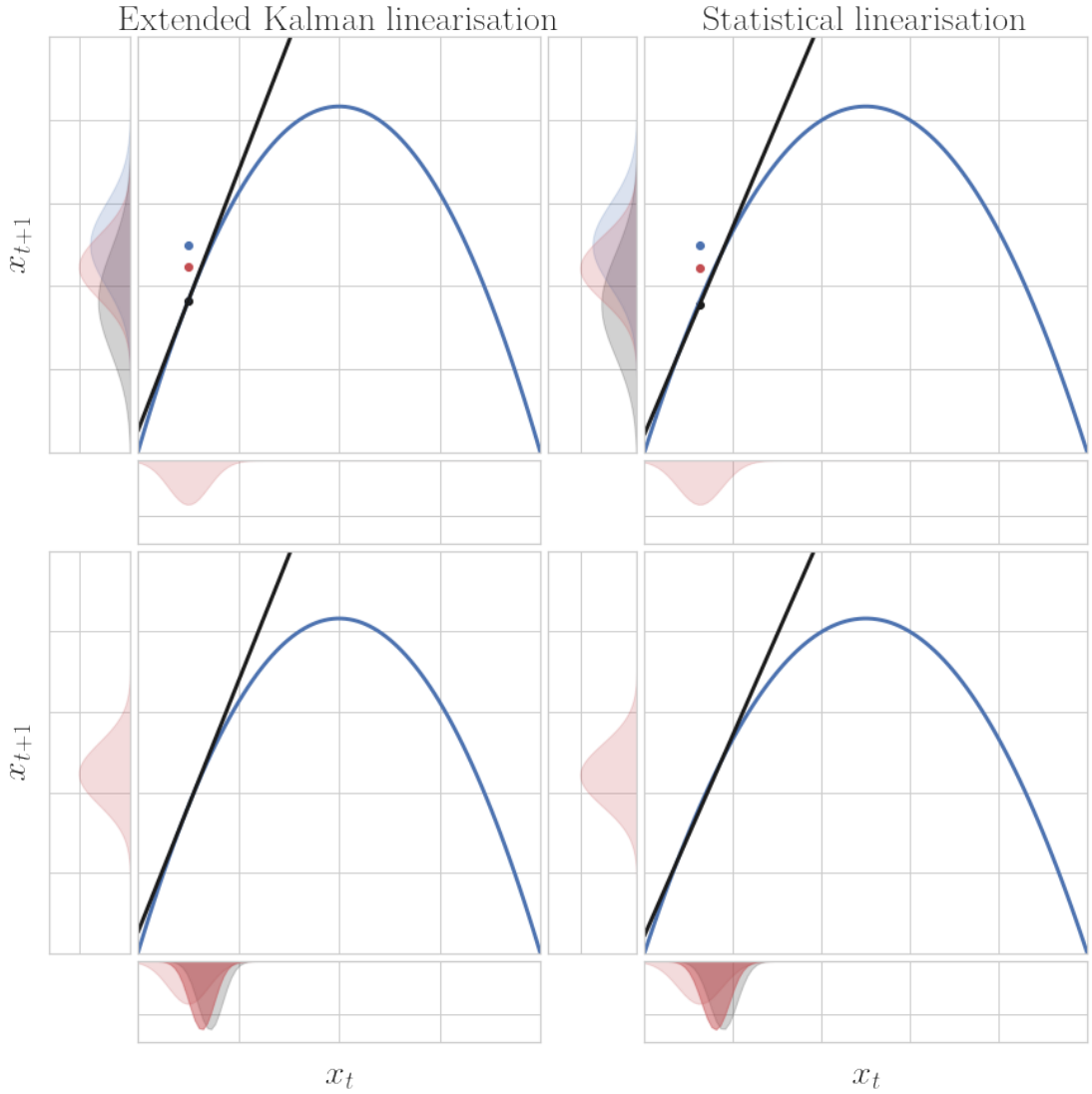


Figure 1.8 Gaussian filtering (top row) and smoothing (bottom row). For filtering, the current state distribution (red, below) is passed through the dynamics to generate the prediction (red, left hand side) according to Equation (1.71). This is combined with the implied distribution from the measurement (blue, left hand side) using Equation (1.74) to get complete the filtering step (black, left hand side). In smoothing, work backwards: the current state distribution (red, left hand side) is pushed back through the dynamics (black, below) and combined with the filtering distribution (light red, below, as before) to get the updated smoothing distribution (dark red, below), according to Equation (1.75). If the dynamics are not linear, as the blue curve, then locally linearise them, as the black line (Paragraphs 1.4.1.22 and 1.4.1.24)

1.4.1.15 For the marginal likelihood update, note

$$p(y_t|y_{1:t-1}) = \mathcal{N}(y_t|c_t^+, \mathsf{S}_t^+) \text{ with } \begin{cases} c_t^+ = \mathsf{C}_t\mu_t^+ + d_t \\ \mathsf{S}_t^+ = \mathsf{C}_t\Sigma_t^+\mathsf{C}_t^\top + \mathsf{R}_t. \end{cases} \tag{1.72}$$

1.4.1.16 Following from Equation (1.56), let $\mathcal{L}_t = \log p(y_t|y_{1:t})$, then $\mathcal{L}_T$ is the log marginal likelihood $\mathcal{L}$, and initialising $\mathcal{L}_0 = 0$, the update rule is just

$$\mathcal{L}_t = \mathcal{L}_{t-1} + \log \mathcal{N}(y_t|\mathsf{c}_t^+, \mathsf{S}_t^+). \tag{1.73}$$

1.4.1.17 To construct the filtering posterior, incorporate the observation (m_{ot}), weighting the influence of the observation and the prediction according to their respective uncertainties (Figure 1.8, top row, red and blue to black).

$$p(x_t|y_{1:t}) = \mathcal{N}(x_t|\vec{\mu}_t, \vec{\Sigma}_t) \text{ with } \begin{cases} \vec{\mu}_t = \mu_t^+ + \Sigma_t^+\mathsf{C}_t^\top\mathsf{S}_t^{+^{-1}}(y_t - c_t^+) \\ \vec{\Sigma}_t = \Sigma_t^+ - \Sigma_t^+\mathsf{C}_t^\top\mathsf{S}_t^{+^{-1}}\mathsf{C}_t\Sigma_t^+ \end{cases} \tag{1.74}$$

This recursion should be initialised with $\vec{\mu}_1 = \mu_1, \vec{\Sigma}_1 = \Sigma_1$.

1.4.1.18 To compute the smoothing posteriors $p(x_t|y_{1:T}) = (x_t|\mu_t, \Sigma_t)$, multiply by the backward message m_{bt} (Figure 1.8, bottom row, red to black, then light red and black to dark red).

$$p(x_t|y_{1:T}) = \mathcal{N}(x_t|\mu_t, \Sigma_t) \text{ with } \begin{cases} \mu_t = \vec{\mu}_t + \vec{\Sigma}_t\mathsf{A}_t^\top\Sigma_{t+1}^{+^{-1}}(\mu_{t+1} - \mu_{t+1}^+) \\ \Sigma_t = \vec{\Sigma}_t - \vec{\Sigma}_t\mathsf{A}_t^\top\Sigma_{t+1}^{+^{-1}}(\Sigma_{t+1}^+ - \Sigma_{t+1})\Sigma_{t+1}^{+^{-1}}\mathsf{A}_t\vec{\Sigma}_t \end{cases} \tag{1.75}$$

This recursion runs backwards from the state at the final observation, whose smoothing posterior should be initialised at the filtering posterior.

1.4.1.19 If the state transition and observation functions depend linearly on deterministic control inputs also, similar update equations exist. We just need to augment the x_t and mean vectors with the control inputs, and conformally pad the covariance matrices with zeroes.

1.4.1.20 **Approximate inference** For nonlinear systems, the broad options for state inference are as follows, with examples of well-known methods.

- Linearise f, g, (this assumes differentiability of f and g) and proceed as with the linear case (extended and unscented Kalman filter/smoother, statistical linearisation, σ-point filters/smoothers).
- Directly approximate the message passing integrals, assuming Gaussian state marginals, and iterate ((power) expectation propagation, variational inference).
- Propagate Monte Carlo approximations without assuming Gaussian state marginals (sequential Monte Carlo, particle filtering).

1.4.1.21 As earlier, I focus on the analytical approximations. Direct linearisation is based on the first terms of a Taylor series expansion. In these methods, an approximating LGSSM is formed, and exact inference is performed on the approximate parameters.

$$\begin{aligned} \hat{q}(x_t|x_{t-1}) &= \mathcal{N}(x_t|\hat{\mathsf{A}}_{t-1}x_{t-1} + \hat{b}_{t-1}, \hat{\mathsf{Q}}_t) \\ \hat{q}(y_t|x_t) &= \mathcal{N}(y_t|\hat{\mathsf{C}}_t x_t + \hat{d}_t, \hat{\mathsf{R}}_t) \end{aligned} \tag{1.76}$$

1.4.1.22 The straightforward version, known as the extended Kalman filter (EKF) or smoother (EKS) is linearisation at the mean.

$$x_t = f(x_{t-1}) + \kappa_{t-1} \tag{1.77}$$

$$\approx f(\vec{\mu}_{t-1}) + \left.\frac{df}{dx}\right|_{x=\vec{\mu}_{t-1}} (x_{t-1} - \vec{\mu}_{t-1}) + \kappa_{t-1} \tag{1.78}$$

$$\Rightarrow \hat{\mathsf{A}}_{t-1} = \left.\frac{df}{dx}\right|_{x=\vec{\mu}_{t-1}}, \; \hat{b}_{t-1} = f(\vec{\mu}_{t-1}) - \hat{\mathsf{A}}_{t-1}\vec{\mu}_{t-1}, \; \hat{\mathsf{Q}}_{t-1} = \mathsf{Q}_{t-1} \tag{1.79}$$

$$y_t = g(x_t) + \rho_t \tag{1.80}$$

$$\approx g(\mu_t) + \left.\frac{dg}{dx}\right|_{x=\vec{\mu}_t} (x_t - \mu_t) + \rho_t \tag{1.81}$$

$$\Rightarrow \hat{\mathsf{C}}_t = \left.\frac{dg}{dx}\right|_{x=\vec{\mu}_t}, \; \hat{d}_t = g(\vec{\mu}_t) - \hat{\mathsf{C}}_t\vec{\mu}_t, \; \hat{\mathsf{R}}_{t-1} = \mathsf{R}_t \tag{1.82}$$

1.4.1.23 This propagates the means through the nonlinearities, without any use of the higher order moments. The variance is propagated based on the linearisation at the mean with, for instance, no attention to the curvature of the nonlinearities. Intuitively this should only work well while the function is close to linear over lengthscales on the order of the standard deviation of the state posterior, which tends to grow (due to

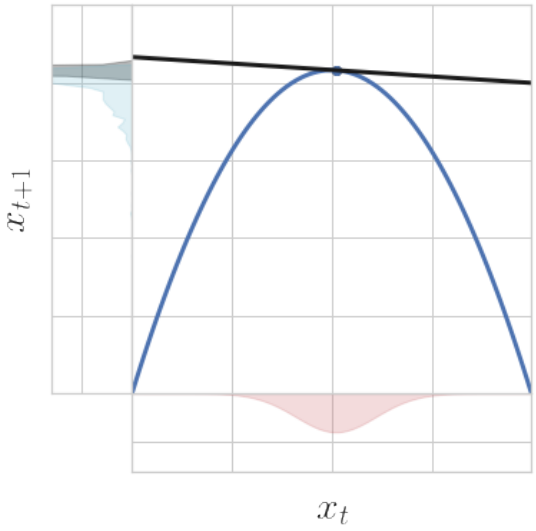


Figure 1.9 Linearisation where nonlinearity has a significant effect. The black distribution is the Kalman prediction, and the blue distribution is a sample-based estimate of the true one-step predictive. The linearised dynamics have nearly zero slope, so the uncertainty is suppressed. But the true predictive distribution has a large lower tail.

process noise) over time; hence long term predictions are expected to be particularly poor.

1.4.1.24 Another option is so-called statistical linearisation (the statistically linearised filter (SLF) or statistically linearised smoother (SLS)), where the exact predictive mean is used, and the Jacobian is also averaged over the input distribution.

$$x_t = \mathbb{E}_{x_{t-1}}[f(x_{t-1})] + \mathbb{E}_{x_{t-1}}\left[\frac{df}{dx}\right](x_{t-1} - \mu_{t-1}) + \kappa_{t-1} \tag{1.83}$$

$$\Rightarrow \hat{\mathsf{A}}_{t-1} = \mathbb{E}_{x_{t-1}}\left[\frac{df}{dx}\right],\ \hat{b}_{t-1} = \mathbb{E}_{x_{t-1}}[f(x_{t-1})] - \hat{\mathsf{A}}_{t-1}\vec{\mu}_{t-1},\ \hat{\mathsf{Q}}_{t-1} = \mathsf{Q}_{t-1} \tag{1.84}$$

$$y_t = \mathbb{E}_{x_t}[g(x_t)] + \mathbb{E}_{x_t}\left[\frac{dg}{dx}\right](x_t - \mu_t) + \rho_t \tag{1.85}$$

$$\Rightarrow \hat{\mathsf{C}}_t = \mathbb{E}_{x_t}\left[\frac{dg}{dx}\right],\ \hat{d}_t = \mathbb{E}_{x_t}[g(x_t)] - \hat{\mathsf{C}}_t\vec{\mu}_t,\ \hat{\mathsf{R}}_t = \mathsf{R}_t \tag{1.86}$$

1.4.1.25 The expectations are taken over the filtering distribution. This is similar to the previous case except the mean is propagated exactly, and we use the mean linearisation rather than the local linearisation. This is the direct linearisation which minimises the one-step predictive's mean squared error (Särkkä et al, 2013, Chapter

5), and it is also worth noting that

$$\hat{\mathsf{A}}_t = \mathbb{E}_{x_t}\left[\frac{df}{dx}\right] = \frac{d}{d\vec{\mu}_t}\mathbb{E}_{x_t}[f(x_t)] = \frac{d\vec{\mu}_{t+1}}{d\vec{\mu}_t} \tag{1.87}$$

so statistical linearisation is equivalent to EKF linearisation of a smoothed version of f which propagates means.

1.4.1.26 Either of these may be reasonable if the uncertainty in the state is quite small or the dynamics are quite close to linear, but where there are significant nonlinearities, the approximation may be very poor (Figure 1.9).

1.4.1.27 **Assumed density filtering** In assumed density filtering (ADF), we instead directly propagate moments. This is the same as the SLF for the propagation of the means, but differs in propagation of the covariance. EP can be viewed as a kind of assumed density smoothing. This iteratively refines the posterior approximation, trying to reconcile the inconsistent marginals and conditional (see Appendix A.2). The direct linearisation methods can be viewed as a certain way of approximating the moment computations (Wilkinson et al, 2020). In the EKS and SLS, the approximating distributions are not iteratively refined, in contrast to EP.

1.4.1.28 **VI for Gaussian approximations** VI offers a global approximation, which can be optimised jointly with the parameters in a gradient based way. A helpful result for the time series case is the following.

Lemma 1.1 (Optimal $q(x)$ for variational state-space approximations.) *For a joint distribution*

$$p(x, f, y) = \left[\prod_{t=1}^{T} p(y_t|x_t)\right] p(x_1) \left[\prod_{t=2}^{T} p(x_t|x_{t-1}, f)\right] p(f),$$

where y is the observation, the variational objective $\mathcal{L}' = \int q(x,\theta) \log \frac{p(x,f,y)}{q(x,f)}\, dx$ can be minimised by $q(x|f)$ factoring like $p(x|f)$.

Proof Following from Equation (1.12), and letting

$$H_q(x) = -\int q(x) \log q(x)\, dx \tag{1.88}$$

be the entropy of x and similarly the conditional entropy

$$H_q(x|f) = -\int\int q(x|f)\log q(x|f)\,dx\,df \tag{1.89}$$

then it follows that

$$\begin{aligned}\mathcal{L}' &= \int q(x)\log p(y|x,f)\,dx\,df + \int q(f)\log p(f)\,df \qquad (1.90)\\ &\quad + \int\int q(x|f)\log p(x|f)\,dx q(f)\,df + H_q(x|f) + H_q(f)\\ &= H_q(f) + \int\sum_{t=1}^{T}\int q(x_t|f,\theta)\log p(y_t|x_t,f)\,dx_t q(f)\,df + \int q(x_1)\log p(x_1)\,dx_1 \qquad (1.91)\\ &\quad + \int q(f)\int q(x_t,x_{t-1}|f)\log p(x_t|x_{t-1},f)\,dx_{t-1:t}\,df + H_q(x|f).\end{aligned}$$

So the objective depends only on singleton and pairwise marginals of $q(x|f)$, except in $H_q(x|f)$. But

$$H_q(x|f) = H_q(x_1|f) + \sum_{t=2}^{T} H_q(x_t|x_{1:t-1},f) \tag{1.92}$$

$$\leq H_q(x|f) = H_q(x_1|f) + \sum_{t=2}^{T} H_q(x_t|x_{t-1},f) \tag{1.93}$$

using the standard bound on conditional entropy $H(a|b) \leq H(a)$ (Cover & Thomas, 2006, Chapter 2). □

1.4.1.29 So we can optimally parameterise the joint distribution as

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t|\hat{\mathsf{A}}'_t x_{t-1} + \hat{b}'_t, \hat{Q}'_t). \tag{1.94}$$

1.4.1.30 This is similar form the direct linearisation methods, except that this is the posterior factor after smoothing. The smoothed pairwise marginals in the case of the Kalman smoother are

$$q(x_t, x_{t+1}) = \mathcal{N}\left(\begin{bmatrix}x_t\\ x_{t+1}\end{bmatrix}\middle|\begin{bmatrix}\mu_t\\ \mu_{t+1}\end{bmatrix}, \begin{bmatrix}\Sigma_t & \overrightarrow{\Sigma}_t \mathsf{A}_t^\top \Sigma_t^{+-1}\Sigma_{t+1}\\ \Sigma_{t+1}\Sigma_t^{+-1}\mathsf{A}_t\overrightarrow{\Sigma}_t & \Sigma_{t+1}\end{bmatrix}\right) \tag{1.95}$$

$$= \mathcal{N}\left(\begin{bmatrix}x_t\\ x_{t+1}\end{bmatrix}\middle|\begin{bmatrix}\mu_t\\ \hat{\mathsf{A}}'_t\mu_t + \hat{b}'_t\end{bmatrix}, \begin{bmatrix}\Sigma_t & \Sigma_t\hat{\mathsf{A}}_t{}'^\top\\ \hat{\mathsf{A}}'_t\Sigma_t & \hat{\mathsf{A}}'_t\Sigma_t\hat{\mathsf{A}}_t{}'^\top + \hat{\mathsf{Q}}'_t\end{bmatrix}\right) \tag{1.96}$$

so solving for $\hat{\mathsf{A}}'_t, \hat{b}'_t, \hat{\mathsf{Q}}'_t$ yields

$$\hat{\mathsf{A}}'_t = \Sigma_{t+1}\Sigma_t^{+^{-1}}\hat{\mathsf{A}}_t\overrightarrow{\Sigma}_t\Sigma_t^{-1} \tag{1.97}$$
$$\hat{b}'_t = \mu_{t+1} - \hat{\mathsf{A}}'_t\mu_t \tag{1.98}$$
$$\hat{\mathsf{Q}}'_t = \Sigma_{t+1} - \mathsf{A}'_t\Sigma_t{\mathsf{A}'_t}^\top. \tag{1.99}$$

1.4.1.31 VI can be used for learning even if the posterior is estimated by another method, such as approximate Kalman filtering (Särkkä et al, 2013, Chapter 12); alternating between smoothing the posterior and updating the parameters can be a form of variational EM (Paragraph 1.2.3.6). Alternatively, it is possible to directly obtain the VI-optimal Gaussian posterior, either by a direct iterative smoothing algorithm (Appendix A.2) or by a gradient-based method.

1.4.1.32 **Approximating moments** The integrals for filtering/smoothing or moment matching are not in general available in closed form, and I briefly detail how to estimate them. I focus on the Kalman prediction step, but the methodology generalises to the other operations straightforwardly. If we assume Gaussian states, then for the forward message we need

$$\int h(x_t)\mathcal{N}(x_t|\vec{\mu}_t, \overrightarrow{\Sigma}_t)\,dx_t = \int h(\vec{\mu}_t + \overrightarrow{\Sigma}_t^{\frac{1}{2}}\xi)\mathcal{N}(\xi|0,\mathsf{I})\,d\xi \tag{1.100}$$

where $h(x_t) = f(x_t)$ for the mean, and

$$h(x_t) = (f(x_t) - \mathbb{E}_{x_t}[f(x_t)])(f(x_t) - \mathbb{E}_{x_t}[f(x_t)])^\top + \mathsf{Q}_t$$

for the covariance.

1.4.1.33 These integrals can be approximated directly and deterministically by using so-called σ-points, which are values of ξ at which to evaluate h. The integral is approximated as the finite sum

$$\sum_j w_j h(\vec{\mu}_{t-1} + \overrightarrow{\Sigma}_{t-1}^{\frac{1}{2}}\xi_j) \tag{1.101}$$

where w_j are weights determined by choice of the σ points, and for any matrix K, $\mathsf{K}^{\frac{1}{2}}$ is any matrix which satisfies $\mathsf{K}^{\frac{1}{2}}{\mathsf{K}^{\frac{1}{2}}}^\top = \mathsf{K}$. The ℓ^{th} order σ-point filter/smoother refers to the points being selected so that the integral is exact whenever h is a polynomial of order $\leq \ell$, in which case the mean will be exact if f (or f^{-1} for

backward messages, or g^{-1} for observation messages) is polynomial up to order ℓ, but the variance will only be correct up to half this order (Särkkä et al, 2013, Chapters 6 and 10).

1.4.1.34 An alternative is to use stochastic approximations, where we use the Monte Carlo approximation to the integral as

$$\sum_j h(\vec{\mu}_{t-1} + \vec{\Sigma}_{t-1}^{\frac{1}{2}} \xi_j) \tag{1.102}$$

which will work for any h but will need many more samples to be accurate.

1.5 Thesis in brief

1.5.1 This thesis has two main parts:

- improving GP regression in the low dimensional setting (motivated by Example 1.1; Chapters 2 to 4);
- improving the suitability of GPSSMs to tackle the system identification task, in discrete or continuous time settings (motivated by Example 1.2; Chapter 5).

1.5.2 The main themes of both parts is that choosing the variational approximation carefully leads to better (faster or less biased) learning, and that the approximate inference method should not excessively constrain the choice of prior.

1.5.3 In the first part, we are in a setting where with enough features, we can make the variational objective arbitrarily close to the log marginal likelihood. The goal is to choose the features in such a way that the number of features needed is not too large, and that they have convenient structure so that the cost can be cut from $\mathcal{O}(NM^2)$ to $\mathcal{O}(M^3)$.

1.5.4 In the second part, it is no longer the case that enough features will make the variational approximation close to the log marginal likelihood, and it is unavoidable that there are going to be parameter biases. My focus is on constraining the approximations so that the biases are reduced without excessive extra cost.

1.5.5 The secondary theme is that it is important to have tools which work well for a broad range of priors, so that modellers can choose priors which better encode their assumptions. In Chapter 3, this is about designing an approximation which works for a broad class of stationary covariance functions. In Chapter 4, I consider the non-stationary setting, which is far less well explored, and so the focus is both on what makes a good approximation and on what makes a good prior. Finally, in Chapter 5, I consider how far discrete time SSMs are suitable for approximating systems governed by differential equation models.

1.5.6 **Precomputable approximations** Chapter 2 picks up where Section 1.3 left off, providing the more detailed technical background for understanding different variational GP approximations. The $M \times M$ matrix to invert (Equation (1.48)) contains a term costing $\mathcal{O}(NM^2)$, but if the variational approximation is chosen carefully with respect to the covariance function, this term can be made invariant to the parameters, and so precomputed before optimisation. This cuts the cost to $\mathcal{O}(M^3)$, and if we can also show that $M \ll N$, this leads to a major improvement.

1.5.7 The machinery needed is a more general definition of GPs (Definition 2.1) and the inducing variables u as linear functional evaluations (Section 2.2). The feature efficiency of a variational approximation – the rate at which M needs to grow for $\mathcal{L}' \to \mathcal{L}$ – can be characterised by bounding the trace

$$t = \operatorname{tr}(\mathsf{K}_{\mathsf{ff}} - \mathsf{K}_{u\mathsf{f}}^\top \mathsf{K}_{uu} \mathsf{K}_{u\mathsf{f}}) \tag{1.103}$$

following Burt et al (2020b). The relationship between this and the quantities of real interest – bounds on $\mathcal{L} - \mathcal{L}'$ and the corresponding growth rate of M with N – are summarised in Theorem 2.10 and Corollaries 2.1 to 2.3 in general, and in Section 2.3.1 for compositions of covariance functions, so that the rest of the work can proceed just by analysing t.

1.5.8 Work on precomputable approximations has been ongoing since the work of Hensman et al (2017). Ideally, we would have precomputable and feature efficient methods for any choice of covariance function, so that a modeller can be free in the important choice of prior (Figure 1.1).

1.5.9 In Chapter 2 I consider pre-existing methods, which are suitable for stationary covariance functions restricted to certain classes of compact sets. Generally, these methods are limited to low dimensional tensor products of 1D Matérn covariance functions.

1.5.10 The exception is the work of Dutordoir et al (2020) on spherical domains, which I extend both by extending to a broader class of manifolds (Definition 2.16 and Theorem 2.18), and by showing feature efficiency for covariance functions constructed using the restriction method of Solin & Särkkä (2020) (Theorem 2.19). But these are also limited in that they have much more symmetry than is typically assumed in regression (such as the same lengthscale in each dimension).

1.5.11 **Fourier series features for precomputable stationary approximations** The overall premise of precomputable methods is promising but exact variational methods do not seem very fruitful. In Chapter 3, I approximate stationary covariance functions on $\mathbb{R}^D$ by an approximate Fourier series (AFS). This can be viewed as first approximating the prior with a periodic one, and then approximating the corresponding posterior with a low frequency variational approximation. To believe that this is a good approximation, it is necessary to both show the prior approximation is good in a region including all of the data, and that the posterior approximation is feature efficient. I do this for two settings:

- where the Fourier series is that of the periodised covariance function (Theorems 3.1 and 3.2);
- where the Fourier series is estimated using the discrete Fourier transform (Theorems 3.5 and 3.7).

1.5.12 The guarantees for the posterior are asymptotically as strong as those for inducing points (Table 3.1), and the results in practice outperform previous work, though it is applicable to a broader class of covariance functions. There are numerous possible extensions of this method, and in particular in the case of gridded data, the computational cost is reduced to $\mathcal{O}(M)$ (Lemma 3.1).

1.5.13 **Non-stationary approximations: sparse Gibbs' and multiresolution covariance functions** Nonetheless, these methods are limited to lower dimensions and stationary covariance functions. They could be very useful for spatial or

spatiotemporal modelling tasks (Example 1.1) but the stationarity assumption, which is widespread in the GP literature, is problematic in many settings. In Chapter 4, I consider non-stationary GP modelling, and note that a hierarchical GP model using Gibbs' covariance function is the state of the art. I present sparse variational approximations to this model (Section 4.1.2). Furthermore, I introduce multiresolution GPs based on independent discrete wavelet features as a class of non-stationary covariance functions motivated by the signal processing literature (Section 4.3). I provide a precomputable variational approximation for these, and show that in practice, they offer a meaningful improvement over stationary models, with far less computational cost for learning than Gibbs' covariance function (Table 4.1).

1.5.14 **Kalman filtering in the GPSSM** In Chapter 5, I return to dynamical systems modelling. Variational approximations to GPSSMs have been studied previously, and authors have noted significant issues with the learnt parameters, particularly the noise models. These tend to be inflated by model mismatch, which significantly limits the predictive capability of the model, and is a well-known issue with variational approximations for time series models in general.

1.5.15 Previous work has pushed the introduction of more flexible variational approximate posteriors to reduce biases. This is successful, but comes at substantial computational cost. The main source of this is redundant parameterisation: the variational distribution has inducing value parameters to summarise $q(f)$ as well as the transition parameters of $q(x_t|x_{t-1}, f)$. But since $x_{t+1} \approx f(x_t)$, I suggest instead to parameterise only $q(f)$, and let $q(x)$ be determined by an inference algorithm other than VI, for which I use Kalman smoothing. This serves as a useful constraint, avoiding the biases VI is prone to, with far less cost (Section 5.2.1).

1.5.16 **Comparing discrete and continuous time models** In this work, I am also motivated by the continuous time setting; the Kalman filter based methods extend straightforwardly to continuous time, whereas methods that directly parameterise $q(x_t|x_{t-1}, f)$ suffer from an explosion of parameters. I briefly discuss in which kinds of settings discrete time SSMs are a suitable, computationally cheaper, substitute for continuous time ones in Section 5.1.

1.5.17 Chapter 6 provides some brief reflections on the content of this thesis, and outlines some suitable future directions.

1.A Reference notes and further reading

1.A.1 **Gaussian processes, variational inference, and convergence** Rasmussen & Williams (2006) provide a thorough introduction to the use of GPs for probabilistic modelling. The inducing point approximations have a long history, and were first placed into the variational framework presented here by Titsias (2009), with a more complete convergence theory provided by Burt et al (2020b). A good overview of the multi-output setting with variational approximations is provided by van der Wilk et al (2020). One can also use a hierarchical VI approach to aid in sampling the parameters (Lalchand et al, 2022a).

1.A.2 **Online updates** Most of the research on sparse variational methods for GP regression focuses on the batch setting, where the all of the inducing features are optimised to approximate all of the data, and jointly optimised with the hyperparameters.

1.A.3 The online setting was treated in general by Bui et al (2017), specifying a training objective for the distribution of a new set of inducing features $q(u_2)$ and the parameters θ using new data, given the distribution of the old set of inducing features $q(u_1)$, without revisiting the old data. In particular, this allows for efficient sequential updating, and controlling the cost of the approximation by choice of inducing features.

1.A.4 **Other analytical approximations** This thesis focuses on variational approximations, but the wider GP literature contains many other approaches, of which a brief selection is given here. EP is another alternative (Bui, 2017), and this can be combined with VI to create a hybrid method (Adam et al, 2021). For variational objectives, in the non-conjugate setting, the learning of the variational distribution can be accelerated by using natural gradients (Adam et al, 2020; Salimbeni et al, 2018; Wilkinson et al, 2020).

1.A.5 One alternative which is popular in the spatial setting is aggregating GPs using a Bayesian committee machine (Liu et al, 2018).

1.A.6 **Non-variational low rank approximation** One major class of approximations is finite rank approximations, which replace the covariance matrix with one approximated using a sum of basis functions. These could be chosen randomly, so that they form a simple Monte Carlo approximation to the covariance matrix (Rahimi & Recht, 2007, for example). Alternatively, they could be a fixed low rank approximation (Solin & Särkkä, 2020, for example).

1.A.7 It is doubtful that these are a better choice than variational approximations. Note that the variational objective has two terms: the first term is the log marginal likelihood with a low rank approximation to the covariance matrix, whilst the second term is an additional trace penalty. Low rank approximations in general miss out on this trace penalty. They may generally suffer from poor predictive covariances, and be much worse at approximating the posterior than the prior. They are, however, good for prior samples, which can be a useful part of a recipe for sampling from variational posteriors (Wilson et al, 2020).

1.A.8 **Exploiting structure in the kernel matrix** If the training inputs themselves have special structure, this can lead to special structure in the covariance matrix. This has been leveraged for stationary covariance functions with gridded inputs in one dimension (Cunningham et al, 2008, $\mathcal{O}(N \log N)$) and tensor products of stationary covariance functions with multidimensionally gridded inputs (Saatçi, 2011, Chapter 5; $\mathcal{O}(DN^{1+1/D})$).

1.A.9 These ideas were extended by the structured kernel interpolation (SKI) method of Wilson & Nickisch (2015) and Gardner et al (2018b), which approximates the covariance matrix by linearly interpolating from a grid of approximating points. This is efficient in low dimensions, reducing the cost to $\mathcal{O}(N + M \log M)$. However, it shares the pathology of low rank approximations; see Figure 3.12.

1.A.10 **Nearest neighbours** Nearest neighbour approximations use the fact that many covariance functions decay fast over space to constrain predictions to depend only on the nearest training points (Tran et al, 2021; Wu et al, 2022).

1.A.11 **Iterative approximations** Instead of producing feature-based approximations, an alternative is to directly approximate the linear solve and log determinant evaluations, and comparable operations in prediction (Gardner et al, 2018a; Pleiss et al, 2018). The log determinant necessitates a stochastic approximation. This may be a good fallback idea, but it is not clear that it is better than inducing points. Well-chosen inducing points, for example, amount to selecting a partial, pivoted Cholesky decomposition of the covariance matrix (Burt et al, 2020b), but unlike the conjugate gradient approximation to the linear solve, the variational approximation optimises its approximation in terms of the posterior KL, not in terms of the quality of the linear solve.

1.A.12 **GP regression by state-space inference** Another family of methods is to note that the GP is the solution to a linear stochastic differential equation. Then inference can be framed as a smoothing problem (Chang et al, 2020; Hartikainen & Särkkä, 2010; Reece & Roberts, 2010; Wilkinson et al, 2020). This is quite efficient if the data is gridded, since the equivalent SSM parameters are not time-varying. If the data is not gridded, then the process of calculating the SSM parameters dominates the cost. This can be made more efficient using inducing points (Tebbutt et al, 2021).

1.A.13 **Carefully designed covariance functions** A totally different way of improving scalability is to propose covariance functions which are inherently well structured for scalability, though this approach may run into trouble if the prior is not well suited to the data (Cohen et al, 2022; Jørgensen & Osborne, 2022).

1.A.14 **PAC-Bayes objective** An alternative approach to log marginal likelihood approximation is to use a PAC-Bayes objective, which targets stronger learning theoretic guarantees (Reeb et al, 2018).

1.A.15 **Latent inputs** The case where the inputs are not observed is thoroughly addressed in the variational setting by Damianou et al (2016).

1.A.16 **GP PDE approximations**Several works involve approximating differential equation models' solutions as GPs. Typically, these can be put in the form

$$\mathcal{D}f = g \tag{1.104}$$

for some differential operator $\mathcal{D}$. This could be a partial differential equation (PDE), for example. Then the solution for f is a spatial field, so a GP is a natural choice. If the forcing function g is given a GP prior, then this is known as the latent force model (Álvarez et al, 2009, 2013). if $\mathcal{D}$ is linear, then f is jointly Gaussian with g, and the inverse relationship is described in terms of the corresponding Green's functions $G(\bullet, s)$.

$$f(x) = \int G(x, s)g(s)\, ds \tag{1.105}$$

1.A.17 If the covariance function for g is k_g, then the corresponding covariance function for f is given by

$$k(\bullet, \star) = \int\int G(\bullet, s)G^*(\star, s')k_g(s, s')\, ds\, ds' \tag{1.106}$$

where the integrals can be approximated using low rank rank approximations to k_g where they are expensive to calculate or numerically unstable (Guarnizo & Álvarez, 2018; Moss et al, 2022), or alternatively one can use a low rank approximation to the adjoint model (Gahungu et al, 2022). Sequential inference, of particular interest for assimilation, has also been explored (Hartikainen & Särkkä, 2011).

1.A.18 This offers an approach to mix physical knowledge and simulated or measured data, but has two main shortcomings. Firstly, it assumes a linear PDE, and extensions to nonlinear systems make the inference more challenging (Ward et al, 2020). Secondly it assumes exact knowledge of the equation. The LFM has been used in a number of applications involving ODEs (Särkkä et al, 2019), but its use is more limited with spatiotemporal models.

1.A.19 Often, the model may be known only approximately, or some properties of the solution (such as being divergence-free) may be known. One way to to incorporate a regularisation term or additional likelihood term to penalise deviation from $\mathcal{D}f = g$. Note that training a model on the output of a simulator encourages a fit which approximately interpolates the simulated points, whilst this *soft-constraint* method encourages a fit which also has approximately correct derivatives.

1.A.20 **Physics-informed neural networks** This is the approach taken with physics informed neural networks (PINNs; Raissi et al, 2019), which have proven very successful at learning to solve well-known and nonlinear PDEs (Cuomo et al, 2022). However, PINNs suffer from a very hard to optimise loss landscape, even on

relatively simple problems (Krishnapriyan et al, 2021) and quantifying uncertainty is challenging. This is typical of non-Bayesian approaches to get parametric models to incorporate prior knowledge, though there is some interest in developing Bayesian PINNs (Yang et al, 2021).

1.A.21 Work to soft-constrain GP models in a similar way has received less attention, but has demonstrated capability on synthetic problems (Long et al, 2022; Raissi et al, 2017, 2018).

1.A.22 Note that the above focuses mainly on soft-constraining solutions based on physics. Pförtner et al (2023) looked at GP approximations to PDE solutions more generally, through the lens of models informed by physics. For a wider view of physics-informed machine learning for physical problems, see Karniadakis et al (2021); for weather and climate applications see Kashinath et al (2021).

1.A.23 **More on state-space approximations** For further details on state-space inference and learning, including sample-based methods, see Särkkä et al (2013) for the discrete time case and Särkkä & Solin (2019) for the continuous time case. Methods exist for parallelising Kalman filtering and smoothing (Särkkä & García-Fernández, 2020; Yaghoobi et al, 2021). For complementary approaches to GP modelling in the time series context, see the thesis of Turner (2011).

1.A.24 **State-space methods meet spatial fields** Often, for example in modelling climate or the weather, we are interested in modelling systems governed by PDEs. These are spatial fields evolving over time. When viewed as a state-space model, the state is the entire infinite dimensional spatial field. Then filtering and smoothing become very challenging tasks.

1.A.25 The task of combining simulated latents and measured data is often referred to as *data assimilation* (Rabier & Liu, 2003), and a widely used solution is the ensemble Kalman filter (Carrassi et al, 2018; Mandel, 2006). Like sparse variational GP regression, this makes a low rank approximation to the covariance matrix – in this case, the state's covariance matrix. Kamthe et al (2022) developed an approximate EP alternative. An interesting direction is to develop a sparse GP regression base alternative, where each marginal $q(x_t)$ is a GP.

Chapter 2

Fast inference and learning for stationary fields

2.0.1 In Example 1.1 and Section 1.3, Gaussian process models were motivated for spatial modelling problems where high quality uncertainty estimates are desirable, with the ability to input prior knowledge or inductive biases via the design of the mean and covariance functions. Yet, even with sparse variational approximations, the computational cost of learning remains in $\mathcal{O}(NM^2)$ for each optimisation step, which quickly becomes prohibitively large for large datasets.

2.0.2 When the measurement noise distribution can be modelled as white and Gaussian, more careful design of the variational features can lead to *precomputable* methods, where the bulk of the $\mathcal{O}(N)$ work can be done once before optimisation, reducing the cost to $\mathcal{O}(N + M^3)$. This is particularly convenient when the covariance function is *stationary* – on $\mathbb{R}^D$, that means invariant to translations. This leads to the remaining linear term being precomputable, leading to $\mathcal{O}(M^3)$ cost, as well as a number of convenient formalisms.

2.0.3 In this chapter, I introduce the general theoretical framework (Sections 2.1 to 2.3), describe and extend existing precomputable methods and describe their various shortcomings (Section 2.4 and **??**). The next chapter combines some of these ideas with additional approximations for a more widely applicable and practical scheme.

2.0.4 **Contributions** Sections 2.1 and 2.2 provide background material. Elements of the related literature are mentioned throughout. Sections 2.3 and 2.4 are novel

contributions. Proposition 2.7 is reproduced from my previous work (Cheema & Rasmussen, 2024a).

2.1 Mathematical preliminaries

2.1.1 In what follows, we will need a deeper technical understanding of several of the notions introduced in the first chapter, as well as to slightly generalise the standard results. Some of what follows is slightly abstract, but provides the necessary machinery for later developments. Additional technical details are given in Appendix B.

2.1.2 **General Gaussian processes** Firstly, let $\mathcal{X}$ be some general, metrisable[1], input space, generalising from $\mathbb{R}^D$. Let $\mathcal{H}$ be a vector space of functions defined on $\mathcal{X}$ – where usually we will be thinking of something like the space of continuous functions $C^0(\mathcal{X})$ (for precise conditions see Lifshits, 2012, Section 1.3). Then use $\mathcal{H}^*$ for the dual space of continuous linear functionals on $\mathcal{H}$, and for any $\varphi \in \mathcal{H}^*$ denote the evaluation of the linear functional as $\langle f, \varphi \rangle$.

Definition 2.1 (Gaussian process.) A random function $f \in \mathcal{H}$ is called a Gaussian process if for every $\varphi \in \mathcal{H}^*$, $\langle f, \varphi \rangle$ is a Gaussian random variable.

2.1.3 Generally, the scalar $\langle f, \varphi \rangle$ will be real. Occasionally it will be necessary to take complex-valued linear functionals, in which case the definition of a complex Gaussian random variable follows Definition 2.1, but with $\mathcal{H} = \mathbb{C} = \mathcal{H}^*$. In this, complex, case the covariance function k should be conjugate symmetric:

$$k(x, x') = k^*(x', x). \tag{2.1}$$

2.1.4 Let δ_x be the evaluation functional at $x \in \mathcal{X}$, that is

$$\langle f, \delta_x \rangle = f(x); \tag{2.2}$$

then it is clear how the above condition translates to the point evaluations of a Gaussian process being jointly Gaussian, as introduced in Section 1.3.

[1] A metrisable space is one whose topology can be induced by a metric (distance function), such as $\mathbb{R}^D$, or any other metric space.

2.1.5 **The covariance operator** Now, let $\mathcal{K} : \mathcal{H}^* \to \mathcal{H}$ be the (self-adjoint, positive definite) covariance operator corresponding to k (Bogachev, 1998, Chapter 2; Lifshits, 2012). This is a linear operator which describes the covariance of linear functional evaluations. That is, for any $\varphi, \psi \in \mathcal{H}^*$

$$\operatorname{Cov}[\langle f, \varphi\rangle, \langle f, \psi\rangle] = \langle \mathcal{K}\psi, \varphi\rangle \tag{2.3}$$

from which it follows (by setting $\varphi = \delta_x$, and applying the definition of the covariance function) that

$$[\mathcal{K}\psi](x) = \langle k(\bullet, x), \psi\rangle^* \tag{2.4}$$

where * denotes the complex conjugate where needed. This yields the following expressions for the statistics of functionals.

$$\begin{aligned} \langle f, \varphi\rangle &\sim \mathcal{N}(0, \langle \mathcal{K}\varphi, \varphi\rangle) \\ \operatorname{Cov}[\langle f, \psi\rangle, f(x)] &= \langle k^*(x, \bullet), \psi\rangle = \langle k(\bullet, x), \psi\rangle && \text{(by (2.1) and (2.2))} \\ \operatorname{Cov}[\langle f, \varphi\rangle, \langle f, \psi\rangle] &= \langle \mathcal{K}\psi, \varphi\rangle = \langle \langle k(\bullet, \star), \psi(\bullet)\rangle^*, \varphi(\star)\rangle && \text{(by (2.4))} \\ &= \langle \operatorname{Cov}[\langle f, \psi\rangle, f], \varphi\rangle \end{aligned} \tag{2.5}$$

2.1.6 Here and elsewhere $\bullet$ and $\star$ are used compatibly to mark the arguments of functions, to disambiguate functionals of bivariate functions.

2.1.7 **Inner products** So far, except the point evaluation functional δ_x, these linear functionals are quite abstract. For concreteness, consider the case where $\mathcal{H}$ contains only continuous functions which vanish at infinity. Then the functional has a realisation as integration with respect to a complex measure.

Theorem 2.2 (Riesz representation theorem.) *(Rudin, 1987, Theorem 6.19) Let $\mathcal{X}$ be a locally compact Hausdorff space (such as $\mathbb{R}^D$, or a compact subset), and $\mathcal{H}$ the space of continuous functions on $\mathcal{X}$ which vanish at infinity (Rudin, 1987, Definition 3.16). Then for any continuous linear functional φ and any $f \in \mathcal{H}$, there exists a unique regular countably additive complex Borel measure ν_φ on $\mathcal{X}$ such that*

$$\forall f \in \mathcal{H} : \quad \langle f, \varphi\rangle = \int_{\mathcal{X}} f(x)\, d\nu_\varphi(x).$$

2.1.8 In particular, in many cases, for example if $\mathcal{H}$ is a Hilbert space (where $\mathcal{H}^* = \mathcal{H}$), the linear functional can be realised in terms of an inner product, such as the

standard L^2 inner product. In this case, we identify the function for the inner product with the same symbol φ as the linear functional.

$$\langle f, \varphi \rangle = \int_{\mathcal{X}} \varphi^*(x) f(x) \, d\nu(x) \tag{2.6}$$

2.1.9 Generally, sample functions from a GP almost surely do not vanish at infinity, so the conditions of the representation theorem do not hold, and the inner product in Equation (2.6) is generally not well-defined. However, these formulations will suffice for most of the calculations which follow (for example, calculating linear functionals of the covariance function).

2.1.10 **The Mercer representation** Now let us look more closely at the covariance function. Consider the Hilbert space $L^2(\mathcal{X}, \nu)$ where ν is some finite positive measure on $\mathcal{X}$, equipped with the standard inner product

$$\langle \varphi, \psi \rangle_\nu = \int \psi^*(x) \varphi(x) \, d\nu(x). \tag{2.7}$$

2.1.11 Then the covariance operator of Equation (2.4) can be explicitly written as an integral operator by applying the Riesz representation theorem (Theorem 2.2).

$$[\mathcal{K}\psi](x) = \int \psi^*(x') k(x, x') \, d\nu(x)^* = \int k(x', x) \psi(x') \, d\nu(x) \tag{2.8}$$

2.1.12 If it is bounded and continuous, then it has countably many eigenvalues $\alpha_{1:\infty}$, all nonnegative, and the eigenfunctions $\psi_{1:\infty}$ can be chosen to be orthonormal in $L^2(\mathcal{X}, \nu)$. Then Mercer's theorem gives the relationship between the operator eigendecomposition and the covariance function.

Theorem 2.3 (Mercer's theorem.) *(Steinwart & Scovel, 2012, Corollary 3.5) Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{C}$ be a continuous covariance function, with $\mathcal{X}$ a Hausdorff space. Let the corresponding covariance operator have the representation of Equation (2.8) with ν a finite positive measure. Let* $\operatorname{supp} \nu$ *be the support of the measure – that is, the set of points $x \in \mathcal{X}$ such that any open set S containing x has $\nu(S) > 0$. Assume also that*

$$\int_{\mathcal{X}} k(x, x) \, d\nu(x) < \infty.$$

Then k admits the decomposition

$$k(x, x') = \sum_{m=1}^{\infty} \alpha_m \psi_m(x') \psi_m(x)^*$$

with uniform and absolute convergence on any $S^2 \subset [\operatorname{supp} \nu]^2$, and also $\sum_m \alpha_m < \infty$.

2.1.13 Mercer's theorem has been widely used in analysing the achievable performance of kernel-based approximations and learning algorithms. For the context of this chapter, note that the eigenvalues decay at least as fast as $1/m$ asymptotically in order that they are absolutely summable. Intuitively, the faster the decay, the fewer features are needed to approximate the covariance function, but both the specific eigenvalues and the rate of decay are not easily determined in general.

2.1.14 **Stationarity and Bochner's theorem** *Stationary* covariance functions on $\mathbb{R}^D$ are those which are invariant to translations:

$$k(x, x') = k(x - a, x' - a) = k(x - x') \stackrel{\text{def}}{=} k(r) \quad \forall a \in \mathbb{R}^D. \tag{2.9}$$

2.1.15 The covariance function can be characterised as a univariate function of the separation, for which I use the same symbol k. Bochner's theorem characterises the function in terms of its Fourier transform.

Theorem 2.4 (Bochner's theorem.) *(Rasmussen & Williams, 2006, Theorem 4.1) For any positive definite function k normalised such that $k(0) = 1$, there is a unique probability measure s such that*

$$k(x) = \int e^{-i2\pi\xi^\top x} ds(2\pi\xi).$$

2.1.16 I will use the same notation s to refer to the spectral density if it exists. Applying linearity of the Fourier transform, for general $k(0) = \sigma_f^2$, the spectral measure is scaled by σ_f^2. Compared to Theorem 2.3, the covariance function has been characterised by a *continuous* eigenbasis.

2.1.17 The Fourier transform of the Gaussian process itself is a (complex-valued) Gaussian white noise process $\mathcal{W}$ whose control measure is s (Lifshits, 2012, Section 3). That

is, formally $\mathcal{W}$ is a Gaussian process on sets of finite spectral measure and

$$\text{Cov}[\mathcal{W}(A), \mathcal{W}(B)] = s(A \cap B) \quad \forall A, B \,|\, s(A) < \infty \tag{2.10}$$

$$f(x) = \int e^{-i2\pi\xi^\top x} \, d\mathcal{W}(2\pi\xi). \tag{2.11}$$

2.1.18 The rate of decay of the spectral density can be related to the rate of decay of the covariance operator's eigenvalues. These relationships are easiest to make explicit in the case where the covariance function is also *isotropic* – that is, dependent only on $||r||$. Seeger et al (2008) bound the eigenvalues by the spectral density in this case for some covariance functions (for example, those whose spectral density decays polynomially and monotonically).

2.1.19 **Isotropic covariance functions on compact sets** Solin & Särkkä (2020) consider isotropic stationary covariances on $\mathbb{R}^D$ restricted to compact subsets, and vanishing at the boundary. The result is framed in terms of the Laplacian.

2.1.20 The Laplacian or Laplace-Beltrami operator ∇^2 is defined as the divergence of the gradient, which can be shown to be negative and self-adjoint.

Theorem 2.5 (Eigenbasis of isotropic covariance functions on compact bounded subsets of $\mathbb{R}^D$.) *Let k_{base} be an isotropic and stationary covariance function on $\mathbb{R}^D$ with spectral density s_{base}, let $\mathcal{X} \subset \mathbb{R}^D$ be a compact bounded set, with smooth enough boundary that the Dirichlet eigenvalues of the negative Laplacian $\lambda_1 \leq \lambda_2 \leq ...$ exist:*

$$-\nabla^2 \psi_m(x) = \lambda_m \psi_m(x) \quad x \in \mathcal{X}$$
$$\psi_m(x) = 0 \quad x \in \partial\mathcal{X}$$

with $\psi_{1:\infty}$ orthonormal. Let ν of Equation (2.8) be a volume measure for $\mathcal{X}$ (for example, Lebesgue measure for a cube). Let k be the projection of k_{base} onto $\text{span}\{\psi_{1:\infty}\}$*; then clearly k shares the eigenfunctions of the Laplacian. The eigenvalues are given by*

$$\alpha_m = s(\sqrt{\lambda_m}). \tag{2.12}$$

2.1.21 This is useful, since the Laplacian's eigenvalues depend only on the geometry of the space, and the spectral density can often be easily evaluated. It can either be used to construct covariance functions on compact bounded subsets (in which case the

projection which forces k to decay to 0 on the boundary is natural) or it can be used as an approximation (which will not be too severe if all of the data points are far from the boundary).

2.1.22 The rate at which the Laplacian's eigenvalues grow is characterised by Weyl's Law.

Theorem 2.6 (Weyl's Law.) *For a bounded domain $\mathcal{X} \subset \mathbb{R}^D$, let $\lambda_{1:\infty}$ be the Dirichlet eigenvalues of the Laplace-Beltrami operator arranged in non-decreasing order. Then*

$$\sqrt{\lambda_m} = C_{D,\mathcal{X}} m^{1/D} + o\left(m^{1/D}\right)$$

where the constant is given by

$$C_{D,\mathcal{X}} = \frac{(2\pi)^D}{\mathrm{Vol}(\mathcal{B}_D)\,\mathrm{Vol}(\mathcal{X})}$$

where Vol *returns the volume of a set, and $\mathcal{B}_D$ is the unit ball in $\mathbb{R}^D$.*

2.1.23 **The RKHS** Another view of the covariance function k is that of its reproducing kernel Hilbert space (RKHS) $\mathcal{H}_k$ (Rasmussen & Williams, 2006, Chapter 6), which is an inner product space for which $k(\bullet, x)$ acts as the evaluation functional δ_x:

$$\langle \varphi, k(\bullet, x) \rangle_{\mathcal{H}_k} = \varphi(x). \tag{2.13}$$

2.1.24 There is a one-to-one correspondence between RKHSs and covariance functions. The space can be viewed as one of smooth functions, for which the norm penalises roughness. One general way of evaluating the inner product emerges from the Mercer expansion (Theorem 2.3):

$$\langle \varphi, \varphi' \rangle_{\mathcal{H}_k} = \sum_m \frac{\langle \varphi, \psi_m \rangle_{L^2(\mathcal{X},\nu)} \langle \varphi', \psi_m \rangle^*_{L^2(\mathcal{X},\nu)}}{\alpha_m}. \tag{2.14}$$

2.1.25 The reproducing property (Equation (2.13)) can be verified by setting $\psi' = k(\bullet, x) = \sum_m \alpha_m \psi_m^*(x) \psi_m$.

$$\langle \varphi, k(\bullet, x) \rangle = \sum_m \frac{\langle \varphi, \psi_m \rangle_{L^2(\mathcal{X},\nu)} (\alpha_m \psi_m^*(x))^*}{\alpha_m} = \sum_m \langle \varphi, \psi_m \rangle_{L^2(\mathcal{X},\nu)} \psi_m^*(x) = \varphi(x) \tag{2.15}$$

2.1.26 The final equality follows assuming $\varphi \in \mathcal{H}_k$, noting $\psi_{1:\infty}$ is a complete orthonormal basis for $\mathcal{H}_k$. As with the Mercer eigenbasis in general, this concrete realisation will only be possible in special cases.

2.1.27 In the next section, I revisit the sparse variational approximation using more general inducing variables, outline the requirements for precomputability, and review theoretical aspects of feature efficiency from the literature.

2.2 General inducing variables

2.2.1 **The generalised model** Now, consider the slightly generalised single output, centred, conjugate model for GP regression:

$$\left.\begin{aligned} y_n &= \langle f, \omega_n \rangle + \rho_n \\ \rho_n &\overset{\text{IID}}{\sim} \mathcal{N}(0, \sigma^2) \end{aligned}\right\} \text{ for } n \in \{1{:}N\} \qquad f \sim \mathcal{GP}(0, k) \tag{2.16}$$

with each $\rho_n, y_n \in \mathbb{R}$, $f \in \mathcal{H}$, each $\omega_n \in \mathcal{H}^*$, and the domain of f is $\mathcal{X}$ which is assumed to be a D dimensional space. The only significant extension on the standard case (Section 1.3) is that the measurements are made using an arbitrary linear functional. The original case is recovered by setting each $\omega_n = \delta_{x_n}$. Now by analogy with the previous definition in Paragraph 1.3.0.18, define

$$\mathsf{f} = \mathsf{f}_{1:N}, \quad \mathsf{f}_n = \langle f, \omega_n \rangle \tag{2.17}$$

and then the formulae for the exact and variational approximate conditional, posterior, and training objective remain unchanged – only the expression for the elements of K_{ff} changes. Previously, it was convenient to assume the inputs $x_{1:N}$ were distinct, so that $\mathsf{K}_{\mathsf{ff}} > 0$. The equivalent condition in this generalised setting is the linear independence condition that for every ω_n, there is no sequence of coefficients $\alpha_{\{1:N\}\setminus n}$ such that

$$\langle f, \omega_n \rangle = \sum_{n' \neq n} \alpha_{n'} \langle f, \omega_{n'} \rangle. \tag{2.18}$$

2.2.2 For convenience, I reproduce the standard results here. Consider the $N \times N$ data covariance matrix.

$$\mathsf{A} = \mathsf{K}_{\mathsf{ff}} + \sigma^2 \mathsf{I} \tag{2.19}$$

Rewriting the posterior predictive (Equation (1.33)) and marginal likelihood (Equation (1.34)) in terms of A yields the following.

$$p(f(x_*)|x, y, \theta) = \mathcal{N}(f(x_*) \mid \mathsf{K}_{*\mathsf{f}}\mathsf{A}^{-1}y,\ \mathsf{K}_{**} - \mathsf{K}_{*\mathsf{f}}\mathsf{A}^{-1}\mathsf{K}_{f*}) \tag{2.20}$$

$$p(y|x, \theta) = \mathcal{N}(y|0, \mathsf{A}) \tag{2.21}$$

$$\mathcal{L}(\theta) \stackrel{\text{def}}{=} \log p(y|x, \theta) = -\frac{1}{2}\log|\mathsf{A}| - \frac{1}{2}y^\top \mathsf{A}^{-1}y \tag{2.22}$$

2.2.3 As mentioned in the previous chapter (Paragraph 1.3.0.30), the $\mathcal{O}(N^3)$ cost of each learning step arises from calculating the inverse and log determinant of this data covariance.

2.2.4 **The variational objective** For the variational approximation, we construct an approximate posterior $q(f) = \int p(f|u)q(u)du \approx p(f|y)$ where $u = u_{1:M}$ is a collection of inducing features with prior distribution $p(u)$. That is, we condition on u instead of y and average over an optimised distribution on u (Section 1.3.1 and Figure 1.3). The classic choice is inducing points, where $u_m = f(z_m) = \langle f, \delta_{z_m}\rangle$ for some $z_m \in \mathcal{X}$. We maximise a lower bound on the log marginal likelihood.

$$\mathcal{L}' = \int q(f) \log \frac{p(y, f)}{q(f)} df = \mathcal{L} - D_{KL}(q(f)||p(f|y)) \leq \mathcal{L} \tag{2.23}$$

2.2.5 More generally, u_m is chosen to be a linear functional $\varphi_m \in \mathcal{H}^*$ of f (Lázaro-Gredilla & Figueiras-Vidal, 2009), with $\langle f, \varphi_m\rangle \in \mathbb{C}$.

$$u_m = \langle f, \varphi_m\rangle \tag{2.24}$$

2.2.6 For features other than inducing points, these are termed *inter-domain* features. For convenience, define the inter-domain and inducing domain covariance functions, applying Equation (2.5), as

$$c_m(x) \stackrel{\text{def}}{=} \mathrm{Cov}[u_m, f(x)] = \langle k(\bullet, x), \varphi_m\rangle. \tag{2.25}$$

$$\bar{k}_{m,m'} \stackrel{\text{def}}{=} \mathrm{Cov}[u_m, u_{m'}] = \langle \mathcal{K}\varphi_{m'}, \varphi_m\rangle = \langle c^*_{m'}, \varphi_m\rangle \tag{2.26}$$

which give the entries of the covariance matrices K_{uf} and K_{uu}. With inducing points, $c = \bar{k} = k$. But it holds more generally that $p(u) = \mathcal{N}(0, \mathsf{K}_{uu})$.

2.2.7 **A natural class of approximations** Are linear functionals the right class of inducing values to consider? They lead to jointly Gaussian (u, f) regardless of the evaluation functionals ω, which in turn leads to a closed form posterior and training objective. We could still retain this with an *affine* transformation – adding a bias term to the linear transformation – but this would simply have to be subtracted when calculating the conditional mean of f given u and would not change any of the results. We could also add independent Gaussian noise to the transformation, but this could not make u a better summary of the data, as information would be lost in general. Hence, linear functionals are essentially the most general transformation of the function we could perform without breaking the joint Gaussianity of (u, f).

2.2.8 **The collapsed objective** Now the structured approximation to the data covariance matrix is

$$\mathsf{A}' = (\mathsf{K}_{uf}^* \mathsf{K}_{uu}^{-1} \mathsf{K}_{uf} + \sigma^2 \mathsf{I}) \tag{2.27}$$

but by exploiting standard linear algebra results (Appendix A.2), the inverse and log determinant can be isolated to K_{uu} and

$$\mathsf{B} = (\mathsf{K}_{uu} + \sigma^{-2} \underbrace{\mathsf{K}_{uf} \mathsf{K}_{uf}^*}_{\stackrel{\text{def}}{=} \mathsf{B}'}). \tag{2.28}$$

2.2.9 This is the covariance of $u + \mathsf{K}_{uf}\rho$ – the inducing features with independent additive noise created by reweighting the original additive noise by the inter-domain covariance. Also, let

$$\bar{y} = \mathsf{K}_{uf} y. \tag{2.29}$$

2.2.10 Then, in terms of this $M \times M$ matrix and length M vector, the closed form variational optimum is

$$q(u) \sim \mathcal{N}(\mu_u, \Sigma_u) \quad \text{with} \quad \begin{cases} \Sigma_u^{-1} = \mathsf{K}_{uu}^{-1} \mathsf{B} \mathsf{K}_{uu}^{-1}, \\ \mu_u = \sigma^{-2} \Sigma_u \mathsf{K}_{uu}^{-1} \bar{y} \end{cases} \tag{2.30}$$

with corresponding training objective (Lázaro-Gredilla & Figueiras-Vidal, 2009; Titsias, 2009)

$$\mathcal{L}'(\mu_u, \Sigma_u) = \log \mathcal{N}(y|0,\ \mathsf{A}') - \frac{1}{2}\sigma^{-2}\operatorname{tr}(\mathsf{A} - \mathsf{A}') \tag{2.31}$$

$$= -\frac{1}{2}\left[N \log 2\pi\sigma^2 + \log \frac{|\mathsf{B}|}{|\mathsf{K}_{uu}|} + \sigma^{-2}||y||_2^2 - \sigma^{-4}\bar{y}^\top \mathsf{B}^{-1}\bar{y}\right. \tag{2.32}$$

$$\left. + \sigma^{-2}\operatorname{tr}(\mathsf{K}_{\mathsf{ff}}) - \sigma^{-2}\operatorname{tr}(\mathsf{K}_{uu}^{-1}\mathsf{B}')\right]. \tag{2.33}$$

2.2.11 The posterior predictive at new points x_* is calculated as

$$q(f(x_*)|x_*, x, y) = \int p(f(x_*)|x, z, u)q(u)\,du$$

$$= \mathcal{N}(f(x_*)|\mathsf{K}_{u*}^*\mathsf{K}_{uu}^{-1}\mu_u,\ \mathsf{K}_{**} - \mathsf{K}_{u*}^*\mathsf{K}_{uu}^{-1}\mathsf{K}_{u*} + \mathsf{K}_{u*}^*\mathsf{K}_{uu}^{-1}\Sigma_u\mathsf{K}_{uu}^{-1}\mathsf{K}_{u*}^*). \tag{2.34}$$

and high quality posterior samples can still be efficiently generated by updating prior samples using samples of the inducing variables (Wilson et al, 2020). That is, there is no change to the structure of the equations for prediction and learning (from Equations (1.44), (1.46) and (1.47)); only the calculation of the elements of the covariance matrices changes.

2.2.12 **The reduced cost** The operations with cubic cost have been isolated to $M \times M$ matrices. If $M \ll N$ then the dominant cost is $\mathcal{O}(NM^2)$ to form $\mathsf{B}' = \mathsf{K}_{u\mathsf{f}}\mathsf{K}_{u\mathsf{f}}^*$ and $\bar{y} = \mathsf{K}_{u\mathsf{f}}y$. During iterative optimisation, these must usually be recalculated at each step, since the covariance function's parameters have been updated. This can only be avoided if $c_m(\bullet)$ does not depend on the covariance function's parameters.

2.2.13 Even if $c_m(\bullet)$ is not dependent on the covariance function's parameters, $\operatorname{tr}(\mathsf{K}_{\mathsf{ff}})$ incurs $\mathcal{O}(N)$ cost in general. But in the stationary setting, $\operatorname{tr}(\mathsf{K}_{\mathsf{ff}}) = N\sigma_f^2$. Finally, the term $||y||_2^2$ can always be precomputed and stored.

2.2.14 Hence, the goal is to construct a set of features which does not depend on the kernel parameters, to cut the computational cost for $\mathcal{O}(M^3)$.

2.2.15 It is also possible to relax the requirement a little further to features which depend only linearly on the kernel parameters. One way to view this is that linear

reparameterisation does not change either the training objective or the predictions, as formalised in the following straightforward result.

Proposition 2.7 (Linear reparameterisation of the variational approximation.) *Let* $\mathsf{T} \in \mathbb{C}^{M \times M}$ *be an invertible matrix, and* $u'_{1:M}$ *a sequence of inducing values generated using linear functionals* $\varphi'_{1:M}$*. Then* $u = \mathsf{T}u'$ *is an equivalent sequence of inducing values in the sense that the optimal variational distribution is unchanged, and at the optimal variational distribution, the value of the training objective and the posterior predictive distributions are unchanged.*

The corresponding linear functionals are given by

$$\varphi_m = \sum_{m'} \mathsf{T}_{mm'} \varphi'_{m'}.$$

PROOF Essentially, we just need to check each computation is unchanged.

Let u' have optimal mean and covariance $\mu_{u'}, \Sigma_{u'}$. But

$$\mathsf{K}_{u\mathsf{f}} = \mathbb{E}[uf(x)^\top] = \mathbb{E}[Tu'f(x)^\top] = \mathsf{T}\mathsf{K}_{u'\mathsf{f}} \tag{2.35}$$

and similarly $\mathsf{K}_{uu} = \mathsf{T}\mathsf{K}_{u'u'}\mathsf{T}^*$. Then from Equation (2.30) it follows that if we optimise after transforming, the optimal $\Sigma_u^{-1} = \mathsf{T}^{-*}\Sigma_{u'}^{-1}\mathsf{T}^{-1}$ and the optimal $\mu_u = \mathsf{T}\mu_{u'}$, as would be expected from optimising before transforming using the functionals as defined in the proposition. Furthermore, the collapsed objective of Equation (2.32) and posterior predictive mean and covariance of Equation (2.34) are left unchanged. □

Example 2.8 (Inducing points are precomputable for a linear covariance function.) For a first example, if we use a linear covariance function $k(x, x') = \theta||x - x'||$ with inducing points, B'/θ and $\bar{y}/\theta$ can indeed be precomputed and stored. This is equivalent to working with the transformed features $\varphi_m = \delta_{z_m}/\theta$ except at $\theta = 0$.

Example 2.9 (Precomputable approximation using the Mercer eigenfunctions.) For a second example, consider using Mercer eigenfunctions as in Paragraph 2.2.19. Then $c_m = \alpha_m \psi_m$. If each ψ_m does not depend on the parameters, then the equivalent precomputable features can be created by setting each $\varphi_m = \psi_m/\alpha_m$, yielding

$$c_m = \psi_m \tag{2.36}$$

$$\bar{k}_{m,m'} = \alpha_m^{-1}\delta_{m-m'}. \tag{2.37}$$

2.2.16 In summary, a computationally efficient set of features has two desiderata:

1. they comprise a *feature efficient* approximation:
   $\mathcal{L} - \mathcal{L}' = D_{KL}(q(f)||p(f|y)) \to 0$ quickly as M increases, so that we can use $M \ll N$ (Figure 1.3);

2. they provide a *precomputable* approximation – so that the $\mathcal{O}(NM^2)$ cost is paid only once, at the beginning of optimisation, with each optimisation step incurring only $\mathcal{O}(M^3)$ cost.

2.2.17 **Asymptotic feature efficiency** Consider the first requirement. Using features which are similar to measurement (as with inducing points in the standard setting) will require at most N features to take $D_{KL}(q(f)||p(f|y))$ to 0, but this offers no assurance that we can set $M \ll N$. We can consider the behaviour of KL as $M, N \to \infty$ in the setting where the data really was generated by Equation (2.16) and see what rate of M emerges. In particular, define

$$t = \operatorname{tr}\left(\mathsf{K}_{\mathsf{ff}} - \mathsf{K}_{u\mathsf{f}}^{*}\mathsf{K}_{uu}^{-1}\mathsf{K}_{u\mathsf{f}}\right) = \operatorname{tr}(\mathsf{A} - \mathsf{A}'). \tag{2.38}$$

Theorem 2.10 (Bounds in expectation on the approximation error.) *(Burt et al, 2020b, Lemma 4) If the data is sampled from Equation (2.16), then*

$$\frac{t}{2\sigma^2} \leq \mathbb{E}\left[\mathcal{L} - \mathcal{L}'\right] \leq \frac{t}{\sigma^2}.$$

Corollary 2.1 (Probabilistic bounds on the approximation error.) *If the data is sampled from Equation (2.16), for any probability level $c \in (0, 1)$,*

$$\mathbb{P}\left[\mathcal{L} - \mathcal{L}' \geq \frac{t}{c\sigma^2}\right] \leq c.$$

Proof Apply Markov's inequality.

$$\mathbb{P}[\mathcal{L} - \mathcal{L}' \geq a] \leq \frac{\mathbb{E}[\mathcal{L} - \mathcal{L}']}{a} \leq \frac{t}{a\sigma^2} = c \tag{2.39}$$

where the final equality is achieved by setting $a = t/c\sigma^2$. □

2.2.18 Then if $t \to 0$ with $M \in o(N)$ then the requirement is met in the average and worst cases. Generally t will grow linearly in N. I consider the two cases where it is bounded polynomially or exponentially.

Corollary 2.2 (Approximation error when t is polynomially bounded.) *If $t \in \mathcal{O}(NM^{-\kappa})$ for some $\kappa > 1$, then for any probability level $c_1 \in (0,1)$ and any $c_2 > 0$, for sufficiently large M, N with*

$$M \in \mathcal{O}(N^{\frac{1}{\kappa}})$$

we have

$$\mathbb{P}\left[\mathcal{L} - \mathcal{L}' \geq c_2\right] \leq c_1.$$

Corollary 2.3 (Approximation error when t is exponentially bounded.) *If $t \in \mathcal{O}(Ne^{-\kappa_1 M^{\kappa_2}})$ for some $\kappa_1 > 0, \kappa_2 > 0$, then for any probability level $c_1 \in (0,1)$ and any $c_2 > 0$, for sufficiently large M, N with*

$$M \in \mathcal{O}\left(\log^{\frac{1}{\kappa_2}} N\right)$$

we have

$$\mathbb{P}\left[\mathcal{L} - \mathcal{L}' \geq c_2\right] \leq c_1.$$

Proof *(of Corollaries 2.2 and 2.3)* From Corollary 2.1, it follows that

$$\mathbb{P}\left[\mathcal{L} - \mathcal{L}' \geq \frac{t}{c_1 \sigma^2}\right] \leq c_1. \tag{2.40}$$

But in Corollary 2.2, for sufficiently large M, N, $t \leq aNM^{-\kappa}$ by assumption. Then, if M is sufficiently large that

$$c_2 \geq \frac{aNM^{-\kappa}}{c_1 \sigma^2} \geq \frac{t}{c_1 \sigma^2} \tag{2.41}$$

then it follows that

$$\mathbb{P}\left[\mathcal{L} - \mathcal{L}' \geq c_2\right] \leq c_1. \tag{2.42}$$

M sufficiently large means for any fixed N

$$M \geq \left(\frac{aN}{c_1 c_2 \sigma^2}\right)^{\frac{1}{\kappa}} \tag{2.43}$$

so it suffices to set M equal to the right hand side rounded up, for example. Then indeed, $M \in \mathcal{O}(N^{\frac{1}{\kappa}})$.

The exponential case follows similarly, except now sufficiently large means

$$M \geq \kappa_1 \log \frac{a}{c_1 c_2 \sigma^2} + \kappa_1 \log^{\frac{1}{\kappa_2}} N \tag{2.44}$$

from which the result follows. □

2.2.19 **Example application to eigenfunction features** For example, this can be used to make the relationship between the decay rate of the operator eigenvalues and the convergence rate more precise. If we select $\varphi_m = \psi_m$, taking inner products in $L^2(\mathcal{X}, \nu)$, assume k is stationary, and $x_n \overset{\text{IID}}{\sim} \nu$ then it follows that

$$\mathbb{E}_x[t] = \mathbb{E}_x \left[\sum_n \left[k(x_n, x_n) - \Sigma_{m=1}^M \alpha_m |\psi_m(x_n)|^2 \right] \right] \tag{2.45}$$

$$= \sum_n \sum_{m=M+1}^{\infty} \alpha_m \mathbb{E}_x[|\psi_m(x_n)|^2] \tag{2.46}$$

$$= N \sum_{m=M+1}^{\infty} \alpha_m. \tag{2.47}$$

2.2.20 Then if the rate of the decay of the eigenvalues is known, this can be combined with Corollaries 2.2 and 2.3 to bound the rate of convergence. This can be generalised to a different input density to ν under mild conditions (Burt, 2022, Chapter 3, Assumption 2).

2.2.21 **The empirical eigenvectors** But the eigenfunctions are difficult to compute in general. Closely related to the eigenfunctions are the eigenvectors of the actual covariance matrix. Theorem 2.11 shows that using the first M eigenvectors of K_{ff} as φ will yield a similar result.

Theorem 2.11 (Eigenvalues of the covariance matrix.) *(Shawe-Taylor et al, 2005, Burt, 2022, Lemma 3.18) Let $\lambda_m(\mathsf{K})$ denote the mth largest eigenvalue of the matrix K. Let ν be the input probability measure of Equation (2.8) and $\alpha_{1:\infty}$ the corresponding eigenvalues, and let each $x_n \sim \nu$ IID.*

$$\frac{1}{N} \mathbb{E}_x \left[\sum_{m=M+1}^{\infty} \lambda_m(\mathsf{K}_{\mathsf{ff}}) \right] \leq \sum_{m=M+1}^{\infty} \alpha_m$$

2.2.22 **Inducing points** However, calculating the partial eigendecomposition of the covariance matrix is prohibitively expensive ($\Omega(N^2M)$). But Burt et al (2020b) showed that, under mild technical conditions, it is possible to efficiently select a subset of the data points as inducing points while yielding a similarly good approximation to the partial eigendecomposition. Concretely, this leads to convergence with $M \in \mathcal{O}(\log^D N)$ for the squared exponential covariance function, and with $M \in \mathcal{O}(N^{D/2\nu-D})$ for Matérn-ν covariance functions in low dimensions ($D < 2\nu$).

2.2.23 **On the nature of these bounds** Before continuing with the main developments of this chapter, it is worth remarking on what sort of properties are being described here. The feature efficiency bounds are merely upper bounds, which may be loose. Also, these bounds are on average over the prior, but we may not be convinced that the data was really generated from the prior. A posteriori bounds of a similar nature, and lower bounds on the number of features needed, were explored by Burt (2022). Here, I show via upper bounds that candidate precomputable methods are comparable in feature efficiency to inducing points, which are a well-established and competitive approximation. This analysis does not make strong guarantees about predictive performance on any specific dataset, but shows which approximation strategies reduces the computational cost of learning.

2.2.24 Using inducing points, or more generally using a subset of the canonical features, is appealing from a feature efficiency perspective. However, they do not in general satisfy precomputability. For example, with inducing points, $k(\bullet, z_m)$ clearly depends directly on the parameters. The remainder of the this chapter focuses on precomputable features.

2.3 Biorthogonal features

2.3.0.1 In this section, I provide a slightly different perspective on sparse variational approximations in terms of the feature functions $k(\bullet, x_n)$ and c_m^*.

2.3.0.2 Consider sets of features for which $\bar{k}_{m,m'} = \delta_{m-m'}$ – that is, the features are independent a priori. This also assumes that the features have unit variance, without loss of generality (if the features had any other positive variance, we could linearly

rescale the features by Proposition 2.7). As discussed in the preceding section, this is not critical for improved performance, but it significantly simplifies the analysis. Now consider the case where $\mathcal{H}$ is equipped with an inner product.

Theorem 2.12 (Characterising biorthogonal features.) *For any sequence of linear functionals $\varphi_{1:\infty}$ for which the corresponding inducing features (Equation (2.24)) are independent a priori with unit variance, the linear functionals and the inter-domain covariance functions c_m^* form a biorthogonal pair, in the sense that*

$$\langle c_m^*, \varphi_m' \rangle = \delta_{m-m'}. \tag{2.48}$$

Moreover, for a finite sequence of features $m \in \{1{:}M\}$, if the data is sampled from Equation (2.16), then the expected KL divergence between the approximate and exact posterior is bounded above and below by terms proportional to the residual from projecting the canonical features $k(\bullet, x_n)$ onto $\operatorname{span}\{c_m^*\}$*, evaluated at the data points x_n.*

PROOF The first part follows immediately from the definition of $\bar{k}_{m,m'}$.

For the second part, consider projection using a biorthogonal pair of bases.

$$\hat{k}(\bullet, x) = \sum_m \langle k(\bullet, x), \varphi_m \rangle c_m(\bullet) \tag{2.49}$$

$$= \sum_m c_m(x) c_m^*(\bullet) \tag{2.50}$$

If $\hat{k}(\bullet, x)$ is in the closure of $\operatorname{span}\{c_m^*\}$ then $\hat{k}(\bullet, x) = k(\bullet, x)$, and otherwise the residual $\hat{k}(\bullet, x) - k(\bullet, x)$ is orthogonal to $\operatorname{span}\{c_m^*\}$.

Now consider the particular point $k(x, x)$.

$$\hat{k}(x, x) = \sum_m |c_m(x)|^2 \tag{2.51}$$

Since each $|c_m(x)|^2 \geq 0$, the sum converges monotonically.

Then the trace term is

$$t = \operatorname{tr}(\mathsf{K}_{\mathsf{ff}} - \mathsf{K}_{\mathsf{uf}}^* \mathsf{K}_{uu}^{-1} \mathsf{K}_{\mathsf{uf}}) \tag{2.52}$$

$$= \sum_n [k(x_n, x_n) - \sum_m |c_m(x_n)|^2] \tag{2.53}$$

from which the claim follows by applying Theorem 2.10. □

2.3.0.3 This result is just to provide perspective: independent variational features replace the canonical features $k(\bullet, x_n)$ with the alternative features $c_m(\bullet)$. With inducing points placed at training inputs, this is a subset of the same features.

2.3.0.4 For any finite collection of non-independent features with linear functionals $\varphi_{1:M}$, we can construct biorthogonal features by replacing K_{uf} with $\mathsf{K}_{uu}^{-1/2}\mathsf{K}_{uf}$, which is equivalent to constructing new linear functionals $\tilde{\varphi}_{1:M}$, each a linear combination of the old ones. This does not change the span of the corresponding feature functions, so the above perspective remains valid.

Theorem 2.13 (The minimum number of features.) *For $M = N$, there exists a set of inducing functionals $\varphi_{1:M}$ such that the variational approximation is equal to the exact posterior. But if k is of infinite rank, then for any finite $M < N$, the expected KL divergence from the approximate to exact posterior is positive.*

PROOF The first part is a straightforward generalisation of the case for inducing points. Let $\varphi_m = \omega_m$. Then the trace $t = 0$, and the variational approximation is equivalent to the exact formulation.

For the second part, suppose there existed a sequence $\varphi_{1:M}$ with $M < N$ such that $t = 0$. Then it follows from Theorem 2.12 that $\langle k(\bullet, x_n), \omega_n \rangle \in \operatorname{span}\{\varphi_{1:M}\}$ for all $n \in \{1{:}N\}$. But then $\operatorname{rank}(\mathsf{K}_{ff}) = M$, which contradicts the assumption that each ω_n is distinct and k is infinite rank. □

2.3.0.5 This statement only describes when it is possible (or not) to make the approximation exact, and the result that sparse approximations really do result in an approximation is not at all surprising. To comment on feature efficiency, we need to analyse how quickly M needs to grow with N as $N \to \infty$ and $t \to 0$.

2.3.0.6 For precomputability, we seek a biorthogonal decomposition

$$c_m^* = \langle k(\bullet, x), \varphi_m \rangle \tag{2.54}$$

for which the evaluations of c_m^* do not depend more than linearly on the parameters. But for feature efficiency we require that the span of c_m^* grows quickly to approach the canonical features $k(\bullet, x_n)$.

2.3.1 Composition of features

2.3.1.1 Often, it is challenging to design the features or analyse their convergence when $D > 1$. If a higher dimensional covariance function is constructed from the composition of one dimensional covariance functions, then we can extend the features in the same way without compromising precomputability, and correspondingly extend the feature efficiency results.

2.3.1.2 **Additive models** Let the model be the sum of independent effects. That is,

$$f = \sum_{a=1}^{A} f_a \tag{2.55}$$

$$k = \sum_{a=1}^{D} k_a \tag{2.56}$$

with a particularly important case being where $A = D$, and each GP is defined on a one dimensional domain (Duvenaud et al, 2011).

Proposition 2.14 (Composition of bounds for additive covariance functions) *Let f be an additive covariance function with A terms. Let $\varphi_{1:\infty}^{(a)}$ be feature functionals for each $a \in \{1 : A\}$. Suppose that for data samples from any f_d with additive white Gaussian noise of variance σ^2, the trace of the variational approximation is bounded according to*

$$t_a \in \mathcal{O}(NM^{-\kappa_a}).$$

Let $\kappa = \min_a \kappa_a$. Then there exists a sequence of feature functionals $\varphi_{1:\infty}$ which produce a variational approximation for which

$$t \in \mathcal{O}(NA^{1+\kappa}M^{-\kappa}).$$

Similarly, if each $\varphi_{1:\infty}^{(a)}$ yields approximations with

$$t_a \in \mathcal{O}(e^{-\kappa_{1,a}M^{\kappa_{2,a}}})$$

and let $a' = \arg\min_a \kappa_{2,a}$, $\kappa_2 = \kappa_{2,a'}$, $\kappa_1 = \kappa_{1,a'}$. Then there exists a sequence of feature functionals $\varphi_{1:\infty}$ which produce a variational approximation for which

$$t \in \mathcal{O}\left(Ae^{-\frac{\kappa_1}{A^{\kappa_2}}M^{\kappa_2}}\right).$$

In particular,

$$\varphi_{\ell d+m} = \varphi_m^{(a)} \qquad \ell, m \in \{1:\infty\}, a \in \{1:A\}$$

suffices.

PROOF Consider the construction in the last part of the statement. Since $f_{1:A}$ are mutually independent, for a finite number of features $M = AM'$, the covariance matrices $\mathsf{K}_{\mathsf{ff}}, \mathsf{K}_{uf}$ and K_{uu} are compatibly block diagonal, and

$$t = \sum_a t_a \leq A \max_a t_a \in \mathcal{O}(ANM'^{-\kappa}) \tag{2.57}$$

but $M'^{-\kappa} = A^\kappa M^{-\kappa}$, which yields the result. The exponential case follows by a similar argument. □

2.3.1.3 **Product models** An alternative construction is a product over dimensions (Saatçi, 2011).

$$k = \prod_{p=1}^{P} k_p \qquad k_p : \mathcal{X}_p \times \mathcal{X}_p \to \mathbb{R} \tag{2.58}$$

$$\mathcal{X} = \bigoplus_p \mathcal{X}_p \tag{2.59}$$

2.3.1.4 The canonical example is where each $\mathcal{X}_p$ is one dimensional, and $P = D$. It is helpful to define

$$f_p \sim \mathcal{GP}(0, k_p). \tag{2.60}$$

Proposition 2.15 (Composition of bounds for tensor products of covariance functions) *Let f be a GP whose covariance function is a product over dimensions with P terms. Let $\varphi_{1:\infty}^{(p)}$ be feature functionals for each $p \in \{1:P\}$ such that $\varphi_{1:\infty}^{(p)}, c_{1:\infty}^{(p)}$ are biorthogonal. Suppose that for data samples from any f_d with additive white Gaussian noise of variance σ^2, the trace of the variational approximation is bounded according to*

$$t_p \in \mathcal{O}(NM^{-\kappa_p}).$$

Let $\kappa = \min_p \kappa_p$. Then there exists a sequence of feature functionals $\varphi_{1:\infty}$ which produce a variational approximation for which

$$t \in \mathcal{O}(NPM^{-\frac{\kappa}{P}}).$$

Similarly, if each $\varphi_{1:\infty}^{(p)}$ *yields approximations with*

$$t_p \in \mathcal{O}(e^{-\kappa_{1,p}M^{\kappa_{2,p}}})$$

and let $p' = \arg\min_a \kappa_{2,p}$, $\kappa_2 = \kappa_{2,p'}$, $\kappa_1 = \kappa_{1,p'}$. *Then there exists a sequence of feature functionals* $\varphi_{1:\infty}$ *which produce a variational approximation for which*

$$t \in \mathcal{O}\left(Pe^{-\kappa_1 M^{\frac{\kappa_2}{P}}}\right).$$

In particular,

$$\varphi_{m_{1:P}} = \bigotimes_p \varphi_{m_p}^{(p)} \quad m_p \in \{1:\infty\}, p \in \{1:P\}$$

suffices.

PROOF Consider the construction in last part of the statement. For a finite number of features $M = M'^P$, let the covariance matrices $\mathsf{K}_{\mathsf{ff}}^{(p)}, \mathsf{K}_{\mathsf{uf}}^{(p)}$ and $\mathsf{K}_{uu}^{(p)}$ be the covariance matrices for f_p using features $u_m^{(p)} = \langle f_p, \varphi_m^{(p)} \rangle$. Then it follows that

$$\mathsf{K}_{\mathsf{ff}} = \bigotimes_p \mathsf{K}_{\mathsf{ff}}^{(p)} \tag{2.61}$$

$$\mathsf{K}_{uu} = \bigotimes_p \mathsf{K}_{uu}^{(p)} = \bigotimes_p \mathsf{I} = \mathsf{I} \tag{2.62}$$

$$\mathsf{K}_{u\mathsf{f}} = *_p \mathsf{K}_{\mathsf{uf}}^{(p)} \tag{2.63}$$

where $\bigotimes$ represents the Kronecker product, and $*$ represents the Khatri-Rao product (column-wise Kronecker product).

$$\begin{aligned}
t &= \sum_n \left(\prod_p k_p(x_n^{(p)}, x_n^{(p)}) - \sum_m |c_m(x_n)|^2 \right) \\
&= \sum_n \left(\prod_p k_p(x_n^{(p)}, x_n^{(p)}) - \sum_m \prod_p |c_{m_p}(x_n^{(p)})|^2 \right) \\
&= \sum_n \left(\prod_p k_p(x_n^{(p)}, x_n^{(p)}) - \prod_p \sum_{m_p} |c_{m_p}(x_n^{(p)})|^2 \right) \\
&\leq \sum_n \sum_p \left(k_p(x_n^{(p)}, x_n^{(p)}) - \sum_{m_p} |c_{m_p}(x_n^{(p)})|^2 \right) \\
&= \Sigma_p t_p \\
&\leq P \max_p t_p
\end{aligned}$$

Note $M' = M^{\frac{1}{P}}$, and then the results follow by applying a similar argument to Proposition 2.14. □

2.3.1.5 Similarly, results are often available for simple covariance functions, but the covariance functions used in practice are sums or products of these. While the sum case followed immediately in Proposition 2.14, the general product case requires a little more care. Let

$$\tilde{X} = \mathcal{X}^P \tag{2.64}$$
$$\tilde{f}_p \sim \mathcal{GP}(0, k_p) \qquad \text{for } p \in \{1{:}P\} \text{ independently} \tag{2.65}$$
$$f(x) = \tilde{f}(x, x, ..., x). \tag{2.66}$$

Then Proposition 2.15 can be applied.

2.3.1.6 Note that these bounds require using a very specific construction which is likely to be suboptimal. If the same sequence of functionals $\varphi_{1:M}$ would work well for each component individually, it is likely that using them for the composed covariance function will also work well. However, these composition rules can be useful where different sets of features are efficient or precomputable for each component covariance function.

2.3.1.7 Two families of precomputable methods have received attention in previous work. The first is harmonic approximations, which lead to precomputable biorthogonal features on certain manifolds. This is the focus of the next section, where I generalise this from the sphere (Dutordoir et al, 2020), and provide a convergence result.

2.3.1.8 The second is the closely related idea of using RKHS inner products (Paragraphs 2.1.23, 2.1.24 and 2.1.25), which I briefly review here. Formally define

$$u_m = \langle \varphi_m, f \rangle_{\mathcal{H}_k} \tag{2.67}$$

though the samples of f are not in $\mathcal{H}_k$. But then the reproducing property (Equation (2.13)) guarantees precomputability, as long as the functions φ_m do not depend on the parameters.

$$c_m = \varphi_m \tag{2.68}$$

$$\bar{k}_{m,m'} = \langle \varphi_m, \varphi_{m'} \rangle_{\mathcal{H}_k} \tag{2.69}$$

2.3.1.9 For the purposes of designing precomputable features, this is convenient, since the inner product function φ_m is also the feature function c_m. But $\bar{k}_{m,m'}$ requires calculating explicit RKHS inner products, which is not straightforward in general.

2.3.1.10 One special case is one dimensional Matérn covariance functions on the bounded interval $[a, b]$, for which the RKHS has a convenient explicit characterisation. This can be extended to higher dimensional settings for sums of products (over dimensions) of Matérn covariance functions, and to sums and products of Matérn covariance functions in one dimension, using the composition constructions of Section 2.3.1.

2.3.1.11 Hensman et al (2017) used a Fourier basis on $[a, b]$. They refer to this as variational Fourier features (VFF). I refer to this as RKHS Fourier series (RKHS-FS) features.

2.3.1.12 Cunningham et al (2023) used a B-spline basis. This was chosen since the compact support of the basis functions leads to a banded form for B'.

2.3.1.13 Although these methods provide convenient precomputable methods, they have two significant drawbacks. Firstly, they extend to higher dimensions only with exponentially increasing cost unless an additive structure can be assumed. This limits their applicability to low dimensions, say $D \lesssim 4$. This may not be a severe limitation for the motivating spatial modelling applications.

2.3.1.14 Secondly, they do not extend readily to a broader class of covariance functions, which is much more problematic.

2.4 Harmonic approximations

2.4.0.1 In this section, I consider existing precomputable approximations for stationary covariance functions based on Fourier features, including some generalisations.

2.4.0.2 Recall from Theorem 2.4 and Paragraph 2.1.16 that a stationary covariance function with well-defined spectral density s can be viewed as having a continuous eigenbasis

$$\left(e^{i2\pi\xi^\top x}, s(2\pi\xi)\right) \quad \xi \in \mathbb{R}^D. \tag{2.70}$$

and from Example 2.9 that we can create precomputable and biorthogonal features by setting φ_m to the eigenfunctions normalised to the eigenvalues, or in this setting

$$\varphi_\xi = \frac{e^{i2\pi\xi^\top x}}{s(2\pi\xi)}. \tag{2.71}$$

2.4.0.3 But the continuous basis causes issues. A first way of viewing this is that the variance of the features is unbounded (Lázaro-Gredilla & Figueiras-Vidal, 2009; Lifshits, 2012, Chapter 3), which can be shown as follows, using $\delta(\bullet)$ for the Dirac delta.

$$c(x', \xi) = \int k(x', x) \frac{e^{-2\pi\xi^\top x}}{s(\xi)}\, dx = e^{-i2\pi\xi^\top x'} \tag{2.72}$$

$$\bar{k}(\xi, \xi') = \int c(x', \xi) \frac{e^{-2\pi\xi'^\top x'}}{s(\xi')}\, dx' = \int \frac{e^{-2\pi x'^\top(\xi - \xi')}}{s(\xi')}\, dx' = \frac{\delta(\xi - \xi')}{s(\xi)} \tag{2.73}$$

2.4.0.4 If we select any finite subset of points $z_{1:M}$ then the conditional prior $p(f(x)|u)$, and hence the posterior, will revert to the prior. Informally, the prior feature precision K_{uu}^{-1} vanishes but K_{uf}^* is finite, so the product vanishes (Equation (2.34); $\Sigma_u = 0$, $\mu_u = \sigma^{-2}\bar{y}$ from Equation (2.30)).

2.4.0.5 A second view is that u is M points sampled from the Fourier transform of f, say $\tilde{f} = \mathcal{F}f$ where $\mathcal{F}$ is the Fourier transform operator. Then

$$p(f|u) = \mathbb{E}_{\tilde{f}}[p(f|\tilde{f}, u)|u]. \tag{2.74}$$

2.4.0.6 Now recall from Paragraph 2.1.17 that $\tilde{f}$ is a Gaussian white noise process whose control measure is the spectral measure. Then $\tilde{f}(2\pi\xi)$ for $\xi \notin \{z_{1:M}\}$ is not affected by conditioning on u. So, for the purposes of the distribution inside the expectation,

$$f = \mathcal{F}^{-1}\tilde{f}' \tag{2.75}$$

$$\tilde{f}'(2\pi\xi) = \begin{cases} u_m & \xi = 2\pi z_m \\ \tilde{f}(2\pi\xi) & \text{otherwise} \end{cases} \tag{2.76}$$

but changing the value of the function at isolated points (that is, on sets of zero measure) does not change the inverse Fourier transform, so $p(f|\tilde{f}, u) = p(f|\tilde{f})$. Moreover, $\tilde{f}$ has not been changed in distribution except on $\bigcup_m z_m$ which has zero spectral measure. Hence we recover

$$p(f|u) = p(f). \tag{2.77}$$

2.4.0.7 This deficiency is illustrated in Figure 2.1b.

2.4.0.8 Modifications to Fourier features include applying a Gaussian window (Lázaro-Gredilla & Figueiras-Vidal, 2009) which gives finite variance but loses biorthogonality and precomputability, and variational orthogonal features (VOF), where

$$\langle f, \varphi_m \rangle = \int \int \frac{e^{i2\pi\xi^\top x}}{\sqrt{s(\xi)}} \tilde{\varphi}_m(\xi)\, d\xi f(x)\, dx \tag{2.78}$$

for pairwise orthogonal $\tilde{\varphi}_m$. This approach yields biorthogonal features, but it is challenging to find sets of orthogonal functions which have desirable properties, such as retaining precomputability and easily calculable c_m (Burt et al, 2020a).

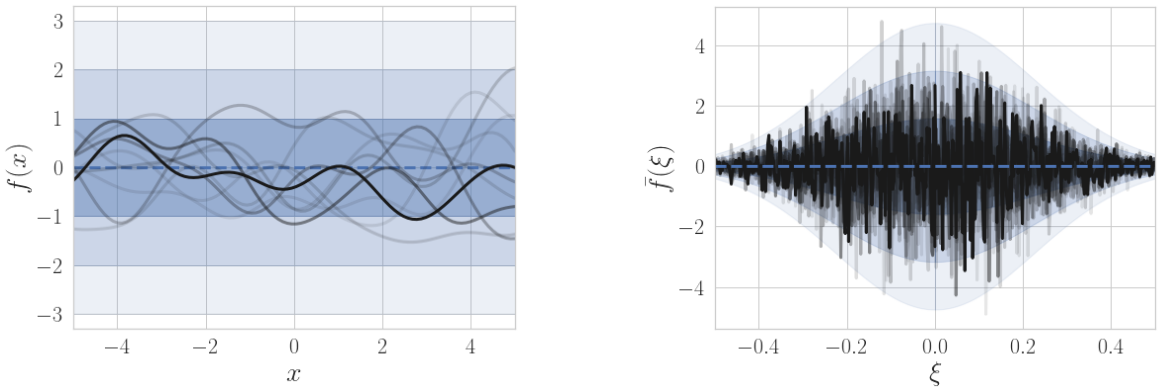
2.4.1 Compact homogeneous Riemannian manifolds

2.4.1.1 Dutordoir et al (2020) used spherical harmonic features for rotationally invariant covariance functions on the sphere, which they applied to $\mathbb{R}^D$ by mapping the data onto a hemisphere. In this section, motivated by that work, I generalise the setting to a broader class of stationary covariance functions, and provide a convergence result for covariance functions constructed using Theorem 2.5.

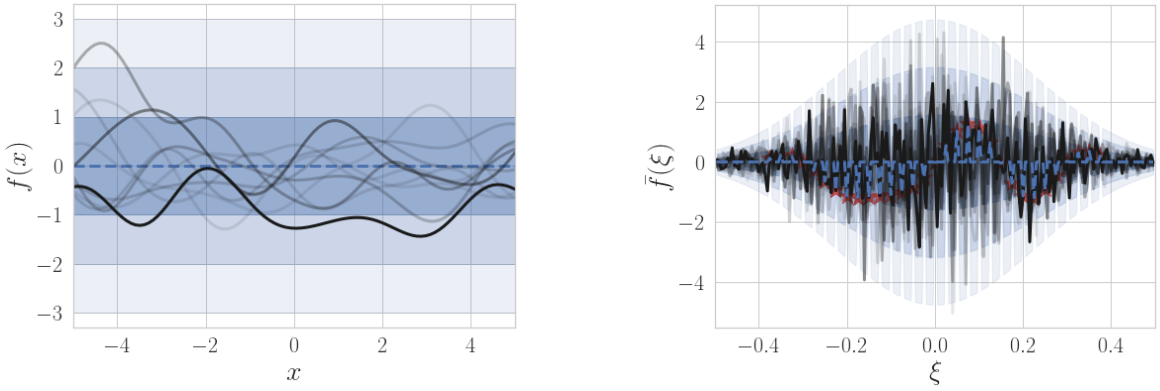
2.4.1.2 Firstly, in order to make progress, we need a more general notion of stationarity, which includes the translation invariance on $\mathbb{R}^D$ discussed in the preceding section and the rotational invariance on the sphere mentioned above.

2.4.1.3 Let $h : \mathcal{X} \to \mathcal{X}$ be a diffeomorphism. Then we could say k is stationary with respect to H if for any $h \in H$ and for all $x, x' \in \mathcal{X}$,

$$k(h(x), h(x')) = k(x, x'). \tag{2.79}$$



(a) Prior



(b) Conditioning on Fourier features

Figure 2.1 The ineffectiveness of conditioning on Fourier features. I plot the mean function (dashed), between one and three standard deviations (shaded) and sample functions in both the data domain (left column) and frequency domain (right column) for a squared exponential kernel with unit lengthscale. The sample functions in the data and frequency domains correspond to one another. (a) The prior's Fourier transform is white Gaussian noise whose variance is given by the spectral density. (b) We cannot condition meaningfully on some finite collection of frequencies (red stars), as this gives no information about the other frequencies – the conditional prior $p(f|u)$ in the data domain is unchanged.

2.4.1.4 There is always a nonempty set H which k is stationary with respect to, since the identity mapping is always included. Since h is a diffeomorphism, h^{-1} always exists, and it is clear that $h^{-1} \in H$ whenever h is since

$$k(h^{-1}(x), h^{-1}(x')) = k(h(h^{-1}(x)), h(h^{-1}(x))) = k(x, x') \tag{2.80}$$

where the first equality follows since $h \in H$ and the second since hh^{-1} is identity. Hence, $(H, \circ)$ is always a group, where $\circ$ is function composition.

2.4.1.5 Since there is always an H which satisfies this, it is clear this is not a satisfactory definition for stationarity. Any covariance function would be stationary to a group containing only the identity mapping, for example. To capture the notion of stationarity common to translation invariant kernels on $\mathbb{R}^D$ and rotationally invariant kernels on the sphere, I use the following definition.

Definition 2.16 (Stationarity.) Let $(H, \circ)$ be a group which acts transitively on $\mathcal{X}$ – that is, for any $x, x' \in \mathcal{X}$ there is an $h \in H$ such that $h(x) = x'$. Then k is a proper stationary covariance function with respect to H (or an H-stationary covariance function) if k is stationary with respect to H.

2.4.1.6 For classical stationary covariance functions on $\mathbb{R}^D$, the group is translations. On the sphere, the group is rotations about its centre. Since H is a group of diffeomorphisms acting transitively on $\mathcal{X}$, it is (by definition) an isometry on $\mathcal{X}$.

2.4.1.7 An immediate consequence of this definition is that for a proper stationary covariance function, the marginal variance $k(x, x)$ is constant. Let

$$\sigma_f^2 \stackrel{\text{def}}{=} k(x, x). \tag{2.81}$$

2.4.1.8 Now, consider the distance parameterisation of the kernel

$$k(x, x') = k(d(x, x')) \tag{2.82}$$

where d is some distance function. Such a distance function always exists, as it can be constructed from the kernel (exploiting the fact that k is an inner product).

$$d_{\text{ind}}^2(x, x') = k(x, x) + k(x', x') - 2k(x, x') = 2(\sigma_f^2 - k(x, x')) \tag{2.83}$$

2.4.1.9 Using the induced distance, $k(d_{\text{ind}})$ has the explicit form

$$k(d_{\text{ind}}) = \sigma_f^2 - \frac{1}{2} d_{\text{ind}}^2. \tag{2.84}$$

2.4.1.10 For what follows, we will need the following well-known theorem.

Theorem 2.17 (Isometries commute with the Laplacian.) *(Judge, 2022) Let $(\mathcal{X}, d)$ be Riemannian manifold, and ∇^2 the corresponding Laplace-Beltrami operator. Then a diffeomorphism h is an isometry iff*

$$\nabla^2(g \circ h) = (\nabla^2 g) \circ h$$

for every $g \in C^\infty(\mathcal{X})$.

2.4.1.11 Next, I use this result to characterise the eigenbasis of stationary covariance functions on compact Riemannian manifolds.

Theorem 2.18 (Stationary covariance functions commute with the Laplacian.) *Let k be an H-stationary covariance function on $\mathcal{X}$. Let $(\mathcal{X}, d)$ be a Riemannian manifold without boundary, or whose boundary has zero measure, with d such that $k(x, x') = k(d(x, x'))$, and let ∇^2 be the corresponding Laplace-Beltrami operator. Then the corresponding covariance operator commutes with ∇^2. In particular, they share the same eigenbasis.*

Proof Commuting operators share the same eigenbasis (see Appendix B). It remains to show the operators commute. Firstly, suppose

$$[\nabla^2 k(\bullet, x')]|_x = \nabla^2 [k(x, \bullet)]_{x'}. \tag{2.85}$$

Then the covariance operator commutes with ∇^2, which is shown as follows, using an arbitrary function g, with $d\Omega(x)$ being the volume form of the manifold, $d\Gamma(x)$ the volume form of its (nonexistent) boundary $\partial\mathcal{X}$, with n the unit surface normal vector. I use the standard notation $\nabla\cdot$ for the divergence operator, so $\nabla^2 = \nabla \cdot \nabla$.

$$\mathcal{K}[\nabla^2 g](x') = \int_{\mathcal{X}} k(x, x') \nabla^2 g(x) \, d\Omega(x) \tag{2.86}$$

$$= \int_{\mathcal{X}} k(x, x') \nabla \cdot \nabla g(x) \, d\Omega(x) \tag{2.87}$$

$$= \cancelto{0 \because \text{no boundary}}{\int_{\partial\mathcal{X}} k(x,x')\nabla g(x) \cdot \mathsf{n}\, d\Gamma(x)} - \int_{\mathcal{X}} [\nabla k(\bullet, x')]|_x \cdot \nabla g(x)\, d\Omega(x) \tag{2.88}$$

$$= -\left[\cancelto{0 \because \text{no boundary}}{\int_{\partial\mathcal{X}} g(x)[\nabla k(\bullet, x')]|_x \cdot \mathsf{n}\, d\Gamma(x)} - \int_{\mathcal{X}} g(x)[\nabla^2 k(\bullet, x')]_x\, d\Omega(x)\right] \tag{2.89}$$

$$= \int_{\mathcal{X}} g(x)[\nabla^2 k(\bullet, x')]_x\, d\Omega(x) \tag{2.90}$$

$$= \int_{\mathcal{X}} g(x)\nabla^2 [k(x, \bullet)]_{x'}\, d\Omega(x) \tag{2.91}$$

$$= \left[\nabla^2 \int_{\mathcal{X}} g(x)k(x, \bullet)\, d\Omega(x)\right](x') \tag{2.92}$$

$$= \nabla^2[\mathcal{K}g](x') \tag{2.93}$$

The integrals over the boundary also vanish under the milder condition that the boundary has zero measure.

Next, I show that the condition in Equation (2.85) is a consequence of k being H-stationary for some H. Firstly, exploit the symmetry of the covariance function to write Equation (2.85) as

$$[\nabla^2 k(\bullet, x')]|_x = \nabla^2 [k(\bullet, x)]_{x'}. \tag{2.94}$$

Then we have $k(\bullet, x') = k(\bullet, h(x')) \circ h$ for any $h \in H$ by stationarity. But by symmetry of k, we also have

$$k(\bullet, x')|_x = k(\bullet, h^{-1}_{x \to x'}(x')) \circ h_{x \to x'} \tag{2.95}$$

where $h_{x \to x'}$ is the isometry which maps x to x', which must be included in H since it acts transitively on $\mathcal{X}$. Then

$$[\nabla^2 k(\bullet, x')]|_x = \nabla^2 [k(\bullet, h^{-1}_{x \to x'}(x')) \circ h_{x \to x'}](x) \tag{2.96}$$

$$= [(\nabla^2 k(\bullet, h^{-1}_{x \to x'}(x'))) \circ h_{x \to x'}](x) \tag{2.97}$$

$$= [\nabla^2 k(\bullet, x)]|_{x'} \tag{2.98}$$

as required.

□

2.4.1.12 However, the harmonic expansion of k on the domain must be known. The eigenfunctions are only available in closed form in a limited number of cases (such as the torus), though there are well established numerical routines for approximating them in general. A similar statement holds for the eigenvalues, but for *isotropic* kernels on $\mathbb{R}^D$ restricted to the manifold, these can be computed from the spectral density using Theorem 2.5. For this setting, I show that the convergence rate appears to be comparable to inducing points.

Theorem 2.19 (Trace bound for restrictions of isotropic covariance functions.) *Let k be an H-stationary covariance function on $\mathcal{X} \subset \mathbb{R}^{D'}$ with distance d such that $k(x, x') = k(d(x, x'))$ and $(\mathcal{X}, d)$ is a Riemannian manifold without boundary, and let k also be constructed as the restriction of an isotropic covariance function on $\mathbb{R}^{D'}$ to $\mathcal{X}$. Let the Dirichlet eigenfunctions of the Laplacian on $\mathcal{X}$ be given by $\psi_{1:\infty}$. Let the spectral density of the isotropic covariance function be s. Then for inducing features constructed using $\varphi_m = \psi_m$, if $s(\xi) \in \mathcal{O}(\xi^{-\kappa})$ for some $\kappa > 0$*

$$t \in \mathcal{O}\left(\frac{D'}{\kappa - D'} N M^{-\frac{\kappa - D'}{D'}}\right).$$

If $s(\xi) \in \mathcal{O}(e^{-\xi})$, then

$$t \in \mathcal{O}(D' N e^{-M^{\frac{1}{D'}}}).$$

2.4.1.13 Note that the dimensionality D' is in general larger than that of the manifold. For example, with a sphere, $D' = D + 1$. For the case of polynomially decaying spectral density, we require the decay to be faster in higher dimensions ($\kappa > D'$) for the bound to decay with M.

Example 2.20 (Convergence for restricted isotropic Matérn covariance functions.) If k is a Matérn-ν covariance function restricted to $\mathcal{X}$, then the spectral density is polynomially bounded with $\kappa = 2\nu + D'$, yielding

$$t \in \mathcal{O}(DNM^{-2\nu/D'}).$$

In particular, applying Corollary 2.2 yields convergence with

$$M \in \mathcal{O}((D'N)^{\frac{D'}{2\nu}}).$$

Example 2.21 (Convergence for restricted squared exponential covariance function.) If k is a squared exponential covariance function restricted to $\mathcal{X}$, then the spectral density is exponentially bounded. Then appling Corollary 2.3 yields convergence with

$$M \in \mathcal{O}(\log^{D'} N)$$

assuming $D' \ll N$.

2.4.1.14 In comparison to the inducing point bounds (Paragraph 2.2.22), the bound for the squared exponential covariance function is the same, but the bound for Matérn covariance functions is more favourable here. It is not clear if this is due to being essentially more feature efficient, or due to excess slack in the inducing point bound.

2.4.1.15 However, note that unlike in the inducing point case, this does not converge for finite M in general. The intuitive picture is that as M is increased, $\operatorname{span} c^*_{1:M}$ gets closer to every $k(\bullet, x_n)$ simultaneously, whereas inducing point style features can select from a subset of the canonical features, matching one $k(\bullet, x_n)$ exactly (and refining the approximation for the rest) for each additional feature.

2.4.1.16 For the proof, the following technical result will be needed.

Theorem 2.22 (Addition formula.) *(Giné M, 1975) Let $\mathcal{X}$ be a homogeneous Riemannian manifold. Let λ_ℓ be the distinct Dirichlet eigenvalues of the Laplace-Beltrami operator on $\mathcal{X}$, ν_ℓ their multiplicities, and $\psi_{j\ell}, j \in \{1 : \nu_\ell\}$ the corresponding eigenvectors. Then for any $x \in \mathcal{X}$,*

$$\sum_{j=1}^{\nu_\ell} |\psi_{j\ell}(x)|^2 = \nu_\ell.$$

Proof *(of Theorem 2.19)* Assume that

$$M = \sum_{\ell=1}^{L} \nu_\ell$$

for some L. If not, we can merely discard features until we reduce M to such a value to produce a valid upper bound on t. Now, I will proceed using either the index $m \in \{1{:}M\}$, or $\ell \in \{1{:}L\}, j \in \{1{:}\nu_\ell\}$ as appropriate, and for the eigenvalues I will

drop the j index, since the value does not depend on it. To avoid cluttering the notation, I will assume that $D' = D$ without loss of generality.

$$t = \sum_n \left[k(x_n, x_n) - \sum_{m=1}^{M} \alpha_m |\psi_m(x_n)|^2 \right] \tag{2.99}$$

$$= \sum_n \sum_{m=M+1}^{\infty} \alpha_m |\psi_m(x_n)|^2 \tag{2.100}$$

$$= \sum_n \sum_{\ell=L+1}^{\infty} \alpha_\ell \underbrace{\sum_{j=1}^{\nu_\ell} |\psi_{j\ell}(x_n)|^2}_{\nu_\ell \text{ by Theorem 2.22}} \tag{2.101}$$

$$= \sum_n \sum_{\ell=L+1}^{\infty} \nu_\ell \alpha_\ell \tag{2.102}$$

$$= N \sum_{m=M+1}^{\infty} \alpha_m \tag{2.103}$$

$$= N \sum_{m=M+1}^{\infty} s(\sqrt{\lambda_m}) \tag{2.104}$$

$$= N \sum_{m=M+1}^{\infty} s(C_{D,\mathcal{X}} m^{1/D} + o(m^{1/D})) \tag{2.105}$$

Now, applying $s(\xi) \leq \beta \xi^{-\kappa}, q > 1$, for sufficiently large ξ, there exists M_0 such that for $M > M_0$, by absorbing other constants into β and bounding the sum with an integral,

$$t \leq N\beta' \sum_{m=M+1}^{\infty} m^{-\kappa/D} + o(m^{-\kappa/D}) \tag{2.106}$$

$$\leq N\beta' \int_M^{\infty} m^{-\kappa/D} + o(m^{-\kappa/D})\, dm \tag{2.107}$$

$$\leq N\beta' \frac{D}{\kappa - D} M^{-(\kappa-D)/D} + o(M^{-(\kappa-D)/D}). \tag{2.108}$$

Or, rearranging,

$$t \in \mathcal{O}\left(\frac{D}{\kappa - D} N M^{-(\kappa-D)/D} \right). \tag{2.109}$$

Alternatively, applying $s(\xi) \leq \beta e^{-\kappa\xi}$,

$$t \leq N\beta' \sum_{m=M+1}^{\infty} e^{-\kappa m^{1/D} + o(m^{1/D})} \tag{2.110}$$

$$\leq N\beta' \int_M^\infty e^{-\kappa' m^{1/D}} dm \quad \text{for sufficiently large } M \tag{2.111}$$

$$\leq N\beta' \left[-\frac{m(\kappa' m^{1/D})^{-D}\Gamma(D, \kappa' m^{1/D})}{1/D} \right]_M^\infty \tag{2.112}$$

$$= N\beta' D\kappa'^{-D}\Gamma(D, \kappa' M^{1/D}) \tag{2.113}$$

where Γ is the generalised Gamma function, with $\Gamma(D, \infty) = 0$. Also, $\Gamma(D, \kappa' M^{1/D}) \in \mathcal{O}(e^{-M^{1/D}})$. Hence

$$t \in \mathcal{O}(DNe^{-M^{1/D}}) \tag{2.114}$$

as required. □

2.4.1.17 **Limitations** The construction here required isotropy in the containing space $\mathbb{R}^{D'}$, which is too limiting an assumption. One can effectively incorporate different lengthscales in each dimension by learning a linear mapping onto $\mathcal{X}$, but K_{uf} in general also depends on this mapping, so this violates precomputability and the cost returns to $O(NM^2)$.

2.4.1.18 The pragmatic strategy of Dutordoir et al (2020) is to learn the linear mapping using a small subset of the data only. But this is not really reducing the cost of learning – it is discarding the information from most of the data points.

2.5 Experimental evaluations

2.5.1 In this section, I introduce datasets and metrics which will be used repeatedly in subsequent chapters, and show that batched approximations often do not yield faster learning.

2.5.2 **Metrics** I use two standard test metrics. Given a Gaussian prediction at N_* points f_* for test data (x_*, y_*), the root mean squared error (RMSE) is

$$\text{RMSE}(q(\mathsf{f}_*)) = \frac{1}{N_*} ||\mathbb{E}[f_*] - y_*||_2 \tag{2.115}$$

while the negative log predictive density is

$$\text{NLPD}(\mathsf{f}_*) = \frac{1}{N_*} \sum_{n=1}^{N_*} \int p(y_{*,n}|\mathsf{f}_{*,n})\, q(\mathsf{f}_{*,n})\, d\mathsf{f}_{*,n} \tag{2.116}$$

2.5.3 which for a Gaussian predictive is

$$\text{NLPD}(q(\mathsf{f}_*)) = -\frac{1}{2N_*} \sum_{n=1}^{N_*} \log 2\pi(\sigma^2 + \text{Var}[\mathsf{f}_{*,n}]) - \frac{||y_{*,n} - \mathsf{f}_{*,n}||_2^2}{\sigma^2 + \text{Var}[\mathsf{f}_{*,n}]}. \tag{2.117}$$

2.5.4 This is the *marginal* negative log predictive density, which does not take into account the full covariance of f_*. For the avoidance of doubt, the expectation and variance are with respect to the distribution q.

2.5.5 The RMSE checks the quality of the predictive mean, whereas the NLPD also takes into account the marginal predictive variances, scoring a high error well if the corresponding variance is sufficiently large (it is okay to be wrong when uncertain).

2.5.6 Both metrics are better when lower. A low RMSE only shows good central estimates; a good NLPD also accounts for uncertainty estimates. Using both metrics is important, since if one consider only the NLPD, then when comparing two models, it is not immediately clear if improved NLPD is due to improvement in the central estimate or better calibrated uncertainty estimates, or both.

2.5.7 **Datasets** To evaluate the regression performance of different approximations, I use three example geospatial datasets of increasing size. In each case, the output and each dimensions of the input is shifted and rescaled to have zero mean and unit standard deviation. The predictive model then predicts normalised variables which need to be shifted and rescaled. The RMSE and NLPD are reported on the unnormalised scale. If the standard deviation of the data is originally σ_y, then the normalised metrics are transformed into unnormalised metrics by doing

$$\text{RMSE} \mapsto \sigma_y \text{RMSE}, \tag{2.118}$$

$$\text{NLPD} \mapsto \text{NLPD} + \log \sigma_y. \tag{2.119}$$

2.5.8 The values of N, N_* and σ_y for each dataset are given in Table 2.1.

Table 2.1 Summary of real world datasets.

Dataset	N	N_*	σ_y
Temperature	12 947	3 236	2.766
Precipitation	23 144	5 785	58.855
House price	106 875	26 718	0.642

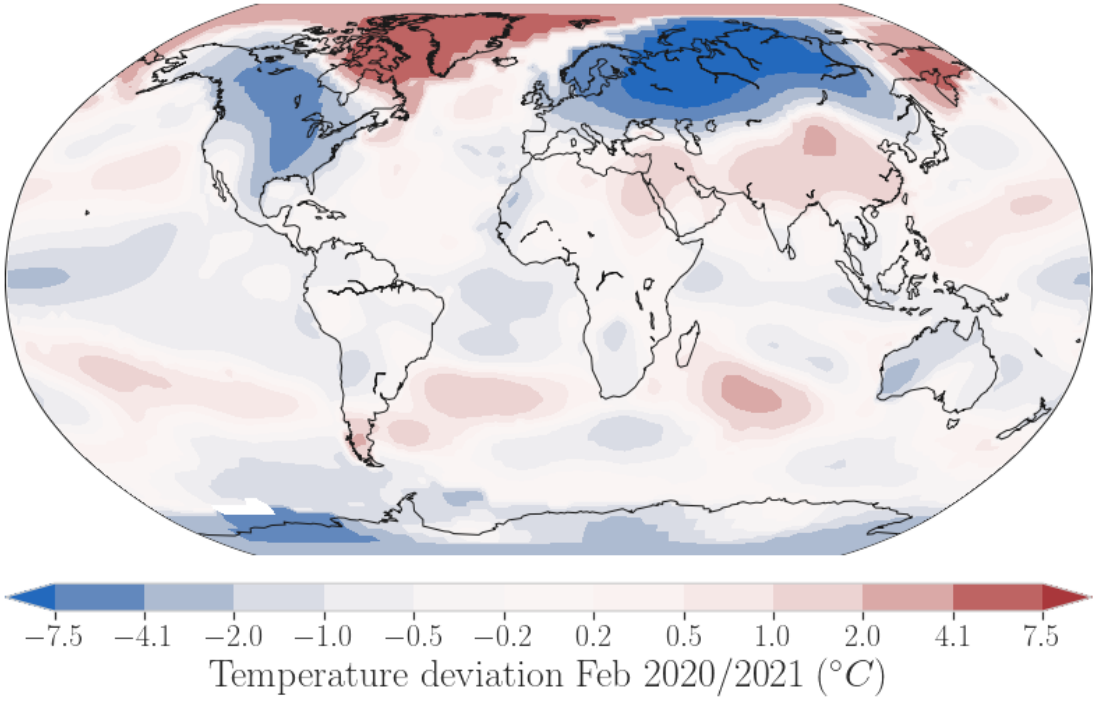


Figure 2.2 Temperature dataset, groundtruth

2.5.9 **Temperature** The temperature dataset is the change in mean land surface temperature (°C). over the year ending February 2021 relative to the base year ending February 1961.[2] It is the smallest of the datasets used, almost gridded over latitude and longitude, and fairly stationary. The data is shown in Figure 2.2.

2.5.10 **Precipitation** The precipitation dataset also contains nearly regularly gridded data, which are the modelled precipitation normals in mm in the contiguous United States for 1 January 2021.[3] The dataset is downsampled by 16 (by 4 in each dimension) to create a dataset of intermediate size. This is shown in Figure 2.3.

[2]Publicly available from https://data.giss.nasa.gov/gistemp/maps.

[3]publicly available with further documentation at https://water.weather.gov/precip/download.php; note the data at the source is in inches.

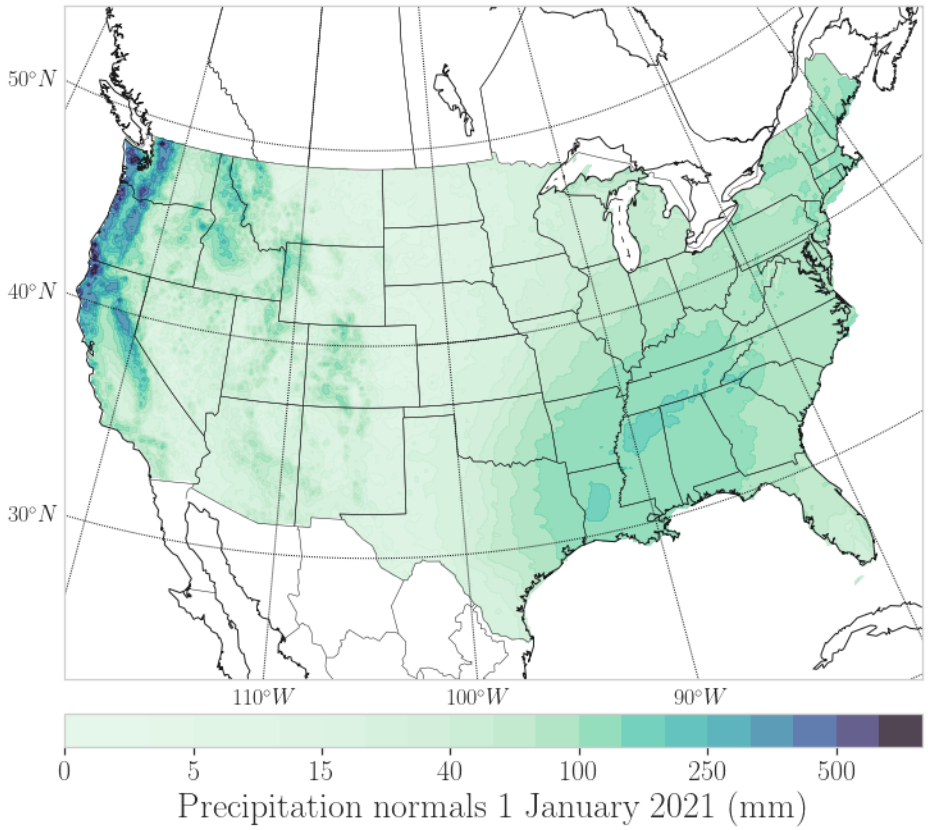


Figure 2.3 Precipitation dataset, groundtruth

2.5.11 The data is highly non-stationary, and so a challenging target for GP regression with typically used, usually stationary, covariance functions. In particular, the lengthscales are fairly large across the plains and in the southeast in general, but quite small near the Pacific coast, especially in the northwest. For a stationary model, the high frequency content in that region leads to a globally low lengthscale.

2.5.12 **House prices** The house price dataset is a snapshot of house prices in England and Wales, which is not regularly gridded (Figure 2.4). I use a random 20% slice of the full dataset, and target the log price to compress the dynamic range. It is based on the publicly available UK house price index[4]. A version of this dataset was used by Hensman et al (2013) to demonstrate scaling up GPs.

2.5.13 These datasets are chosen to be representative of geospatial regression problems of various sizes, and with different properties. For instance, the first have regularly

[4] https://landregistry.data.gov.uk/app/ukhpi

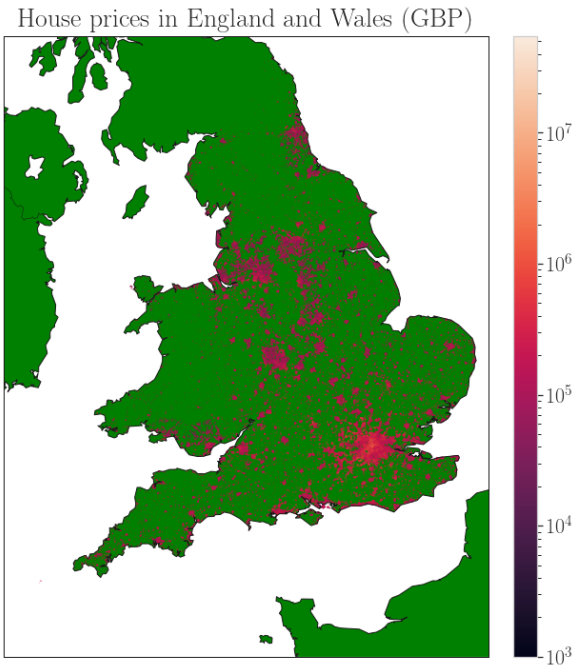


Figure 2.4 House price dataset, groundtruth

spaced inputs, whereas the last does not. The precipitation dataset is a good example of a highly non-stationary dataset as discussed. The houseprice dataset has variation on different spatial scales (large scale variations across the country, and local variations within cities, for example). Often, practitioners are interested in using GPs to model such datasets, with a view to making predictions, or solving downstream tasks such as optimally choosing where to collect data.

2.5.14 **Comparison with batched training** Many would argue that constructing new variational approximations is not the way to go, but instead that one should use stochastic learning with batched data (Hensman et al, 2013), which already removes the scaling with N. However, this requires learning the variational parameters as well as the model parameters, and generally leads to using a first order optimiser (for example, due to poor performance of stochastic second order optimisers, or the large number of variational parameters being optimised). As a result, in the conjugate setting, even for larger datasets (N on the order of 10^4 or 10^5), the closed form variational approach is faster than the mini-batched version.

2.5.15 This is demonstrated on the example datasets in Figure 2.5: the performance as measured by the training objective, the test RMSE and test NLPD are plotted as a

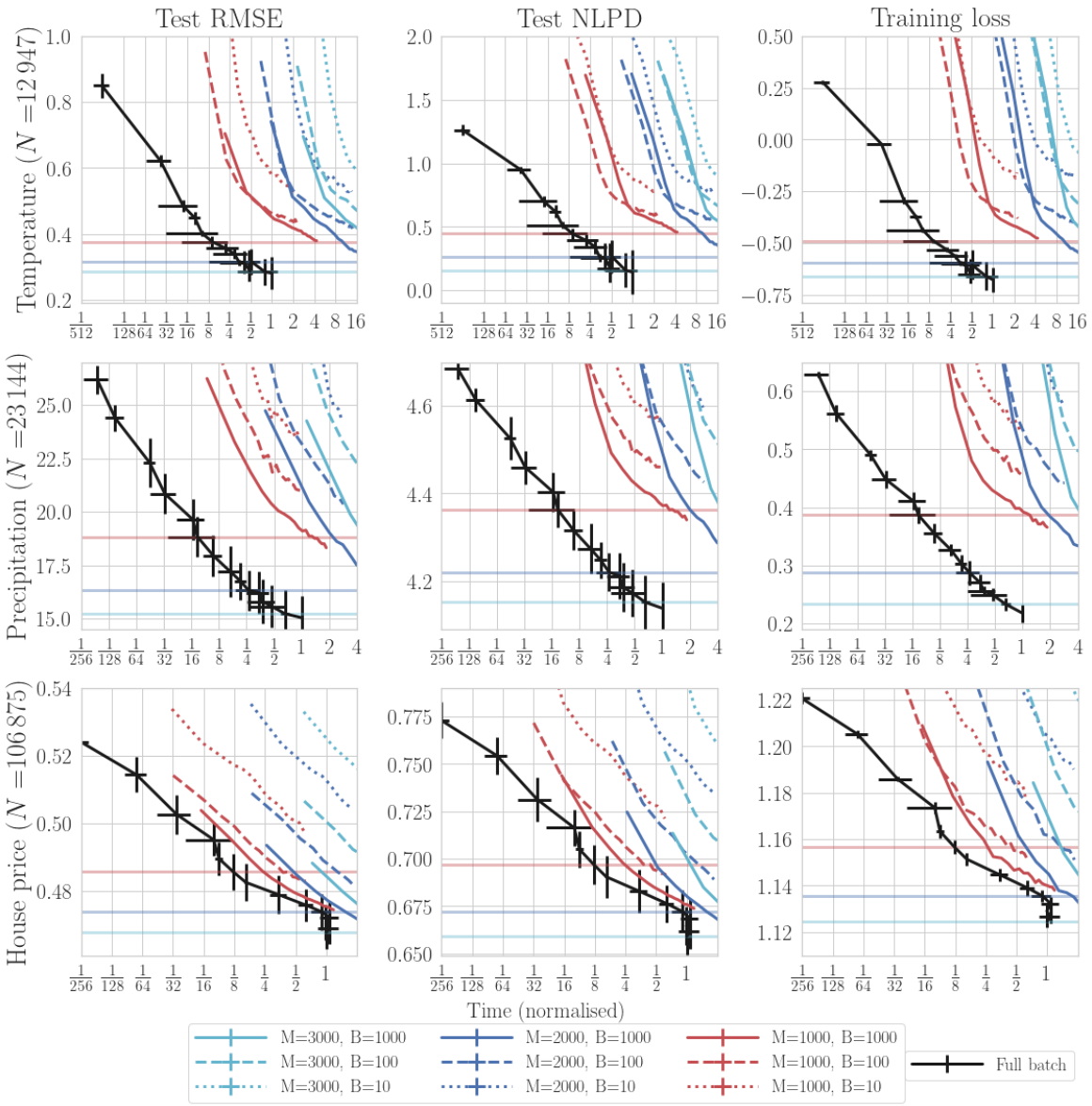


Figure 2.5 Collapsed, or full batch, conjugate sparse GP regression with inducing points (black) compared to stochastic variational GP regression. For batched learning, colours show variation in the number of inducing points, and the horizontal lines show the average full batch performance for the same number of inducing points. For example, the horizontal red line shows the performance achieved by full batch learning at $M = 1\,000$. The error bars for the stochastic results are suppressed to avoid cluttering the plot. The full batch curve is a parametric function of M, using up to 3 000 for the largest dataset, and 3 600 elsewhere. The stochastic results are parametric functions of the number of iterations of Adam (Paragraph 1.2.2.6). The learning rate (10^{-3}) was chosen as the best of several in initial, exploratory tests. Note the logarithmic time axis – even for the largest dataset, stochastic optimisation is much slower than full batch.

function of training time, using three uniformly random 80/20 train/test splits. In full batch, collapsed, training, μ_u, Σ_u are set in closed form. In this case, the training time is controlled implicitly by the number of of inducing points. In the case of batched learning, the training time is controlled more explicitly, by evaluating the performance after fixed numbers of iterations of the learning algorithm.

2.5.16 Another way to think about this comparison is that in the stochastic method, each pass through the data (each epoch) will come with a cost which scales linearly in N, and multiple epochs would be needed for training. On the other hand, with precomputable methods, only one pass through the data is needed; the rest of the learning process scales only as $\mathcal{O}(M^3)$. The full batch curves of Figure 2.5 are the same as those which appear in the next chapter, where it is shown that precomputable methods fulfill the promise of much faster learning.

2.5.17 **Mean functions** Finally, note that if there is a mean function with learnable parameters, we must consider whether or not it is precomputable. If not, we have to pay $\mathcal{O}(N)$ cost to recalculate it in each step. Often, the mean function is zero or identity, in which case no recalculation is required.

2.6 Summary

2.6.1 In this chapter, I set out the general requirements for and potential advantages of precomputable variational approximations. Examples in Sections 2.3 and 2.4 showed that the cost of learning can be lowered from $\mathcal{O}(NM^2)$ to $\mathcal{O}(M^3)$ with a comparable asymptotic feature efficiency (bound on M in terms of N) to using inducing points. Moreover, I showed empirically in Section 2.5 that on various geospatial datasets, minibatching the training data is not a faster alternative. However, the main families of methods described here all suffer from some basic limitations.

2.6.2 Generally, these are that they require the RKHS inner product has a convenient explicit representation due to some combination of the covariance function and the domain. This could be a product of 1D Matérn covariance functions on a rectangular subset $\prod_d [a_d, b_d]$ as in the work of Hensman et al (2017), or using eigenfunctions, which works out particularly conveniently on compact homogeneous Riemannian manifolds (Section 2.4) as long as the harmonic expansion of the covariance function

is known. But in this latter case, typically the harmonic expansion is not known and must be estimated numerically with unclear error, or the covariance function has too many symmetries (for example, it is isotropic).

2.6.3 It is desirable to have a scheme which works a little more flexibly, admitting a broader class of covariance functions, for example in order that a practitioner can freely select the best covariance function for the dataset. In the next chapter, I abandon the search for *exact* variational methods which meet these requirements, and add additional approximations to produce a more flexible, approximate scheme. Of course, where additional approximations are introduced, it is necessary to analyse how the approximations impact learning.

2.A Reference notes and further reading

2.A.1 The main part of the background development and convergence results are based upon Burt (2022).

2.A.2 The proof of Theorem 2.18 follows that of Dutordoir et al (2020), generalising to a broader class of manifolds. Non-stationary Matérn class covariance functions have also been developed for Riemannian manifolds (Borovitskiy et al, 2020; Hutchinson et al, 2021); these also share their eigenfunctions with the Laplacian, so the same precomputable methods can be used. There is a cost in $\mathcal{O}(N)$ in each step associated with estimating the function variance at each training location. Nevertheless, these could be promising methods for data on quite general Riemannian manifolds.

2.A.3 The projection-based presentation of Section 2.3 was inspired by the descriptions of Hensman et al (2017).

Chapter 3

Fast stationary modelling with approximate Fourier series

3.0.1 The previous chapter described precomputable and feature efficient variational approximations for conjugate sparse Gaussian process regression. These provide much faster learning than log marginal likelihood maximisation with little difference in performance.

3.0.2 But the methods set out to achieve this were limited in the sense that they were applicable only to a narrow subset of covariance functions. In this chapter, by introducing some additional approximations, I produce a practical method based on approximate Fourier series representations which works with a broad class of stationary covariance functions on $\mathbb{R}^D$. The effect of the additional approximations is accounted for in the theoretical results (Sections 3.2 and 3.3).

3.0.3 As with several of the methods in the previous chapter, the features in higher dimensions have a product structure, which leads to a cost exponential in the dimension. Nonetheless, the method remains applicable to the low dimensional, large data settings common in spatial modelling, particularly with some additional pragmatic variations (Section 3.4). The faster learning is demonstrated with real-world geospatial datasets (Section 3.5).

3.0.4 Finally, approximate Fourier series methods have further advantages when the data has additional structure to leverage. In particular, if the training inputs form a rectangular lattice within rectangular boundaries, then the computational cost per

optimisation step is reduced from $\mathcal{O}(M^3)$ to $\mathcal{O}(M)$ (Section 3.6). This setting, commonly referred to as gridded data, is not unusual in settings where the data is generated from a simulator. Minor irregularities, such as missing grid points and irregular boundaries, are discussed.

3.0.5 **Contributions** I described the method developed here, and the bulk of the empirical results of Section 3.5 previously (Cheema & Rasmussen, 2024a). The exposition and proofs there approached the problem differently and gave weaker guarantees; see Appendix A.3 for an overview and discussion. The exposition and main theoretical results of this chapter, and the extension to gridded data, are my own contribution except where noted, and new to this thesis.

3.1 Overview

3.1.1 Recall Figure 2.1 (reproduced in Figure 3.1) and the discussion in Section 2.4: conditioning on Fourier features is ineffective since the spectrum is continuous. Recall that otherwise, Fourier features yielded precomputable inducing features. In Figure 3.1c, I plot the conditional in the case that the covariance function is periodic, which yields a discrete spectrum (a Fourier series; FS). Then the problem with conditioning on Fourier features vanishes.

3.1.2 **An idealised setting** To make this precise in a simple setting, suppose that $\mathcal{X} = \mathbb{R}$ and let

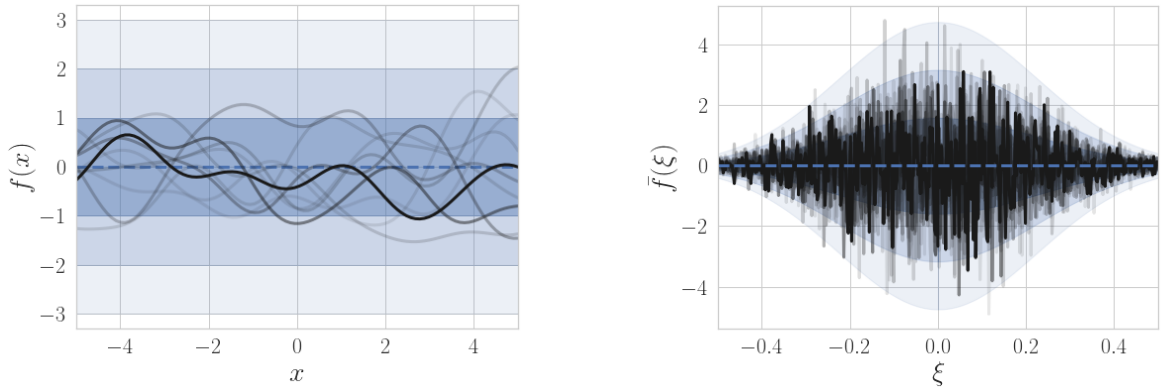
$$W_x = \max_n x_n - \min_n x_n \tag{3.1}$$

be the width of the dataset, and assume that the midpoint $\min_n x_n + \frac{W_x}{2} = 0$ (this assumption can easily be relaxed by shifting the data). Suppose also that there exists some $W \geq W_x$ such that

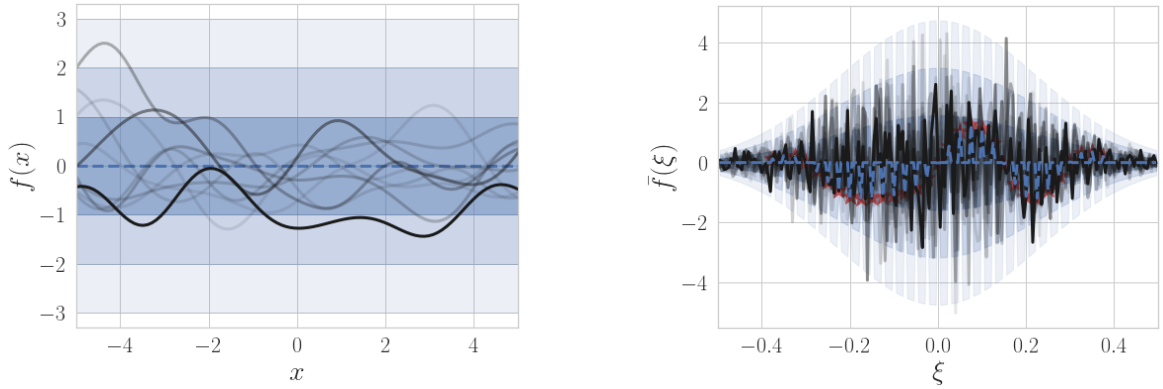
$$k(r) = 0 \quad \forall r \,\big|\, |r| > W. \tag{3.2}$$

3.1.3 Of course, this is not a very realistic case – widely used covariance functions do not vanish except at infinity. But this will be dealt with in Sections 3.2 to 3.4 by approximation, so for now I proceed with the idealised case.

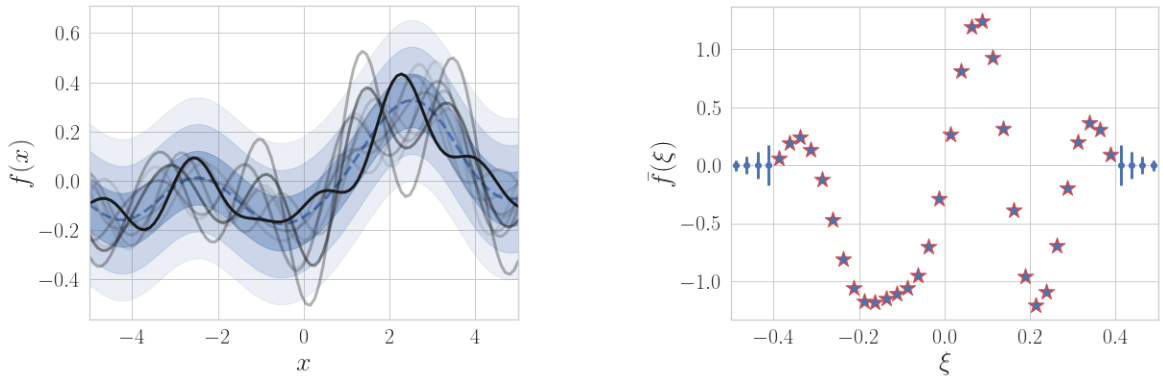
3.1.4 Now consider the model being applied on the domain $\tilde{\mathcal{X}} = [-W/2, W/2]$. Then on this domain, k (as a function of separation) is equal to its $2W$-periodic repetition k_P.



(a) Prior



(b) Conditioning on Fourier features



(c) Conditioning on a Fourier series

Figure 3.1 Illustrating Fourier series approximations; sequel to Figure 2.1. Recall that the samples in the frequency domain (right) and spatial domain (left) correspond to one another. (a) The prior's Fourier transform is white Gaussian noise whose variance is given by the spectral density. (b) We cannot condition meaningfully on come finite collection of frequencies (red stars), as this gives no information about the other frequencies – the conditional prior $p(f|u)$ in the data domain is unchanged. (c) Now conditioning is performed on a Fourier series formed by discretising and rescaling the spectral density. The conditional prior is now meaningful, and the residual uncertainty is due to high frequency content not included in the features (visible at the edges).

This can be represented by its Fourier series.

$$k_P(r) = \frac{1}{2W} \sum_{j \in \mathbb{Z}} s_j e^{i2\pi r j (2W)^{-1}} \tag{3.3}$$

3.1.5 The Fourier coefficents are just regular samples of the spectral density, since the j^{th} Fourier coefficient is given by

$$\begin{aligned} s_j &= \int_{-W}^{W} k_P(r) e^{-i2\pi r j (2W)^{-1}} \, dr && (3.4) \\ &= \int_{-W}^{W} k(r) e^{-i2\pi r j (2W)^{-1}} \, dr \quad (k_P = k \text{ on } [-W, W]) && (3.5) \\ &= s(2\pi j (2W)^{-1}). && (3.6) \end{aligned}$$

3.1.6 Often, the normalisation factor $(2W)^{-1}$ is absorbed into the definition of s_j. The convention used here is to yield the above property.

3.1.7 This is illustrated in Figure 3.2, using the toy example of a triangular covariance function. The periodisation of k can be viewed as the original k plus *aliases* centred at $2jW, j \in \mathbb{Z}$. These aliases do not distort the covariance function on $[-W, W]$, due to the finite width property.

3.1.8 This result is closely related to the work on stationary covariance functions on compact homogeneous Riemannian manifolds in Section 2.4. The Laplacian's eigenvalues and eigenvectors on a domain of width $2W$ are respectively

$$\left\{ (2\pi j (2W)^{-1})^2, \frac{1}{\sqrt{2W}} e^{i2\pi r j (2W)^{-1}} \right\}_{j \in \mathbb{Z}}. \tag{3.7}$$

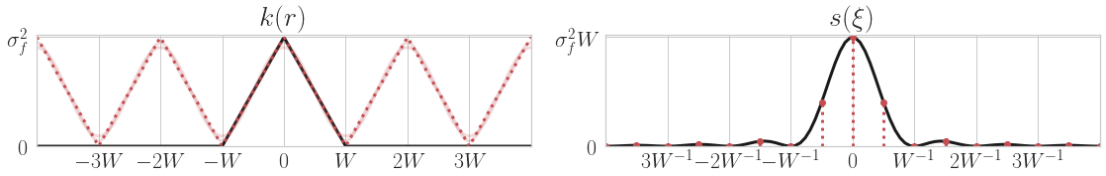


Figure 3.2 Toy example triangular covariance function (left, black) which follows Equation (3.2), and the corresponding spectral density (right, black). The periodisation and its Fourier series are shown in red.

The FS decomposition can therefore alternatively be arrived at by combining Theorem 2.18 with Theorem 2.5.

3.1.9 **The precomputable features** Now, let M be odd for convenience. Then precomputable features can be constructed by setting

$$\varphi_{m_1}(x) = \frac{1}{s(2\pi m_1(2W)^{-1})} e^{i2\pi x m_1 (2W)^{-1}} \quad m_1 \in \left\{ -\frac{M-1}{2} : \frac{M-1}{2} \right\} \tag{3.8}$$

and the convergence rate depends on the spectral density; it is given by applying Theorem 2.19 with $D' = 1$. In particular, we get vanishing error in the marginal likelihood approximation with $M \in \mathcal{O}(\log N)$ for sub-Gaussian spectral densities, and with $M \in \mathcal{O}(N^{\frac{1}{2\nu}})$ for Matérn-ν covariance functions. It can also be shown that M should scale up with W^{-1} for a specified error tolerance.

3.1.10 **The white noise process perspective** An equivalent view, mirroring that of Paragraphs 2.4.0.5 and 2.4.0.6, is that instead of doing inference using the original model, for which the Fourier transform of the function $\tilde{f}$ is a white noise process, we do inference using an approximate model $\hat{f}$ which is periodic; its Fourier transform $\tilde{\hat{f}}$ is still white but is now supported only on countably many points. But by construction this approximate prior is equivalent to the original on any length W interval, and since $W \geq W_x$, the posterior is unchanged.

3.1.11 Of course, the predictions outside of $\tilde{\mathcal{X}}$ are substantially different and should not be used. If a test point arises outside of $\tilde{\mathcal{X}}$, we should increase W – but there is no need to relearn the parameters.

3.1.12 This method can straightforwardly be extended to higher dimensions by replacing the one dimensional Fourier series of Equation (3.3) with a multidimensional Fourier series. Then M will scale up with W^{-D}.

3.1.13 **The new approximate log marginal likelihood** However, usually k does not vanish except at infinity, so in the next sections I introduce additional approximations to deal with this. Recall that just as k is used to calculate the elements of K_{ff}, so c is used to calculate the elements of K_{uf} and $\bar{k}$ is used to calculate the elements of K_{uu} (Equations (2.25) and (2.26)). I will consider an approximate covariance function $\hat{k}$ to which the linear functionals are applied to

compute approximations $\hat{c}$ to c and $\hat{\bar{k}}$ to $\bar{k}$. In practice, I mix k with $\hat{c}, \hat{\bar{k}}$ in calculating the variational objective and the posterior predictive distribution. Then we can no longer apply Theorem 2.10.

3.1.14 In order to clarify the situation, some additional notation will be useful. Let $\mathcal{L}$ and $\hat{\mathcal{L}}$ be the log marginal likelihood if $f \sim \mathcal{GP}(0, k)$ and if $f \sim \mathcal{GP}(0, \hat{k})$ respectively.

$$\mathcal{L} = \log \mathcal{N}(y|0, \mathsf{K}_{\mathsf{ff}} + \sigma^2 \mathsf{I}) \tag{3.9}$$
$$\hat{\mathcal{L}} = \log \mathcal{N}(y|0, \hat{\mathsf{K}}_{\mathsf{ff}} + \sigma^2 \mathsf{I}) \tag{3.10}$$

3.1.15 Now, let $\mathcal{L}'$ be the variational objective computed using inducing features, which is a lower bound to $\mathcal{L}$ (as in Section 2.2). Let the variational lower bound analogously constructed for $\mathcal{L}'$ be $\hat{\mathcal{L}}'$. Then these have the following explicit expressions.

$$\mathcal{L}' = \log \mathcal{N}(y|0, \mathsf{K}_{uf}^* \mathsf{K}_{uu}^{-1} \mathsf{K}_{uf} + \sigma^2 \mathsf{I}) - \frac{1}{2\sigma^2} \operatorname{tr}(\mathsf{K}_{\mathsf{ff}} - \mathsf{K}_{uf}^* \mathsf{K}_{uu}^{-1} \mathsf{K}_{uf}) \tag{3.11}$$
$$\stackrel{\text{def}}{=} \log \mathcal{N}(y|0, \mathsf{Q}_{\mathsf{ff}}) + \underbrace{\operatorname{tr}(\mathsf{K}_{\mathsf{ff}} - \mathsf{Q}_{\mathsf{ff}})}_{t} \tag{3.12}$$
$$\hat{\mathcal{L}}' = \log p(y|0, \hat{\mathsf{K}}_{uf}^* \hat{\mathsf{K}}_{uu}^{-1} \hat{\mathsf{K}}_{uf}) - \frac{1}{2\sigma^2} \operatorname{tr}(\hat{\mathsf{K}}_{\mathsf{ff}} - \hat{\mathsf{K}}_{uf}^* \hat{\mathsf{K}}_{uu}^{-1} \hat{\mathsf{K}}_{uf}) \tag{3.13}$$
$$\stackrel{\text{def}}{=} \log p(y|0, \hat{\mathsf{Q}}_{\mathsf{ff}}) - \frac{1}{2\sigma^2} \underbrace{\operatorname{tr}(\hat{\mathsf{K}}_{\mathsf{ff}} - \hat{\mathsf{Q}}_{\mathsf{ff}})}_{\hat{t}} \tag{3.14}$$

3.1.16 The shorthands $t, \hat{t}, \mathsf{Q}_{\mathsf{ff}}, \hat{\mathsf{Q}}_{\mathsf{ff}}$ have been introduced or reintroduced for convenience. In the chapter so far, I have been building up to using $\hat{\mathcal{L}}'$ as a learning objective, under the assumption that the approximate prior does not differ too much from the true prior, at least at the training locations. In practice, we do not usually have an explicit expression for $\hat{k}$ – it could be the periodisation of k, for example. Hence, I replace $\hat{\mathsf{K}}_{\mathsf{ff}}$ with K_{ff}, which yields the following learning objective.

$$\hat{\mathcal{L}}'' = \log p(y|0, \hat{\mathsf{Q}}_{\mathsf{ff}}) - \frac{1}{2} \operatorname{tr}(\mathsf{K}_{\mathsf{ff}} - \hat{\mathsf{Q}}_{\mathsf{ff}}) \tag{3.15}$$
$$= \hat{\mathcal{L}}' - \frac{1}{2\sigma^2} \underbrace{\operatorname{tr}(\hat{\mathsf{K}}_{\mathsf{ff}} - \mathsf{K}_{\mathsf{ff}})}_{\tau} \tag{3.16}$$

Table 3.1 Summary of the two approximate Fourier series methods and main results of Sections 3.2 and 3.3, with ε any positive value.

	subsampled AFS	AFS DFT
$\widehat{k}(r)$	$k_P(r)$	$(2W)^{-D}\sum_{m_{1:D}} \hat{s}_{T,m_{1:D}} e^{i2\pi z_{m_{1:D}}^\top r} \approx k_{TP}$
$\hat{\tilde{k}}^{-1}_{mm}$	$(2W)^{-D}s(2\pi z_m)$	$(2W)^{-D}\hat{s}_{T,m_{1:D}}$ (Equation (3.61))
SE	$M \in \mathcal{O}(\log^D N)$	$M \in \mathcal{O}(\log^D N)$
Matérn-ν	$M \in \mathcal{O}(N^{\frac{D}{2\nu}})$	$M \in \mathcal{O}(N^{\frac{D}{(1-\varepsilon)[\nu]}})$
$\lvert k(r) - \widehat{k}(r)\rvert$	Theorem 3.1	Theorem 3.5
$t + {}^{\lvert\tau\rvert}/_2$	Theorem 3.2	Theorem 3.7

3.1.17 It immediately follows from Theorem 2.10 that

$$\mathbb{E}[|\widehat{\mathcal{L}} - \widehat{\mathcal{L}}''|] \leq \frac{1}{\sigma^2}\hat{t} + \frac{|\tau|}{2\sigma^2}. \tag{3.17}$$

3.1.18 Ideally we would like to check the convergence rate of $|\mathcal{L} - \widehat{\mathcal{L}}''|$. So far, if $W_x \leq W$ then $\mathsf{K}_{\mathsf{ff}} = \hat{\mathsf{K}}_{\mathsf{ff}}$, so $\widehat{\mathcal{L}} = \widehat{\mathcal{L}}'$. But in the next sections, we will only be able to guarantee that each element of $\mathsf{K}_{\mathsf{ff}} - \hat{\mathsf{K}}_{\mathsf{ff}}$ is small when $W_x < W$. This generally leads to bounds which are quadratic in N, rather than linear as in Corollary 2.3.

3.1.19 However, it is intuitively reasonable to expect that a small element-wise error in the covariance matrix should not change the marginal likelihood's sensitivity to the parameters too much. This is comparable to saying the marginal likelihood is fairly robust to numerical errors, for example.

3.1.20 Furthermore, $\mathcal{L} - \widehat{\mathcal{L}}''$ is no longer the posterior KL, and we might reasonably be concerned about the impact of using the same inconsistent approximations to compute the posterior predictive. But since we have a good posterior approximation to a model constructed with $\widehat{k}$, if k is close to $\widehat{k}$ then the posterior predictions should be fairly good.

3.1.21 **Approximate by ignoring aliasing** The first method (Section 3.2) is to ignore aliasing. Then $\widehat{k} = k_P$ and $s_j = s(2\pi j(2W)^{-1})$ as in Equation (3.4). But now k_P is not equal to k on $[-W, W]$; there is aliasing distortion, which will reduce as W is increased, and be less for covariance functions with faster decay. This is illustrated in

Figure 3.3. This is a convenient approximation when the spectral density is available in closed form, as for the squared exponential and Matérn covariance functions.

3.1.22 **Approximate the periodised truncation** Define k_T as

$$k_T(r) = \begin{cases} k(r) & |r| < W \\ 0 & \text{otherwise} \end{cases} \tag{3.18}$$

then its periodisation $k_{TP} = k$ throughout $[-W, W]$. The FS of k_{TP} can be conveniently estimated using a discrete Fourier transform (DFT) (Section 3.3) with cost in $\mathcal{O}(M^2)$. This is illustrated in Figure 3.4. This is convenient when the spectral density is not available in closed form.

3.1.23 I refer to these methods, and those with additional approximations introduced in Section 3.4, as variational approximate Fourier series (AFS).

3.1.24 **Higher dimensions** In order to easily generalise to higher dimensions, some additional notation will be useful. Firstly, use z_m to denote the frequencies we evaluate at, and ζ_j the shifted centres of k_P (not to be confused with the GP mean function, which is assumed 0 in this chapter). The canonical case used for proofs will be where M is a power of D, and $M^{1/D}$ is odd.

3.1.25 Where it is convenient, instead of using the single index $m \in \{1{:}M\}$, I will use the multi-index $m_{1:D}$ where each $m_d \in \{-(M^{1/D}-1)/2{:}(M^{1/D}-1)/2\}$. Using the same multi-index infinitely extended, the collection of all possible frequencies forms a square lattice.

$$z_{j_{1:D}} = (2W)^{-1} j_{1:D} \tag{3.19}$$

3.1.26 For the avoidance of doubt, here $j_{1:D}$ is being treated as a vector in $\mathbb{R}^D$. Similarly, the centres of the periodised covariance function ζ_j form a dual lattice.

$$\zeta_{j_{1:D}} = 2W j_{1:D} \tag{3.20}$$

3.1.27 The higher dimensional periodisation is

$$k_P(r) = \sum_{j_{1:D} \in \mathbb{Z}^D} k(r - 2W j_{1:D}). \tag{3.21}$$

3.1.28 The higher dimensional FS is defined by the following forward and inverse transforms.

$$k(r) = \frac{1}{(2W)^D} \sum_{j_{1:D} \in \mathrm{Z}^D} s_{j_{1:D}} e^{-i2\pi z_{j_{1:D}}^\top r} \tag{3.22}$$

$$s_{j_{1:D}} = \int_{[-W,W]^D} k(r) e^{-i2\pi r^\top z_{j_{1:D}}} \, dr \tag{3.23}$$

3.1.29 For concreteness, the feature functions are

$$\varphi_{m_{1:D}}(x) = \frac{1}{s(2\pi z_{m_{1:D}})} e^{i2\pi x^\top z_{m_{1:D}}} \quad \text{for } m_{1:D} \in \left\{ -\frac{M^{\frac{1}{D}} - 1}{2} : \frac{M^{\frac{1}{D}} - 1}{2} \right\}^D \tag{3.24}$$

and the inducing variables are defined by

$$u_{m_{1:D}} = \langle \hat{f}, \varphi_{m_{1:D}} \rangle \tag{3.25}$$

where $\hat{f} \sim \mathcal{GP}(0, \hat{k})$.

3.1.30 The next two sections provide the element-wise bounds on K_{ff} and the convergence rates for $|\widehat{\mathcal{L}} - \mathcal{L}''|$; the approximations and results are summarised in Table 3.1.

3.1.31 Section 3.4 covers a variety of practical variations which reduce the number of frequencies used in practice, and moves from complex-valued features to real-valued ones. The empirical evaluation is covered in Section 3.5. Section 3.6 covers the additional improvements possible in the gridded setting.

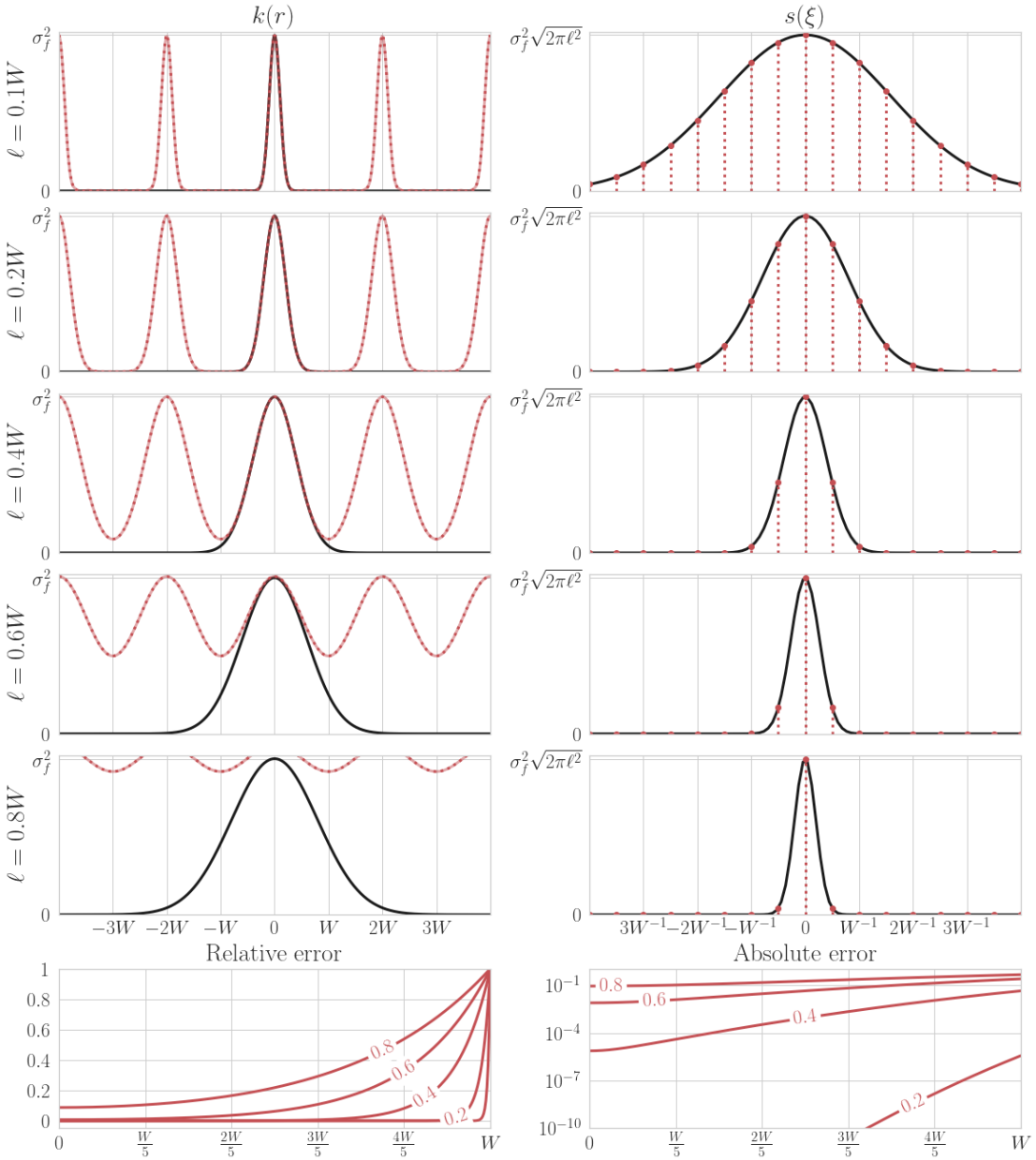


Figure 3.3 Approximating by ignoring aliasing (subsampled AFS). The original covariance function and spectral density are plotted in black; k_P and its FS are in red dashed. The continuous red lines are the low frequency ($M = 17$) approximation. The error $|k_P - k|$ is shown in the bottom panels. The covariance function is squared exponential with lengthscale ℓ, and the contours are marked with ℓ/W; the approximation deteriorates as ℓ gets closer to W.

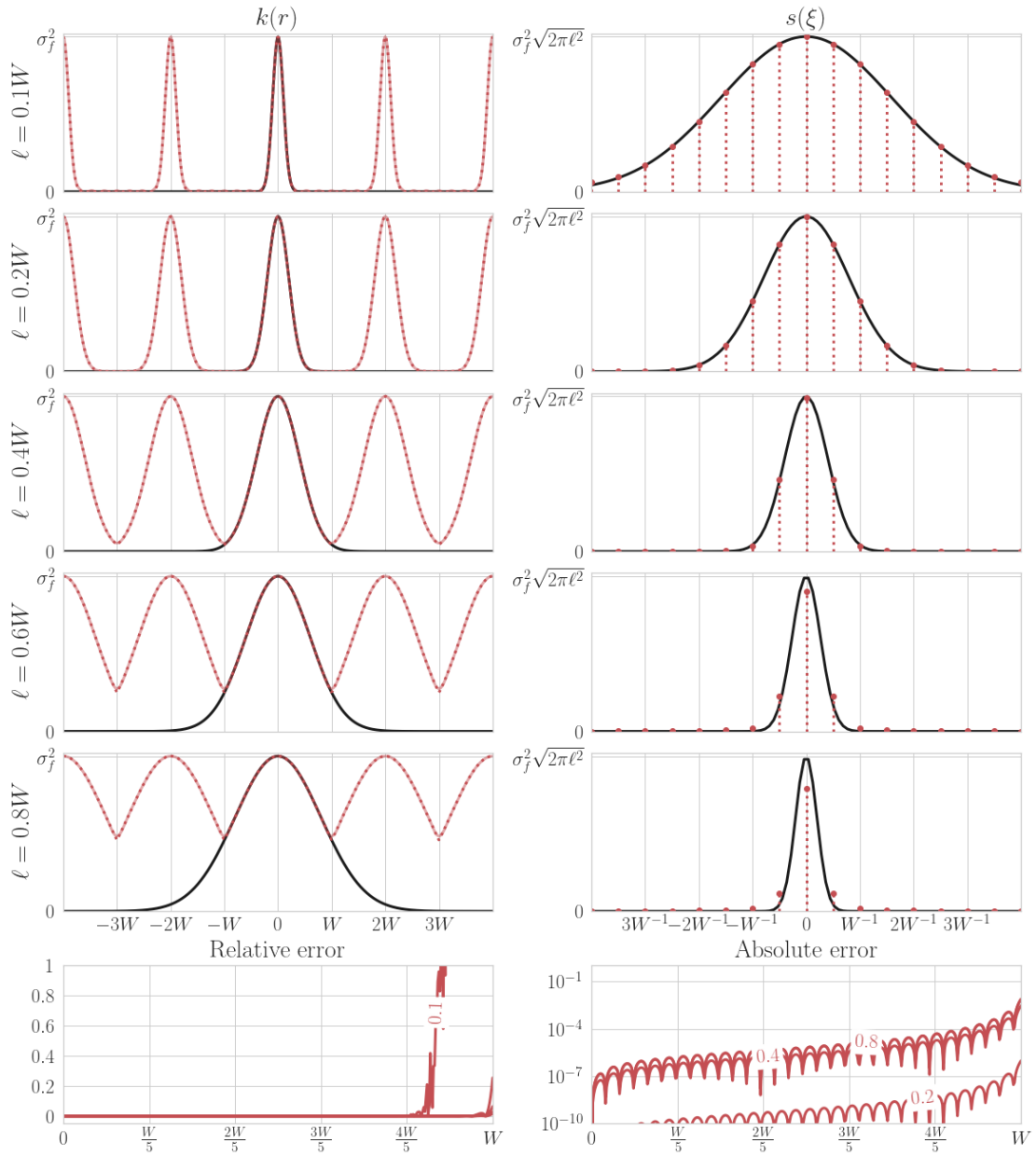


Figure 3.4 Approximating the periodised truncation using the DFT (AFS DFT). The original covariance function and spectral density are plotted in black; k_{TP} and its FS in red dashed. The error is 0 everywhere in $[-W, W]$ – the bottom panels show the error for the $M = 30$ approximation. The covariance function is squared exponential with lengthscale ℓ, and the contours are marked with ℓ/W.

3.2 Approximate Fourier series by subsampling

3.2.1 If the covariance function does not have finite width, then regularly sampling the spectral density as in Equation (3.3) still yields a periodisation of the covariance function, but now there is aliasing distortion. This will be more severe if k has not decayed close to zero by $\pm W$ – see Figure 3.3.

3.2.2 To show the periodisation result, let Ш_a be the Dirac comb with spacing a, or

$$\text{Ш}_a(x) = \sum_{j_{1:D} \in \mathbb{Z}^D} \delta(x - j_{1:D} a) \tag{3.26}$$

where δ is the Dirac delta. This periodises functions by convolution, and has the Fourier transform self-similarity (Vetterli et al, 2012, Chapter 5)

$$[\mathcal{F}\text{Ш}_a](\xi) = a^{-D}\text{Ш}_{a^{-1}}(\xi). \tag{3.27}$$

3.2.3 The Dirac comb as a linear functional performs regular sampling, from which the result will follow. Below, the operator $\star$ denotes convolution.

$$\begin{aligned}
\hat{k}(r) &= \frac{1}{(2W)^D} \sum_{j_{1:D} \in \mathbb{Z}^D} s(2\pi r j_{1:D} (2W)^{-1}) e^{i2\pi r^\top j_{1:D} (2W)^{-1}} && (3.28)\\
&= \int_{\mathbb{R}^D} s(2\pi\xi) \frac{1}{(2W)^D} \text{Ш}_{(2W)^{-1}}(\xi) e^{i2\pi r^\top \xi} \, d\xi && (3.29)\\
&= \mathcal{F}^{-1}[(2W)^{-1} \text{Ш}_{(2W)^{-1}} s] && (3.30)\\
&= k \star \text{Ш}_{2W} && (3.31)\\
&= \sum_{j_{1:D} \in \mathbb{Z}^D} k(r - j_{1:D} 2W) = k_P. && (3.32)
\end{aligned}$$

3.2.4 If the spectral density decays fast, then so does the approximating Fourier series. Then $\hat{t}$ will converge quickly to zero as M grows. If k has fast decay, then the aliasing distortion will reduce to 0 quickly as W grows. This reduces the element-wise error in the kernel matrix, and reduces $|\tau|$.

3.2.5 Define the tail integral of the covariance function as

$$I_k(r) = \int_{r' \notin [-r,r]} |k(r')|\, dr' \tag{3.33}$$

or, in higher dimensions,

$$I_k(r) = \int_{r' \notin [-r,r]^D} |k(r')|\, dr'. \tag{3.34}$$

Similarly, define the tail integral of the spectral density as

$$I_s(\xi) = \int_{\xi' \notin [-\xi,\xi]^D} s(\xi')\, d\xi'. \tag{3.35}$$

Generally, I will assume exponential or polynomial bounds on these tails. The aliasing distortion will shrink faster with W if I_k has fast decay, and the features will converge faster with M if I_s has fast decay.

Theorem 3.1 (Covariance function approximation error for subsampled AFS.) *Assume $k(r) \in \mathcal{O}(||r||^{-(1+\kappa_1)D})$, and so $I_k(r) \in \mathcal{O}(||r||^{-\kappa_1 D})$. Then if $\hat{k} = k_P$ (Equation (3.21)),*

$$|\mathsf{K}_{\mathsf{ff}} - \hat{\mathsf{K}}_{\mathsf{ff}}|_{n,n'} \in \mathcal{O}(W^{-(1+\kappa_1)D}).$$

If instead $k(r) \in e^{-||r||}$ and $I_k(r) \in \mathcal{O}(e^{-||r||})$, then

$$|\mathsf{K}_{\mathsf{ff}} - \hat{\mathsf{K}}_{\mathsf{ff}}|_{n,n'} \in \mathcal{O}(e^{-W}).$$

Proof The objective is to bound

$$|\mathsf{K}_{\mathsf{ff}} - \hat{\mathsf{K}}_{\mathsf{ff}}|_{n,n'} = |k(x_n - x_{n'}) - k_P(x_n - x'_n)|. \tag{3.36}$$

The general idea is given by the 1D case. First, let k be monotonically decreasing. By assumption $r = |x_n - x_{n'}| < W$.

$$|k(r) - k_P(r)| = \sum_{j \neq 0} k(\zeta_j - r) = \sum_{j_1=1}^{\infty} k(2Wj_1 - r) + \sum_{j_1=1}^{\infty} k(-2Wj_1 - r) \tag{3.37}$$

$$\leq 2(k(W) + \sum_{j_1=2}^{\infty} k(2W(j_1+1))) \tag{3.38}$$

$$\leq 2k(W) + 2(2W)^{-1} \int_W^{\infty} k(r)\, dr \tag{3.39}$$

The second step follows by maximising each of the first and second sum by letting r be W and $-W$ respectively. The second step is a standard upper bound for sums by integrals for monotonically decreasing functions. Now the second term is recognisable as $(2W)^{-1}I_k(r)$, and the result follows. If the covariance function is not monotonically decreasing, replace k everywhere by a monotonically increasing upper bound k' with the same asymptotic rate of decay. The closest aliases could not be included in the integral part of the bound, else the integral would start at 0, and would not decrease with W.

The higher dimensional case follows similarly. But now we exclude from the integral every alias which is at most $2W$ away from the origin in any dimension.

$$|k(r) - k_P(r)| \leq \sum_{j\neq 0 | \text{each } |j_d|\leq 1} k(\zeta_j) + (2W)^{-1} \int_{r\notin[-W,W]^D} I_k(r)\, dr \tag{3.40}$$

The sum has $3^D - 1$ elements, but D is assumed constant.

For the exponential case, the proof is the same, but the scaling of the $I_k(r)$ by $(2W)^{-1}$ does not change the order of the bound. □

Theorem 3.2 (Convergence for subsampled AFS.) *Let k have spectral density s, and let M and z be as set out in Paragraph 3.1.24, let $\hat{k} = k_P$ (Equation (3.21)) and assume $I_k(r) \in \mathcal{O}(||r||^{-\kappa_1 D})$, $I_s(\xi) \in \mathcal{O}(||\xi||^{-\kappa_2 D})$, and $k(r) \in \mathcal{O}(||r||^{-(1+\kappa_1)D})$. Using the approximate FS features (Paragraph 3.1.29)*

$$\hat{t} + \frac{|\tau|}{2} \in \mathcal{O}(NM^{-\kappa_2 \frac{\kappa_2 - 1}{\kappa_1 + \kappa_2}}).$$

If instead $I_k(r) \in \mathcal{O}(e^{-||r||}), I_s(\xi) \in \mathcal{O}(e^{-||\xi||}), k(r) \in e^{-||r||}$, then

$$\hat{t} + \frac{|\tau|}{2} \in \mathcal{O}(Ne^{-M^{\kappa_2}}).$$

Proof In the first instance, assume that the spectral density decreases monotonically in every dimension. Then we can show the following technical bounds on the spectral density. Let the largest frequency in each dimension be w:

$$w = \frac{M^{1/D} - 1}{2}(2W)^{-1}. \tag{3.41}$$

In the 1D case, the multidimensional form of the index m_1 is the same as the single index m, but shifted by $(M-1)/2$. Then,

$$(2W)^{-1}\sum_{m>M} s(2\pi z_m) = 2(2W)^{-1}\sum_{m_1>\frac{M-1}{2}} s(2\pi(2W)^{-1}m_1) \tag{3.42}$$

$$\leq 2\int_{(2W)^{-1}\frac{M-1}{2}}^{\infty} s(2\pi\xi)\, d(2\pi\xi) \tag{3.43}$$

$$= \int_{\xi\notin[-w,w]} s(2\pi\xi)\, d(2\pi\xi) = I_s(w) \tag{3.44}$$

$$= I_s\left(\frac{M-1}{2}(2W)^{-1}\right) \tag{3.45}$$

with the higher dimensional analogue

$$(2W)^{-1}\sum_{m>M} s(2\pi z_m) \leq \int_{\xi\notin[-w,w]^D} s(2\pi\xi)\, d(2\pi\xi) = I_s(w) \tag{3.46}$$

$$= I_s\left((2W)^{-1}\frac{M^{1/D}-1}{2}\right). \tag{3.47}$$

The covariance function is bounded by Theorem 3.1. Combining these yields the trace bound, for W, M sufficiently large.

$$\hat{t}+\frac{1}{2}\tau = \sum_n \left(\hat{k}(x_n,x_n) - (2W)^{-D}\sum_{m\leq M} s(2\pi z_m) + \frac{1}{2}|(k(0)-\hat{k}(0))|\right) \tag{3.48}$$

$$= \sum_n \left(k_P(x_n,x_n) - (2W)^{-D}\sum_{m\leq M} s(2\pi z_m) + \frac{1}{2}|(k(0)-k_P(0))|\right) \tag{3.49}$$

$$= \sum_n \frac{1}{2}|(k(0)-k_P(0))| + (2W)^{-D}\sum_{m>M} s(2\pi z_m) \tag{3.50}$$

$$\leq N\left(\frac{1}{2}|(k(0)-k_P(0))| + I_s\left((2W)^{-1}\frac{M^{\frac{1}{D}}-1}{2}\right)\right) \tag{3.51}$$

$$\leq N\left(\frac{1}{2}\beta_1(2W)^{-(1+\kappa_1)D} + \beta_2(M^{1/D}(2W)^{-1})^{-\kappa_2 D}\right) \tag{3.52}$$

Equation (3.50) follows since $\sum_{m\in\mathbb{Z}} s(2\pi z_m) = k_P(0)$, and β_1, β_2 are suitable constants.

For convergence with M only, we can set W to grow with M, say $W = W_0 M^{p/D}$.

$$\hat{t} + \frac{|\tau|}{2} \leq N \left(\frac{1}{2} \beta_1 (2W_0)^{-(1+\kappa_1)D} M^{-(1+\kappa_1)p} + \beta_2 M^{-\kappa_2(1-p)} (2W_0)^{\kappa_2 D} \right) \tag{3.53}$$

Increasing W reduces the aliasing distortion, but it packs the frequencies closer together, which reduces the rate of decay of the Fourier series.

We can optimise the trade-off. Assuming $\kappa_2 > 1$, the optimal trade-off is when both rates are equal, where

$$p = \frac{\kappa_2 - 1}{\kappa_1 + \kappa_2} \tag{3.54}$$

yielding

$$\hat{t} + \frac{1}{2}\tau \leq (\beta_1 (2W_0)^{-(1+\kappa_1)D} + \beta_2 (2W_0)^{\kappa_2 D}) N M^{-\kappa_2 \frac{\kappa_1+1}{\kappa_1+\kappa_2}} \tag{3.55}$$

as required.

For the case where the spectral density does not decay monotonically, we need only make the following alteration: introduce some monotone $s'(\xi)$ which bounds $s(\xi)$ from above, with the same asymptotic rate of decay.

Now, consider the case where both k and s have exponentially bounded tails. Then, again, optimally p is set so that the rates are the same, which means the final bound is in $\mathcal{O}(Ne^{-M^{1/D}})$. □

Example 3.3 (Subsampled AFS with a Matérn-ν covariance function.) The covariance function itself decays faster than any polynomial, which we can identify with $\kappa_1 \to \infty$, and the spectral density decays polynomially with rate $2\nu + D$. Then the spectral mass in the tail will decay with rate $\kappa_2 D = 2\nu$ (due to integrating over D dimensions). So altogether it appears

$$\hat{t} + \frac{1}{2}\tau \in \mathcal{O}(NM^{-\frac{2\nu}{D}\frac{D\kappa_1+D}{D\kappa_1+2\nu}}) \tag{3.56}$$

but since D and ν are finite and $\kappa_1 \to \infty$, we have

$$\hat{t} + \frac{1}{2}\tau \in \mathcal{O}(NM^{-\frac{2\nu}{D}}). \tag{3.57}$$

In particular, applying Corollary 2.2 yields convergence with

$$M \in \mathcal{O}(N^{\frac{D}{2\nu}}). \tag{3.58}$$

Example 3.4 (Subsampled AFS with a squared exponential covariance function.) In this case, both the covariance function and the spectral density have exponential tails with $\kappa_2 = 1$. So it immediately follows that

$$\hat{t} + \frac{1}{2}\tau \in \mathcal{O}(Ne^{-M^{\frac{1}{D}}}) \tag{3.59}$$

and hence by Corollary 2.3,

$$M \in \mathcal{O}(\log^D N). \tag{3.60}$$

3.2.6 Recall that these are the same bounds as in Examples 2.20 and 2.21 which restricted isotropic covariance functions to a compact homogeneous Riemannian manifold *without* any approximations. Recall also that the bound for the squared exponential covariance function is the same as with inducing points, and the bound for the Matérn covariance function is lower. But the number of features required in practice really does grow exponentially in D – it does not cap at N.

3.2.7 Also, unlike Examples 2.20 and 2.21, this bound is of the type in Equation (3.17); this difference will not be significant if W is set sufficiently large that the discrepancy between $\widehat{\mathcal{L}}$ and $\mathcal{L}$ does not significantly impact learning.

3.2.8 For this to be the case, W should be set large enough to be well into the tail of k; then approximation error will be largest near the boundaries within $[-W, W]$ (Theorem 3.1, and see the bottom panel of Figure 3.3). In practice, if the lengthscales are much lower than W_x, it will suffice to set W a little larger than W_x.

3.2.9 For predictions, we know that they will be good if $\hat{k} \approx k$, which should be true for any pair of points $(x_{*,1}, x_{*,2}) \in [-W/2, W/2]^{2D}$ when W is set suitably large for good training, at least sufficiently far from the boundaries.

3.3 Numerically estimated coefficients

3.3.1 In many cases, the spectral density is not available in closed form. For example, consider the case where a covariance function is constructed as the product of standard covariance functions. Even if the component covariance functions have spectral densities available in closed form, the product's spectral density will be the

convolution of these, which generally will not be available in closed form. We could use the composition results of Section 2.3.1, but these will be highly feature inefficient in general. In particular, they scale up the effective dimensionality by the number of components, which is problematic due to the exponential scaling of the computational complexity with the dimensionality.

3.3.2 Of course, it is not s itself which we are interested in approximating, but, focusing on the 1D case, $s_{T,m} = s_T(2\pi m(2W)^{-1})$ for $m \in [-(M-1)/2{:}(M-1)/2], M$ odd, which are the lowest frequency terms of the Fourier series of k_{TP} – the covariance function truncated to distances in $[-W, W]$ and periodised.

3.3.3 A natural choice is to approximate these using the discrete Fourier transform (DFT) of regularly sampled points in the input space. To simplify the notation, let the frequency spacing be $b = 2W/M^{1/D}$; then the DFT is defined as

$$\hat{s}_{T,j_{1:D}} = b^D \sum_{m_{1:D} \in \{-\frac{M^{1/D}-1}{2}:\frac{M^{1/D}-1}{2}\}^D} k\left(m_{1:D} b\right) e^{-i2\pi j_{1:D}^\top m_{1:D} b}. \tag{3.61}$$

3.3.4 This can be computed with at most $\mathcal{O}(M^2)$ operations. Of course, this is a straightforward Riemann sum approximation of the true FS coefficients of k_{TP}, but one might reasonably be concerned about the quality of the approximation, since at higher frequencies the period of the oscillations approaches the sample spacing. Figure 3.4 illustrates the approximation.

3.3.5 The error introduced by this approximation can be characterised using the following result from the signal processing literature.

Theorem 3.5 (Covariance function approximation error for AFS DFT.) *(Epstein, 2005, Theorem 3.4 and Corollary 2.2; Lorentz, 1960) Let $\hat{s}_{T,m}$ be defined as in Equation (3.61). Let k_{TP} be the $2W$ periodisation of k_T, and continuous. Then let $\hat{k}$ be the continuous reconstruction, treating $\hat{s}_{T,m}$ as a finite Fourier series (Equation (3.22)). Take any $\varepsilon > 0$.*

If k_T has $q \geq 0$ continuous derivatives, and the q^{th} derivative is Hölder continuous, then

$$|k_T(x) - \hat{k}(x)| \in \mathcal{O}((2W)^D M^{-(1-\varepsilon)\frac{q}{D}}).$$

If k_T has an analytic extension to the strip $\mathcal{S}_a \subset \mathbb{C}$ defined by

$$\mathcal{S}_a = \{z \in \mathbb{C} \mid Im(z) \leq a\}$$

then

$$|k_T(x) - \hat{k}(x)| \in \mathcal{O}((2W)^D e^{-D(1-\varepsilon)M^{\frac{1}{D}}}).$$

Proposition 3.6 (Regularity and spectral decay.) *(Vetterli et al, 2012, Section 4.4.5) Let $s = \mathcal{F}k$. If k has q continuous derivatives, then $s(\xi) \in \mathcal{O}(||\xi||^{-(q+D)})$.*

3.3.6 If $s(\xi) \in \mathcal{O}(||\xi||^{-(q+D)})$, then it also follows that $I_s(\xi) \in \mathcal{O}(||\xi||^{-q})$. However, the converse to Proposition 3.6 is not true; the spectral density may decay faster even if k has only q continuous derivatives. This happens with the Matérn covariance function, for example.

Theorem 3.7 (Convergence for AFS DFT.) *Assume that k has q continuous derivatives. Let $\hat{k}$ be the reconstruction (Equation (3.22)) of the FS estimated by Equation (3.61). Then using AFS features (Paragraph 3.1.29), for any $\varepsilon > 0$,*

$$\hat{t} + \frac{|\tau|}{2} \in \mathcal{O}(NM^{-(1-\varepsilon)qD} \log^D M^{\frac{1}{D}}).$$

If instead k has an analytic extension to $\mathcal{S}_a$ for some $a > 0$, then

$$\hat{t} + \frac{|\tau|}{2} \in \mathcal{O}(Ne^{-(1-\varepsilon)M^{\frac{1}{D}}}).$$

PROOF I follow the general method of the proof of Theorem 3.2.

Now the approximating covariance $\hat{k}$ is constructed using the DFT frequencies only, so $\hat{t}$ will be zero (since $\hat{\mathsf{K}}_{\mathsf{ff}} = \hat{\mathsf{Q}}_{\mathsf{ff}}$). Then it remains only to bound $|\tau|$.

$$\hat{t} + \frac{|\tau|}{2} = \sum_n \left(\hat{k}(x_n, x_n) - (2W)^{-D} \sum_{m \leq M} \hat{s}_{T,m} + \frac{1}{2}|(k(0) - \hat{k}(0))| \right) \tag{3.62}$$

$$= \frac{1}{2} N |(k(0) - \hat{k}(0))| \tag{3.63}$$

Applying Theorem 3.5, the result follows. □

3.3.7 **Comparison to Section 3.2** Since it is assumed that $W > W_x$, there is no further need to increase W in the DFT case. Hence, the decay rate of the covariance function is not important, which is comparable to the setting $\kappa_1 \to \infty$. In general,

the DFT approximation introduces a relatively small $\log M$ factor to the convergence rate. In the analytic case, this is the only discrepancy.

3.3.8 If k is q times continuously differentiable, then Proposition 3.6 guarantees that $I_s(||\xi||) \in \mathcal{O}(||\xi||^{-qD})$, which would correspond to Theorem 3.2 with $\kappa_2 = q$. But the decay rate may be faster than this, in which case Theorem 3.2 provides better convergence rates, though it is not clear if this is due to an essential limitation of the DFT approximation or due to slack in the bound. A slight improvement could be made by using more evaluation points in the DFT.

Example 3.8 (AFS DFT with a Matérn-ν covariance function.) We have $q = \lfloor \nu \rfloor$.

$$\hat{t} + \frac{1}{2}|\tau| \in \mathcal{O}(NM^{-(1-\varepsilon)\frac{\lfloor \nu \rfloor}{D}}) \tag{3.64}$$

Then applying Corollary 2.2, for any $\varepsilon > 0$,

$$M \in \mathcal{O}(N^{\frac{D}{(1-\varepsilon)\lfloor \nu \rfloor}}). \tag{3.65}$$

Example 3.9 (AFS DFT with a squared exponential covariance function) Immediately from Theorem 3.7,

$$\hat{t} + \frac{1}{2}|\tau| \in \mathcal{O}(Ne^{-(1-\varepsilon)M^{\frac{1}{D}}}) \tag{3.66}$$

and hence by Corollary 2.3,

$$M \in \mathcal{O}(\log^D N). \tag{3.67}$$

3.4 Selecting the frequencies in practice

3.4.0.1 The preceding section set out the theoretical details of the approximations. One major issue is that the number of inducing frequencies increases exponentially in D. In Section 3.4.1, I set out some pragmatic modifications which slightly reduce the number of frequencies to improve speed. Then, in Section 3.4.2, I transform the approximating features to their real-valued versions, which is more convenient for implementation.

3.4.1 Trimming the frequencies

3.4.1.1 **Drop every other frequency** From Equations (3.28) and (3.32), if we space the frequencies by W^{-1} instead of $(2W)^{-1}$, the closest aliases of the covariance function will be a distance W from the origin. If $W \approx W_x$ then the covariance between the further apart data points will experience severe error. However, if the covariance function has fast decay, for example because learnt lengthscales are much smaller than W_x, then the error becomes highly localised near the boundaries (Figure 3.5, bottom row).

3.4.1.2 This will reduce the number of features required by 2^D. In practice, by setting W a little larger than W_x, say $W^{-1} = 0.95W_x^{-1}$, the error has limited impact on K_{ff}. Then the reduction is around $(2 \times 0.95)^D$. In spatial regression settings, it is likely that the lengthscales would be much smaller than the width of the dataset; in practice one could check for dimensions in the data which are likely to have large lengthscales, for example by visualising slices of the data, or training a test model on a small subset of the data, before deciding whether to use this additional approximation.

3.4.1.3 **Impact on the log marginal likelihood estimate** To verify that this scheme can approximate the log marginal likelihood well, I sample inputs either from a uniform distribution or a Gaussian distribution, sample y from a GP, and calculate $\mathcal{L}''$ with the parameters fixed at the generating parameters, for a range of settings of M and W (Figure 3.6). The error floor of the approximation drops as W gets much larger than W_x, with the impact of dropping every other frequency much less severe in the case of Gaussian inputs, since there the points are concentrated near the centre.

3.4.1.4 With a Matérn-$^3/_2$ covariance function, the covariance function decays much faster than the spectral density, so the impact of aliasing distortion is generally much smaller than the impact of low M (compare the uniform input columns of Figure 3.6b to Figure 3.6a).

3.4.1.5 **The equivalent covariance function** In choosing to discard half of the frequencies, we could choose to keep the even frequencies (which includes the constant component) or the odd frequencies. Choosing the even frequencies is exactly equivalent to changing the value of W, but choosing the odd frequencies introduces a quarter period antisymmetry in $\hat{k}$ (Figure 3.5).

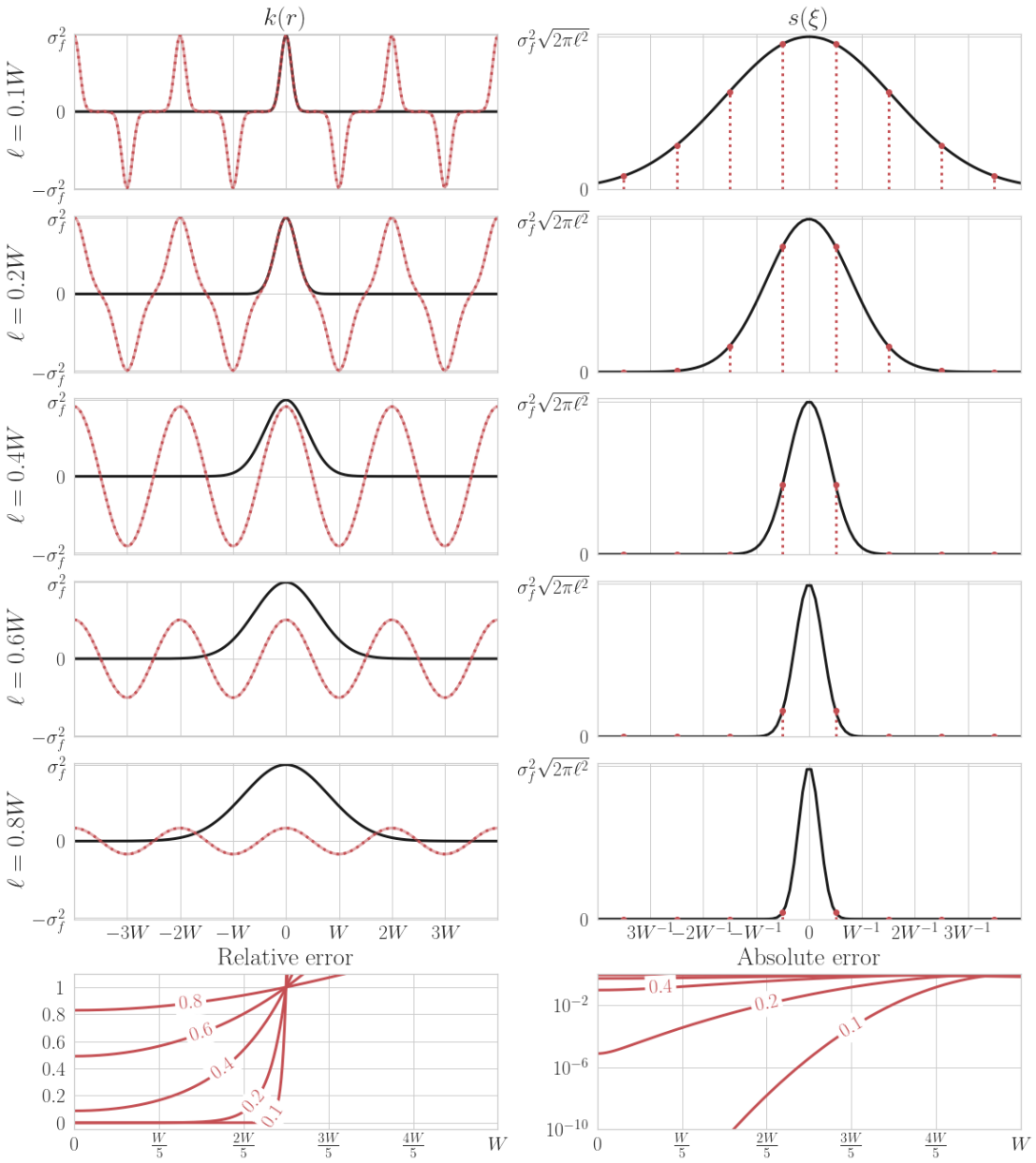
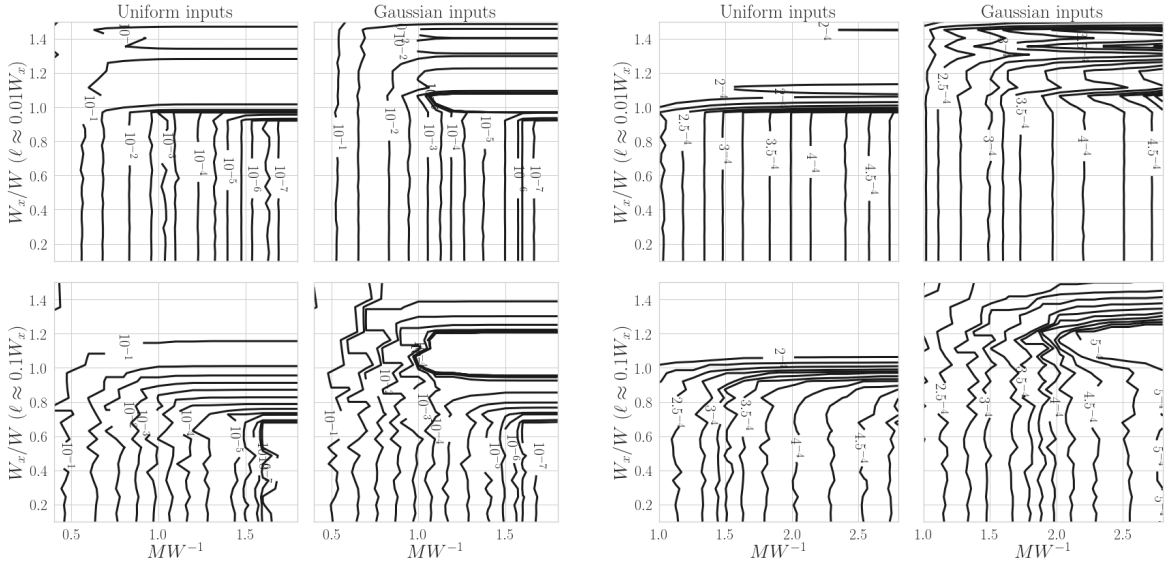


Figure 3.5 The covariance function approximation for subsampled AFS, odd frequencies only. The original covariance function and spectral density are plotted in black; $k_{P'}$ and its FS are in red dashed. The continuous red lines are the low frequency ($M = 17$) approximation. The error $k_{P'} - k$ is shown in the bottom row. The covariance function is squared exponential with scale ℓ, and the contours are marked with ℓ/W. Although the relative error unavoidably gets large by $W/2$, when $\ell \ll W$, $k(W/2)$ is almost zero, and the absolute error is kept low until near W. Compare with Figure 3.3.



(a) SE covariance function (b) Matérn-$^3/_2$ covariance function

Figure 3.6 Gap between the log marginal likelihood and the training objective ($\mathcal{L} - \mathcal{L}''$) for different settings of M, W. The measurements are sampled from the same GP as the approximating model, with a lengthscale of ℓ. W_x is calculated for the sampled inputs.

3.4.1.6 In the 1D case, this is shown as follows.

$$\widehat{k}_{P'}(r) = \widehat{k}_P(r) - \widehat{k}_P(r - W) \tag{3.68}$$

$$= \frac{1}{2W} \sum_m s(2\pi z_m) e^{i2\pi m (2W)^{-1} r} - \frac{1}{2W} \sum_m s(2\pi z_m) e^{i2\pi m (2W)^{-1} (r - W)} \tag{3.69}$$

$$= \frac{1}{2W} \sum_m (1 - e^{-i\pi m}) s(2\pi z_m) e^{i2\pi m (2W)^{-1} r} \tag{3.70}$$

$$= \frac{1}{W} \sum_{m \text{ odd}} s(2\pi z_m) e^{i2\pi m (2W)^{-1} r} \tag{3.71}$$

$$= \frac{1}{W} \sum_m s(2\pi z_{2m+1}) e^{i2\pi (m + \frac{1}{2}) W^{-1} r} \tag{3.72}$$

3.4.1.7 For higher dimensions, the sign of the extra aliases will depend on their location in the following way. Let $\mathsf{e}_j, j \in \{1{:}2^D\}$ be the D dimensional vector whose elements are given by the binary representation of j. Then we have one extra set of aliases for each j, which is shifted by $W\mathsf{e}_j$, and the sign is equal to $(-1)^{||\mathsf{e}_j||_1}$ (that is, it is

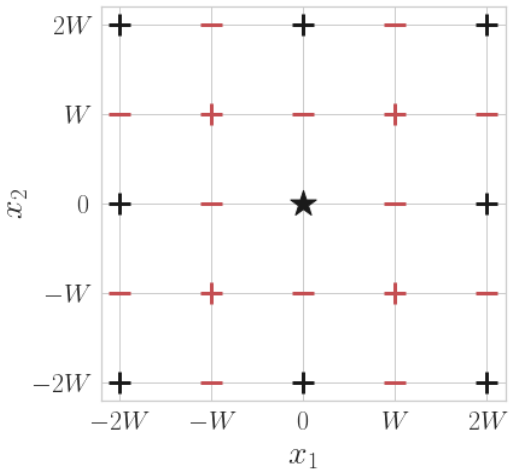


Figure 3.7 The pattern of signs of aliases generated by the odd AFS approximation. The ⋆ shows the original centre, + shows aliases present in the original FS, +, − show the positive and negative new aliases respectively.

positive if e_j has an even number of 1s, and negative otherwise). This pattern is illustrated for the 2D case in Figure 3.7.

3.4.1.8 To show the result, note that

$$1+\sum_{j=1}^{2^D}(-1)^{||\mathsf{e}_j||_1}e^{-i\pi\sum_d \mathsf{e}_{jd}m_d} = 1-\sum_d e^{-i\pi m_d}+\sum_{d,d'} e^{i2\pi(m_d+m_{d'})}+\cdots+e^{-i\pi\sum_d m_d}$$
$$=\prod_d(1-e^{-i2\pi m_d}). \tag{3.73}$$

3.4.1.9 Then the multidimensional result follows by a similar method to the 1D case.

$$\hat{k}_{P'}(r) = \hat{k}_P(r)+\Sigma_{j=1}^{2^D}(-1)^{||\mathsf{e}_j||_1}\hat{k}_P(r-W\mathsf{e}_j) \tag{3.74}$$
$$= \frac{1}{(2W)^D}\sum_{m_{1:D}} s(2\pi z_{m_{1:D}})e^{i2\pi\sum_d m_d(2W)^{-1}r_d} \tag{3.75}$$
$$+\frac{1}{(2W)^D}\sum_{m_{1:D}} s(2\pi z_{m_{1:D}})\sum_{j=1}^{2^D}(-1)^{||\mathsf{e}_j||_1}e^{i2\pi\sum_d m_d(2W)^{-1}(r_d-W\mathsf{e}_{jd})}$$
$$= \frac{1}{(2W)^D}\sum_{m_{1:D}}\prod_d(1-e^{-i\pi m_d})s(2\pi z_m)e^{i2\pi\sum_d m_d(2W)^{-1}r_d} \tag{3.76}$$
$$= W^{-D}\sum_{m_{1:D}\text{ each odd}} s(2\pi z_m)e^{i2\pi\sum_d m_d(2W)^{-1}r_d} \tag{3.77}$$

$$= W^{-D} \sum_{m_{1:D}} s(2\pi z_{2m_1+1,\ldots,2m_D+1}) e^{i2\pi \sum_d (m_d+\frac{1}{2}) W^{-1} r_d} \tag{3.78}$$

3.4.1.10 It does not matter much whether the even or the odd frequencies are used; I proceed with the odd frequencies due to the symmetry it introduces in z and the partial cancellation of the aliasing errors when k is positive everywhere, though with exponentially decaying covariance functions, the benefit of this is negligible.

3.4.1.11 **Specialising by dimension** For convenience, Sections 3.2 and 3.3 proceeded by treating every dimension identically – using $M^{1/D}$ frequencies and spacing them by $(2W)^{-1}$. It should be clear from a quick examination of those sections that variable $W_d, d \in \{1{:}D\}$ will not essentially change the main results. For practical purposes, it is best to set

$$W_{xd} = \max_n x_{nd} - \min_n x_{nd} \tag{3.79}$$

and then set W_d a little larger than W_{xd}; this will lead to some additional small savings where the data has different ranges in each dimension.

3.4.1.12 **Trimming the boundaries** We can consider a more general setting than varying M_d over d. We can choose any subset of the frequencies on the lattice, rather than cubic boundaries; convergence still follows if asymptotically all of them are included. However, note that the transformation to real-valued features follows only when the subset of frequencies chosen is antisymmetric about every axis; see Section 3.4.2.

3.4.1.13 The optimal choice is to select the frequencies with the highest spectral density. For many covariance functions, the spectral density decays with distance from the origin, so choosing the lowest frequencies is a reasonable choice. This mimics exactly the frequencies chosen for the 1D RKHS Fourier series features by Hensman et al (2017).

3.4.1.14 However, for covariance functions such as squared exponential or Matérn in higher dimensions, the spectral density in higher dimensions has ellipsoidal level sets (Paragraph 1.3.0.6). Then it is more sensible to pick the frequencies which are contained in that ellipsoid (Figure 3.8). However, note that to maintain precomputability, z must be fixed throughout learning, else K_{uf} and hence B will have to be recalculated. But the relative sizes of the axes of the ellipsoid are given by the inverse lengthscales.

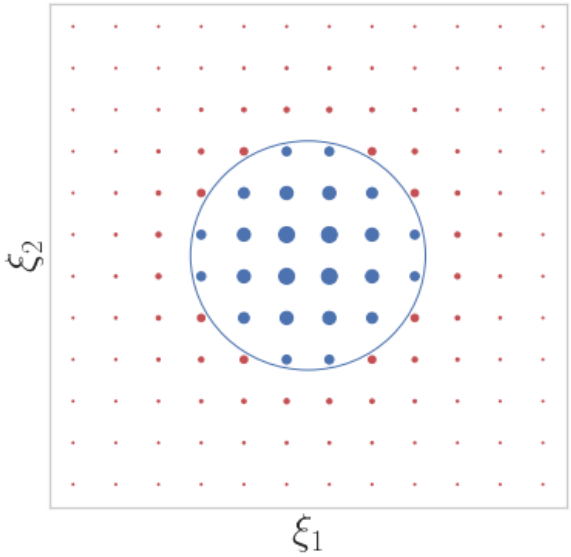


Figure 3.8 Selecting the best frequencies in 2D for a squared exponential covariance function. Each dot's size is an affine function of the spectral density at that location; thresholding $||z_m||^2$ is selecting the blue points within the circle.

3.4.1.15 A reasonable choice which will work well is to pick frequencies within a sphere. This is optimal on average if the lengthscales are assumed to be independently and identically distributed across dimensions. Note from the figure that the improvement in frequency selection due to this method may be very small.

3.4.2 Real-valued features

3.4.2.1 So far, I have used complex-valued features. This is very convenient both for presentational purposes, and for the proofs. However, it is more convenient in implementation to use real-valued features, since the automatic differentiation libraries required for the learning algorithms have poor support for complex variables. It should be clear from the theory of Fourier series that one can replace the complex exponentials with sines and cosines, but for completeness I provide a short technical argument.

3.4.2.2 Recall Proposition 2.7: applying an invertible linear transformation to the features does not change inference or learning. Hence, I show that replacing the complex exponentials with suitable products of cosines and sines evaluated at the same frequencies is an invertible linear transformation.

3.4.2.3 To simplify the presentation, use multidimensional indices. Let the complex valued features be given by inner products with $\varphi_{m_{1:D}}$ which is the feature corresponding to the frequency $z_{m_{1:D}}$, and let z be antisymmetric in every axis (that is, $z_{-m_1,m_2,\ldots,m_D} = -z_{m_1,m_2,\ldots,m_D}$, and similarly for every other $d \in \{1{:}D\}$). For example, the rectangular and spherical lattices have this antisymmetry. The purpose of this is that features which form conjugate pairs are always included, which allows them to be transformed into real-valued features.

3.4.2.4 Now, note that each $\varphi_{m_{1:D}}$ is proportional to a complex exponential (Paragraph 3.1.29). In general, this yields the expansion

$$\varphi_{m_{1:D}}(x) \propto e^{i2\pi x^\top z_m} = \prod_{d=1}^{D} e^{i2\pi x_d z_{m_d}} = \prod_{d=1}^{D} (\cos(2\pi x_d z_{m_d}) + i \sin(2\pi x_d z_{m_d})) \tag{3.80}$$

$$= \sum_{j=0}^{D} (-i)^j \sum_{\substack{S \subseteq \{1:D\} \\ |S|=j}} \prod_{d \in S} \sin 2\pi x_d z_{m_d} \prod_{d \notin S} \cos 2\pi x_d z_{m_d} \tag{3.81}$$

so let $\varphi'_{m_{1:D},S}$ for each $m_d \geq 0$ only, and every $S \subseteq \{1:D\}$, be defined by

$$\varphi'_{m_{1:D},S} \propto \prod_{d \in S} \sqrt{2} \sin 2\pi z_{m_d} x_d \prod_{d \notin S} \sqrt{2} \cos 2\pi z_{m_d} x_d \tag{3.82}$$

with the same constant of proportionality as φ. Then

$$\varphi_{m_{1:D}} = 2^{-\frac{D}{2}} \sum_S i^{|S|} \prod_{d \in S} \operatorname{sgn}(m_d) \varphi'_{|m_1|,\ldots,|m_D|,S}. \tag{3.83}$$

3.4.2.5 The inverse transform also exists: consider the matrix T which transforms from the real to the complex parameterisation. From Equation (3.83), the elements of this matrix are

$$\mathsf{T}_{m_{1:D},(m'_{1:D},S)} = 2^{-\frac{D}{2}} i^{|S|} \prod_{d \in S} \operatorname{sgn}(m_d) \prod_{d=1}^{D} \delta_{m_d - |m'_d|}. \tag{3.84}$$

3.4.2.6 In fact, the matrix is orthonormal. The rows are clearly independent when the magnitudes of $m_{1:D}$ and $m'_{1:D}$ differ, so consider two rows where the magnitudes of the frequencies are the same. Then, denoting the row $\mathsf{T}_{m_{1:D}}$, the inner product of

two rows is

$$\mathsf{T}^*_{m_{1:D}}\mathsf{T}_{m''_{1:D}} = 2^{-D}\sum_S \underbrace{i^{|S|}(-i)^{|S|}}_{1}\sum_S\prod_{d\in S}\operatorname{sgn}(m_{1:D})\operatorname{sgn}(m''_{1:D}) \tag{3.85}$$

$$= 2^{-D}\prod_{d=1}^{D}(\operatorname{sgn}(m_{1:D}) + \operatorname{sgn}(m''_{1:D})) \tag{3.86}$$

$$= \cancel{2^{-D}}\cancel{2^D}\delta_{\operatorname{sgn}(m_{1:D})+\operatorname{sgn}(m''_{1:D})}. \tag{3.87}$$

3.4.2.7 Then not only does T^{-1} exist, but $\mathsf{T}^{-1} = \mathsf{T}^*$

3.4.2.8 **Efficient construction** These are features with a product structure in both the complex and real case. For a rectangular lattice of frequencies, the K_{uf} matrix can be generated by first generating 1D K_{uf} matrices with the cosine and sine features for non-negative frequencies, then taking the Khatri-Rao product (denoted $\star$) over dimensions.

3.4.2.9 To use a subset of the frequencies, we can discard the corresponding rows of K_{uf}. For example, if we want to use a spherical lattice as in Figure 3.8 then we can discard all rows for which $||z_m||^2$ exceeds a threshold. To avoid an explosion in memory cost, we can enforce this one dimension at a time. The procedure is as follows.

1. Construct $\mathsf{K}_{uf,d}$ using z_{m_d}, x_{nd}.
2. Given the running total $\mathsf{K}_{uf,1:d-1}$, $\tilde{\mathsf{K}}_{uf,1:d} = \mathsf{K}_{uf,1:d-1} \star \mathsf{K}_{uf,d}$.
3. Form $\mathsf{K}_{uf,1:d}$ by discarding all rows of $\tilde{\mathsf{K}}_{uf,1:d}$ for which $||z_m||^2$ exceeds the threshold.

3.5 Experimental evaluation

3.5.0.1 In this section, I evaluate the FS variational approximation empirically. I show that it leads to faster learning for large datasets in low dimensions, with a particular focus on spatial modelling. Amongst the other precomputable variational approximations, I compare principally against the RKHS FS of Hensman et al (2017); for the real-world datasets, I also consider B-Spline features (Cunningham et al, 2023).

3.5.0.2 For a conventional sparse baseline, I use inducing points sampled according to the scheme of Burt et al (2020a), alternating between sampling the inducing inputs and optimising the hyperparameters. For the synthetic experiments, I also use inducing points initialised using the k-means algorithm and kept fixed. For the real-world spatial datasets, I also tested SKI (Wilson et al, 2016), due to its reputation for fast performance, and its fairly robust implementation.

3.5.0.3 Further experimental details are in Appendix C.1.

3.5.1 Synthetic datasets

3.5.1.1 First I consider a synthetic setting where the data is indeed sampled from a GP, using either a squared exponential (Figure 3.9) or Matérn-$^5/_2$ (Figure 3.10) covariance function, and compare the speed of variational methods in 1 and 2 dimensions. This is assessed by learning the parameters using LBFGS for a range of values of M, and plotting the empirical gap between $\mathcal{L}$ and the optimised variational objective as a function of time (top row). As a check, the gap to $\mathcal{L}$ evaluated at the groundtruth parameters is also plotted (middle row). An approximation yields faster learning if its curve lies lower and to the left.

3.5.1.2 I use a small ($N = 10\,000$) dataset in order to be able to easily evaluate the log marginal likelihood at the learnt hyperparameters. Where possible, I use the same (squared exponential) model for learning; for RKHS FS, I use a Matérn-$^5/_2$ covariance function in 1D, and a tensor product of Matérn-$^5/_2$ covariance functions in 2D, since this is the best approximation to a squared exponential covariance function which is supported.

3.5.1.3 Note the logarithmic time axis – because the $O(NM^2)$ work is done only once, all the precomputable methods are much faster (8–16×) to run than inducing points. This is solely due to the precomputable structure, in the sense that performance for fixed M is similar between AFS and inducing points (bottom row).

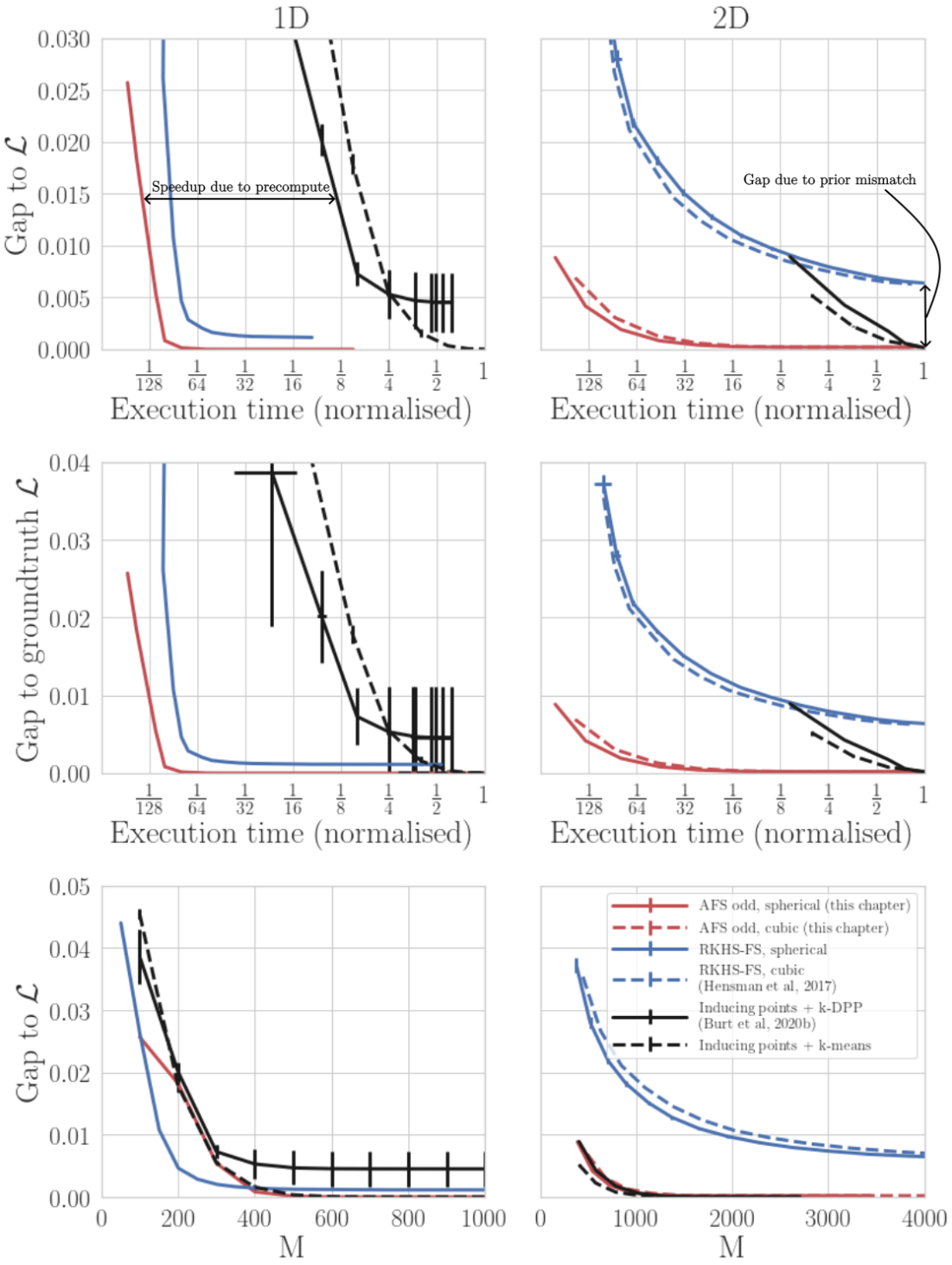


Figure 3.9 Fourier series approximations compared with inducing points, for data sampled from a GP with an SE covariance function. Curves lower and to the left are better performing. The top and bottom rows show $\mathcal{L} - \mathcal{L}''$ at the learnt hyperparameters as a function of learning time and number of features respectively. In the middle row, $\mathcal{L}$ is evaluated at the groundtruth parameters, while $\mathcal{L}''$ is evaluated at the learnt parameters. The bottom row shows feature efficiency, while the upper rows show computational efficiency. The dashed lines use the full cubic lattice; the solid lines use a spherical lattice (Section 3.4). Precomputable approximations yield speedups on the order of 8× (1D) to 16× (2D) (upper rows), though they have the same feature efficiency (bottom row).

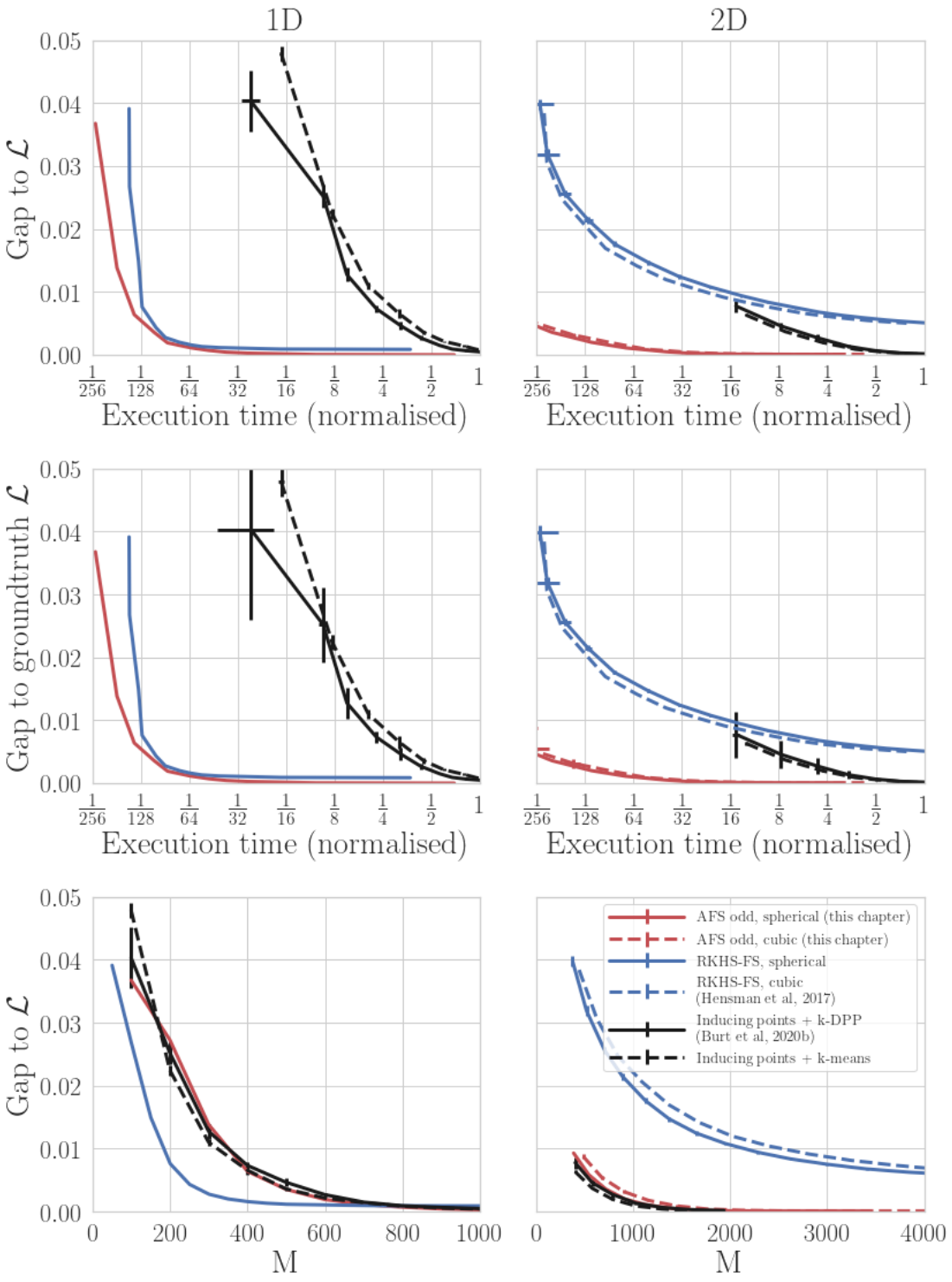


Figure 3.10 As Figure 3.9, but with the data sampled from a Matérn-$^5/_2$ GP. RKHS FS is only compatible with products of lower order Matérn covariance functions, so its model is more closely matching the data generating distribution here than in Figure 3.9. The drop in feature efficiency is therefore far less in higher dimensions.

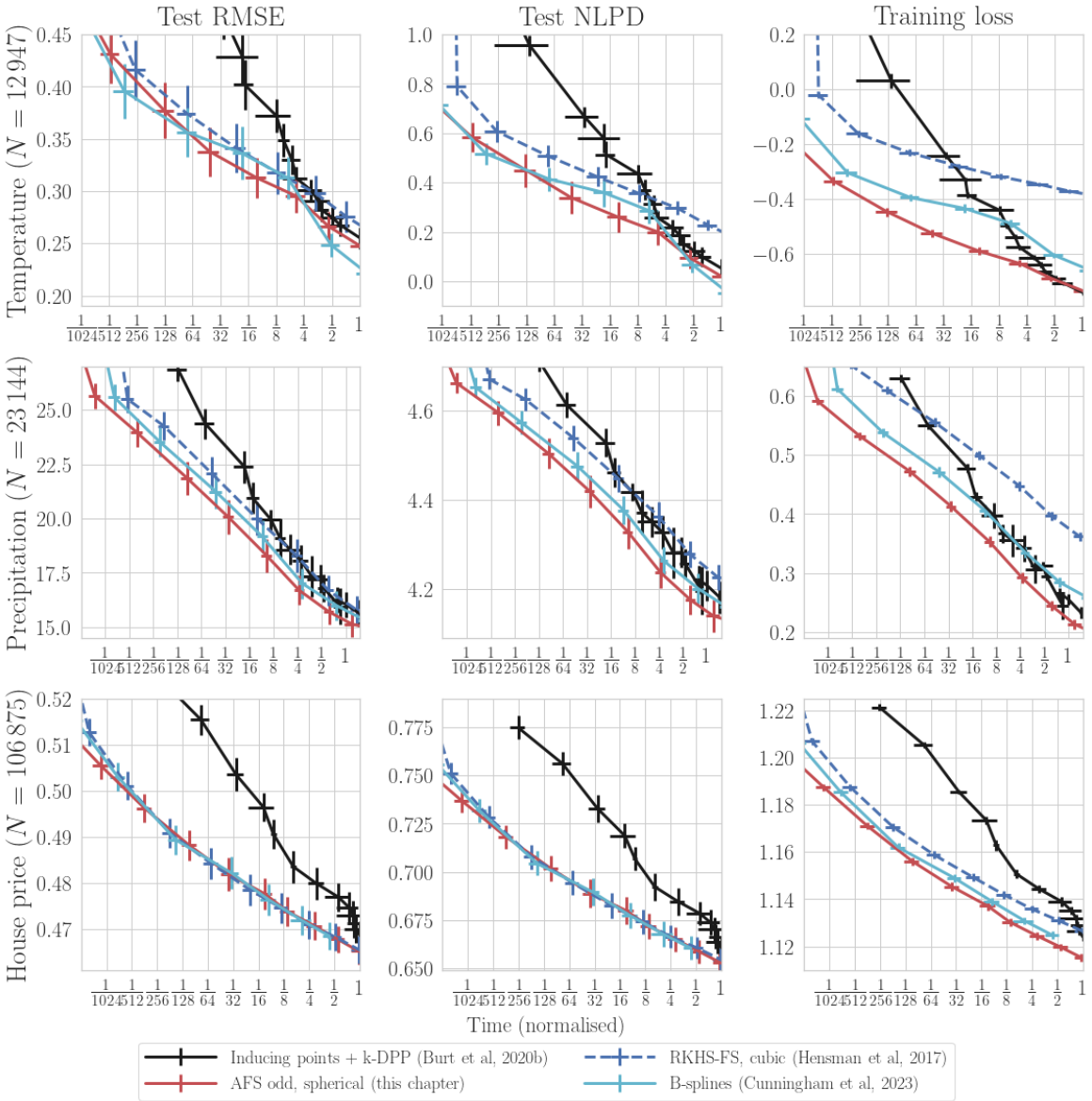


Figure 3.11 Performance curves for real-world datasets of increasing size (the top row is the smallest). Lower and to the left is better. The curves are parameterised by M; to the right, as M becomes very large, all methods take a similar amount of time, since precomputation is no longer advantageous. For moderate error tolerance, all precomputable methods offer a major speedup.

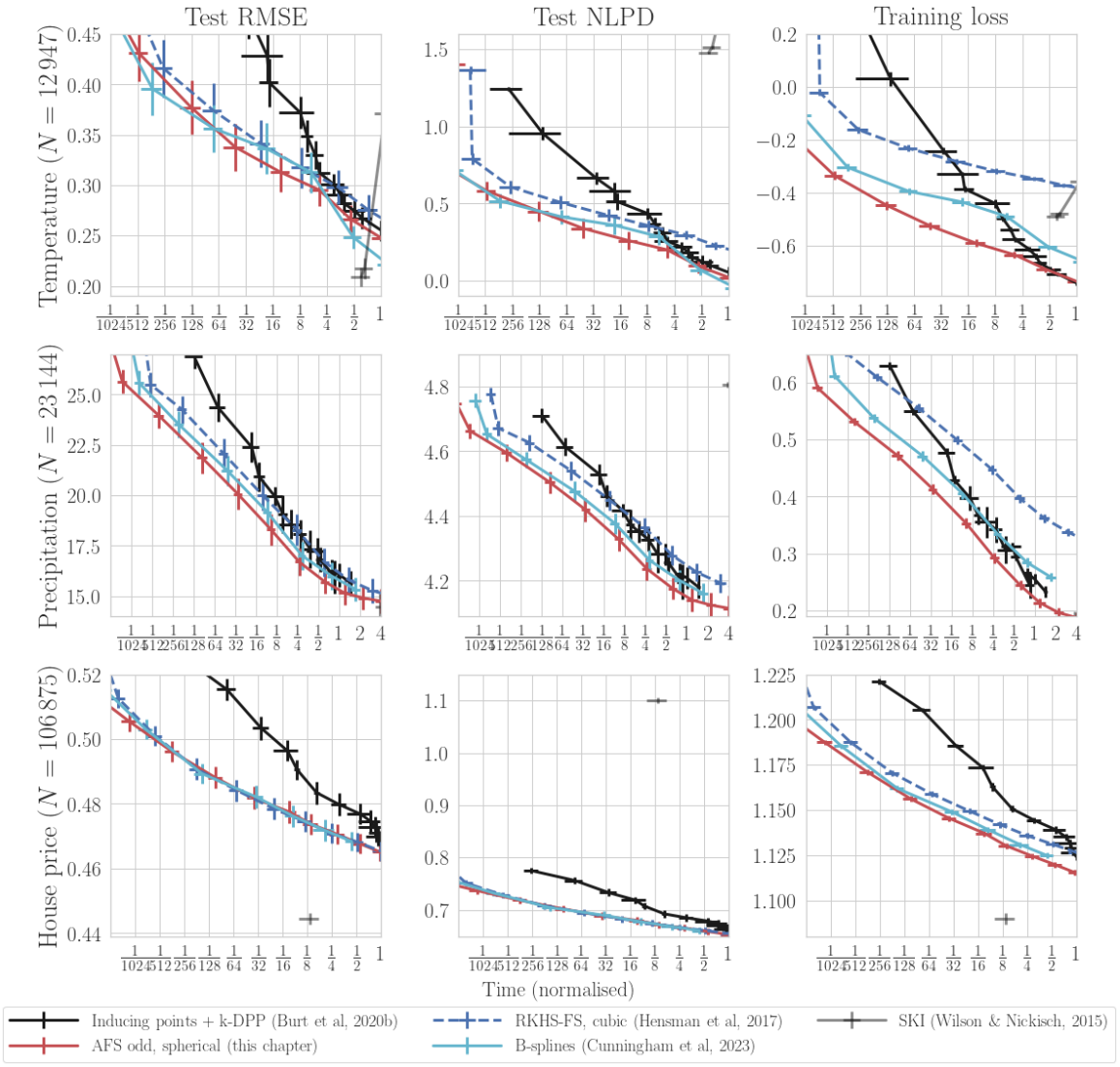


Figure 3.12 As Figure 3.11, but with SKI included. SKI exhibits very favourable and fast predictive mean performance, but its predictive variances are poor, leading to very large NLPD.

3.5.1.4 **Feature efficiency and the prior** Although the setting is highly idealised, the feature efficiency results (bottom row) demonstrate the issue with limited choice of prior. In 1D, the Matérn-$^5/_2$ covariance function matches the prior in Figure 3.10, and is also a good approximation to squared exponential, so the RKHS FS performs similarly to the odd AFS. But in 2D, the product model is a much worse approximation of the groundtruth model.

3.5.1.5 **Impact of trimming the frequencies** Additionally, in the 2D setting, I show the results using both a cubic (dashed lines) and spherical (solid lines) lattice of frequencies (as described in Section 3.4). This is indeed slightly more feature efficient (bottom panels). In practice, I find that the computational overhead involved in selecting better features outweighs the savings from using fewer features for RKHS FS, though not for odd AFS (Figure 3.9, upper and middle right panels).

3.5.2 Real-world datasets

3.5.2.1 I now compare the training objective and test performance on three real-world spatial modelling datasets of increasing size, and of practical interest, as previously introduced in Section 2.5. I plot the root mean squared error (RMSE) and negative log predictive density (NLPD) on the test set along with the training objective and run time in Figures 3.11 and 3.12 using five uniformly random 80/20 train/test splits. For inducing points, I use the method of Burt et al (2020b). The time plotted is normalised per split against inducing points. Further training details are in Appendix C.1.

3.5.2.2 I add SKI to the comparison here, since it is the most widely used alternative to variational methods in this setting. For the variational methods, both the time and performance are implicitly controlled by the number of features M; as M is increased the performance improves and the time increases, that is we move along the curve to the right. This is a very useful property, since we can select M according to our available computational budget and be fairly confident of maximising performance. With SKI, the equivalent parameter is the grid size, and similarly increasing the grid size generally improves performance. However, when the grid size is low, optimisation can take longer, so for example for the temperature dataset, we move to the left as the grid size is increased (Figure 3.12; top row). For the other datasets, I only plot the SKI with the grid size automatically selected by the reference implementation.

3.5.2.3 SKI generally has very good predictive means, leading to low RMSE, and is very fast, but the predictive variances are poor, generally being too low and producing some negative values. Ignoring the negative values, the NLPD is much worse for SKI than for the variational methods, which is highly undesirable in a probabilistic method.

3.5.2.4 I exclude SKI in Figure 3.11 in order to zoom in on the curves for the variational methods. We are interested in the regime where $M \ll N$; as we move to the right and M is similar to N, inducing points will become competitive with the faster methods, since the $O(M^3)$ cost dominates. The odd AFS always performs at least as well as the RKHS FS, and produces a substantially better performance for a given time on the temperature and precipitation datasets, due to a more flexible choice of covariance function. In particular, note that the training objective of odd AFS is substantially lower than RKHS FS on these datasets, but comparable to that of inducing points, which uses the same covariance function.

3.5.2.5 The B-spline features are also limited in the choice of covariance function, but the sparse structure of the covariance matrix leads to a much better performance. Nonetheless, despite involving a dense matrix inverse, odd AFS has a significant advantage on the precipitation dataset (around 25-30% faster typically), and is comparable on the houseprice dataset.

3.5.2.6 Compared to the idealised, synthetic, setting, inducing points become very competitive in the higher resource setting towards the right hand side of each plot, particularly for the smallest dataset (temperature). But for most performance thresholds, we find that fast variational methods offer a substantial improvement, with typically more than a factor of two speedup. For the largest dataset (houseprice, bottom row) for intermediate performance thresholds (the middle region along the vertical axis), speedups in excess of eight times are typical.

3.6 Gridded data

3.6.1 So far in this chapter, I have shown that approximate Fourier series features offer a flexible, precomputable and feature-efficient approximation. In this section, I consider the special case of gridded data, where the rows of K_{uf} become orthonormal, and so the cost per optimisation step is reduced to $\mathcal{O}(M)$. However, there is no

avoiding data points being on the boundary (at $\pm W$ in 1D), so some error is unavoidable.

3.6.2 Assume throughout that the data is on a regular lattice with spacing η_d in dimension $d \in \{1{:}D\}$, with rectangular boundaries. That is,

$$x_{nd} = \alpha_d + n_d \eta_d \quad n_d \in \{1{:}N_d\} \tag{3.88}$$

with $N = \prod_d N_d$. Additionally, assume that the Fourier features are chosen on rectangular lattice with spacing in dimension d of $a_d/N_d\eta_d$ for some $a_d \in \mathbb{Z}$.

Lemma 3.1 (Gridded data yields a diagonal approximation.) *In the gridded setting, with gridded frequencies,* $\mathsf{B} = N\mathsf{I}$*, given* $N_d/a_d \notin \{1{:}M_d - 1\}$ *for any* $d \in \{1{:}D\}$.

PROOF Firstly, consider the case of complex valued features.

$$\begin{aligned}
\mathsf{B}_{mm'} &= \sum_{n_{1:D}} \prod_d e^{-i2\pi(z_{md} - z_{m'd})x_{nd}} && (3.89)\\
&= \prod_d \sum_{n_d} e^{-i2\pi(z_{md} - z_{m'd})x_{nd}} && (3.90)\\
&= e^{-i2\pi \sum_d \alpha_d (z_{md} - z_{m'd})} \prod_d \sum_{n_d} e^{-i2\pi(z_{md} - z_{m'd})\eta_d} && (3.91)\\
&= e^{-i2\pi \sum_d \alpha_d (m_d - m'_d)\frac{a_d}{\eta_d}} \prod_d \sum_{n_d} e^{-i2\pi(m_d - m'_d)\frac{a_d n_d}{N_d}} && (3.92)
\end{aligned}$$

But we have the standard result that

$$\sum_{n_d} e^{-i2\pi(m_d - m'_d)\frac{a_d n_d}{N_d}} = \begin{cases} N_d & \text{if } \frac{a_d(m_d - m_{d'})}{N_d} \in \mathbb{Z} \\ 0 & \text{otherwise} \end{cases}. \tag{3.93}$$

We will be in the first case when $m_d = m'_d$. If $N_d/a_d \notin \{1{:}M_d - 1\}$ for any $d \in \{1{:}D\}$ then for every other pair m_d, m'_d we will be in the second case, which proves the result.

For real valued features, recall the transformation from real to complex features was defined by a unitary matrix T (Equations (3.83) and (3.84)). Then for real features

$$\mathsf{B}' = \mathsf{T}^*\mathsf{B}\mathsf{T} = N\mathsf{T}^*\mathsf{T} = N\mathsf{I}. \tag{3.94}$$

□

3.6.3 The smallest possible frequency spacing is when $a_d = 1$, which means $W = W_x$. This is the spacing of Section 3.4.1; recall that this means the approximation of the covariance between the furthest apart data points will be highly inaccurate. This may not be too severe an issue if the number of datapoints is very large. However, in higher dimensions, the furthest apart points will become an increasingly large fraction of the data. In the synthetic example of Figure 3.13, both the speedup, and the significant error floor in 2D, are illustrated.

3.6.4 One possible heuristic to work around this is to set a_d a little smaller than 1, but still assume that B is diagonal, with the expectation that the error will be small.

3.6.5 A significant limitation is the need for a full grid of data. If only a small number of points are missing, we could pretend the data is gridded, and set $\mathsf{B} = N\mathsf{I}$, knowing that the true value of B differs from this only by a low rank term.

3.6.6 **Partially gridded data** However, it is common in the spatial regression setting to have a mix of gridded and non-gridded data. A first example is where there are a large number of gridded measurements (for example, from a simulator) and a much smaller number of non-gridded measurements. In such a setting, one possibility is to use sequential learning (Bui et al, 2017): Fourier series features can be used to learn from the gridded data, and this can then be updated using the non-gridded data, possibly with different inducing variables.

3.6.7 A second example is where some dimensions of the input are gridded and others are not. For example, a precipitation model might have measurements or simulated values which are gridded in space and time, but might also use the slope of the terrain as an input. This breaks the grid structure, since for each location in space, we only have precipitation for one slope value (the real slope at that location). One option is to use a multi-output model. This will only linearly correct the spatial or spatiotemporal model using the slope, but allows us to leverage the grid structure.

3.7 Further extensions and limitations

3.7.1 **Non-conjugate settings** If the likelihood is not linear and Gaussian, the variational approximation no longer yields a closed form optimal posterior

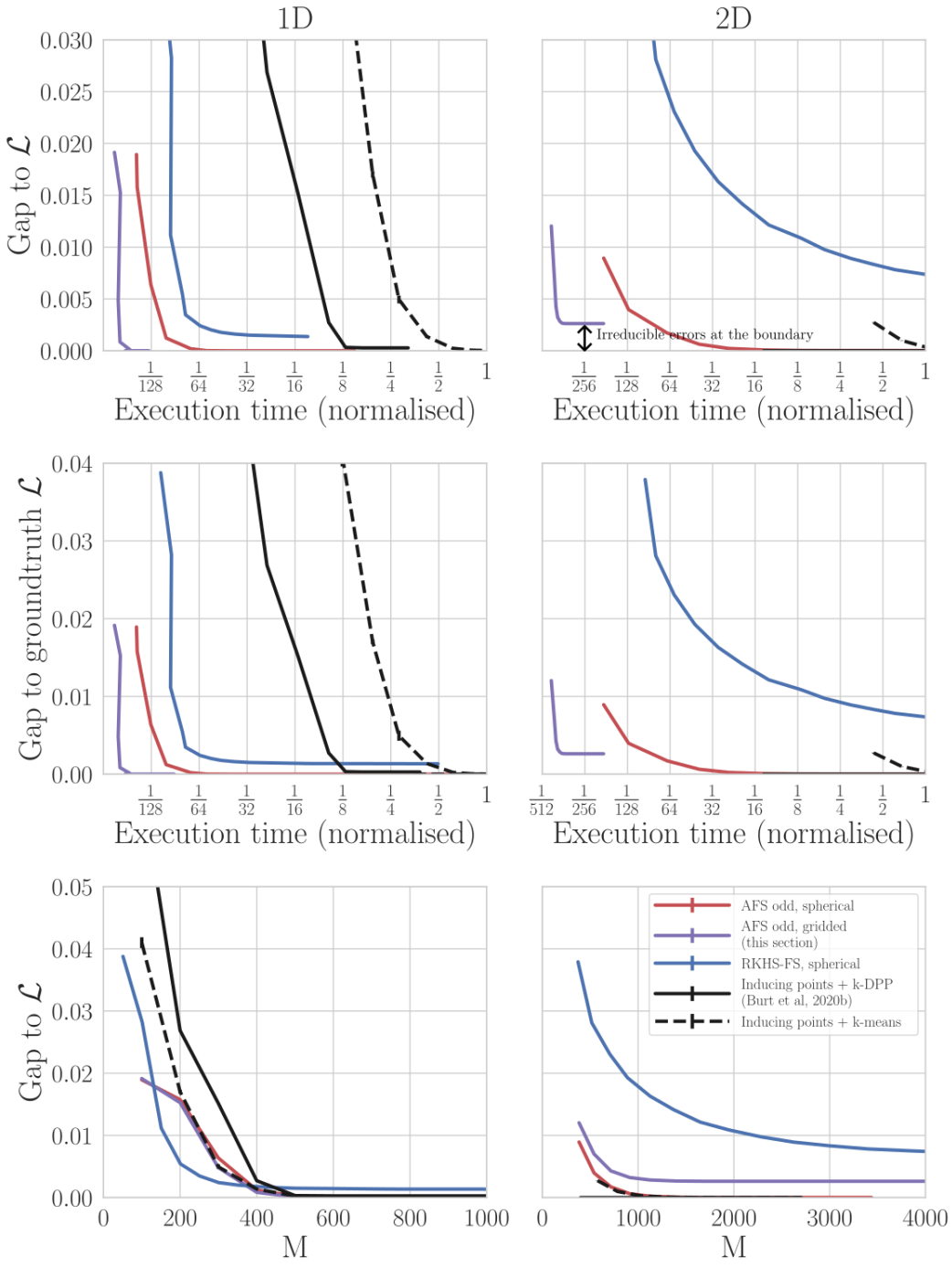


Figure 3.13 Fourier series approximations compared with inducing points, for data sampled from a GP with an SE covariance function and gridded inputs. Curves lower and to the left are better performing. The top and bottom rows show $\mathcal{L} - \mathcal{L}''$ at the learnt hyperparameters as a function of learning time and number of features respectively. In the middle row, $\mathcal{L}$ is evaluated at the groundtruth parameters, while $\mathcal{L}''$ is evaluated at the learnt parameters. The bottom row shows feature efficiency, while the upper rows show computational efficiency. Leveraging grid structure leads to significant speedup, but requires frequency spacing of W^{-1} which leads to a significant error floor.

approximation. As an alternative, one can either learn a parameterised Gaussian distribution for u, or sample from the optimal distribution for u. In either case, computation is dominated by calculating

$$\mathsf{K}_{uf}^{*}\mathsf{K}_{uu}^{-1} \text{ or } \mathsf{K}_{uf}^{*}\mathsf{K}_{uu}^{-1}\mathsf{K}_{uf} \tag{3.95}$$

where in precomputable methods, K_{uf} is fixed but K_{uu} may depend on the covariance function's parameters. To avoid repeating the $O(NM^2)$ computation in each step, let $\mathsf{K}_{uu} = \mathsf{L}\mathsf{L}^{\top}$, and directly update $\mathsf{K}_{uf}^{*}\mathsf{L}^{-\top}$ according to

$$(\mathsf{K}_{uf}^{*}\mathsf{L}_{(\text{new})}^{-\top}) = (\mathsf{K}_{uf}^{*}\mathsf{L}_{(\text{old})}^{-\top})\mathsf{L}_{(\text{old})}^{\top}\mathsf{L}_{(\text{new})}^{-\top}. \tag{3.96}$$

3.7.2 **Periodic covariance functions** It has been assumed throughout that the covariance function has a valid spectral density. If the covariance function is periodic, then clearly it has an exact Fourier series, but usually the period is a learnable parameter, so the FS features are not precomputable.

3.7.3 **Near-periodic covariance functions** Usually, periodic covariance functions are used multiplied by baseband covariance function, such as squared exponential. If the period is fixed, this can be approximated using Proposition 2.15. However, usually, the period is being learnt jointly with the other parameters. In this case, for finite M, W, the spectral density must also be wide enough that the samples do not 'miss' its peaks completely. This can be enforced by placing a lower limit on the lengthscale parameter of the SE component. This is a reasonable restriction – classical signal processing analysis tells us that a function observed for a finite width W_x has its frequencies smeared by about W_x^{-1}.

3.7.4 Forcing the multiplying SE component to have a width at least W_x^{-1} guarantees that the $(2W)^{-1}$ spaced AFS methods of Sections 3.2 and 3.3 and the $\approx 0.95(W)^{-1}$ odd AFS method of Section 3.4 would not miss any peaks.

3.8 Summary

3.8.1 In this chapter, I presented a group of precomputable variational approximations for conjugate GP regression, based on approximate Fourier series (AFS) approximations. These come with some helpful guarantees on the quality of the marginal likelihood

approximation, and perform competitively empirically (learning more than twice as quickly for higher performance thresholds, and in excess of eight times faster for some performance thresholds). Unlike other precomputable methods, they can be used with a broad variety of stationary covariance functions.

3.8.2 There was particularly compelling speedup in the case of gridded data, in which learning takes negligible time compared to using inducing points, but there is a limit to the quality of the approximation which gets worse in higher dimensions.

3.8.3 Further work could be directed at developing on the ideas for non-conjugate settings in Section 3.7, testing the applicability of the gridded approximation to real, partially gridded data as discussed in Section 3.6, and using AFS approximations as a component in algorithms, for example probabilistic numerical ones.

3.8.4 As with other precomputable approximations, AFS only works well in low dimensions, with some improvement if the data has a grid structure. Nonetheless, they are compelling for spatial modelling tasks. They also share another limitation with other precomputable approximations, which is that they are limited to stationary covariance functions. Non-stationary covariance functions are addressed in the next chapter.

3.A Reference notes and further reading

3.A.1 The idea of using a Fourier series approximation was considered by Hensman et al (2017); ultimately they found *exact* L^2 FS features dissatisfactory due to the Gibbs phenomenon, and since they did not produce precomputable features. This is why they developed RKHS FS features.

3.A.2 Approximations based on the spectrum were previously motivated by improving upon the random Fourier feature approximation (Rahimi & Recht, 2007); this line of work is known as the sparse spectrum GP, but does not have any of the desirable features of precomputable methods nor does it outperform inducing points. For completeness, I note Lázaro-Gredilla et al (2010) developed the initial formulation, and Gal & Turner (2015) placed it in the variational framework.

Chapter 4

Scalable learning of non-stationary fields

4.0.1 Chapters 2 and 3 focused almost exclusively on *stationary* priors, where the covariance is invariant to translations (or similar groups on non-Euclidean spaces). They are useful for low dimensional settings, such as geospatial modelling, yet it is well-known that many geospatial datasets exhibit a lot of non-stationary variation.

4.0.2 Non-stationary GP models have received much less research focus, but those which have been relatively well-established are much more complex than the standard conjugate GP regression setting addressed so far in this thesis. Typically, they involve some kind of hierarchical model – either warping the input space before applying GP regression (for example in deep GPs), or varying the covariance function's parameters as a function of spatial position (for example in Gibbs' covariance function).

4.0.3 Not only are these models incompatible with the precomputable variational methods of the preceding chapters, but learning is much more computationally expensive than standard GP regression, and requires careful tuning.

4.0.4 Gibbs' covariance function can perform very well in practice, but work on developing this covariance function and variants have focused on using sample-based inference over the parameters. This scales poorly to very large datasets; in this chapter I will describe the use of inducing points to address this problem.

4.0.5 The Fourier feature methods of the previous chapter essentially used the observation that stationary covariance functions on $[-W/2, W/2]^2$ have the Fourier basis as their eigenbasis. Inspired by the historic signal processing literature, I construct non-stationary covariance functions by choosing the eigenbasis as a discrete wavelet basis. This creates a prior with a compatible precomputable variational approximation. Moreover, when the wavelets are chosen to have compact support, then the covariance matrices are sparse.

4.0.6 In Section 4.1, I review existing non-stationary GP models, discussing sparse approximations to Gibbs' covariance function. In Section 4.2 I provide some context on the design of new non-stationary covariance functions, and on multiresolution approximations. Section 4.3 describes specific examples of wavelet-based multiresolution covariance functions, which are empirically evaluated in Section 4.4. The results show that the hierarchical sparse Gibbs formulation captures non-stationary variation well, facilitating more data efficient learning. The multiresolution covariance function improves on the stationary baseline, but further work is needed to close the gap between these and alternatives, such as Gibbs' covariance function.

4.0.7 **Contributions** Work on the doubly sparse approximation to Gibbs' covariance function was done in collaboration with Vidhi Lalchand. Sections 4.1.1 and 4.2 are background material. I published the work on multiresolution (wavelet) GPs (Sections 4.3 and 4.4) concurrently (Cheema & Rasmussen, 2024b).

4.1 Non-stationary priors

4.1.0.1 In previous chapters, I have extensively used the popular squared exponential (SE) and Matérn class covariance functions, with the former being a prior over infinitely differentiable functions, and the latter a rougher analogue. These were introduced in Paragraphs 1.3.0.25 to 1.3.0.27. The parameters typically include a separate lengthscale for each dimension, which also characterises the inverse of the width of the spectral density.

4.1.0.2 Covariance functions such as these are universal in the sense that they are capable of learning any continuous function given enough data; in particular, a sufficient

condition for universality for a stationary covariance function is that its spectral measure has a well-defined density (Micchelli et al, 2006).

4.1.0.3 Non-stationarity can be characterised as having a lengthscale which varies with input location. That is, the characteristic lengthscale of turning points is shorter in some locations than others. Or, thinking in the frequency domain, there is more high frequency content in some regions than others.

4.1.0.4 Many spatial fields have this kind of non-stationarity, and work on suitable priors has been noted as a worthwhile direction by researchers in applications (Prudden et al, 2021).

4.1.0.5 Straightforwardly modifying SE or Matérn covariance functions by asserting a spatially varying lengthscale does not yield a positive definite covariance function (Rasmussen & Williams, 2006, Chapter 4). One strategy is to subtly modify the expression to guarantee positive definiteness (Section 4.1.2). Another is to leave the covariance function unchanged, but warp the inputs (Section 4.1.1). Before developing novel approaches, I provide a brief overview of these pre-existing alternatives.

4.1.1 Methods related to warped inputs

4.1.1.1 One options is to use an ordinary, stationary covariance function, but on warped inputs. That is, the covariance function is

$$k_{\text{warped}} = k(g(x), g(x')) \tag{4.1}$$

for some stationary base covariance function k. This could be a parametric function, such as a neural network (Sampson & Guttorp, 1992; Wilson et al, 2016), and optimised with respect to the log marginal likelihood. Alternatively, we could follow a Bayesian approach, and place a functional prior on g (Aitchison et al, 2021; Damianou & Lawrence, 2013; Lázaro-Gredilla, 2012; Pfingsten et al, 2006; Schmidt & O'Hagan, 2003; Snoek et al, 2014). When this functional prior is a Gaussian process, the overall model is a single hidden layer deep GP.

4.1.1.2 This idea can be extended to an arbitrary number of layers. Deep GPs are thought to be highly expressive models which can cope with non-stationarity in the data. Unlike with single layer GP regression models, the optimal variational distribution is not available in closed form due to the intermediate layers creating latent inputs to the final layer.

4.1.1.3 Using a simple form for the variational posterior, with independence between hidden outputs and hidden inputs, yields a closed form objective (Damianou & Lawrence, 2013). But this is a strong and performance-limiting simplifying assumption: a richer class of approximate posteriors is one in which the inducing values are independent across layers, but the marginal hidden inputs and outputs are correlated (Salimbeni & Deisenroth, 2017).

4.1.1.4 Deep GPs can warp the input space arbitrarily. For many geospatial settings, we typically expect the warping of the input space to be smooth or approximately diffeomorphic, so that the input warping function stretched out high frequency regions and compacts low frequency regions, but the relative ordering of points is approximately maintained. Then if the final layer's covariance function is stationary and monotone, spurious long range correlations are avoided, but the overall model is non-stationary. Choosing a linear or identity mean function in the hidden layers tends to promote this behaviour (Salimbeni & Deisenroth, 2017), but does not guarantee it.

4.1.1.5 Generally, deep GPs can be challenging to train, with performance sensitive to architecture choices and learning algorithm hyperparameters. Substantial research effort has gone into tweaking these choices to develop improved inference and learning strategies, but these are typically not widely adopted, for example due to being difficult to tune (Havasi et al, 2018; Ober & Aitchison, 2021). They offer an interesting alternative to non-stationary covariance functions such as Gibbs', but it is not clear that their performance is competitive.

4.1.2 Gibbs' covariance function

4.1.2.1 A direct approach to non-stationarity is to construct a spatially varying lengthscale function. Gibbs (1998) re-examined the derivation of the SE covariance function as that arising from Bayesian linear regression with Gaussian basis functions in the

limit of infinite basis functions. By varying the basis functions' covariance as a function of position, Gibbs produced the covariance function

$$k_{\text{Gibbs'}}(x, x') = \sigma_f^2 \prod_{d=1}^{D} \sqrt{\frac{2\ell_d(x)\ell_d(x')}{\ell_d^2(x) + \ell_d^2(x')}} \exp\left(-\sum_{d=1}^{D} \frac{(x_d - x'_d)^2}{\ell_d^2(x) + \ell_d^2(x')} \right) \tag{4.2}$$

wherein $\ell_d(x)$ is the spatially varying positive lengthscale for dimension d evaluated at location $x \in \mathbb{R}^D$. Paciorek & Schervish (2003) extended this construction to a broader class of covariance functions, including Matérn class covariance functions.

4.1.2.2 Unlike naively varying the lengthscale, this construction guarantees positive definiteness. Gibbs also designed the covariance function to have constant marginal variance ($k_{\text{Gibbs'}}(x, x) = \sigma_f^2$), so that the spatial variation in the covariance structure is limited to the lengthscale or frequency content.

4.1.2.3 Spatially varying the lengthscale can lead to unexpected phenomena. In particular, consider the case of regions of large lengthscale separated by a small region of short lengthscale, as illustrated in Figure 4.1. The two regions of large lengthscale will be strongly correlated with one another, even though the intermediate region of short lengthscale will be little correlated with either.

4.1.2.4 In some spatial modelling contexts, this may be physically unreasonable. For example, consider the geospatial case where the measurement variable is a climatic variable, the region of short lengthscale is a mountain range, and the regions of long lengthscales are plains on either side. The mountain range may separate the climate into two uncorrelated regimes (for example, due to a rain shadow), which is poorly captured by Gibbs' construction.

4.1.2.5 Nonetheless, this covariance function is elegant and has been shown to perform well in empirical evaluations and case studies (Heinonen et al, 2016; Lalchand et al, 2022b). Its limited use in practice can be attributed to the fact that it is little known, not available in widely used libraries, and challenging to scale to larger datasets.

4.1.2.6 The lengthscale function is not known in advance, so must be learnt. One approach for learning is to construct a hierarchical model where the log lengthscale has a

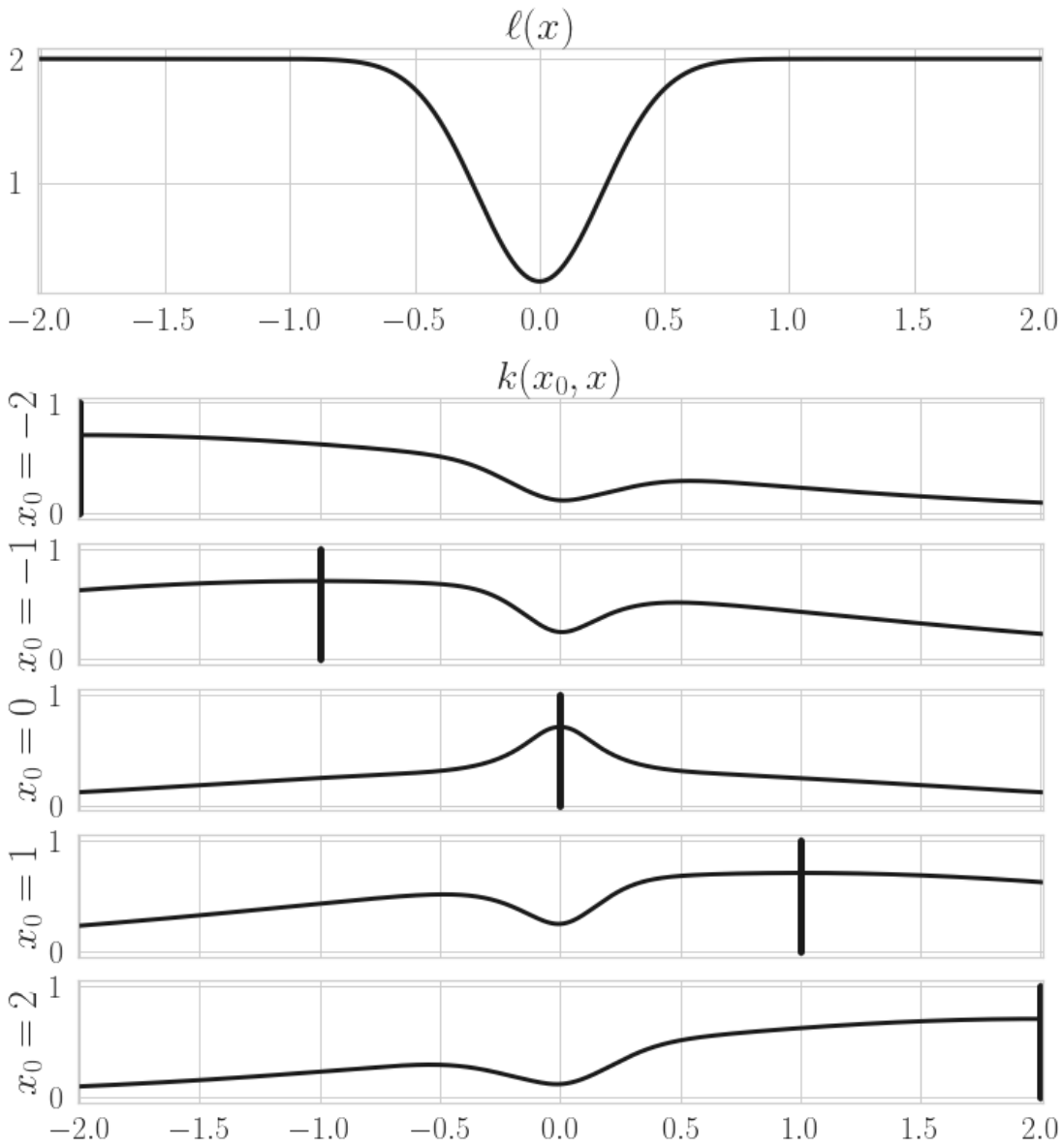


Figure 4.1 Long range correlations with Gibbs' covariance function. The top panel shows the lengthscale function ℓ, which is short near the origin but large elsewhere. The other panels show the covariance function with one argument fixed (at x_0, the thick vertical line). The short lengthscale region has little dependence on the long lengthscale regions (second row), but the long lengthscale regions are correlated with each other (lower rows).

Gaussian process prior (Heinonen et al, 2016, who also place a GP prior on the log variance). In the following, I refer to the prior over parameters as the hyperprior.

4.1.2.7 Good test performance requires the hyperprior to be well tuned; the need to tune the hyperprior can lead to the need to run the learning algorithm many times. In the rest of this section, I develop a sparse maximum a posteriori (MAP) approximation routine suitable for scaling up inference and learning in a hierarchical Gibbs model.

4.1.2.8 Remes et al (2017) used computationally intensive HMC to integrate out the spatially varying parameters; a cheaper alternative is to perform approximate MAP estimation. Compared to approximate log marginal likelihood maximisation, we just need to add the log prior. The optimal variational distribution for the inducing values $q(u)$ is not available in closed form, but we can use the stochastic formulation. This follows by applying Equation (1.42) to Equation (1.14).

$$\begin{aligned}
\log p(y, \theta) &= \log p(y|\log \ell) + \log p(\theta) && (4.3)\\
&= \mathcal{L} + \log p(\log \ell) && (4.4)\\
&\geq \log \mathcal{N}(y \mid \mathsf{K}_{uf}^* \mathsf{K}_{uu}^{-1} \mu_u,\, \sigma^2 \mathsf{I}) - \frac{1}{2\sigma^2} \operatorname{tr}(\mathsf{K}_{ff} - \mathsf{K}_{uf}^* \mathsf{K}_{uu}^{-1} \mathsf{K}_{uf}) && (4.5)\\
&\quad - \frac{1}{2\sigma^2} \operatorname{tr}(\mathsf{K}_{uf}^* \mathsf{K}_{uu}^{-1} \Sigma_u \mathsf{K}_{uu}^{-1} \mathsf{K}_{uf}) + \log p(\log \ell)\\
&= \mathcal{L}' && (4.6)
\end{aligned}$$

4.1.2.9 This lower bound is estimated stochastically by taking batches of the training data. The prior for the log lengthscales is

$$\log \ell_d \sim \mathcal{GP}(\upsilon_{pd}, k_{pd}) \tag{4.7}$$

independently for $d \in \{1{:}D\}$. Assuming that the inducing features used are inducing points, then the parameters which we are interested in estimating the posterior mode of are the lengthscales at the data points, $\ell_{1:N}$, as these are the only lengthscales used in the calculation of the log marginal likelihood. Hence we should optimise the objective with respect to $\ell_{1:N}, \mu_u, \Sigma_u$, and the other, non-spatially varying, hyperparameters.

4.1.2.10 But we expect the number of data points to be large, which has motivated the variational formulation. Then $\ell_{1:N}$ will be very large, with a high computational cost

and memory footprint associated with computing gradients. A suitable scheme is to make the model doubly sparse by using inducing features for the log lengthscales also. Then the log lengthscales at the data can be recovered in distribution by conditioning on log lengthscale inducing features. We can then optimise the approximate log posterior with respect to $\ell_{1:M}$.

4.1.2.11 However, the log prior term is not available in closed form. It could be approximated by simple Monte Carlo, but instead, I make the lower variance approximation

$$\log \ell_{1:N} = \mathbb{E}[\log \ell_{1:N} \mid \log \ell_{1:M}] \tag{4.8}$$

which will be shown to produce satisfactory results in the experimental section. This would only perform very poorly if the lengthscales at the training locations have substantial variance given the lengthscales at the inducing points, which should not occur if a good number of well-placed inducing points are used.

4.1.2.12 I choose to use inducing points as the variational features, and use the same inducing inputs for f and the log lengthscales for simplicity. One option to improve performance is to allow the inducing points to differ.

4.1.2.13 The performance of this approach is sensitive to the choice of parameters for the hyperprior, which requires careful tuning.

4.1.3 Converting between warped inputs and lengthscales

4.1.3.1 Suppose that we are given an input-warping function $g(x)$ for a covariance function with a simple static lengthscale and wish to construct an equivalent spatially varying lengthscale function for unwarped inputs. Then consider some $x' = x + \delta$, and we seek

$$\frac{|x - x'|}{\ell(x)} = \frac{|g(x) - g(x')|}{\ell} \tag{4.9}$$

$$\Rightarrow \ell(x) = \frac{\ell\delta}{|g(x) - g(x+\delta)|} \approx \frac{\ell}{\frac{dg}{dx}}. \tag{4.10}$$

4.1.3.2 The final approximation follows via a first order approximation.

4.2 Space-frequency analysis

4.2.1 The issue with stationary covariance functions is that their spectral characteristics do not vary over the input domain: they have a fixed spectral density. Non-stationary covariance functions have spatially varying spectral characteristics.

4.2.2 One approach to try to design new and useful non-stationary covariance functions would be to try to write down some kind of spatially varying spectral density. But unlike stationary covariance functions, non-stationary covariance functions cannot be straightforwardly characterised in terms of a density, which is the focus of the discussion in this section.

4.2.3 Consider the following quadratic transform.

Definition 4.1 (Wigner-Ville transform.) (Mallat, 2009, Chapter 4) Let $f \in L^2(\mathbb{R})$ with Fourier transform $\tilde{f}$. Then the Wigner-Ville transform of f is

$$\begin{aligned}[\mathcal{S}_{WV} f](x,\xi) &= \int f\left(x + \frac{x'}{2}\right) f^*\left(x - \frac{x'}{2}\right) e^{-i2\pi\xi x'} dx' \\ &= \int \bar{f}\left(\xi + \frac{\xi'}{2}\right) \bar{f}^*\left(\xi - \frac{\xi'}{2}\right) e^{i2\pi x\xi'} d\xi'\end{aligned}$$

where the equality between the first and second forms follows by applying the Fourier transform.

4.2.4 Now if $f \sim \mathcal{GP}(0, k)$, it is not guaranteed that $f \in L^2(\mathbb{R})$ almost surely, but we can take $f \in L^2(\mathbb{R}, P)$ where P is the prior measure of f. Then if its corresponding Fourier transform has covariance function $\tilde{k}$, the expectation of the Wigner-Ville transform can be expressed as an integral transform of the covariance function.

$$\omega(x,\xi) \overset{\text{def}}{=} \mathbb{E}[[\mathcal{S}_{WV} f](x,\xi)] = \int k(x + x'/2, x - x'/2) e^{-i2\pi\xi x'} dx' \tag{4.11}$$

$$= \int k_x(x') e^{-i2\pi\xi x'} dx' = [\mathcal{F} k_x](\xi) \tag{4.12}$$

where $k_x(x') = k(x + x'/2, x - x'/2)$. And, similarly, $\omega(x,\xi) = [\mathcal{F}\tilde{k}_\xi](x)$. This generalises the spectral density in the sense that if k is stationary, then $k_x(x') = k(x')$ everywhere, and $\omega(x,\xi) = s(\xi)$ (if the spectral measure has a

density). It also contains the intuitive energy conservation property

$$\int \omega(x,\xi)\, d\xi = k(x,x) \tag{4.13}$$

$$\int \omega(x,\xi)\, dx = \tilde{k}(\xi,\xi). \tag{4.14}$$

4.2.5 However, unlike the spectral density, ω is not guaranteed to be positive everywhere: it typically contains oscillating interference terms which integrate out. The source of this issue is essentially that the integral is quadratic in f: if $f = f_1 + f_2$ there are undesirable terms such as $\int f_1(x + x'/2) f_2^*(x - x'/2) e^{-i2\pi x'\xi}\, dx'$.

4.2.6 Indeed, there is no positive quadratic energy that satisfies the energy property, and the only covariance functions which have positive Wigner-Ville transforms are translated and frequency modulated Gaussians (Mallat, 2009, Chapter 4). This means that in contrast to stationary covariance functions, we cannot design non-stationary covariance functions by specifying some spatially varying density.

4.2.7 In signal processing, smoothed versions of the Wigner-Ville transform, such as the spectrogram, are sometimes used to get insight into the spatial variations in a signal's properties. But these are generally not used for quantitative purposes, such as approximation and compression, for which it is generally preferable to use an orthonormal basis which is selective in both space and frequency, such as discrete wavelet bases.

4.2.8 **Covariance functions from basis functions** Those same bases are also useful here: suppose that we are given a spatially and frequency selective orthonormal basis $\{\varphi_{j,t}\}_{j,t\in\mathbb{Z}}$, where j is the frequency index, and the t is the spatial index. Then we can construct a covariance function which has these as its Mercer eigenfunctions.

$$k(x,x') = \sum_{j,t} \beta_{j,t} \varphi_{j,t}(x) \varphi_{j,t}^*(x') \tag{4.15}$$

4.2.9 We can then construct inducing features which are inner products with the basis functions.

$$u_{j,t} = \langle f, \varphi_{j,t}/\beta_{j,t} \rangle \tag{4.16}$$
$$c_{j,t}(x) = \varphi_{j,t}(x) \tag{4.17}$$
$$\bar{k}_{j,t,j',t'} = \beta_{j,t}^{-1} \delta_{j-j',t-t'} \tag{4.18}$$

4.2.10 These are precomputable features. As with Fourier features for stationary covariance functions, these will be feature efficient if the expansion decays quickly in the frequency index j.

4.2.11 **Convergence for spatially localised features** But additionally, for $\text{span}\{c_{j,t}\} \to \text{span}\{k(\bullet, x_n)\}_{n \in \{1:N\}}$ as $j \to \infty$ (as motivated by Theorem 2.12), we need to use every t such that $\varphi_{j,t}$ has x_n in its support. The best scenario is where $\varphi_{j,t}$ has compact support, so that we only need to use basis functions spatially centred near the data. Indeed, in this case, the rows of K_{uf} will be sparse, which will yield additional benefits. If the support is not compact, we will need fast decay in t in order to guarantee feature efficiency.

4.3 Multiresolution (wavelet) covariance functions

Definition 4.2 (Multiresolution approximations.) (Mallat, 2009, Chapter 7) A multiresolution approximation $\{V_j\}_{j \in \mathbb{Z}}$ with each $V_j \subset L^2(\mathbb{R})$ satisfies the following.

1. Translation invariance at local scale. $g(x) \in V_j \Leftrightarrow g(x - 2^{-j}t) \in V_j$
2. Telescopic inclusion. $V_{j-1} \subset V_j$.
3. Dilation/contraction. $g(x) \in V_j \Leftrightarrow g(x/2) \in V_{j-1}$.
4. Coarse limit is zero. $\lim_{j \to -\infty} V_j = \bigcap_j V_j = \{0\}$.
5. Fine limit is everything. $\lim_{j \to \infty} V_j = \text{closure} \bigcup_j V_j = L^2(\mathbb{R})$.
6. Scaling function. There exists v, known as the scaling function, orthonormal to its shifts such that $\{v(x-t)\}_{t \in \mathbb{Z}}$ is an orthonormal basis for V_0. (This condition can be significantly weakened.)

4.3.0.1 **Scaling function bases** It can be verified that an orthonormal basis for V_j can be obtained by scaling the scaling function. That is, let

$$v_{j,t}(x) = \frac{1}{\sqrt{2^{-j}}} v\left(\frac{x - 2^{-j}t}{2^{-j}}\right), \tag{4.19}$$

and then $\{v_{j,t}\}_{t\in\mathbb{Z}}$ is an orthonormal basis for V_j. Each V_j is a space for functions in $L^2(\mathbb{R})$ with behaviour no more fine scaled than 2^{-j}. Fine-scaled is a notion that is similar, but not identical to, high frequency. For example, if the scaling function is discontinuous, then even coarse-scaled subspaces contain a lot of high frequency content.

4.3.0.2 **Detail space and wavelet bases** Let the space W_j be the orthogonal complement of V_j in V_{j+1}, or

$$V_{j+1} = V_j \oplus W_j. \tag{4.20}$$

4.3.0.3 W_j is the space of functions in $L^2(\mathbb{R})$ which are of scale exactly 2^{-j}, and $V_j = \text{closure} \bigcup_{j'=-\infty}^{j} W_j$. It can be shown that there exists a wavelet function w whose scaled and shifted versions

$$w_{j,t}(x) = \frac{1}{\sqrt{2^{-j}}} w\left(\frac{x - 2^{-j}t}{2^{-j}}\right) \tag{4.21}$$

are orthonormal, and $\{w_{j,t}\}_{t\in\mathbb{Z}}$ is an orthonormal basis for W_j.

4.3.0.4 In particular, we can take any $\iota \in \mathbb{Z}$, and then it follows that

$$\{v_{\iota,t}, w_{j,t}\}_{j\geq\iota, t\in\mathbb{Z}} \tag{4.22}$$

is an orthonormal basis for $L^2(\mathbb{R})$, and a suitable set of basis functions to use to construct a non-stationary covariance function. I refer to ι as the *canonical scale* of the basis.

4.3.0.5 **Feature efficiency** Now the variational approximation will be feature efficient if each $w_{j,t}$ has narrow support, and if the coefficients $\beta_{j,t}$ can be chosen with fast decay. The latter also depends on the choice of wavelet: for example, if the data is generated by a smoothly oscillating function, a piecewise constant wavelet will need to use very fine scales to produce a reasonable data fit.

4.3.0.6 In general, there tends to be a tension between the support size and the decay rate. This is captured by the notion of vanishing moments.

Definition 4.3 (Vanishing moments.) A wavelet is said to have p vanishing moments if

$$\int x^j w(x)\,dx = 0 \qquad \text{for } j \in \{0{:}p-1\}.$$

4.3.0.7 The intuition for this is that on a finer and finer scale, the function is better and better approximated by a lower and lower order polynomial (for example, a truncation of its Taylor series), and so the coefficient will be closer and closer to 0. In particular, if g is p times differentiable with bounded p^{th} order derivative and w has fast decay and p vanishing moments, then the coefficients decay at worst like $\mathcal{O}((2^{-j})^{p+1/2})$.

4.3.0.8 **Evaluating wavelets** Most of the commonly used wavelets do not have explicit functional forms, but are expressed in terms of a discrete, finite impulse response, filter. In such a case, point evaluations of the wavelet or scaling functions, which are needed to estimate k and compute c_m, must be approximated numerically by using the filter on a constant vector to evaluate $w(2^{-j'})$ for some $j' \in \mathbb{Z}$.

4.3.0.9 **Daubechies wavelets** The Daubechies wavelet family is a widely used family where the number of vanishing moments is maximised for a given support width; db$p$ wavelets have p vanishing moments and a support width of $2p-1$. The db1 wavelet is the well-known piecewise constant Haar wavelet.

4.3.0.10 It remains to determine a suitable parametric form for the coefficients. Rewriting the covariance function with explicit reference to the scaling and wavelet functions,

$$k(x,x') = \sum_t \alpha_t v_{\iota t}(x) v_{\iota t}(x') + \sum_{j=\iota}^{\infty} \sum_t \beta_{jt} w_{jt}(x) w_{jt}(x'). \tag{4.23}$$

4.3.0.11 We want to localise fine-scaled behaviour, so that we avoid smoothing out high frequency phenomena or learning excessively low global lengthscales.

4.3.0.12 **A first attempt** For an almost-stationary version, consider

$$\alpha_t = 2^{-1} \tag{4.24}$$

$$\beta_{jt} = 2^{-1}\frac{2^{1+a_0}-1}{2^{-(1+a_0)}}2^{-(1+a_0)(j-\iota+2)} \tag{4.25}$$

where $a_0 > 0$ is a parameter. The exponent is designed so that β decays as j increases, and the constant is to normalise so that $\alpha_t + \sum_j \beta_{jt} = 1$. Note that the sum is convergent for all $a_0 > 0$. For the Haar wavelet, this sum is exactly $k(x,x)$; more generally, the calculation for $k(x,x)$ is more complex, but fixing the value of this sum means that a controls only the coarse-scaledness of the covariance function, and not the variance, as in Gibbs' covariance function.

4.3.0.13 The issue with this first covariance function is that it does not allow for learning localised fine-scaled regions. An alternative form for β is

$$\beta_{jt} = \frac{1}{Z_{jt}}\left[\frac{2^{1+a_0}-1}{2^{-(1+a_0)}}2^{-(1+a_0)(j-\iota+2)} + \sum_{q=1}^{Q} b_q \frac{2^{1+a_q}-1}{2^{-(1+a_q)}}2^{-(1+a_q)(j-\iota+2)}2^{-\left(\frac{t2^{-j}-c_q}{s_q}\right)^2}\right] \tag{4.26}$$

$$Z_{jt} = 2\left[1 + \sum_{q=1}^{Q} b_q 2^{-\left(\frac{t2^{-j}-c_q}{s_q}\right)^2}\right] \tag{4.27}$$

which has the capacity to add extra fine scaled terms – if $a_q > a_0$ then we have additional fine scaled behaviour within a few multiples of s_q from c_q. The normalisation terms are chosen with the aim of keeping the local variance $k(x,x)$ approximately constant (see Appendix A.4).

4.3.0.14 **Higher dimensions** To extend to higher dimensions, we construct the kernel and the features by taking a product over dimensions.

4.3.0.15 **Convergence considerations** Unlike in Chapters 2 and 3, I do not provide explicit convergence results here. Convergence is guaranteed asymptotically following from Theorem 2.12, but bounding the rate of growth of the number of features becomes challenging for wavelets which do not have disjoint support within a given level t.

4.3.1 Leveraging sparsity

4.3.1.1 For wavelets with compact support, the matrix K_{uf} is sparse, and hence $\mathsf{B} = \mathsf{K}_{uu} + \sigma^{-2}\mathsf{K}_{uf}\mathsf{K}_{uf}^*$ is also sparse. This is the matrix of which we need the log determinant and inverse (Section 2.2); recall also that here K_{uu} is diagonal since the wavelets are orthogonal. Compared to the stationary case, we expect the number of inducing features to be much larger, because they are spatially localised. That is, while using scales j up to some $j_{\max}$ may suffice to approximate the function, at each scale j we will need many shifts t. By leveraging sparsity, we can reduce the severity of the impact of this.

4.3.1.2 There is an ordering of the data such that for 1D inputs it is possible to express K_{uf} as vertically stacked banded matrices, and the higher dimensional version as the Khatri-Rao product of such matrices (from Proposition 2.15). Then we may be able to use specialised routines for banded matrices, as with the B-spline basis methods of Cunningham et al (2023). However, it would be more convenient and flexible to use general sparse routines.

4.3.1.3 We can efficiently calculate the linear solve and log determinant given the Cholesky decomposition of B, for which sparse routines with heuristics to minimise the infill are well established, most notably `cholmod` (Chen et al, 2008). Unfortunately, these routines do not have differentiable implementations available in widely-used automatic differentiation packages, which means they are not immediately compatible with standard gradient-based optimisation routines.

4.3.1.4 In this subsection, I briefly develop the calculation for the adjoint, so that standard sparse Cholesky routines can be used in the loss function calculation when packaged with a custom backward pass.

4.3.1.5 One could directly implement the algorithm to compute the adjoint sensitivity of the Cholesky algorithm, following Murray (2016). This will be able to leverage sparsity only if the computations are carried out using blocked sparse linear algebra routines.

4.3.1.6 However, in this context, the overall sequence of computations (in the style of Figure 1.2), omitting irrelevant details, is

$$\theta \to (\mathsf{B}, b) \to (\mathsf{L}, b) \to (\log|\mathsf{L}|, \mathsf{L}^{-1}b) \to \mathcal{L}' \tag{4.28}$$

where $b = \mathsf{K}_{uu}\bar{y}$ is a vector-valued function of the parameters, and L is the Cholesky root of B. But really, the intermediate Cholesky calculation is only used to simplify calculations relating to the original matrix B. In particular, $(\log|\mathsf{L}|, \mathsf{L}^{-1}b)$ is used for $(\log|\mathsf{B}|, \mathsf{B}^{-1}b)$. The simplified sequence is

$$\theta \to (\mathsf{B}, b) \to (\log|\mathsf{B}|, \mathsf{B}^{-1}b) \to \mathcal{L}'. \tag{4.29}$$

4.3.1.7 Then I package the forward function $(\mathsf{B}, b) \mapsto (\log|\mathsf{B}|, \mathsf{L}^{-1}b) = (\delta, \gamma)$ with a custom adjoint update $(\overleftarrow{\delta}, \overleftarrow{\gamma}) \mapsto (\overleftarrow{\mathsf{B}}, \overleftarrow{b})$.

$$\overleftarrow{\mathsf{B}} = \mathsf{B}^{-1}\overleftarrow{\delta} - \overleftarrow{b}\gamma^\top \tag{4.30}$$

$$\overleftarrow{b} = \mathsf{B}^{-1}\overleftarrow{\gamma} \tag{4.31}$$

4.3.1.8 These follow from standard results for the log determinant and linear solve, and we can leverage `cholmod`'s implementation for both parts, since it offers efficient routines for computing linear solves, the log determinant, and the explicit inverse. Finally, to avoid dense computation, note that we only need the adjoint sensitivities at the non-zero locations in B, which we can form directly rather than going through the outer product. The sparsity pattern of the matrix does not change during training, so the expensive infill-minimising analysis step can be performed once during precomputation. An example implementation is given in Appendix C.2.

4.4 Experimental evaluation

4.4.0.1 To evaluate the non-stationary covariance functions, I compare a stationary, SE, baseline with Gibbs' covariance function using the doubly sparse hierarchical approach of Section 4.1.2, and multiresolution covariance functions using both Haar (db1) and db4 wavelets. As described earlier, the Haar wavelet will need a low decay rate in order to approximate even smoothly varying functions well, due to the small number of vanishing moments.

4.4.0.2 Firstly, I consider the ability of the models to generalise in low lengthscale regions in a basic regression task, and later I test the performance in an active learning task, where no parameter learning takes place, and the model is only used to select new training inputs.

4.4.0.3 For a real-world dataset, I use the precipitation dataset introduced in Section 2.5. Recall that this dataset exhibits non-stationarity: in particular there are short lengthscale variations near the coasts and especially in the Pacific northwest, with low lengthscale regions dominating the inland regions. To reduce the computational burden of tasks such as tuning the hyperprior, I further downsample the dataset (by 10 in each dimension, compared to be 4 in each dimension previously).

4.4.0.4 To develop easy to visualise intuition, I introduce a toy non-stationary dataset, constructed by sampling from a squared exponential GP (lengthscale 0.4, unit signal variance, noise standard deviation 0.05) with warped inputs. The input warping function is a combination of linear and logistic. For inputs shifted and scaled to the range $[0, 1]$, the warping function is defined as

$$g(x) = 0.3x + 0.7\frac{1}{1 + e^{-21(x-0.7)}} \tag{4.32}$$

and I sample inputs in the range $[-5, 5]$. This produces a low lengthscale region centred on $x = 2$, with higher lengthscales elsewhere.

4.4.0.5 I exclude methods deep GPs from the comparison, since the Gibbs model serves as a state of the art example, and there are many difficulties in training deep GP models. For example, following a standard setup of GPflux (Dutordoir et al, 2021) did not make progress towards a reasonable fit on the precipitation dataset. In these cases, the lengthscale appears to vary smoothly across in the input domain, so deep GPs are a very poor prior for the data, yet the issues discussed in Section 4.1.1 come into play.

4.4.0.6 Additional experimental details are documented in Appendix A.4

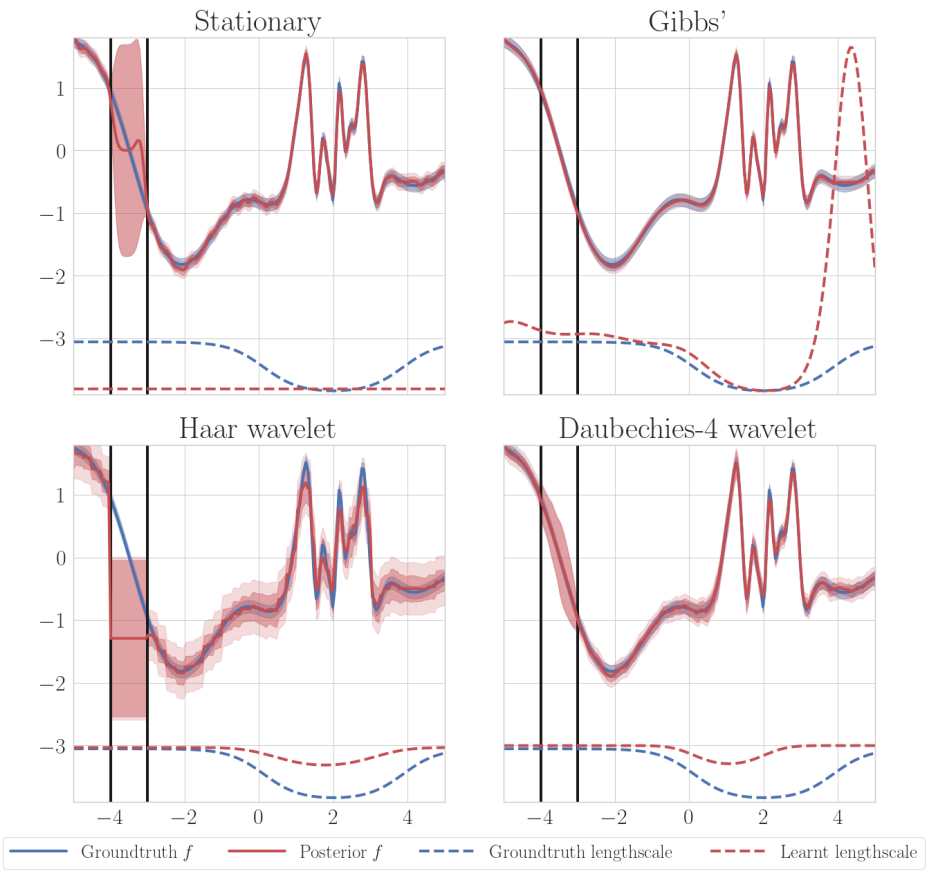


Figure 4.2 Comparing predictions on 1D toy data. No training data is used within the black lines. The groundtruth is in blue (two standard deviations of measurement noise shaded) and the posterior predictive in red (two standard deviations of function variance shaded darkly, two standard deviations of combined function and learnt noise variance shaded lightly). An estimate of the lengthscales is shown in dashed red. The stationary model learns a globally low lengthscale, therefore generalising poorly in the censored region. Using Gibbs' covariance function or a db4 wavelet multiresolution prior learns the non-stationary behaviour much better. The Haar multiresolution model localises the shorter lengthscale region very well but still generalises poorly since the prior is a poor match for the data generating process (piecewise constant compared to smooth).

4.4.1 The regression patch test

4.4.1.1 I use training inputs which are regularly spaced, except in a patch in a low lengthscale region (highlighted with a black outline in Figure 4.2 for the synthetic case, and in Figures 4.3 and 4.4 for the precipitation case). To fit the data in the short lengthscale region, a stationary model will need to learn a low lengthscale, which will lead to poor generalisation in the censored patch. In contrast, a good

nonstationary model should learn that the low lengthscales are localised, and hence generalise well without compromising the data fit at the training locations.

4.4.1.2 In Figure 4.2, lengthscale functions are also plotted: for the SE and Gibbs' covariance functions, these are the actual learnt lengthscales; the groundtruth lengthscale (dashed blue line) is approximated using the routine of Section 4.1.3.

4.4.1.3 For the multiresolution covariance functions, I estimate a local lengthscale according to the following heuristic method. In a squared exponential covariance function, when $x' = x + \ell$, the correlation falls by a factor of $e^{-1/2}$ compared to $k(x, x)$. Taking a multiresolution covariance function, and setting $Q = 0$ for now, the coefficients tail over j of the coefficients β_{jt} describe how much a point at distance proportional to $t2^{-j}$ is correlated with the current position. This tail is reduced by a factor $e^{-1/2}$ when

$$j = j' \stackrel{\text{def}}{=} \iota + \frac{-\log_2 e^{-1/2}}{1 + a_0} \tag{4.33}$$

so define the characteristic lengthscale of the covariance function as $2^{-j'}$. For $Q > 0$, we just need to weight the different components, so use

$$t = \frac{x}{2^{-j'}} \tag{4.34}$$

$$j'_q = \iota + \frac{1}{1 + a_q} \tag{4.35}$$

$$\ell_{\text{eff}} = Z^{-1}_{j't} \left[2^{-j'} + \sum_{q=1}^{Q} b_q 2^{-\left(\frac{t2^{j'_q} - c_q}{s_q}\right)} 2^{-j'_q} \right]. \tag{4.36}$$

4.4.1.4 Unlike the warped lengthscale of Section 4.1.3, this lengthscale is not directly comparable to the static lengthscale parameter, or the spatially varying lengthscale of Gibbs' covariance function: it is only useful for qualitative comparisons, to determined whether fine-scaled behaviour is identified in the correct locations. Therefore, in the plots of Figure 4.2, the multiresolution effective lengthscales are scaled and shifted to have similar values to the groundtruth.

4.4.1.5 From Figure 4.2, it can be seen that the stationary model learns a globally low lengthscale to fit the low lengthscale region, which leads to rapid prior reversion in the censored region. In contrast, he non-stationary models learn the local variation

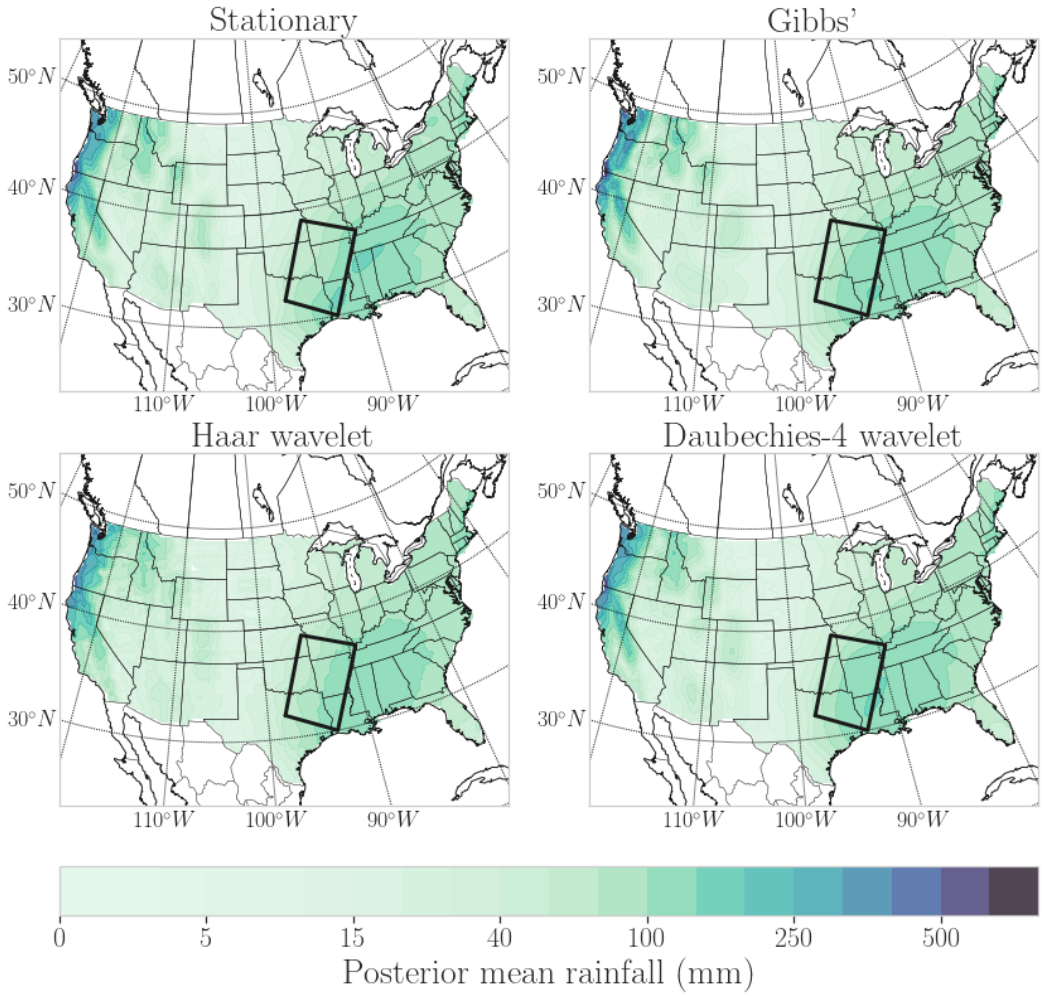


Figure 4.3 Posterior predictive means for the patch test.

in scale well. However, for multiresolution covariance functions, the performance is highly sensitive to the choice of wavelet – the db4 wavelet generalised well in the low lengthscale region, whereas the Haar wavelet does not.

4.4.1.6 This is also corroborated by the test metrics (Table 4.1). In addition to the test RMSE and NLPD, I also provide the learnt signal to noise ratio (SNR), defined as

$$\sqrt{\frac{\mathrm{Var}[f(x)]}{\sigma^2}}. \tag{4.37}$$

4.4.1.7 For the multiresolution covariance functions ($Q = 1$), the function variance varies slightly across in domain; the value reported in the table is from the centre of the censored region. The groundtruth noise standard deviation is 0.05 and the

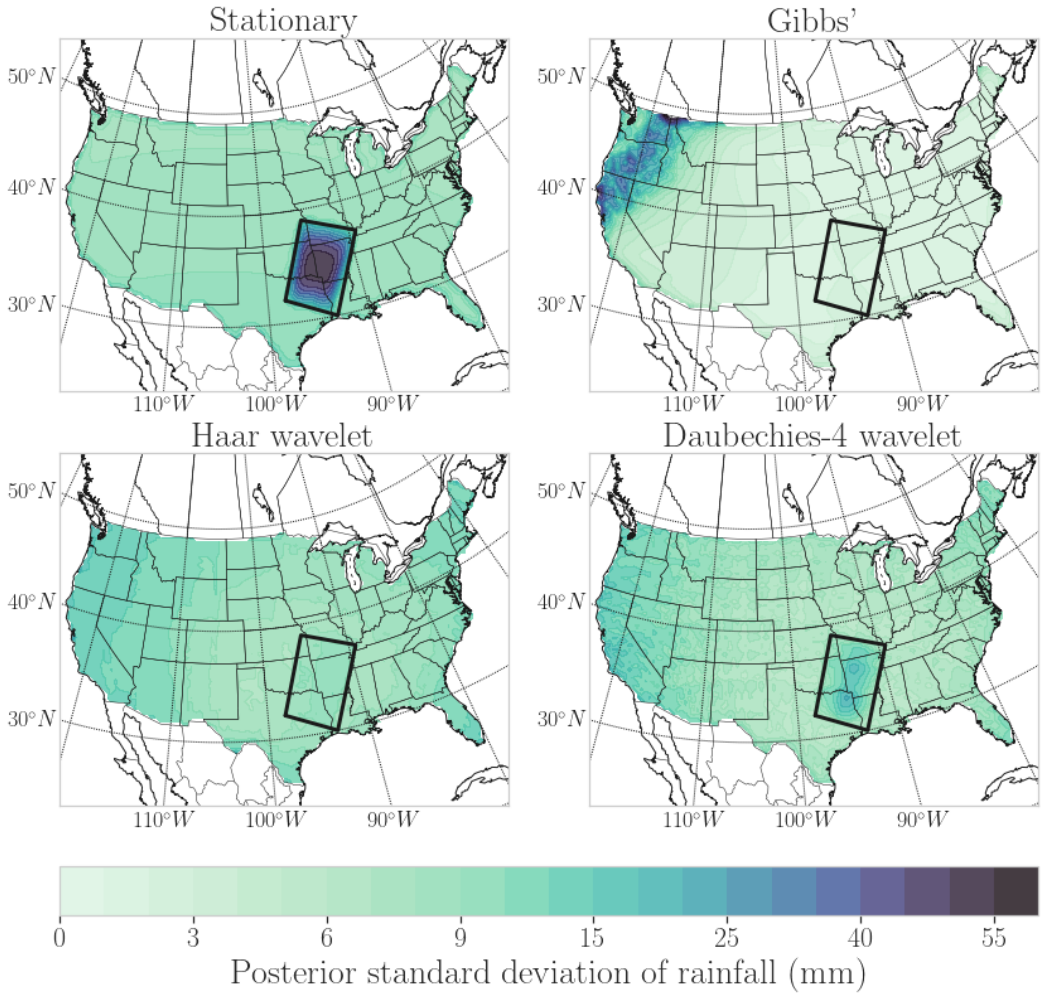


Figure 4.4 Predictive standard deviations for the patch test. The globally low lengthscale learnt by the stationary model leads to unreasonably large uncertainty in the censored region (bordered by a black rectangle). The non-stationary priors exhibit this phenomenon far less, especially the Gibbs model. Since the region is fairly low frequency, a good fit should be able to generalise well from the surrounding training examples.

groundtruth SNR is 20; ideally, the learnt SNR should be close to the true SNR and the learnt RMSE should be close to the groundtruth noise standard deviation, to which the Gibbs' model is the closest, followed by db4.

4.4.1.8 For the precipitation dataset, I use $Q = 3$. Overall, there is a similar pattern – see the predictive means (Figure 4.3) and standard deviations (Figure 4.4). By also training and testing using a uniformly random 80/20 train/test split (metrics in Table 4.1), we can see that indeed the stationary model learns an excessively low lengthscale – it generalises poorly in the patch test, but fairly well in the uniform

Table 4.1 Non-stationary regression test metrics. For the precipitation results, the mean and standard error over five runs are given. For the synthetic data, the groundtruth noise standard deviation is 0.05 and the groundtruth SNR is 20. Best values are in bold (lowest for RMSE and NLPD, closest to 20 for SNR).

Test		Stationary	Gibbs'	Wavelet–Haar	Wavelet–db4
Patch (synthetic)	RMSE	0.425	**0.041**	1.46	0.046
	NLPD	0.755	**-2.09**	1.51	-0.544
	SNR	15.2	**21.4**	5.51	14.95
Uniform (precipitation)	RMSE	**21.19** (0.87)	23.52 (1.33)	26.03 (0.72)	22.27 (0.78)
	NLPD	4.484 (0.04)	**4.07** (0.016)	4.60 (0.02)	4.47 (0.03)
Patch (precipitation)	RMSE	21.38 (0.00)	**7.606** (0.135)	18.15 (0.88)	8.287 (0.282)
	NLPD	4.69 (0.000)	**3.734** (0.043)	4.430 (0.025)	4.212 (0.013)

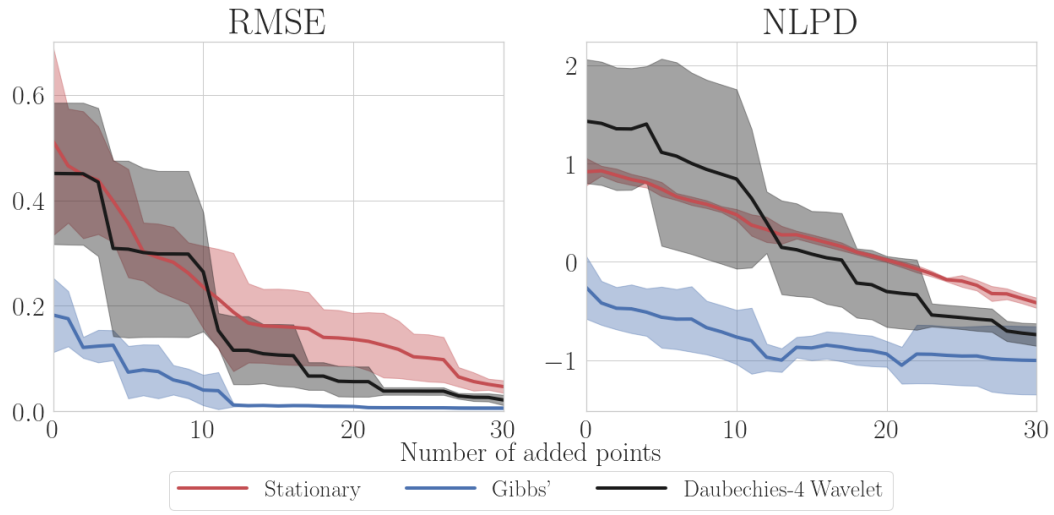
test. The db4 multiresolution model exhibits similar robustness in the patch test, but performs a little less well than the hierarchical Gibbs model, which exhibits excellent performance.

4.4.1.9 The regression tests provide evidence of the advantages in generalisation of non-stationary models. The multiresolution model exhibits some of these advantages, though they require improvement to close the gap to Gibbs' covariance function (for example, by changing the form of the coefficients, or by more careful choice of wavelet).

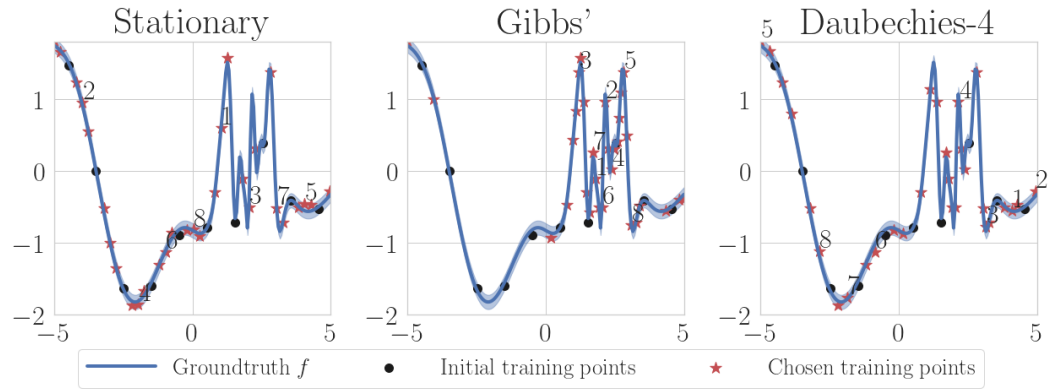
4.4.2 Active learning

4.4.2.1 Learning local variations in lengthscale should permit more efficient selection of new training points. This active learning task is as follows. We are given some initial training points, and at a candidate set of new training inputs. This sensor placement problem is a specific variant of the active learning problem. Detailed background on the problem and strategies is given by Krause et al (2008); here I briefly summarise the strategies used in this chapter. We select the training point which maximises an acquisition function – this is a greedy strategy, insofar as we consider only the immediate benefit of adding one new training location.

4.4.2.2 There are many popular choices of acquisition function. The most compelling is to maximise the mutual information between the measurement at the candidate



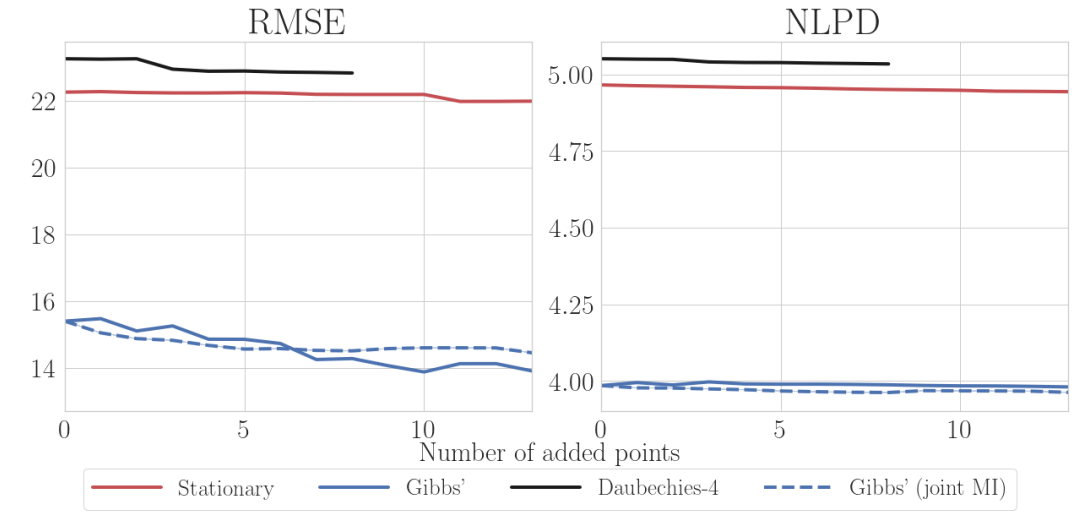
(a) Test metrics



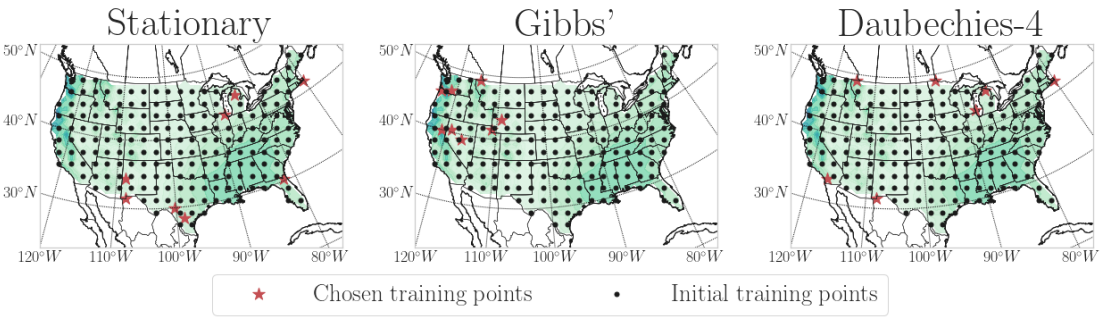
(b) Chosen points

Figure 4.5 (top) RMSE and NLPD for the synthetic experiment; faster decay is better. The shaded region shows two standard errors over the three randomly sampled datasets. (bottom) The earliest points selected (stars) in the synthetic experiment. The initial training set is shown in black. The Gibbs model selects points concentrated in the low lengthscale region, which improves its performance rapidly. The multiresolution covariance function selects points a little less evenly spread than the stationary model.

location and function ($I(f; y_a)$ for candidate action y_a) conditional on the existing training set. To create a practical scheme, the function is limited to its point evaluations at test locations x_*, and the measurement value y_a is imputed using the predictive mean. In the GP case, maximising the mutual information reduces to minimising the log determinant of the predictive covariance of the test outputs y_*.



(a) Test metrics



(b) Chosen points

Figure 4.6 (top) RMSE and NLPD for the precipitation dataset, showing the results for March only. (bottom) The earliest points selected (stars) in the precipitation experiment, using March data. The initial training set is shown in black. The Gibbs model selects points in the low lengthscale northwest, which explains its better performance than the other two.

4.4.2.3 Sometimes, particularly for large and dense datasets, the log determinant can be ill-conditioned. A simple alternative is to minimise the marginal entropy of the test outputs, which is equivalent to the maximising the mutual information when the off-diagonals of the covariance matrix are zeroes.

4.4.2.4 I assume the parameters are pre-learnt on a larger dataset, and active learning is deployed on a new dataset, with the parameters kept fixed. In particular, for the synthetic dataset, the new data is generated by resampling the same data generating process. Note that there is no censored patch in this case. For the precipitation

dataset, parameter learning takes place on the data from January, and active learning is performed on data from the subsequent three months (as three separate experiments).

4.4.2.5 In the synthetic dataset, the initial training set is one tenth original dataset, the test set is half of the original dataset, and any of the original inputs can be selected as a new addition to the training point (an action). In the precipitation dataset, the test and action sets are formed by thinning the data by 20 in each dimension (rather than by 10 as in the pretraining set), and the initial training set is formed by thinning the data by 49. The initial training sets are shown with black dots in Figure 4.5b (synthetic data) and Figure 4.6b (precipitation). I exclude the Haar model here since its weaknesses have been clearly demonstrated in the previous section.

4.4.2.6 On the precipitation dataset, the joint mutual information was prone to numerical errors except with Gibbs' covariance function, so I use the marginal mutual information with that dataset. The joint mutual information results are also plotted for Gibbs' covariance function (the dashed lines in Figures 4.6a and 4.7).

4.4.2.7 The test RMSE and NLPD after each action is reported in Figures 4.5a, 4.6a and 4.7. Better selection of points should lead to a faster decrease in the metrics. Indeed, we see that the Gibbs model has better performance across the experiments. In the precipitation case, the NLPD stays steady as the RMSE drops, which means the uncertainty was high where there were larger errors, but the model is selecting the right points to reduce that uncertainty. The performance is quite different for each month's data, so the metrics for March are plotted individually (Figure 4.6a) to accompany Figure 4.6b.

4.4.2.8 We can examine the earlier points selected qualitatively in Figures 4.5b and 4.6b (for one random sample of the synthetic dataset, and for one particular month for the precipitation dataset). This confirms that the stationary model does not select points in a meaningful way (it is affected only by distance from the existing training points), whereas the Gibbs model focuses heavily on low lengthscale regions, which it has identified well.

4.4.2.9 The multiresolution covariance function presents a more mixed picture. It leverages samples in the low lengthscale regions better (see the staircase pattern in the

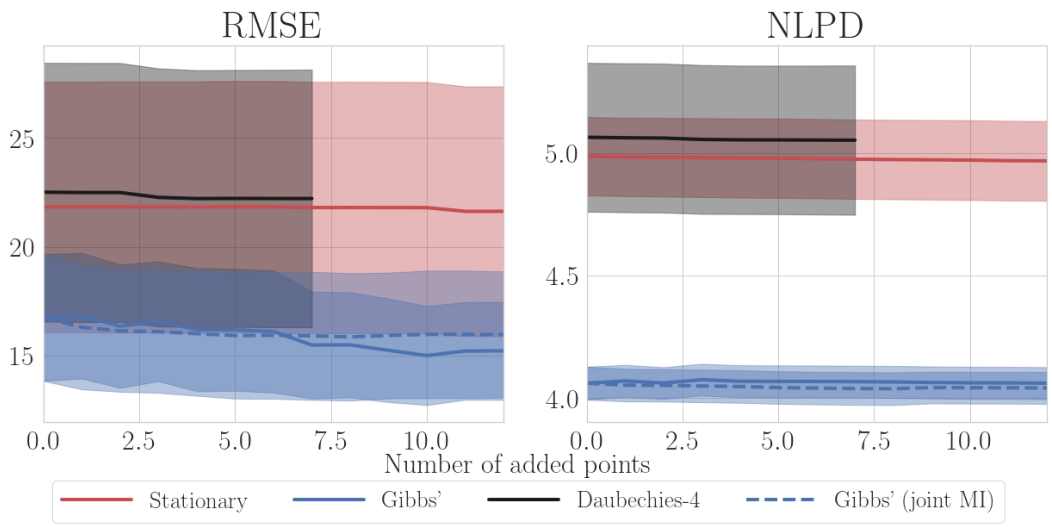


Figure 4.7 RMSE and NLPD for the precipitation experiment; faster decay is better. The shaded region shows two standard errors over the three months' datasets. The shorter run for db4 is due to the large cost of changing the training set repeatedly.

decrease of the metrics in Figure 4.5a). The points it selects are influenced by more than just distance from the existing training points – they are less concentrated in the negative region in the synthetic case, and mainly near the boundaries in the precipitation case. However, they are not much better choices compared to the stationary case.

4.4.2.10 Also, there is no parameter learning during the active learning loop, which means that precomputable methods do not have much less computation cost than non-precomputable methods in general. For the multiresolution covariance function, the computational cost becomes prohibitive, since it relies on the sparse structure of the covariances, which in turn depends on the training set. Hence, the fill-reducing permutation for the sparse Cholesky algorithm must be recalculated to evaluate the acquisition function for each action.

4.4.2.11 **Effects of imputation** The acquisition function requires imputing the value of y_a (since in deployment, we would not have access to the true value), and so the quality of the predictive mean could dominate the outcome of the experiments. To check for this, I also ran the active learning loop with the imputation replaced by the true measurement value. This made no difference to the actions selected in the synthetic case, and the differences in the precipitation case were sufficiently small so as to not affect the overall nature of the results.

4.5 Summary and outlook

4.5.1 This chapter noted that, unlike in the stationary case, there is no suitable generalisation of the spectral density to use as a design space for non-stationary covariance functions (Section 4.2). A practical covariance function to use is Gibbs' covariance function with a GP prior on the lengthscale process, which can be made more scalable with variational approximations (Section 4.1.2). This was shown to generalise better on non-stationary datasets, and to facilitate better actions in a sensor placement task.

4.5.2 A possible way to design precomputable non-stationary covariance functions is to use discrete wavelet bases (Section 4.3). This is a broad family, since both the basis (the wavelet) and the parameterisation can be varied to obtain different properties.

4.5.3 The specific parameterisation suggested here showed improvements over stationary approximations, but this is highly sensitive to the choice of wavelet, and more work is needed to close the gap to Gibbs' covariance function. This could be either by more careful selection of the wavelet or by more careful design of the parameterisation. There are some major limitations: these are designed for fast parameter learning, but changing the training data (such as in the sensor placement task) incurs a large computational cost. This problem could be mitigated by finding a suitable heuristic to update the fill-reducing permutation, for example.

4.A Reference notes and further reading

4.A.1 Opfer (2006) explored the use of similar multiresolution approximations in the context of kernel methods. Ferkous et al (2021) used a wavelet decomposition of the measurements as a preprocessing step before fitting a GP.

4.A.2 This chapter focused on the design of non-stationary covariance functions which capture the notion of local variation in scale, or frequency content. Of course, other non-stationary constructions exist; for example the stationary covariance functions on compact homogeneous Riemannian manifolds of Chapter 2 can be combined with a mapping between Euclidean space and the manifold. This covariance function is stationary on the manifold, but induces a non-stationary approximation in Euclidean

space. For an overview of other examples, see Rasmussen & Williams (2006), Chapter 4.

Chapter 5

Bias in learning flow and velocity fields

5.0.1 SSMs were motivated by a continuous time model in Example 1.2, but the bulk of the treatment in Section 1.4 was in discrete time. This is also true of much of the previous work with GPSSMs, since working with continuous time GPSSMs comes with considerable computational challenges. In Section 5.1, I explore how far discrete time GPSSMs are a good prior when the data is generated by a continuous time process.

5.0.2 When functions are modelled in state space, neither the input nor the output of the function are directly observed, unlike in the regression setting. This makes inference and learning much more challenging – it will no longer be possible to guarantee the marginal likelihood approximation becomes tight as the number of variational parameters goes up, nor that the approximate posterior becomes close to the true posterior.

5.0.3 In these settings, parameter estimates can be badly biased, and the relationship between the structure of the approximate posterior and these biases can be subtle (Turner & Sahani, 2011). Previous work has seen that the typical approximation strategy – which treats f and the latent trajectory x as independent – tends to bias the model towards high noise, which significantly reduces predictive performance (Ialongo et al, 2019).

5.0.4 Correlated approximations perform better but with higher computational cost. In Section 5.2, I show that constructing the state posterior using Kalman smoothing offers a far cheaper alternative, which works well to mitigate the noise bias and can also easily be used in continuous time.

5.0.5 **Contributions** The discussion on the results of Section 5.1 are from my previous work (Cheema & Rasmussen, 2022). I previously published the basic ideas of using Kalman smoothing to construct the state posteriors in the GPSSM (Cheema, 2021); the presentation and results of Section 5.2.1 differ from there in a two main ways. Firstly, here I focus mainly on correlated approximations, which allows for much faster learning, whereas there I used only uncorrelated approximations. Secondly, the experiments here focus on a couple of carefully selected toy examples to focus on understanding the bias in the learnt noises, whereas the experiments there focus on standard metrics on real-world datasets.

5.1 GPSSMs as priors for continuous time processes

5.1.1 Discrete-time SSMs were introduced in some detail in Section 1.4. There, I noted that the sorts of trajectories modelled by *linear* SSMs do not contain many of the interesting phenomena observed in real systems, necessitating a nonlinear approach.

5.1.2 **Accounting for nonlinear measurements** For system identification – learning the dynamics of the system from measured trajectories – placing a flexible prior on both the dynamics function and the measurement function (f and g in Equation (1.53)) introduces excessive non-identifiability. As a first example, f can be freely scaled up if g is scaled down. In general, complexity in g can be shifted to f, and it suffices to make g linear.

5.1.3 Concretely, suppose the 'true' generative model had nonlinear f and g. Then it has an equivalent model which has a linear measurement model as follows.

$$\begin{aligned} \tilde{x}_{t+1} = \begin{bmatrix} x_{t+1} \\ y_{t+1} \end{bmatrix} &= \begin{bmatrix} f(x_t) \\ g(f(x_t)) \end{bmatrix} + \begin{bmatrix} \kappa_t \\ 0 \end{bmatrix} \stackrel{\text{def}}{=} \tilde{f}(\tilde{x}_t) + \tilde{\kappa}_t \\ y_t &= \begin{bmatrix} 0 & \mathsf{I} \end{bmatrix} \tilde{x}_t + \rho_t \end{aligned} \tag{5.1}$$

5.1.4 Hence, using only a linear observation model will at worst require increasing the state dimension by Δ.

5.1.5 **A discrete time GPSSM** GPs offer a promising prior for dynamics models, and combined with the linear measurement model, the overall model is

$$\begin{aligned} x_{t+1} &= f(x_t) + \mathsf{L}\kappa_t \\ y_{t_j} &= \mathsf{C}x_{t_j} + \mathsf{P}\rho_{t_j} \quad j \in \{1{:}T\} \\ f_d &\overset{\text{IID}}{\sim} \mathcal{GP}(\zeta_d, k_d) \quad d \in \{1{:}D\} \end{aligned} \tag{5.2}$$

with $x_t \in \mathbb{R}^D, y_{t_j} \in \mathbb{R}^\Delta$ as introduced in Section 1.4; κ and ρ are white Gaussian noise processes with zero mean and identity covariance; L, P, and C are parameters and often the mean function $\zeta_d = 0$. The observation times have been generalised slightly to allow for times without measurements.

5.1.6 This parameterisation of the noise yields the effective noise covariances (using the notation of Section 1.4)

$$\mathsf{L}\kappa_t \sim \mathcal{N}(0, \mathsf{Q}) \qquad \mathsf{Q} = \mathsf{L}\mathsf{L}^\top \tag{5.3}$$

$$\mathsf{P}\rho_t \sim \mathcal{N}(0, \mathsf{R}) \qquad \mathsf{R} = \mathsf{P}\mathsf{P}^\top. \tag{5.4}$$

5.1.7 In this discrete time setting, the GP is a prior on the *flow field* of the state, which is assumed time invariant. This is the setting which has dominated previous work (Bui, 2017; Curi et al, 2020; Deisenroth & Mohamed, 2012; Doerr et al, 2018; Eleftheriadis et al, 2017; Fan et al, 2023; Frigola-Alcalde, 2014; Ialongo et al, 2019).

5.1.8 More generally, f is a multi-output GP, and could be chosen to correlate state dimensions. Asserting independence across dimensions makes calculations simpler, and is not too limiting: $f_d(x_{t-1})$ is independent across d given x_{t-1}, but $y_t|x_t$ can have correlations introduced by C, and the marginals of x_t are also correlated across dimensions.

5.1.9 By a similar motivation, Q will usually be constrained to be diagonal.

5.1.10 **The continuous time GPSSM** However, often the motivation for a state-space approach, and the form of our prior knowledge, is as a mechanistic stochastic

differential equation (SDE) model, as motivated in Example 1.2. Concretely, this model is

$$\begin{aligned} dx_t &= f(x_t)\,dt + \mathsf{L}\,d\beta_t \\ y_{t_j} &= \mathsf{C}x_{t_j} + \mathsf{P}\rho_{t_j} \quad j \in \{1{:}T\} \\ f_d &\overset{\text{IID}}{\sim} \mathcal{GP}(0, k_d) \quad d \in \{1{:}D\} \end{aligned} \tag{5.5}$$

where β_t is standard Brownian motion; that is to say the increments are Gaussian

$$\beta_t - \beta_{t'} \sim \mathcal{N}(0, |t - t'|\mathsf{I}) \tag{5.6}$$

and any increments over disjoint time intervals are independent. The SDE as written in differential form in Equation (5.5) is understood as defining x_t as a solution in the Itô sense (Øksendal, 2013; Särkkä & Solin, 2019). Informally, a continuous time model learns the *velocity field* rather than the flow field, and the process noise $\mathsf{L}\,d\beta_t$ is a white noise process added to this velocity at each time instant.

5.1.11 Both the function f and the process noise scaling matrix L therefore play different roles compared to discrete time. Where necessary to disambiguate them, I will use the subscripts DT for discrete time and CT for continuous time.

5.1.12 There appear to be several practical advantages to modelling in continuous time.

5.1.13 **No constraints on measurement times** Firstly, there is no physical reason for the measurement times to be constrained to a regular grid, and in some applications, particularly in biological systems, data is highly irregularly sampled. A continuous time model, which learns the *velocity field* rather than the flow field, can handle this straightforwardly by varying the integration time.

5.1.14 Closely related to this is the idea of invariance to the measurement times. In continuous time, the learnt f is an object which is invariant to the measurement times, whereas in discrete time the learnt flow field depends on the frequency of measurement.

5.1.15 **Learning sparse systems** Secondly, it preserves sparse structures in systems. Consider the van der Pol oscillator ($D = 2$, with parameter r), a well known toy nonlinear system (Hirsch et al, 2004, Chapter 12). Suppressing reference to the

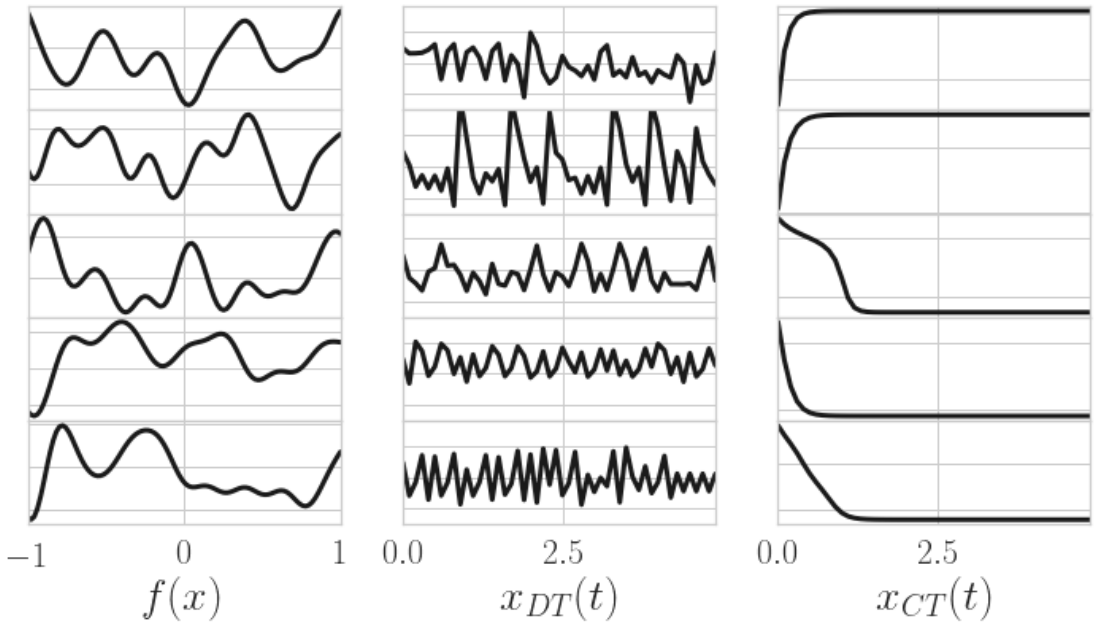


Figure 5.1 Prior samples from GPSSMs (without noise). Each row is one sample; f is on the left, and $x(t)$ for the discrete time (centre) and continuous time (right) cases are shown.

process noise, its dynamics follow

$$\begin{aligned} dx_{t,1} &= x_{t,2}\, dt \\ dx_{t,2} &= [r(1 - x_{t,1}^2)x_{t,2} - x_{t,1}]\, dt. \end{aligned} \tag{5.7}$$

5.1.16 The true dynamics have a sparse structure in the sense that $dx_{t,1}$ depends only on $x_{t,2}$, but when integrated over a time step of δ to generate the discrete time updates, both update equations will depend on both variables.

5.1.17 This is not an exceptional example: many real systems are composed of sparsely interconnected subsystems. For example, consider the toy example of a mechanism made up of several interconnected rigid links. Each link has a fairly low state dimension: its instantaneous dynamics are only influenced by the state of its neighbours (such as the velocities of the neighbours, or forces exerted by their accelerations). But after integrating over even a very small time step, the influence of any link reaches every other link.

5.1.18 These sparse structures could be advantageous for scaling up system identification methods to higher state dimensions, since the effective input dimension of each f_d is kept low. Without this advantage, inference and learning in high dimensional state spaces is very challenging. This may be in part due to non-identifiability which is exacerbated with more input dimensions to each function.

5.1.19 **Qualitative behaviour of the prior** Thirdly, there is a qualitative difference between discrete and continuous time priors. Ignoring the process noise, continuous time dynamics generate trajectories which do not cross over, whereas in discrete time they may (Hirsch et al, 2004). Discrete time systems often exhibit more complex and challenging to analyse behaviour than continuous time dynamics. It is well-known in the dynamical systems literature that these more challenging behaviours arise when f_{DT} is not diffeomorphic (that is, smooth and bijective with smooth inverse; Losert & Akin, 1983). In our case, f_{DT} is a GP, and will typically not be guaranteed to be diffeomorphic. The difference between sample trajectories from discrete and continuous time GPSSM trajectories is illustrated in Figure 5.1.

5.1.20 There are two common approaches to construct diffeomorphisms for machine learning purposes.

- Construct the function as the solution to an ODE (for example, Walder & Schölkopf, 2008).
- Construct the function to by composing simpler diffeomorphisms (for example, Papamakarios et al, 2021).

5.1.21 If we were to do the former, we may as well work with a continuous time model to begin with. The latter is not immediately applicable here, since we would need a fundamentally different probabilistic model for f.

5.1.22 **Continuous time modelling is expensive** These arguments have motivated work on continuous time GPSSMs (Duncker et al, 2019), but this comes with a number of challenges. Details of inference and learning in GPSSMs will be given later; in brief, previously established methods generally require many more variational parameters, with more computationally expensive and less numerically stable learning.

5.1.23 This is typical of the situation for continuous time methods in machine learning algorithms in general – there are numerous challenges to overcome in the implementation, but they can bring compelling advantages.

5.1.24 **Can we get away with discrete time priors?** Intuitively, when modelling using a Bayesian approach, it should be advantageous to select prior distributions which align with our beliefs about the data generating process. In the case of data generated from a continuous time process, the question is whether it matters if we use the discrete time prior of Equation (5.2).

5.1.25 In both cases, the model f is quite flexible, so either model should do well under benign circumstance, such as when there is adequate, high quality data. When the data is lower quality – for example, noisier – then the priors play a more important role, and we might expect to see the discrete time model perform worse.

5.1.26 I test this hypothesis qualitatively by training both a discrete time and continuous time GPSSM on data sampled from a van der Pol oscillator (Equation (5.7)) with $r = 0.5$. This is a minimal working example in the sense that we require $D \geq 2$ to exhibit interesting nonlinear phenomena in continuous time, such as the stable oscillations of this system. I train the models in lower ($\mathsf{P} = 10^{-1}$) and higher ($\mathsf{P} = 10^{-\frac{1}{2}}$) observation noise settings and qualitatively assess the posterior dynamics (Figure 5.2) and the estimated latent trajectory (Figure 5.3).

5.1.27 In both cases, I allow full state observations ($\mathsf{C} = \mathsf{I}$) and keep the process noise fixed to a low value ($\mathsf{L}_{DT} = 10^{-\frac{3}{2}}, \mathsf{L}_{CT} = \delta^{-\frac{1}{2}} 10^{-\frac{3}{2}}$), where $\delta = 0.3$ is the time for a discrete time step, generally treated as 1 in the notation of this section. For the continuous time method, I use a fourth order Runge-Kutta integrator with a fixed step size of 0.1.

5.1.28 In addition, I estimate the root mean squared error (RMSE) between the learnt mean function and the ground truth dynamics. For a fair comparison, I convert the continuous time dynamics functions to discrete time by integrating: $\int_0^\delta f(x_t)dt$. Further details are available in Appendix C.3.

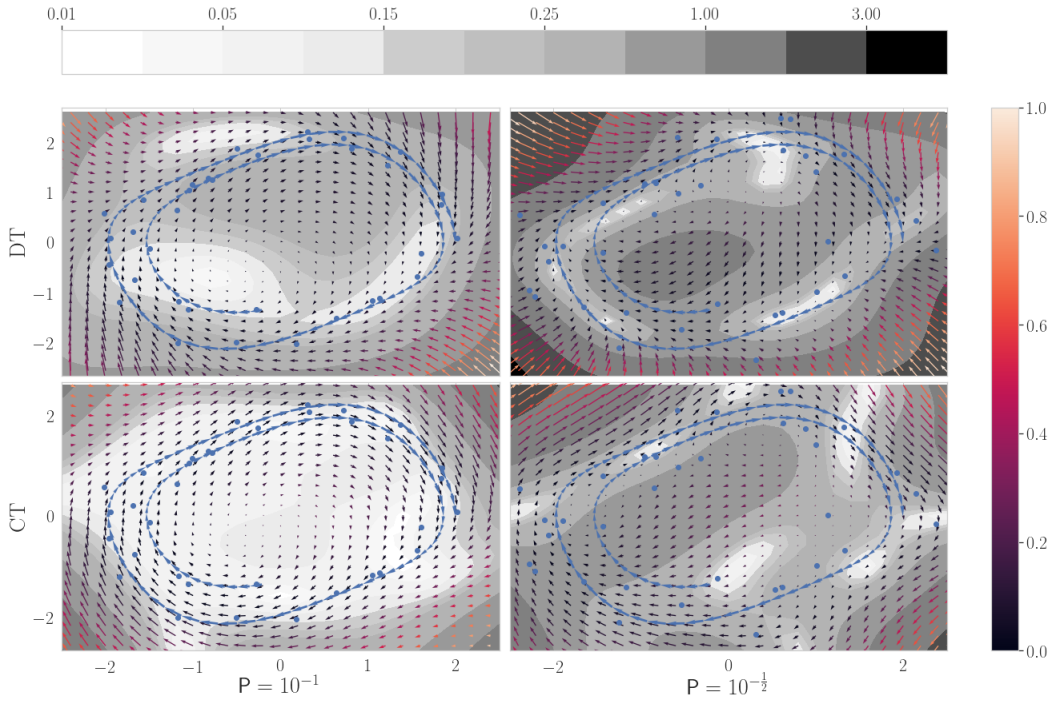


Figure 5.2 RMSE (grey contours) and posterior predictive flow field (arrows, shaded by 2σ confidence) for the van der Pol fit. The blue dots are the training data, and the blue line is the generating latent process. The CT model produces a visually better result, in the sense that its flow field contains cycles even outside the region covered by the training data. In contrast, under higher noise (right hand column), the DT field generally flows radially into the region covered by the training data.

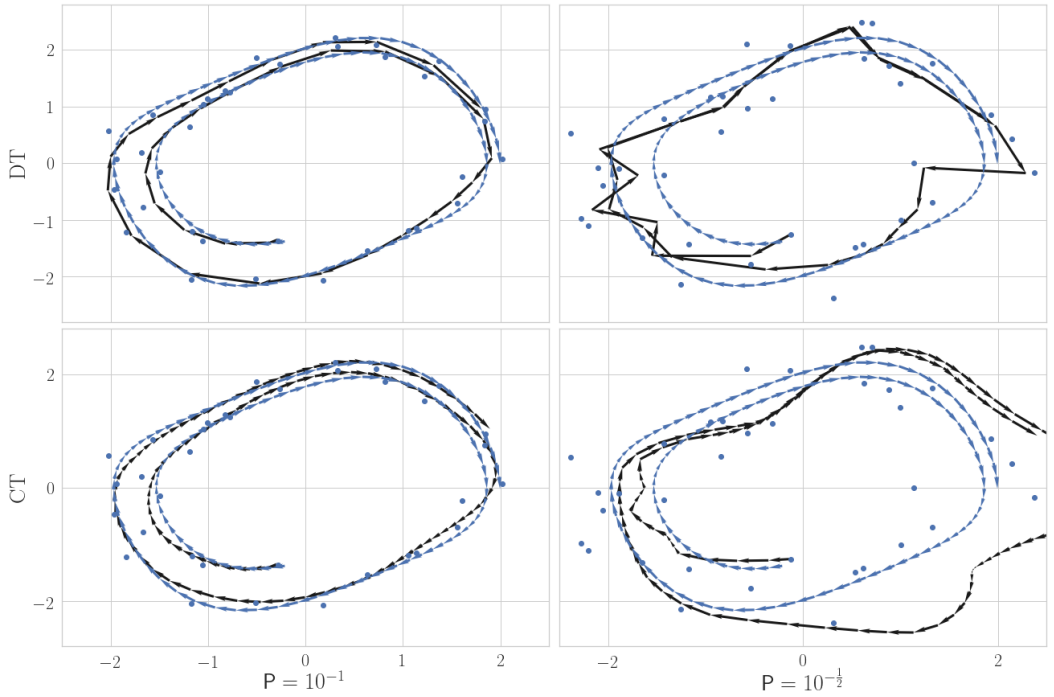


Figure 5.3 Mean of the approximate posterior trajectory $q(x)$ in the van der Pol fits as black arrows; rest as in Figure 5.2. Note that the DT model picks up the smooth, non-self-intersecting properties of the generating process in low noise, but violates these under high noise. The continuous time fit tracks the data less well in higher noise settings, but it captures the general form of the limit cycle.

5.1.29 In the low noise case, both models perform well, tracking the state of the system closely, learning dynamics qualitatively similar to the groundtruth, and with similar RMSEs (0.51 in continuous time; 0.46 in discrete time). Although the discrete time model has a slightly lower RMSE, the continuous time model produces a qualitatively better fit: the states track the generating trajectory more closely, and the error is lower both near the training data and in the central region between the training data.

5.1.30 In high noise, both models struggle, but the RMSE is much worse for the discrete time model (0.84 in continuous time; 1.37 in discrete time). Qualitatively, the continuous time model performs much better, with similar behaviour at the edges of the plots to low noise. However, the continuous time model is more computationally demanding, requiring around five times as long to train, and care to avoid numerical errors.

5.1.31 **A pragmatic compromise** This suggests that in challenging, noisy settings, it would be useful to use a continuous time prior for a continuous time process. But staying with the Itô interpretation, it can be shown that if we set

$$\begin{aligned} f_{DT}(x) &= x + \delta f_{CT}(x) \\ \mathsf{L}_{DT} &= \delta \mathsf{L}_{CT} \end{aligned} \tag{5.8}$$

then the limit as $\delta \to 0$ of the DT system is the CT system, and we see (for example) that the trajectories stop intersecting as we make δ smaller in Figure 5.4. This is equivalent to adding an identity mean function ($\zeta_d(x) = x_d$) to the GP prior.

5.1.32 It is important to note that this discretisation – Euler-Maruyama integration – is of low order, and using a continuous time model permits the use of higher order solvers, such as the fourth order one used in this experiment. Nonetheless, if the data is on a subset of a grid, and it is thought that the data is well explained by a differential equation system, I recommend using a discrete time GPSSM with an identity mean function as a first attempt.

5.1.33 In the next section, I turn to a detailed consideration of inference and learning for the GPSSM. Motivated by the high level findings of this section, I treat both discrete and continuous time GPSSMs.

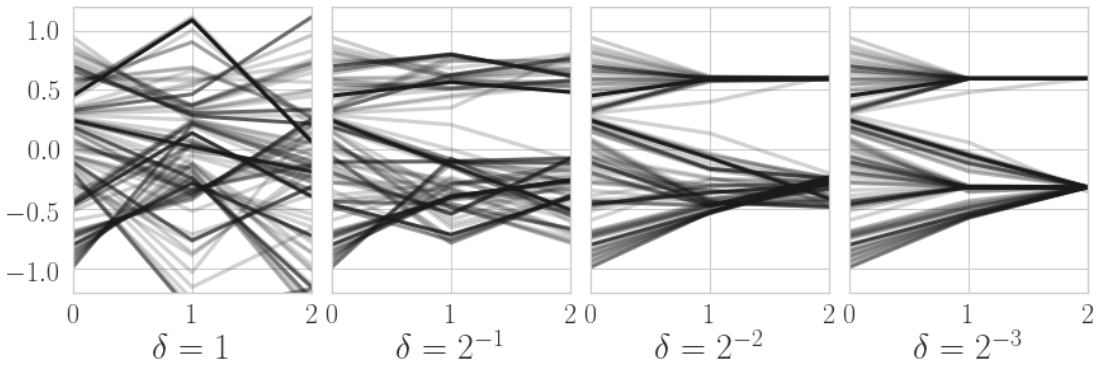


Figure 5.4 Trajectories generated by a discrete time system with transition function $x + \delta f_{CT}(x)$ for decreasing δ. As it converges, the trajectories exhibit the characteristic qualities of continuous time systems (non-intersecting trajectories, all points converging to equilibria in 1D).

5.2 Variational inference and learning with Kalman filtering

5.2.0.1 In the regression setting, the training data $(x_{1:N}, y_{1:N})$ is of the form of inputs and noisy outputs of f, but here neither the GP's inputs nor its outputs are observed directly. In discrete time, the inputs and noisy outputs are respectively x_t, x_{t+1} for $t \in \{1{:}t_T - 1\}$. In continuous time, they are $x_t, \frac{dx}{dt}(t)$ for every $t \in [t_1, t_T]$. These are only observed indirectly by the measurements y_t. (It is assumed that the latent process is modelled from the time of the first observation to the time of the last observation in each case, though this can straightforwardly be extended in either direction, for instance if there is a natural choice for the initial state prior at some earlier time.)

5.2.0.2 This makes learning more challenging, and in particular we have an additional latent process x. Inference is calculating the full latent posterior $p(x, f|y)$. Unlike in the regression setting, and in common with the more general GP latent variable model setting (Damianou et al, 2016), the full posterior is not available in closed form.

5.2.0.3 **Inducing variables** As with GP regression, we can introduce inducing variables $u_m, m \in \{1{:}M\}$ where usually $u_{dm} = f(z_{dm})$ for some $z_{dm} \in \mathbb{R}^D$. Like in Section 1.3.1, u and z can be interpreted as a surrogate data distribution which is

optimised to include as much of the useful information in the training data as possible. But now they are not necessarily used to reduce the computational cost, but also to transform the inference problem into an approximate one with tractable closed form solutions.

5.2.0.4 In the conjugate regression setting, this variational optimisation problem yielded a closed form solution which could be made an arbitrarily good approximation to the true posterior by increasing M. But here, in common with non-conjugate regression and the general latent variable setting, there is no such guarantee, and in particular the optimisation for general $q(x, f)$ does not yield a closed form solution.

5.2.0.5 **Choice of approximate posterior** We could construct a sample-based approximation to the posterior directly (Frigola-Alcalde, 2014, Chapter 4), or, more efficiently, a sample-based approximation to the variational optimum (Fan et al, 2023). Alternatively, we could restrict $q(f, x)$ to be in a convenient family of distributions such that the optimisation produces a closed form distribution. This would be much faster, but generally we would expect the quality of the approximation to be worse, negatively impacting learning.

5.2.0.6 Generally, it appears that the most severe limitation of analytical approximations is that for $q(u)$ to emerge in closed form, x and f must be treated independent in the posterior, which has been shown empirically to lead to learning a pathologically high process noise (Ialongo et al, 2019), which significantly reduces the predictive capability of the model.

5.2.0.7 **Why use process noise at all?** The process noise can be viewed as modelling fast processes which are not resolved at the time resolution of measurements (Leander et al, 2014). It also absorbs the error observed due to the model not explaining the measurements well. If the model is too simple or learnt poorly, then the noise will have a high upward bias, significantly weakening the predictive capabilities of the model. But if there is no process noise in the prior, then the likelihood will be very sensitive to such model mismatch. This is particularly important early in training, when no progress is made if the noise is initialised too low.

5.2.0.8 These problems motivated a large amount of work on correlated approximations (Curi et al, 2020; Doerr et al, 2018; Ialongo et al, 2019). But it is not necessarily the

case that a more flexible variational approximation will lead to better learning; bias in parameter learning is due to the variational objective having different maxima to the log marginal likelihood, and reducing the slack in the bound does not necessarily bring the maxima closer together. This has been well demonstrated previously in toy time series settings (Turner & Sahani, 2011).

5.2.0.9 Moreover, in such correlated approximations, $q(x_{t+1}|f, x_t)$ will typically be a nonlinear Gaussian model, and so $q(x_t|f)$ will not be available in closed form, necessitating sampling over x, and a higher cost of learning overall.

5.2.0.10 This motivates a careful study of the parameter bias, and a search for good quality closed form approximations. In Section 5.2.1, I briefly review the details of inference and the calculation of the variational objective, handling the discrete and continuous time settings together; full calculation details are provided in Appendix A.5. I propose constructing the approximate state posterior using the Kalman smoothers of Section 1.4.1, which requires far fewer trainable parameters than alternatives.

5.2.0.11 Then, in Section 5.2, I characterise the optimal value of the likelihood and process noise parameters. I show empirically that using Kalman smoothing improves parameter learning, even without using correlated approximations.

5.2.1 Gaussian variational inference

5.2.1.1 **The variational objective** The objective is

$$\mathcal{L}' = \int q(x, f) \log \frac{p(y, x, f)}{q(x, f)} \, dx \, df \tag{5.9}$$

$$= \int q(x) \log p(y|x) \, dx - D_{KL}\left(q(f)||p(f)\right) - \int q(f) D_{KL}\left(q(x|f)||p(x|f)\right) \, df. \tag{5.10}$$

5.2.1.2 The rearrangement uses the general factorisation $q(x, f) = q(f)q(x|f)$. Compared to conjugate GP regression, only the last term is new. From Lemma 1.1, we know that $q(x|f)$ optimally has a Markovian structure, so the trajectory KL term is summed over time. That is,

$$\text{either} \quad \int q(f) D_{KL}\left(q(x|f)||p(x|f)\right)\, df = \frac{1}{2}\sum_{t=1}^{t_T-1} \mathcal{G}_t \tag{5.11}$$

$$\text{or} \quad \int q(f) D_{KL}\left(q(x|f)||p(x|f)\right)\, df = \int_{t_1}^{t_T} \mathcal{G}_t\, dt \tag{5.12}$$

5.2.1.3 in discrete time or continuous time respectively, where $\mathcal{G}_t$ is the KL divergence between the (instantaneous) transition distributions. At time t this is $D_{KL}\left(q(x_{t+1}|f, x_t)||p(x_{t+1}|f, x_t)\right)$ in discrete time, or (informally) $D_{KL}\left(q(\dot{x}_t|f, x_t)\right)$ in continuous time, using the standard shorthand $\dot{x} = \frac{dx}{dt}$.

5.2.1.4 **Gaussian approximate posterior transitions** Requiring the approximation to have Gaussian transitions like the prior, it can be given the general form

$$\text{either} \quad q(x_{t+1}|f, x_t) = \mathcal{N}(x_{t+1}|\hat{f}_t(x_t), \hat{\mathsf{Q}}'_t) \tag{5.13}$$

$$\text{or} \quad q(\dot{x}_t|f, x_t) = \mathcal{N}(\dot{x}_t|\hat{f}_t(x_t), \hat{\mathsf{Q}}'_t) \tag{5.14}$$

5.2.1.5 in discrete and continuous time respectively. If we define the quadratic

$$\mathcal{E}_t = \int q(x_t, f)(f(x_t) - \hat{f}_t(x_t))(f(x_t) - \hat{f}(x_t))^\top\, dx_t\, df + \hat{\mathsf{Q}}'_t \tag{5.15}$$

$$\text{then} \quad \mathcal{G}_t = \log \frac{|\mathsf{Q}|}{|e\hat{\mathsf{Q}}'_t|} + \mathrm{tr}\left(\mathsf{Q}^{-1}\mathcal{E}_t\right). \tag{5.16}$$

5.2.1.6 The analysis here treats $\dot{x}$ as a well-defined random variable and ignores the subtleties introduced by the Brownian motion. Just as with the KL for f, a formal treatment would calculate the KL as the integral over time of the Radon-Nikodym derivative by applying Girsanov's theorem, but the final result is essentially the same (Särkkä & Solin, 2019, Chapter 7; Øksendal, 2013, Chapter 8; Archambeau et al, 2007; Matthews et al, 2016). The only caveat is that we must have $\hat{\mathsf{Q}}' = \mathsf{Q}$ in continuous time or else the KL diverges.

5.2.1.7 **The log conditional likelihood term** The linear Gaussian conditional likelihood's expectation over q depends only on the first two moments of the

approximate posterior marginals. Let $q(x_t)$ have mean μ_t and covariance Σ_t, and then

$$\int q(x) \log p(y|x)\, dx = \sum_{j=1}^{T} \int q(x_{t_j}) p(y_{t_j}|x_{t_j})\, dx_{t_j} \tag{5.17}$$

$$= -\frac{T}{2} \log |2\pi \mathsf{R}| - \frac{1}{2} \operatorname{tr}\left(\mathsf{R}^{-1} \sum_{j=1}^{T} [(y_{t_j} - \mathsf{C}\mu_{t_j})(y_{t_j} - \mathsf{C}\mu_{t_j})^\top + \mathsf{C}\Sigma_{t_j}\mathsf{C}^\top]\right). \tag{5.18}$$

5.2.1.8 **Gaussian approximations for u** As with conjugate GP regression, we have

$$q(f|u) = \mathcal{GP}(\zeta(\bullet) + \mathsf{K}_{(\bullet)u}\mathsf{K}_{uu}^{-1}(u - \zeta_u),\ k(\bullet, \star) - \mathsf{K}_{\bullet u}\mathsf{K}_{uu}^{-1}\mathsf{K}_{u\star}) \tag{5.19}$$

and in Equation (5.15), this provides the expressions for $\int q(f) f(x_t)\, df$ and $\int q(f) f(x_t) f(x_t)^\top\, df$ which are the approximate posterior marginal first and second moment respectively, evaluated at x_t. To make progress, we need to establish the form of $\hat{f}_t$. If it is chosen independently of f (that is, $q(x, f) = q(x)q(f)$) then $\mathcal{E}_t$ is quadratic in u, and so it follows that the whole of $\int q(f) D_{KL}(q(x)||p(x|f))$ is quadratic in u.

5.2.1.9 At the same time, the KL between $q(f)$ and $p(f)$ is (Matthews et al, 2016)

$$D_{KL}(q(f)||p(f)) = D_{KL}(q(u)||p(u)) \tag{5.20}$$

and these are the only u terms. Then

$$\arg\max_{q(u)} \mathcal{L}' = \arg\min_u D_{KL}(q(u)||p(u)) + \int q(f) D_{KL}(q(x)||p(x|f))\, df \tag{5.21}$$

$$= \arg\min_u \int q(u) \log \frac{q(u)}{p(u)}\, du + \int q(u) \int q(f|u) D_{KL}(q(x)|p(x|f))\, df\, du$$

$$= \arg\min_u \int q(u) \log \frac{q(u)}{p(u) e^{-\int q(f|u) D_{KL}(q(x)|p(x|f))\, df}}\, du \tag{5.22}$$

which is like the KL divergence except that the denominator is not normalised. Then the solution is $q(u)$ proportional to the denominator, and the minimum value is the normalising constant of the denominator. But $p(u)$ is Gaussian, and the exponent in the denominator is quadratic in u, so the solution is overall Gaussian, say $q(u) = \mathrm{N}(\mu_u, \Sigma_u)$.

5.2.1.10 Assuming Q is diagonal, and using the expected covariances

$$\left.\begin{aligned}
\Psi_{ff,t} &= \int q(x_t) k(x_t, x_t)\, dx_t \\
\tilde{\Psi}_{ff,t} &= \int q(x_t) \mathsf{K}_{uf(x_t)} \mathsf{K}_{uf(x_t)}^\top\, dx_t \\
\Psi_{f\hat{f},t} &= \int q(x_t) \mathsf{K}_{uf(x_t)} (\hat{f}_t(x_t) - \zeta(x_t))\, dx_t = \Psi_{\hat{f}f,t}^\top \\
\Psi_{\hat{f}\hat{f},t} &= \int q(x_t) (\hat{f}_t(x_t) - \zeta(x_t))(\hat{f}_t^\top(x_t) - \zeta(x_t))\, dx_t,
\end{aligned}\right\} \tag{5.23}$$

let $\Psi_{ff} = \sum_{t=1}^{t_T - 1} \Psi_{ff,t}$ in discrete time, or $\Psi_{ff} = \int_{t_1}^{t_T} \Psi_{ff,t}\, dt$ in continuous time, and similarly for the other three quantities. Then it is straightforward to show that

$$\mu_u = \zeta_u + \mathsf{K}_{uu} \mathsf{G}^{-1} \Psi_{f\hat{f}} \tag{5.24}$$

$$\tilde{\mu}_u = \mu_u - \zeta_u \tag{5.25}$$

$$\Sigma_u = (\mathsf{Q} \otimes \mathsf{I}_M) \mathsf{K}_{uu} \mathsf{G}^{-1} \mathsf{K}_{uu} \tag{5.26}$$

$$\begin{aligned}
\operatorname{tr}(\mathsf{Q}^{-1} \mathcal{E}_t) = \operatorname{tr}\Big(&(\mathsf{Q}^{-1} \otimes \mathsf{I}_M)(\mathsf{K}_{uu}^{-1} \tilde{\Psi}_{ff,t} \mathsf{K}_{uu}^{-1} (\tilde{\mu}_u \tilde{\mu}_u^\top + \Sigma_u - \mathsf{I}) - 2 \mathsf{K}_{uu}^{-1} \Psi_{f\hat{f},t} \tilde{\mu}_u^\top)\Big) \\
&+ \operatorname{tr}(\mathsf{Q}^{-1} \Psi_{ff,t})
\end{aligned} \tag{5.27}$$

$$\text{with} \qquad \mathsf{G} = (\mathsf{Q} \otimes \mathsf{I}_M) \mathsf{K}_{uu} + \tilde{\Psi}_{ff}. \tag{5.28}$$

5.2.1.11 Detailed calculations are deferred to Appendix A.5. The form in Equation (5.27) is suitable for accumulating the objective in a forward pass, but for some settings it is more efficient to simplify the total over t directly, for which the expression is given in Appendix A.5.

5.2.1.12 Even if $q(x|f) \neq q(x)$, we can set $q(u) = \mathcal{N}(\mu_u, \Sigma_u)$ as an approximation, and optimise μ_u, Σ_u jointly with the parameters. In either case it follows that

$$q(f) = \mathcal{GP}(\zeta(\bullet) + \mathsf{K}_{(\bullet)u} \mathsf{K}_{uu}^{-1} (\mu_u - \zeta_u),\ k(\bullet, \star) - \mathsf{K}_{\bullet u} (\mathsf{K}_{uu}^{-1} - \Sigma_u) \mathsf{K}_{u\star}) \tag{5.29}$$

and the KL between this and $p(f)$ emerges exactly as in the GP regression case.

$$\begin{aligned}
D_{KL}(q(f)||p(f)) = \frac{1}{2} \big[&\operatorname{tr}(\mathsf{K}_{uu}^{-1} (\mathsf{K}_{uu} \Sigma_u \mathsf{K}_{uu} + (\mu_u - \zeta_u)(\mu_u - \zeta_u)^\top)) \\
&- \log |e \Sigma_u \mathsf{K}_{uu}| \big].
\end{aligned} \tag{5.30}$$

5.2.1.13 Then the training objective overall is calculated by putting together Equations (5.11), (5.12), (5.18) and (5.30).

5.2.1.14 **Gaussian approximations for x** Suppose that x is indeed treated as independent of f. Then one option for $q(x)$ is as jointly Gaussian, or equivalently as generated by an approximating time varying LGSSM.

$$\hat{f}_t(x_t) = \hat{\mathsf{A}}'_t x_t + \hat{b}'_t \tag{5.31}$$

5.2.1.15 **The direct parameterisation** We can make the identity mean function setting essentially equivalent to a zero mean function by later shifting $\hat{\mathsf{A}}'_t$ by I. Then there is an iterative smoothing algorithm which calculates the optimal variational parameters $(\hat{\mathsf{A}}'_t, \hat{b}'_t, \hat{\mathsf{Q}}'_t)$ (Archambeau et al, 2007; Duncker et al, 2019), though it is slow to converge and not particularly stable, so it is typically preferable to use gradient-based optimisation jointly with the model parameters (Frigola-Alcalde, 2014).

5.2.1.16 In either case, one issue is that the number of variational parameters scales poorly with the length of the trajectory, and becomes infinite in continuous time. Some pragmatic ways to deal with this are to optimise the parameters at only a subset of times and then interpolate (Archambeau et al, 2007) or train a parametric recognition network to produce recover the variational parameters by filtering the measurements (Eleftheriadis et al, 2017).[1]

5.2.1.17 **Kalman filtering** I suggest a different solution: use the Kalman smoothing algorithms of Section 1.4.1, which are well established as suitable methods for nonlinear Gaussian state inference. Kalman filtering can be straightforwardly extended to continuous time with almost identical update equations (Särkkä & Solin, 2019, Chapter 10).

5.2.1.18 Kalman smoothing can be performed in a few different ways in continuous time, for example taking the limit of the discrete time smoother, or directly forming a Gaussian approximation to the solution to the exact smoother SDE.

[1] The latter scheme is often referred to as amortised variational inference. The origin of this term is a move from learning variational parameters for each measurement to learning variational parameters which pool information from many observations. It is a little inappropriate to use this term here, since the naive variational parameters already influence all subsequent measurements.

5.2.1.19 Since f is stochastic, $\hat{\mathsf{A}}', \hat{b}'$ are computed by linearising the mean function (from Equation (5.29); equivalently, these can be viewed as the expectation of the linearisation parameters over f).

5.2.1.20 It is then also possible to extend this model in various ways by leveraging the rich literature on Kalman filtering. For example, these methods can be efficiently paralellised using a parallel scan approach (Särkkä & García-Fernández, 2020; Yaghoobi et al, 2021).

5.2.1.21 **Correlated approximations** Since including correlations between f and x will make the posterior approximation more flexible, it may also improve the quality of learning. With Kalman filtering, this can be performed by linearising $f|u$ (Equation (5.19)). Then $\mathcal{E}_t$ is estimated by simple Monte Carlo over u.

5.2.1.22 **Nonlinear correlated approximations** Arguably, it is preferable to also use a nonlinear Gaussian transition distribution. One option is to directly leverage samples from f (Doerr et al, 2018; Mattos & Barreto, 2019) though this has been shown to be pathologically sensitive to noise in the data by Ialongo et al (2019) who proposed sampling from a parameterised linear modification of Equation (5.19).

5.2.1.23 The common issue with all of these methods is that learning becomes orders of magnitude slower than analytical methods; if it is really necessary to produce a closer approximation to the dynamics then it would be preferable to use a fully sample-based approximation to the variational optimum (Fan et al, 2023).

5.2.1.24 Hence, I focus on the analytical approximations provided by Kalman smoothing, and whether they can be used to deliver satisfactory learning. Note that these constrain the variational maximisation problem by restricted the possible $q(f, x)$. This could improve learning if the restriction removes optima with problematic biases.

5.2.1.25 **The necessary covariance function expectations** The expectations in Equation (5.23) are fairly straightforward to evaluate in this case. Assuming that inducing points are used, all that is needed is Gaussian expectations of the covariance functions, and of $k_d(z_m, x_t)x_t, d \in \{1{:}D\}$, $m \in \{1{:}M\}$. In the case of the squared exponential covariance function, these are available in closed form (Appendix A.5); a more general recipe is to use Gaussian quadrature or σ-points (Paragraph 1.4.1.32).

5.2.1.26 **A natural choice?** Recall that (z, u) together act as a proxy for the inputs and output of f. This is what x is, so separately optimising $q(u)$ and $q(x)$ should not be necessary. In any case, it seems far more natural to determine the state distribution automatically from the function distribution: after all, $p(x|f, y)$ is the solution to the smoothing problem for which Kalman smoothing is a widely used solution.

5.2.1.27 The uncorrelated version should provide good quality inference when samples of f do not locally have much difference in their linearisation (for example, the variance in f is low, or the uncertainty in the states is much less than the lengthscales of f). The correlated version should work well for systems where local linearisations are a good approximation.

5.2.1.28 **Reduced parameterisation** In the uncorrelated case, I update $q(u)$ in closed form using the states, then update the states using Kalman smoothing, meaning that the only variational parameters are the inducing points z. In the correlated case, $q(u)$ is optimised using a gradient-based method, and (μ_u, Σ_u) are added to the variational parameters. In either case, there are no parameters for the latent trajectory, so there is no increase for longer trajectories or continuous time (unless it needs higher M).

5.2.2 Learning with less bias

5.2.2.1 **Learning C and R** Given an estimate of the marginal means and covariances, the VI-optimal likelihood parameters can be determined from Equation (5.18).

$$\mathsf{C} = \frac{1}{T}\left(\sum_{j=1}^{T} y_{t_j}\mu_{t_j}^\top\right)\left(\sum_{j=1}^{T}(\mu_{t_j}\mu_{t_j}^\top + \Sigma_{t_j})\right)^{-1} \tag{5.32}$$

$$\mathsf{R} = \frac{1}{T}\sum_{j=1}^{T}\left[(y_{t_j} - \mathsf{C}\mu_{t_j})(y_{y_j} - \mathsf{C}\mu_{t_j})^\top + \mathsf{C}\Sigma_{t_j}\mathsf{C}^\top\right] \tag{5.33}$$

5.2.2.2 However, in practice, the stability and speed of convergence may be better if we optimise these parameters directly using gradient based optimisation, rather than using this EM approach.

5.2.2.3 **Learning Q** From Equations (5.12) and (5.15) we can also characterise the optimal Q. If all the transition parameters $\hat{f}, \hat{Q}_t$ are treated as fixed, then the unconstrained

optimum is at

$$\text{either} \quad \mathsf{Q} = \frac{1}{T}\sum_{t=1}^{t_T-1} \mathcal{E}_t \tag{5.34}$$

$$\text{or} \quad \mathsf{Q} = \frac{1}{t_1 - t_T}\int_{t_0}^{t_T} \mathcal{E}_t \, dt \tag{5.35}$$

in discrete time and continuous time repsectively. The main component is the average squared error between f and $\hat{f}$. Then it is clear that making $\hat{f}$ more flexible reduces the bias: if the squared error goes to zero, then the optimum collapses to $\mathsf{Q} = \hat{\mathsf{Q}}'_t$. Otherwise, the process noise appears to be inflated by the model mismatch.

5.2.2.4 If $\hat{f}_t$ is uncorrelated with f, this is conflating two different kinds of uncertainty. If there is high uncertainty about f (high posterior variance), it should manifest as uncertainty between trajectories, as different samples of f will lead to different dynamics. But this will inflate the squared error term, increasing the learnt Q. But higher Q means more uncertainty in the evolution of each trajectory, even for a fixed f.

5.2.2.5 If $\hat{f}_t$ is nonlinear, it will be able to match f better, even when there is high uncertainty about the state near significant nonlinearity (Figure 1.9).

5.2.2.6 The situation for the observation noise is similar: it is inflated by the marginal variance of the states at the observation times.

5.2.2.7 Nonetheless, state inference by Kalman smoothing may be sufficient to significantly reduce the bias. This will constrain the state trajectory to be reasonable in the context of the measurements and the current estimate of $q(f)$, rather than following the variational optimum (though the optimum for Q is still characterised by Equations (5.34) and (5.35)).

5.2.2.8 **The cost of learning** When $q(x)$ is Gaussian, each evaluation of the objective requires recovering the marginals from the parameters, which requires a forward pass through the trajectory, with each time step's cost in $\mathcal{O}(D^3)$, due to matrix

multiplications such as $\hat{\mathsf{A}}'_t\hat{\boldsymbol{\Sigma}}_t\hat{\mathsf{A}}'^{\top}_t$. Accumulating the expected log conditional likelihood terms adds $\mathcal{O}(\Delta^3)$ per observation.

5.2.2.9 Suppose that there are $\tilde{T}$ time steps in the latent trajectory. In discrete time, $\tilde{T} = t_T - t_1$, but in continuous time, $\tilde{T}$ depends on the choice of integrator. Then the total cost of the forward pass, accounting also for calculations related to the variational approximation to the GP, is $\mathcal{O}(\tilde{T}D^3 + T\Delta^3 + DM^3)$. If Kalman smoothing is used, the cost is increased, since a few extra matrix operations are needed, and there is an additional backward pass over time. Typically, nonlinear correlated schemes have cost of a similar order (Ialongo et al, 2019).

5.2.2.10 The cost of calculating gradients by automatic differentiation will be proportional to the cost of calculating the objective itself. For Kalman smoother based schemes, the negative impact of this can be reduced slightly by not differentiating the smoothing algorithm, but treating the means and covariances as fixed.

5.2.2.11 Altogether the cost of a single objective function evaluation is not much different between schemes. Those which have fewest parameters – such as the uncorrelated Kalman smoother schemes – would typically converge more quickly. It remains to determine if their optima are competitive with more freely parameterised schemes.

5.2.2.12 **The experimental data generating processes** I consider two illustrative toy examples in discrete time. The first is a linear model, where inference should be easy, with (before process noise)

$$x_{t+1} = e^{-0.1}x_t. \tag{5.36}$$

The second is the well-known logistic model, a discrete time nonlinear 1D system which exhibits a wide variety of interesting phenomena (Hirsch et al, 2004, Chapter 15).

$$x_{t+1} = 3.33x_t(1 - x_t) \tag{5.37}$$

The parameter value is chosen so that the trajectories are stable oscillations. This is that same dynamics function used in Figures 1.8 and 1.9.

5.2.2.13 **The models** I fix the measurement model with $\mathsf{C} = 1$, and vary the groundtruth P_{gt} and L_{gt}, the square roots of the noise variances. I consider learning using

1. *(DP)* a directly parameterised baseline where $\hat{\mathsf{A}}'_t, \hat{b}'_t, \hat{\mathsf{Q}}'_t$ are learnt freely;

2. *(DP, Q matched)* a variant on DP where $\hat{\mathsf{Q}}'_t = \mathsf{Q}$;
3. *(EKS-SG, SLS-SG)* Kalman smoothing using the EKS or SLS, where for computational efficiency the smoothing algorithm is not differentiated (the parameters of $q(x)$ are treated as fixed);
4. *(SLS)* Kalman smoothing using the SLS, with the smoothing algorithm differentiated;
5. *(Conditional SLS)* Kalman smoothing using the SLS conditional on samples of u (a correlated approximation).

5.2.2.14 The purpose of DP, Q matched is to check that it is not just this aspect of the posterior construction which makes smoothing methods perform better. Since it only performs modestly better than DP, the experiments suggest that the construction of $\hat{\mathsf{A}}', \hat{\mathsf{b}}'$ is more significant.

5.2.2.15 **Training** All of the models are trained using the Adam optimiser (Paragraph 1.2.2.6); for optimisation to progress well, it is generally necessary to fix the noise parameters to a moderate value, and train the other parameters for a while before starting to train the noise. This initial value cannot be too small, since when the noise is low, a small error in the predictions leads to a large penalty.

5.2.2.16 To start the latent trajectory with reasonable values, I set the initial state's mean to the first measurement, $\mathsf{C} = 1$, and a low initial state variance. For both DP variants, I set $\hat{b}'_t$ to y_t, and initialise $\hat{\mathsf{A}}'_t$ small. Further details on the experimental setup are in Appendix C.3.

5.2.2.17 I run the experiments for three randomly generated sequences ($N = M = 10$) for each of $\mathsf{P}_{gt}, \mathsf{L}_{gt}$ in $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ and a very high noise setting (1 for linear dynamics, and $10^{-\frac{1}{2}}$ for the logistic model). Though the training set size is small, and the data generating processes are fairly simple, they serve to illustrate the main points.

5.2.2.18 **Plotting latent quantities** Due to learning C, the scale of the latents, and of L and f, can vary together without changing the likelihood or predictions. In the plots,

I scale the learnt $\mathsf{L}, z, \mu_u, \sqrt{\Sigma_u}$, the test L in Figure 5.7, and the posterior predictive mean and standard deviation by C to ensure they are all comparable.

5.2.2.19 **The learnt noises** Figures 5.5 and 5.6 show how the learnt noise varies with the generating noise. These plots draw an arrow from the ground truth values of (L, P) to the learnt values. Perfect identification would yield arrows of zero length. Perfect identification of the process noise would yield vertical arrows, and perfect identification of the measurement noise would yield horizontal arrows.

5.2.2.20 Across all variational approximations, the model generally conflates measurement and process noise, which are hard to distinguish in this setting. This can be seen in the plots by learnt values concentrated much closer to the line of $\mathsf{P} = \mathsf{L}$ than the ground truth values. The directly parameterised variants both learn high noise, with slightly better identification in the linear case. The methods based on Kalman smoothing do much better at identifying the value of the larger noise. This can be identified on the plots as near-vertical arrows in the lower right triangle (where ground truth process noise is larger than measurement noise), and, conversely, near-horizontal arrows in the upper left triangle. Moreover, when the measurement noise is low, it also conflates them less. This can be identified in the plots with the learnt values being further from the line $\mathsf{P} = \mathsf{L}$ when the ground truth measurement noise is lower.

5.2.2.21 In summary, the Kalman smoothing based approximations yield less biased parameter estimated under lower noise conditions than directly parameterised approximations. This is not a feature of fixing the value of Q (since the behaviour of DP, Q matched is comparable to DP), and can therefore be attributed to the useful constraint that smoothing places on the latent trajectories. Using a correlated model does not appear to be advantageous; indeed, the arrows tend to be better (closer to vertical or horizontal, and terminating further from the line of $\mathsf{P} = \mathsf{L}$) when using an uncorrelated model.

5.2.2.22 **Sensitivity of the learnt noises** To get further insight into these results, in Figures 5.7 and 5.8 I plot the sensitivity of the training objective to changes in L (Figure 5.7) or P (Figure 5.8) only. All other parameters are kept fixed at the learnt optimum. This shows the shift in the learnt optima, but also that in lower noise conditions, the Kalman smoother based objectives (black and blue) have wider

peaks, suggesting that the objective's surface is a little better behaved. This may explain why learning using those objectives is able to reach better optima.

5.2.2.23 The process noise places a penalty on mismatch between the posterior states, and what is predicted by $f(x_t)$. The observation noise places a penalty on mismatch between the observations and what is predicted by Cx_t. As the noise gets smaller, generally this penalty will get larger. Smoothing methods automatically adapt the state distribution to the noise model, which explains these wider peaks.

5.2.2.24 **Examining the posteriors** To understand this in terms of the approximate posteriors, consider the posterior predictive plots in Figures 5.9 to 5.12. These overlay the predictive distributions and uncertainties (in red and black) with the groundtruth (in blue). The means of $q(x)$ and the orientation of $\hat{\mathsf{A}}'_t$, the correlation between states, is shown with black arrows.

5.2.2.25 Even when the function is fairly well identified (the linear case in low noise), DP leads to the correlations not matching up with the function. Then the data must be explained by noise. Yet with Kalman smoothing, the correlations line up with the function automatically, even with an uncorrelated model. Note that this works well even for data points at the turning point of the logistic, where the linearisation is less of a good approximation (Figure 1.9).

5.2.2.26 Even when the function is not well-identified, such as high noise with the logistic, the linearisations do not vary much across samples of u, so the correlated model does not gain much over the uncorrelated one. Of course, the uncorrelated model also has the advantage of being far cheaper computationally.

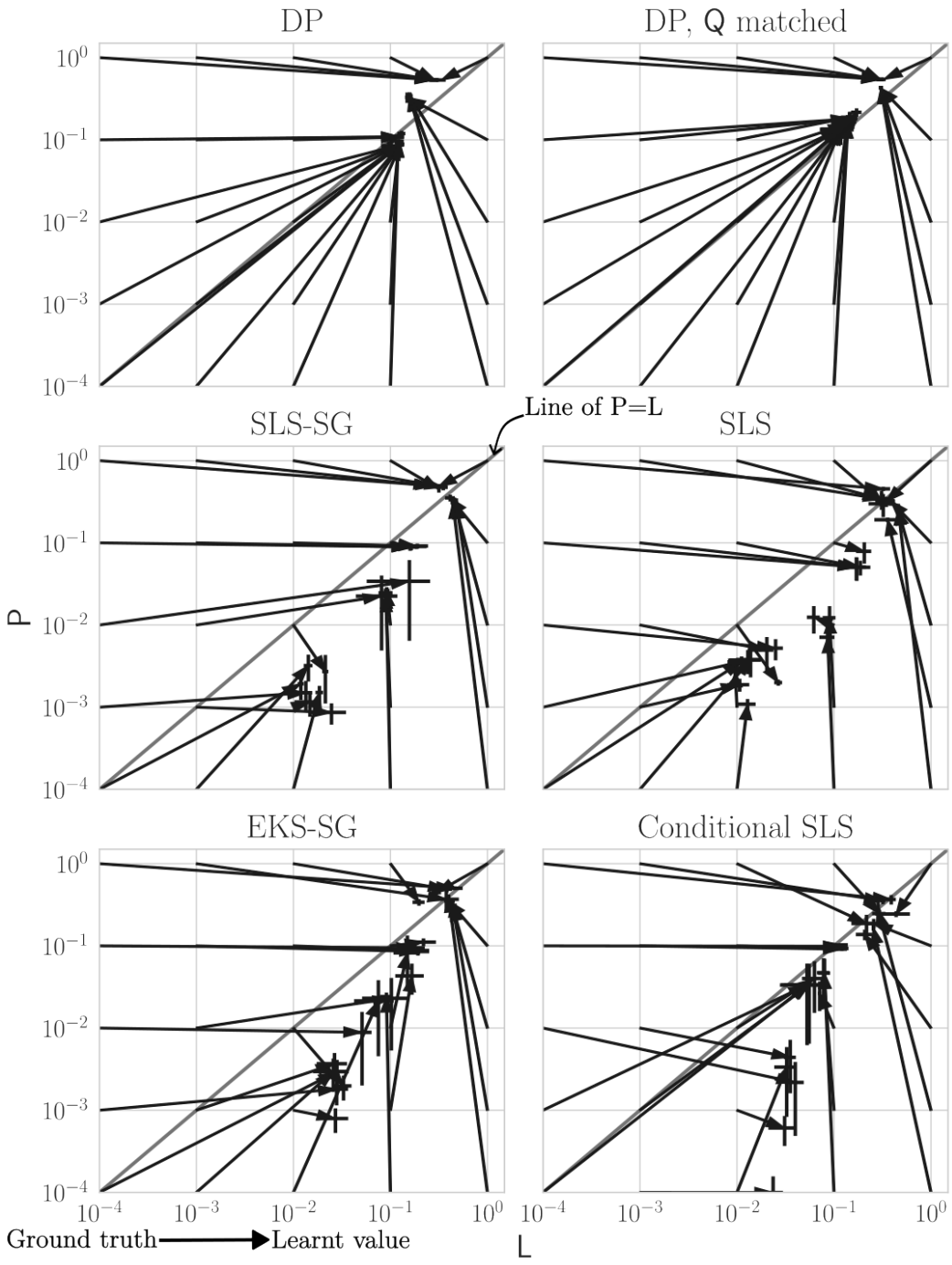


Figure 5.5 (Linear model) Arrows connecting the groundtruth noise standard deviations to the learnt values with 2σ error bars. Vertical arrows show well-identified process noise, and horizontal arrows show well-identified measurement noise. Arrows terminating near the diagonal line show conflation of process and measurement noise. Kalman smoothing methods lead to much lower learnt noise than direct parameterisation, and identifies the process noise (L) well in low measurement noise (P) settings.

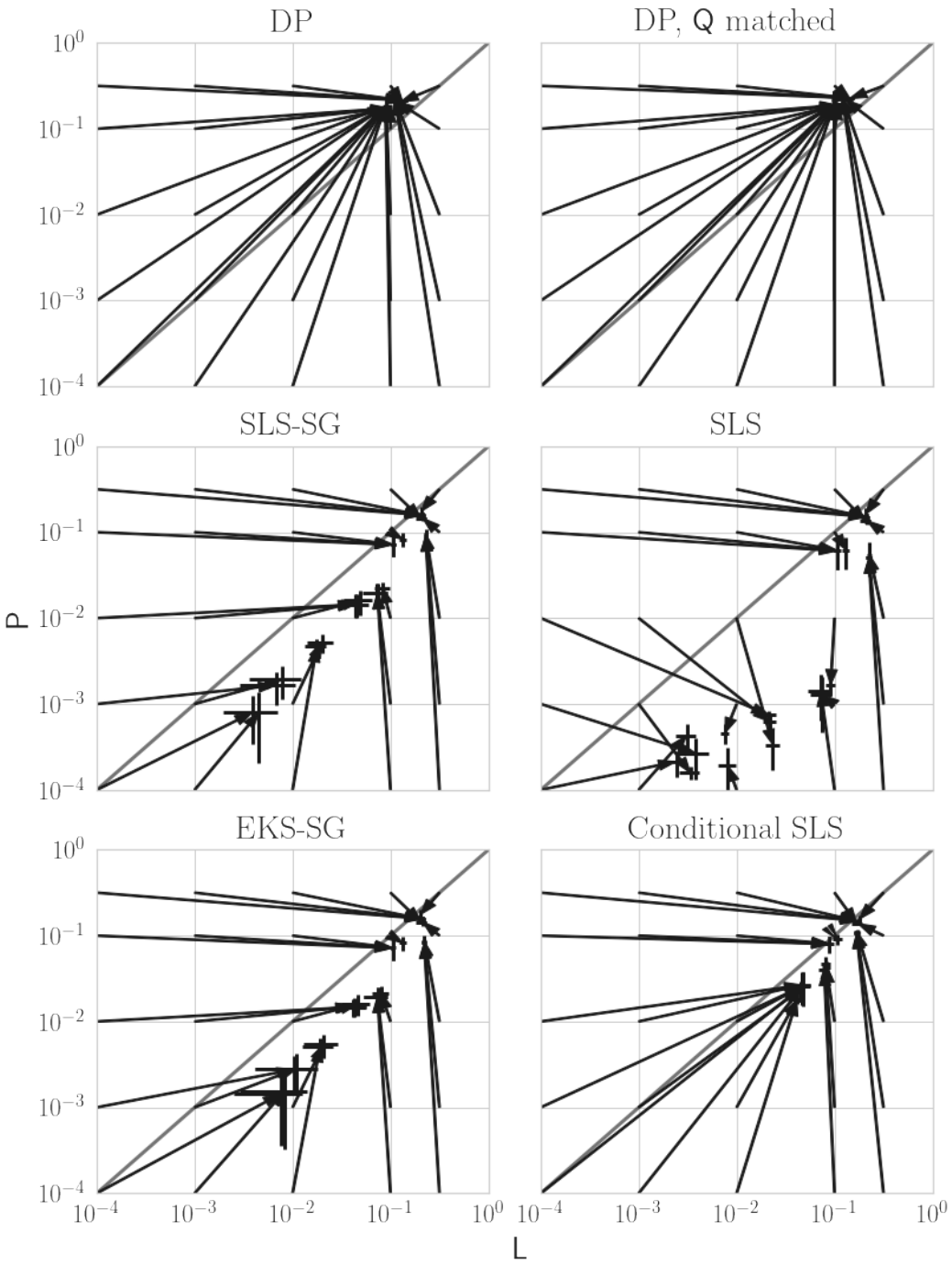
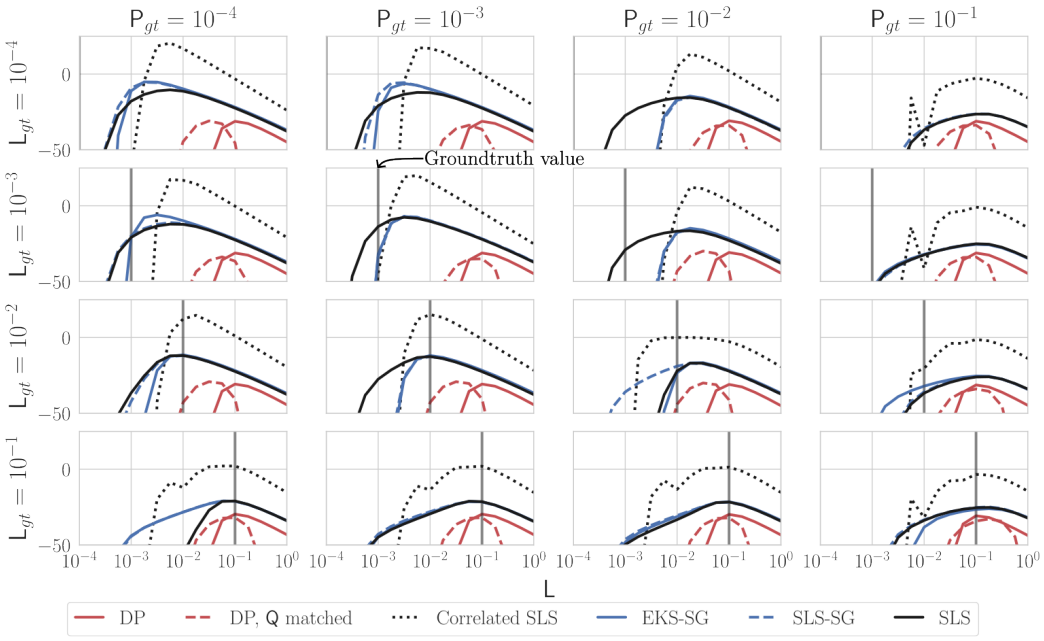
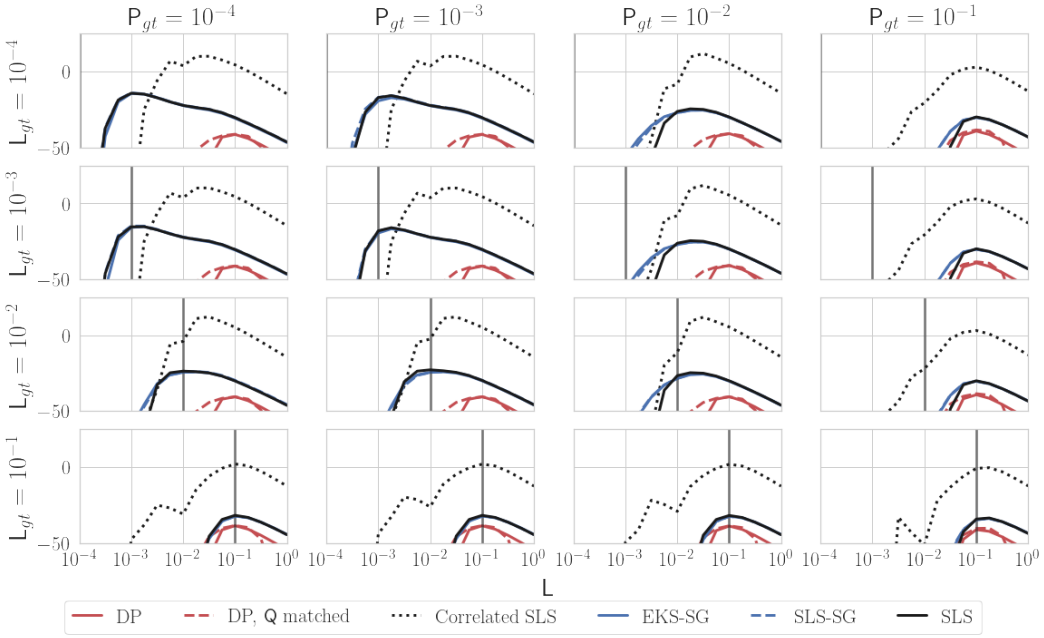


Figure 5.6 (Logistic model) Arrows connecting the groundtruth noise standard deviations to the learnt values with 2σ error bars. Vertical arrows show well-identified process noise, and horizontal arrows show well-identified measurement noise. Arrows terminating near the diagonal line show conflation of process and measurement noise.

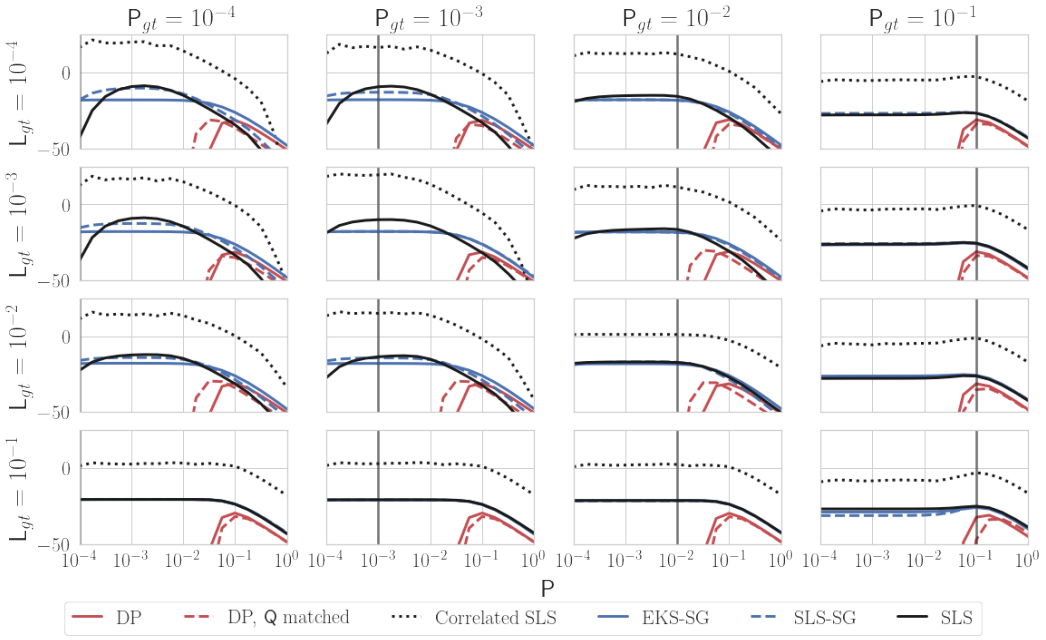


(a) Linear

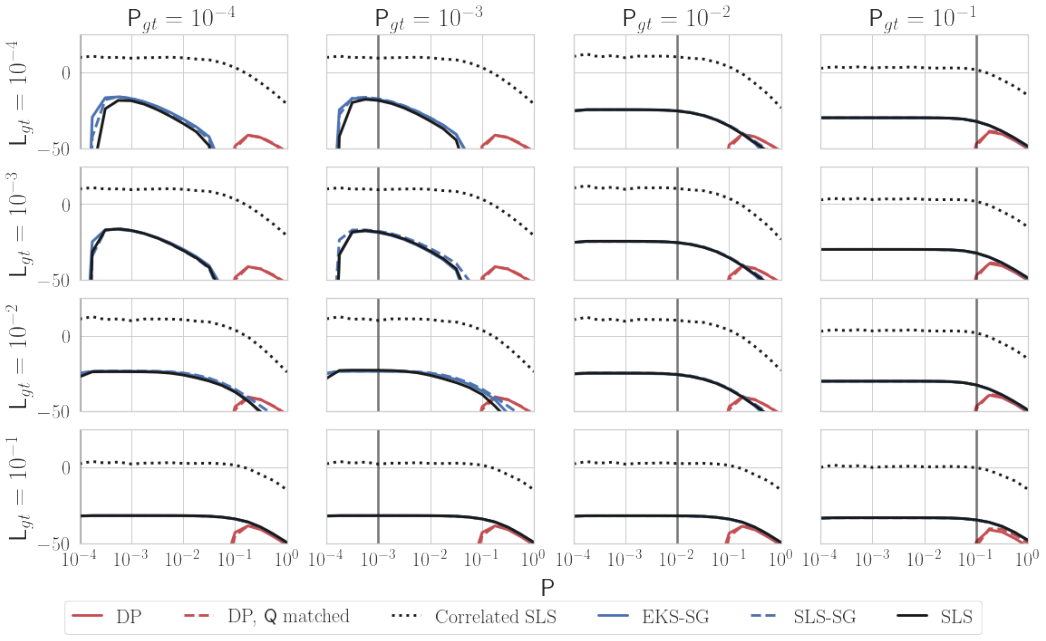


(b) logistic

Figure 5.7 Sensitivity of the learnt optima to variations in L, keeping all other parameters fixed. Flatter curves show lower sensitivity. The groundtruth value is shown as a vertical grey bar. Under lower noise conditions, the optima of Kalman smoothed methods are less biased (closer to the vertical bar), and less sensitive (flatter objective surface); the uncorrelated Kalman smoother in particular has its peaks fairly well aligned with the groundtruth, except when the measurement noise becomes larger than the process noise.



(a) Linear



(b) logistic

Figure 5.8 Sensitivity of the learnt optima to variations in P, keeping all other parameters fixed. Flatter curves show lower sensitivity. The groundtruth value is shown as a vertical grey bar. Under lower noise conditions, the optima of Kalman smoothed methods are less biased. Under high noise conditions, the models learn high process noise. For Kalman smoother methods, this makes them insensitive to the observation noise, since the trajectories will follow the observations.

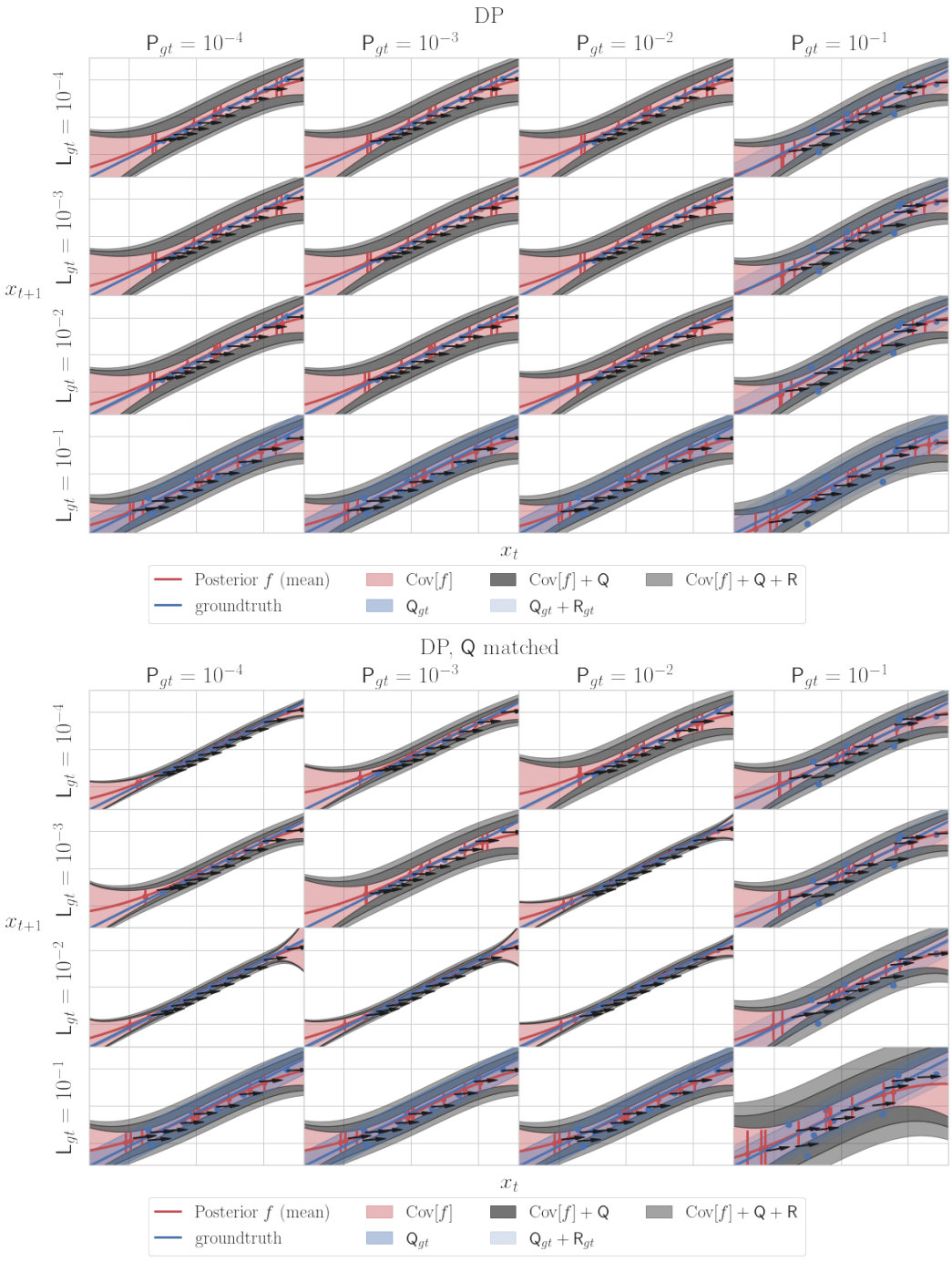


Figure 5.9 (Linear generating model, directly parameterised variational approximations) Learnt dynamics posteriors (red and black) overlaid on the groundtruth (blue) for the directly parameterised methods. The black arrows start at the means of the pairwise marginals, and their orientation shows the correlations between adjacent states. Inducing variables are in red, and the data in blue. Generally, the posterior function variance is high even when the mean fit is good, and the learnt process noise is significant.

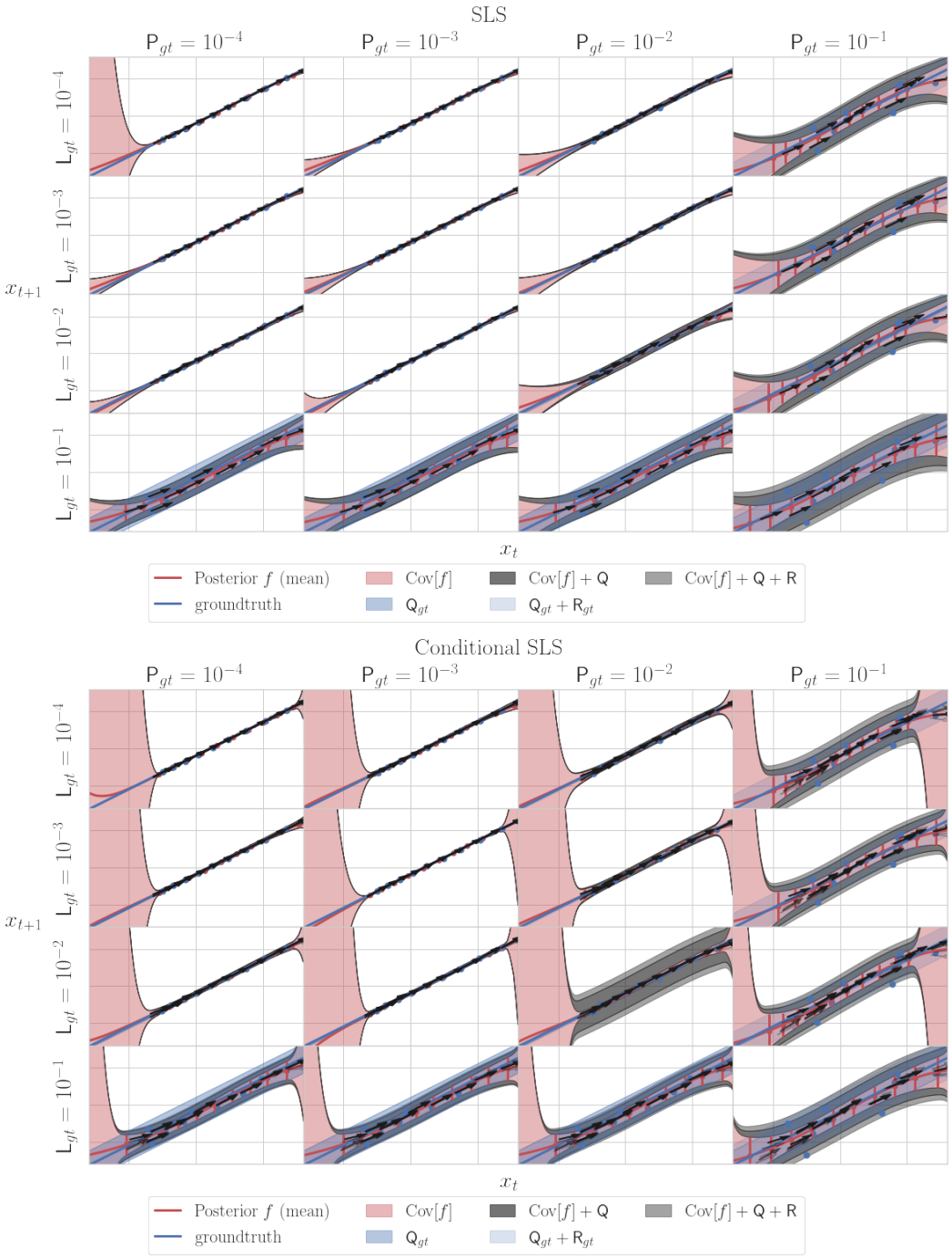


Figure 5.10 (Linear generating model, SLS based variational approximations) Learnt dynamics posteriors (red and black) overlaid on the groundtruth (blue) for methods using the statistically linearised smoother. The black arrows start at the means of the pairwise marginals, and their orientation shows the correlations between adjacent states. Inducing variables are in red, and the data in blue. There is a slight tendency to underestimate the process noise in low measurement noise settings.

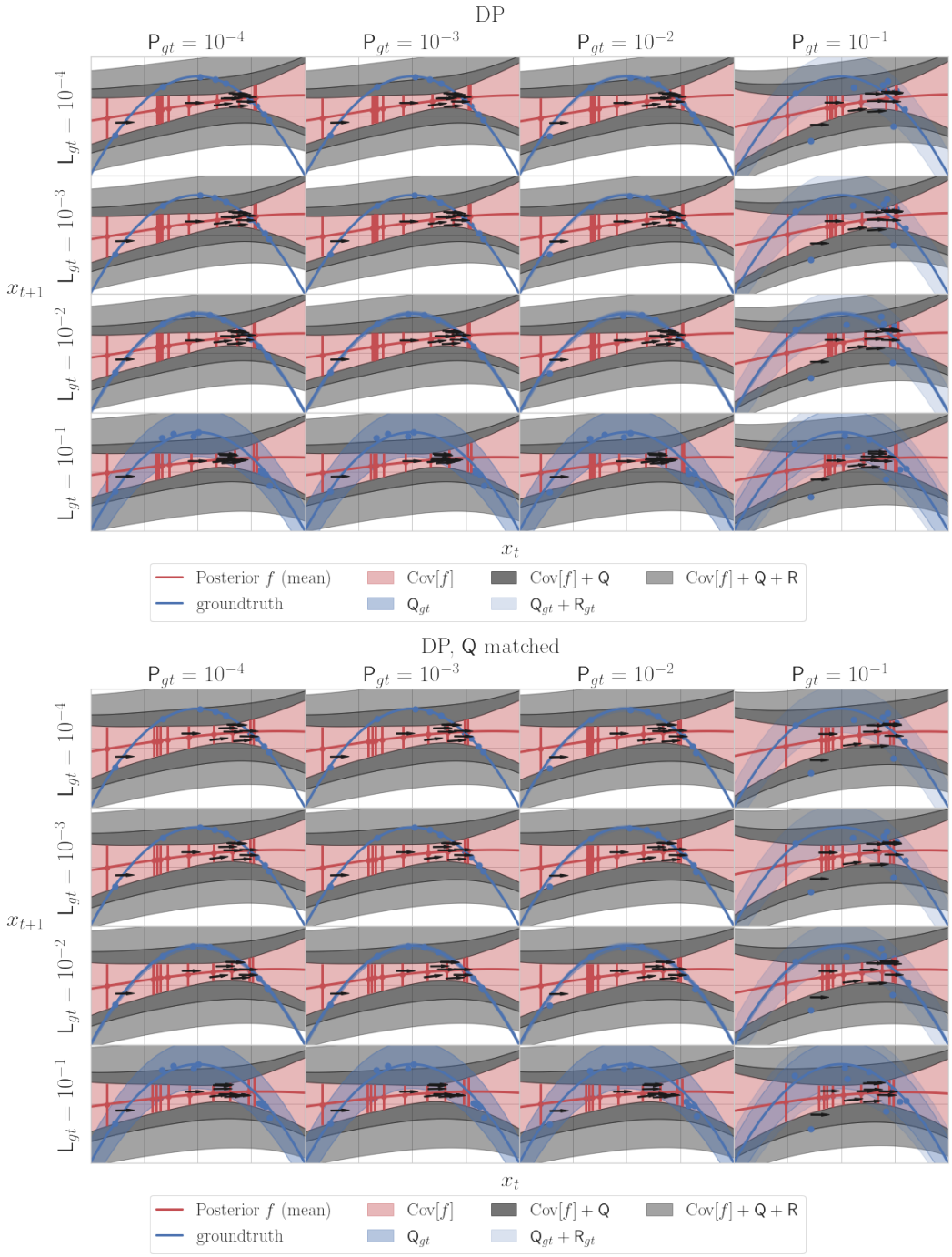


Figure 5.11 (Logistic generating model, directly parameterised variational approximations) Learnt dynamics posteriors (red and black) overlaid on the groundtruth (blue) for directly parameterised methods. The black arrows start at the means of the pairwise marginals, and their orientation shows the correlations between adjacent states. Inducing variables are in red, and the data in blue. The fits are poor, and the mismatch between the fit and the data is explained by noise.

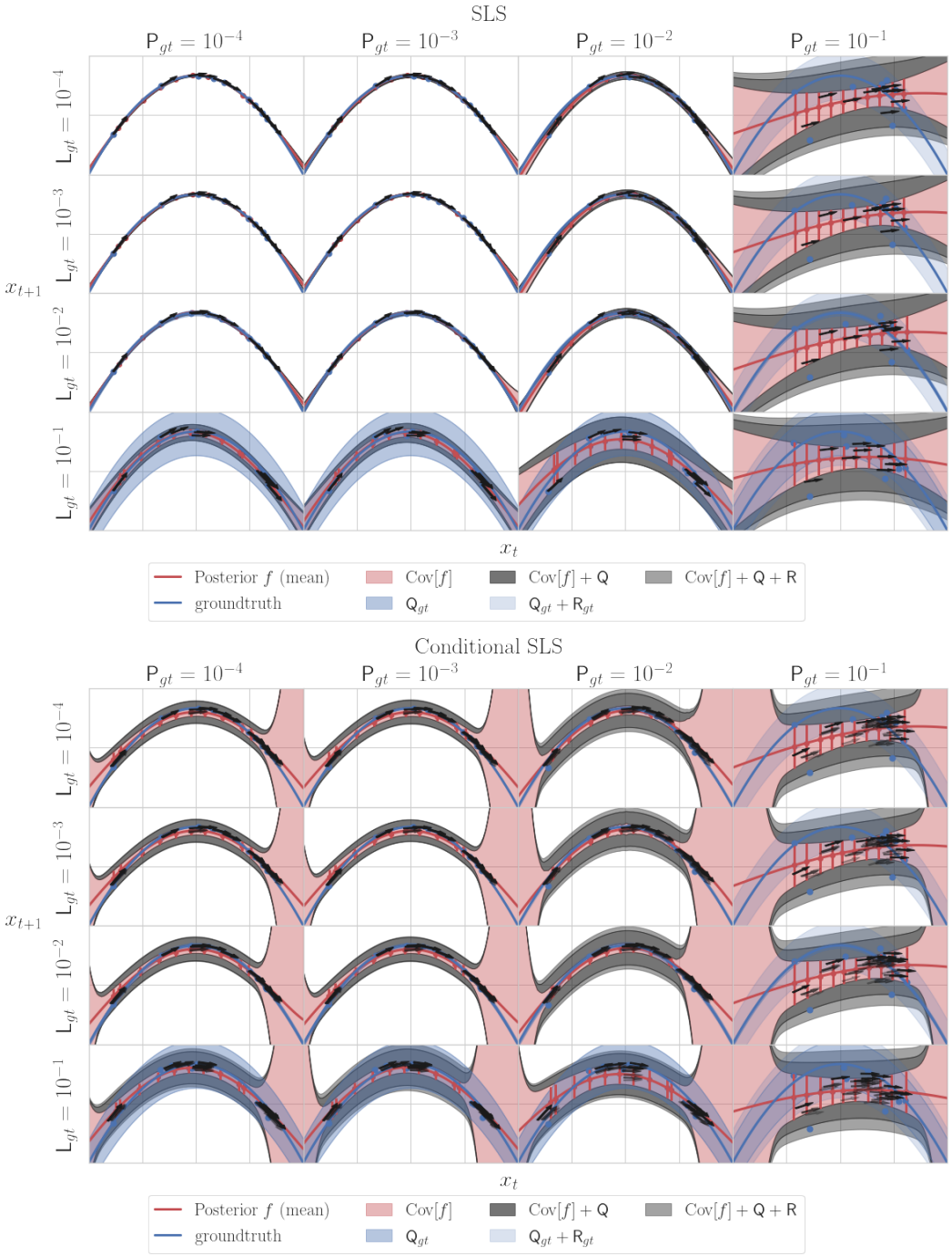


Figure 5.12 (Logistic generating model, SLS based variational approximations) Learnt dynamics posteriors (red and black) overlaid on the groundtruth (blue; logistic) for methods using the statistically linearised smoother. The black arrows start at the means of the pairwise marginals, and their orientation shows the correlations between adjacent states. Inducing variables are in red, and the data in blue.

5.2.2.27 **Computational aspects** Generally, the SLS-SG and EKS-SG variants are the easiest and fastest to train, being several times faster than either directly parameterised variant without taking advantage of recent speedups in Kalman filtering (Yaghoobi et al, 2021). Although differentiating the smoothing algorithm adds significant extra cost, I still found learning was faster than for directly parameterised approximations. The directly parameterised method in turn has been shown to orders of magnitude faster to train than a wide variety of sample-based approximations (Fan et al, 2023).

5.2.2.28 **Cost scaling in continuous time** To avoid having to store every intermediate state evaluation in continuous time, we can leverage the reversibility of the dynamics in between observations to directly define the differential equation for the adjoint (Chen et al, 2018).

5.2.2.29 **Summary of experimental findings** These experiments showed in simple and one dimensional settings that using Kalman smoothing based methods to construct the variational state distribution leads to less biased noise parameter estimates, while also being faster to train. They yielded the same or better benefits as correlated approximations despite requiring far less computational cost. The examination of the objective's sensitivity, and the posterior transition function, has given some useful intuition for why these methods work well. The Kalman smoother provides a useful constraint on the state posterior distribution which leads to a less sensitive objective function. This is the case even near the turning point of the logistic function, where the local linearisation of the Kalman filter is expected to be a worse approximation.

5.2.2.30 **Limitations of the findings** These experiments were limited to simple and one dimensional dynamical systems. It would be worthwhile to test whether these conclusions carry over to higher dimensional systems. Moreover, these experiments focus on the estimates of the important noise parameters. An informative extension for further work would be to explicitly test the impact of this on predictive performance in a real-world task.

5.3 Summary

5.3.1 In this chapter, I first considered the choice of model for Bayesian system identification. There are compelling reasons to model in continuous time if we believe the data is generated by a continuous time process, but this is less significant in favourable conditions – such as lower noise, more data, and regular measurements. Many of the issues may in any case be well addressed by using a discrete time model with an identity mean function, which corresponds to a particular discretisation of the continuous time model.

5.3.2 In the second part of the chapter, I showed that constructing the GPSSM variational posterior approximation by Kalman smoothing leads to favourable properties. In particular, the number of parameters to train is greatly reduced, even in continuous time, and the bias in the noise variance – thought to be a serious issue with uncorrelated variational approximations – is substantially reduced even without correlating the state with the function.

5.3.3 This only tests the parameter estimates in one dimensional synthetic settings, and does not explicitly evaluate the predictive performance in real-world settings. However, it offers a promising approach to scale or speed up GP system identification, at least in the sorts of settings where Kalman smoothing works well. An interesting direction would be to deploy this in an active learning task, as motivated in Example 1.2.

5.A Reference notes

5.A.1 Approximate inference in the GPSSM goes back to the MAP inference of Wang et al (2005), and using EP or power EP has also been studied (Bui, 2017, Chapter 3, Deisenroth & Mohamed, 2012).

5.A.2 Frigola-Alcalde (2014) used both sample-based and modern variational treatments. Another sample-based approach, which uses a reduced rank approximation to the covariance function, was developed by Svensson et al (2016). Eleftheriadis et al (2017) attempted to tackle issues with the variational approximation of Frigola-Alcalde (2014), such as the large number of parameters.

5.A.3 The limitations of the directly parameterised uncorrelated approximation motivated a lot of work using uncorrelated approximations (Curi et al, 2020; Doerr et al, 2018; Fan et al, 2023; Ialongo et al, 2019; Mattos & Barreto, 2019).

5.A.4 Very few have considered the continuous time setting in detail. Gaussian process approximations to SDE solutions were considered by Archambeau et al (2007), and their recipe was used by (Duncker et al, 2019) for the continuous time GPSSM. However, as noted in this chapter, many ideas for the discrete time setting can be readily transferred across to the continuous time setting; one only needs to take care with the scaling of the number of parameters, and various other computational considerations.

5.A.5 The discussion on preserving sparsity in the system was motivated by talk by Vinnicombe (2015).

5.A.6 There is a separate line of work in continuous time where no process noise is used. This does not allow for much model mismatch, and so generally some extra components are needed: either a mapping from the measurements to the latent representation is pretrained, or state observations are assumed (Hegde et al, 2022; Heinonen et al, 2018).

Chapter 6

Conclusions and directions

6.0.1 In this chapter, I conclude the main contributions of this thesis (Section 6.1) and give a partial overview of future research directions (Section 6.2). A longer summary is in Section 1.5.

6.1 Summary and conclusions

6.1.1 The work in this thesis was motivated by two types of applications: modelling spatial fields (Example 1.1) and identifying system dynamics (Example 1.2).

6.1.2 In both settings, GP regression offers a good tool for fitting data, endowed with posterior uncertainty estimates and with a good way of incorporating prior knowledge. Variational approximations offer a convenient analytical formulation for quick learning.

6.1.3 For the spatial modelling setting, this thesis was dominated by the consideration of precomputable approximations. Where stationary models are acceptable, the range of AFS approximations in Chapter 3 offer practical methods for very fast learning.

6.1.4 In that chapter, I showed that this family of approximations yields convergence with feature efficiency asymptotically comparable to that of widely-used inducing points (Table 3.1), and that in low dimensional spatial regression tasks, this can speed up learning by 2 to 8 times for typical performance requirements (Section 3.5); for gridded data and low performance requirements this can be made even faster

(Section 3.6). This comes with very few limits on the possible choice of stationary covariance function. This is immediately applicable to speed up learning in practice on very large geospatial datasets.

6.1.5 A noteworthy limitation of this family of methods is that the computational cost scales exponentially in the input dimension, and it remains an open question whether these can be suitably adapted to higher dimensional problems. Moreover, all the developments and evaluation focused on the conjugate setting. Some particular extensions and generalisations were discussed in Section 3.7.

6.1.6 Since many spatial fields are highly non-stationary, it is highly desirable to construct non-stationary approximations. In Chapter 4, I discussed how designing non-stationary covariance functions is more challenging. Stationary models can be viewed as all sharing a common basis (the Fourier basis), and covariance functions are defined by how they distribute their energy over that basis. There is no natural generalisation of this notion for non-stationary models (Section 4.2).

6.1.7 Gibbs' covariance function remains a very compelling choice for the kinds of non-stationarity which is of interest in spatial modelling. In Section 4.1.2, I described how to carry over the advantages of the sparse inducing points framework to a hierarchical Gibbs model, and demonstrated empirically (Section 4.4) that this leads to useful performance improvements. Those improvements come both in better generalisation in non-stationary data, and faster progress in active learning.

6.1.8 The multiresolution GPs constructed in Section 4.3 offer a promising alternative which can benefit from faster learning due to precomputability. This could be a useful advantage for regression tasks, though the empirical evidence in that chapter shows that further work is needed to find a competitive covariance function in this class, for example by improving the choice of wavelet or the parameterisation. It was also shown that while parameter learning is improved, active learning is prohibitively costly – a problem shared with other precomputable methods which rely on sparsity, such as that of Cunningham et al (2023).

6.1.9 For GP state-space models, in Chapter 5 I recommended not following the variational optimum for the state posteriors, but instead approximating them using Kalman smoothing. Not only does this lead to quicker learning, but it reduces bias

in the noise parameters. It can also straightforwardly be applied to continuous time modelling. The experiments in that chapter focused on synthetic and one dimensional examples to build understanding, and it remains necessary to test whether the advantage carries over to a real task.

6.1.10 Throughout, I have focused on analytical, variational, approximations. This is not because I believe these are always the best choice, or inherently good. Indeed, following the variational methodology led to good posteriors in Chapter 3 when compared to SKI, but not in Chapter 5, where it led to learning pathologically large noise variances.

6.1.11 The focus throughout has been on constructing good quality approximations in such a way that learning can be fast even with very large amounts of data. It has also been my focus to provide approximations that work well for a range of choices of prior – continuous or discrete time dynamics in the GPSSM, or a wider range of stationary priors with AFS – so that the prior can be chosen to best match the problem rather than to match the approximate inference algorithm.

6.2 Future directions

6.2.0.1 This thesis has set out a number of methods for building upon variational approximations for Gaussian process models in the sorts of applications where they are highly usable, such as low dimensional problems, or those where there is significant prior knowledge to exploit. The future directions to build upon this work fall into two groups: further improving methods, and transferring methods to application.

6.2.1 Further improvements

6.2.1.1 **Precomputable methods in higher dimensions** One issue with the precomputable methods of this thesis is that they are generally limited to very low dimensions in Euclidean space. Except in the case of gridded data, it remains an open question how to scale these methods up to higher dimensions, or whether it is possible at all without additional assumptions (such as isotropy). Some ideas for workarounds, such as treating auxiliary inputs as outputs, were given in Chapter 3,

but it would be preferable to have methods which can handle slightly higher dimensional inputs (say, up to around 10-20).

6.2.1.2 **Better multiresolution covariance functions** In Chapter 4, I suggested one particular multiresolution covariance function, and used two example wavelets in practice. It would be worthwhile to more thoroughly explore this class of covariance functions, perhaps guided by physical knowledge of a particular application. Some specific desirable improvements would be to guarantee a spatially invariant marginal variance, and to produce performance more competitive with Gibbs' covariance function.

6.2.1.3 **Beyond conjugate regression** The regression part of this thesis focused on the conjugate setting for simplicity. A sketch of how to use precomputable methods in non-conjugate settings was given in Section 3.7, without any guarantees, which would be worthwhile to investigate in practice.

6.2.1.4 **Refinements of the GPSSM** The Kalman smoother based variational approximations of Chapter 5 offer a practical way to quickly estimate the learning objective. It would be interesting to consider some practical refinements, such as parallelising the Kalman filter, or using a multiple iteration EP approximation for the state posterior.

6.2.2 Applications

6.2.2.1 **Precomputable approximations in probabilistic numerics** Fast learning may be of particular interest in probabilistic numerical algorithms, for example to approximate the solution to a differential equation system (Hennig et al, 2022; Wang et al, 2021). In these settings, rather than receiving measurements, the probabilistic numerical algorithm itself has control over the evaluation points.

6.2.2.2 It is common to use GPs as a prior – for instance, for the solution to a differential equation system, or for an integrand. Using more evaluation points should generally improve the accuracy of the result and reduce uncertainty. But this comes with the usual $\mathcal{O}(N^3)$ cost. Probabilistic numerical algorithms are usually not applied in contexts where they are much more computationally expensive than classical analogues.

6.2.2.3 One response to this is to try to select the evaluation points optimally or approximately optimally. However, an alternative enabled by fast precomputable methods is to select points suboptimally but use far more of them. For example, one option would be to select the evaluation points on a regular grid in order to leverage the speed of gridded AFS ($\mathcal{O}(M)$, not including the precomputation).

6.2.2.4 This is also of particular interest in the case of solving PDEs, where probablistic numerical methods are currently uncompetitive in computational cost with classical numerical algorithms.

6.2.2.5 **Applications of the GPSSM** There is now a rich family of usable approximations, including the faster Kalman smoother approximations in Chapter 5, so applying the GPSSM is of more interest. Bui (2017) reported promising results in initial experiments in active learning problems, and Duncker et al (2019) demonstrated the identification of dynamically interesting features such as equilibria and limit cycles, but neither of these have been transferred to real problems.

6.2.2.6 In both cases, there is an existing methodology available to use, and the approach of Chapter 5 can be plugged in to the parameter learning stage. A particularly interesting real-world problem to consider would be active learning of a simple biological system for scientific purposes, where there is expert knowledge available to help design the model (for example, to determine a suitable state dimension), and a clear gain to be made (a reduction in spend on expensive wet lab experiments).

6.3 Concluding remarks

6.3.1 This thesis has provided a number of usable methods to learn probabilistic models of functions faster or better in important task contexts, like low dimensional spatial modelling, or system identification. Although there is still plenty of room to build upon or refine these methods, they are also ripe for application to use in real engineering or scientific modelling problems.

References

Vincent Adam, Stefanos Eleftheriadis, Artem Artemev, Nicolas Durrande, and James Hensman. Doubly sparse variational Gaussian processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020. URL https://proceedings.mlr.press/v108/adam20a.html. 47

Vincent Adam, Paul Chang, Mohammad Emtiyaz E Khan, and Arno Solin. Dual parameterization of sparse variational Gaussian processes. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. URL https://papers.nips.cc/paper_files/paper/2021/hash/5fcc629edc0cfa360016263112fe8058-Abstract.html. 47

Laurence Aitchison, Adam Yang, and Sebastian W Ober. Deep kernel processes. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2021. URL https://proceedings.mlr.press/v139/aitchison21a.html. 135

Mauricio A Álvarez, David Luengo, and Neil D Lawrence. Latent force models. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2009. URL https://proceedings.mlr.press/v5/alvarez09a.html. 50

Mauricio A Álvarez, Lorenzo Rosasco, and Neil D Lawrence. Kernels for vector-valued functions: A review. *Foundations and Trends in Machine Learning*, 2012. doi: 10.1561/2200000036. URL http://inverseprobability.com/publications/alvarez-vector12.html. 21

Mauricio A Álvarez, David Luengo, and Neil D Lawrence. Linear latent force models using Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013. doi: 10.1109/TPAMI.2013.86. 50

Cédric Archambeau, Dan Cornford, Manfred Opper, and John Shawe-Taylor. Gaussian process approximations of stochastic differential equations. In *Gaussian Processes in Practice Workshop*, 2007. URL https://proceedings.mlr.press/v1/archambeau07a.html. 173, 176, 194, 230, 231

Renato Berlinghieri, Brian L Trippe, David R Burt, Ryan James Giordano, Kaushik Srinivasan, Tamay Özgökmen, Junfei Xia, and Tamara Broderick. Gaussian processes at the Helm(holtz): A more fluid model for ocean currents. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2023. URL https://proceedings.mlr.press/v202/berlinghieri23a.html. 21

Vladimir I Bogachev. *Gaussian Measures*. The American Mathematical Society, 1998. ISBN 978-0-8218-1054-5. 55

Viacheslav Borovitskiy, Alexander Terenin, Peter Mostowsky, and Marc P Deisenroth. Matérn Gaussian processes on Riemannian manifolds. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/hash/92bf5e6240737e0326ea59846a83e076-Abstract.html. 92

Stephen P Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN 978-0-521-83378-3. 9

C G Broyden. The convergence of a class of double-rank minimization algorithms 1. General considerations. *IMA Journal of Applied Mathematics*, 03 1970. doi: 10.1093/imamat/6.1.76. 12

Thang Duc Bui. *Efficient Deterministic Approximate Bayesian Inference for Gaussian Process Models*. PhD thesis, University of Cambridge, 2017. URL https://doi.org/10.17863/CAM.20913. 47, 163, 193, 199

Thang Duc Bui, Cuong Nguyen, and Richard E Turner. Streaming sparse Gaussian process approximations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. URL https://papers.nips.cc/paper_files/paper/2017/hash/f31b20466ae89669f9741e047487eb37-Abstract.html. 47, 129

David R Burt. *Scalable Approximate Inference and Model Selection in Gaussian Process Regression*. PhD thesis, University of Cambridge, 2022. URL https://doi.org/10.17863/CAM.86777. 67, 68, 92

David R Burt, Carl Edward Rasmussen, and Mark van der Wilk. Variational orthogonal features, 2020a. URL https://arxiv.org/abs/2006.13170. 77, 121

David R Burt, Carl Edward Rasmussen, and Mark van der Wilk. Convergence of sparse variational inference in Gaussian processes regression. *Journal of Machine Learning Research (JMLR)*, 2020b. URL http://jmlr.org/papers/v21/19-1015.html. 25, 44, 47, 49, 65, 68, 126, 239, 241

Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 1995. doi: 10.1137/0916069. 12

G Bürger, T Q Murdock, A T Werner, S R Sobie, and A J Cannon. Downscaling extremes – an intercomparison of multiple statistical methods for present climate. *Journal of Climate*, 2012. doi: 10.1175/JCLI-D-11-00408.1. 16

Alberto Carrassi, Marc Bocquet, Laurent Bertino, and Geir Evensen. Data assimilation in the geosciences – an overview on methods, issues and perspectives. *WIREs Clim Change*, 2018. doi: 10.1002/wcc.535. 51

Paul E Chang, William J Wilkinson, Mohammad Emtiyaz Khan, and Arno Solin. Fast variational learning in state-space Gaussian process models. In *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2020. URL https://ieeexplore.ieee.org/document/9231560. 49

Talay M Cheema. Understanding local linearisation in variational Gaussian process state space models. In *Time Series Workshop at the International Conference on Machine Learning (ICML)*, 2021. URL https://roseyu.com/time-series-workshop/submissions/2021/TSW-ICML2021_paper_16.pdf. 162

Talay M Cheema and Carl Edward Rasmussen. Contrasting discrete and continuous time methods for Bayesian system identification. In *Workshop on Continuous Time Methods in Machine Learning at the International Conference on Machine Learning (ICML)*, 2022. URL https://talaych.github.io/files/2022_ctmml/paper.pdf. 162

Talay M Cheema and Carl Edward Rasmussen. Integrated variational fourier features for fast spatial modelling with Gaussian processes. *Transactions on Machine Learning Research (TMLR)*, 2024a. URL https://openreview.net/pdf?id=PtBzWCaCYB. 54, 94, 225

Talay M Cheema and Carl Edward Rasmussen. Fast non-stationary geospatial modelling with multiresolution (wavelet) Gaussian processes. In *Workshop on Tackling Climate Change with Machine Learning at the International Conference on Learning Representations (ICLR)*, 2024b. URL https://www.climatechange.ai/papers/iclr2024/33. 134

Ricky T Q Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. URL https://papers.nips.cc/paper_files/paper/2018/hash/69386f6bb1dfed68692a24c8686939b9-Abstract.html. 192

Yanqing Chen, Timothy A Davis, William W Hager, and Sivasankaran Rajamanickam. Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate. *ACM Trans Math Softw*, 2008. doi: 10.1145/1391989.1391995. 147

Michael K Cohen, Samuel Daulton, and Michael A Osborne. Log-linear-time Gaussian processes using binary tree kernels. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/359ddb9caccb4c54cc915dceeacf4892-Abstract-Conference.html. 49

Thomas M Cover and Joy A Thomas. *Elements of Information Theory*. Wiley, 2nd edition, 2006. ISBN 978-0-471-24195-9. 41

Harry Jake Cunningham, Daniel Augusto de Souza, So Takao, Mark van der Wilk, and Marc Peter Deisenroth. Actually sparse variational Gaussian processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, Proceedings of Machine Learning Research, 2023. URL https://proceedings.mlr.press/v206/cunningham23a.html. 75, 120, 147, 196

John P Cunningham, Krishna V Shenoy, and Maneesh Sahani. Fast Gaussian process methods for point process intensity estimation. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2008. doi: 10.1145/1390156.1390181. 48

Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific machine learning through physics-informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 2022. doi: 10.1007/s10915-022-01939-z. 50

Sebastian Curi, Silvan Melchior, Felix Berkenkamp, and Andreas Krause. Structured variational inference in partially observable unstable Gaussian process state space models. In *Conference on Learning for Dynamics and Control (L4DC)*, 2020. URL https://proceedings.mlr.press/v120/curi20a.html. 163, 171, 194

Andreas C Damianou and Neil D Lawrence. Deep Gaussian processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2013. URL https://proceedings.mlr.press/v31/damianou13a.html. 135, 136

Andreas C Damianou, Michalis K Titsias, and Neil D Lawrence. Variational inference for latent variables and uncertain inputs in Gaussian processes. *Journal of Machine Learning Research (JMLR)*, 2016. URL https://jmlr.org/papers/v17/damianou16a.html. 49, 170

Marc P Deisenroth and Shakir Mohamed. Expectation propagation in Gaussian process dynamical systems. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2012. URL https://proceedings.neurips.cc/paper/2012/hash/e53a0a2978c28872a4505bdb51db06dc-Abstract.html. 31, 163, 193

Marc P Deisenroth and Carl Edward Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2011. doi: 10.5555/3104482.3104541. 16

Andreas Doerr, Christian Daniel, Martin Schiegg, Nguyen-Tuong Duy, Stefan Schaal, Marc Toussaint, and Trimpe Sebastian. Probabilistic recurrent state-space models. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018. URL https://proceedings.mlr.press/v80/doerr18a.html. 163, 171, 177, 194

Lea Duncker, Gergo Bohner, Julien Boussard, and Maneesh Sahani. Learning interpretable continuous-time models of latent stochastic dynamical systems. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2019. URL https://proceedings.mlr.press/v97/duncker19a.html. 166, 176, 194, 199

Vincent Dutordoir, Nicolas Durrande, and James Hensman. Sparse Gaussian processes with spherical harmonic features. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020. 45, 74, 77, 85, 92

Vincent Dutordoir, Hugh Salimbeni, Eric Hambro, John McLeod, Felix Leibfried, Artem Artemev, Mark van der Wilk, Marc P Deisenroth, James Hensman, and ST John. GPflux: A library for deep Gaussian processes. 2021. URL https://arxiv.org/abs/2104.05674. 149

David K Duvenaud, Hannes Nickisch, and Carl Edward Rasmussen. Additive Gaussian processes. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2011. URL https://papers.nips.cc/paper_files/paper/2011/hash/4c5bde74a8f110656874902f07378009-Abstract.html. 71

Ayesha Ekanayaka, Peter Kalmus, Emily Kang, and Amy Braverman. Statistical downscaling of sea surface temperature projections with a multivariate Gaussian process model. In *Workshop on Gaussian Processes, Spatiotemporal Modeling, and Decision-Making Systems (GPSMDMS) at the International Conference on Neural Information Processing Systems (NeurIPS)*, 2022. URL https://gp-seminar-series.github.io/neurips-2022/assets/camera_ready/51.pdf. 16

Stefanos Eleftheriadis, Tom Nicholson, Marc Deisenroth, and James Hensman. Identification of Gaussian Process State Space Models. In *Advances in Neural Information Processing Systems (NeurIPS)*. 2017. URL https://papers.nips.cc/paper_files/paper/2017/hash/1006ff12c465532f8c574aeaa4461b16-Abstract.html. 163, 176, 193

Charles L Epstein. How well does the finite Fourier transform approximate the Fourier transform? *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 2005. doi: 10.1002/cpa.20064. 110

Xuhui Fan, Edwin V Bonilla, Terence O'Kane, and Scott A Sisson. Free-form variational inference for Gaussian process state-space models. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2023. URL https://proceedings.mlr.press/v202/fan23a.html. 163, 171, 177, 192, 194

Khaled Ferkous, Farouk Chellali, Abdalah Kouzou, and Belgacem Bekkar. Wavelet-Gaussian process regression model for forecasting daily solar radiation in the Saharan climate. *Clean Energy*, 06 2021. doi: 10.1093/ce/zkab012. 159

R Fletcher. A new approach to variable metric algorithms. *The Computer Journal*, 1970. doi: 10.1093/comjnl/13.3.317. 12

D H Fremlin. *Measure Theory (Volume 1). The Irreducible Minimum.* Torres Fremlin, 2010. 236

Roger Frigola-Alcalde. *Bayesian Time Series Learning with Gaussian Processes.* PhD thesis, University of Cambridge, 2014. URL https://doi.org/10.17863/CAM.46480. 31, 163, 171, 176, 193

Paterne Gahungu, Christopher Lanyon, Mauricio A Álvarez, Engineer Bainomugisha, Michael T Smith, and Richard Wilkinson. Adjoint-aided inference of Gaussian process driven differential equations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/6dd16c884345ad63e4708367222410e5-Abstract-Conference.html. 50

Yarin Gal and Richard E Turner. Improving the Gaussian process sparse spectrum approximation by representing uncertainty in frequency inputs. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015. URL https://proceedings.mlr.press/v37/galb15.html. 132

Jacob Gardner, Geoff Pleiss, Kilian Q Weinberger, David Bindel, and Andrew G Wilson. GPyTorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018a. URL https://proceedings.neurips.cc/paper_files/paper/2018/hash/27e8e17134dd7083b050476733207ea1-Abstract.html. 49, 241, 242

Jacob Gardner, Geoff Pleiss, Ruihan Wu, Kilian Weinberger, and Andrew Gordon Wilson. Product kernel interpolation for scalable Gaussian processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018b. 48

Roman Garnett. *Bayesian Optimization*. Cambridge University Press, 2023. ISBN 978-1-108-34897-3. 16

Mark N Gibbs. *Bayesian Gaussian processes for regression and classification*. PhD thesis, University of Cambridge, 1998. URL https://www.inference.org.uk/mng10/GP/. 136

Evarist Giné M. The addition formula for the eigenfunctions of the Laplacian. *Advances in Mathematics*, 1975. doi: 10.1016/0001-8708(75)90003-1. 83

Donald Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of Computation*, 1970. doi: 10.1090/S0025-5718-1970-0258249-6. 12

Cristian Guarnizo and Mauricio A Álvarez. Fast kernel approximations for latent force models and convolved multiple-output Gaussian processes, 2018. URL https://auai.org/uai2018/proceedings/papers/295.pdf. 50

Jouni Hartikainen and Simo Särkkä. Sequential inference for latent force models. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2011. URL https://arxiv.org/abs/1202.3730. 50

Jouni Hartikainen and Simo Särkkä. Kalman filtering and smoothing solutions to temporal Gaussian process regression models. In *2010 IEEE International Workshop on Machine Learning for Signal Processing*, 2010. doi: 10.1109/MLSP.2010.5589113. 49

Marton Havasi, José Miguel Hernández-Lobato, and Juan José Murillo-Fuentes. Inference in deep gaussian processes using stochastic gradient hamiltonian monte carlo. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. URL https://papers.nips.cc/paper_files/paper/2018/hash/4172f3101212a2009c74b547b6ddf935-Abstract.html. 136

Pashupati Hegde, Çağatay Yıldız, Harri Lähdesmäki, Samuel Kaski, and Markus Heinonen. Variational multiple shooting for Bayesian ODEs with Gaussian processes. In *Proceedings of the Conference on Uncertainty in Artificial*

Intelligence (UAI), 2022. URL https://proceedings.mlr.press/v180/hegde22a.html. 194

Markus Heinonen, Henrik Mannerström, Juho Rousu, Samuel Kaski, and Harri Lähdesmäki. Non-stationary Gaussian process regression with Hamiltonian Monte Carlo. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 09–11 May 2016. URL https://proceedings.mlr.press/v51/heinonen16.html. 137, 139

Markus Heinonen, Çağatay Yıldız, Henrik Mannerström, Jukka Intosalmi, and Harri Lähdesmäki. Learning unknown ODE models with Gaussian processes. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018. URL https://proceedings.mlr.press/v80/heinonen18a.html. 194

Philipp Hennig, Michael A Osborne, and Hans P Kersting. *Probabilistic Numerics: Computation as Machine Learning*. Cambridge University Press, 2022. ISBN 978-1-316-68141-1. 16, 198

James Hensman, Nicolo Fusi, and Neil D Lawrence. Gaussian processes for big data. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2013. URL https://www.auai.org/uai2013/prints/papers/244.pdf. 25, 88, 89

James Hensman, Alexander G de G Matthews, and Zoubin Ghahramani. Scalable variational Gaussian process classification. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2015. URL https://proceedings.mlr.press/v38/hensman15.html. 25

James Hensman, Nicolas Durrande, and Arno Solin. Variational Fourier features for Gaussian processes. *Journal of Machine Learning Research (JMLR)*, 2017. URL https://jmlr.org/papers/v18/16-579.html. 44, 75, 91, 92, 117, 120, 132

Morris W Hirsch, Stephen Smale, and Robert L Devaney. *Differential Equations, Dynamical Systems, and an Introduction to Chaos*. Elsevier, 2nd edition, 2004. ISBN 978-0-12-349703-1. 28, 30, 164, 166, 180

Peter Holderrieth, Michael J Hutchinson, and Yee Whye Teh. Equivariant learning of stochastic fields: Gaussian processes and steerable conditional neural processes. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2021. URL https://proceedings.mlr.press/v139/holderrieth21a.html. 21

Michael Hutchinson, Alexander Terenin, Viacheslav Borovitskiy, So Takao, Yee Whye Teh, and Marc Deisenroth. Vector-valued Gaussian processes on Riemannian manifolds via gauge independent projected kernels. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/8e7991af8afa942dc572950e01177da5-Abstract.html. 92

Alessandro Davide Ialongo, Mark van der Wilk, James Hensman, and Carl Edward Rasmussen. Overcoming mean-field approximations in recurrent Gaussian process models. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2019. URL https://proceedings.mlr.press/v97/ialongo19a.html. 161, 163, 171, 177, 180, 194

E T Jaynes. *Probability Theory: The Logic of Science*. 1st edition, 2003. ISBN 978-0-511-79042-3. 4

Martin Jørgensen and Michael A Osborne. Bézier Gaussian processes for tall and wide data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/99c80ceb10cb674110f03b2def6a5b76-Abstract-Conference.html. 49

Chris Judge. Why is a diffeomorphism an isometry if and only if it commutes with the Laplacian? Mathematics Stack Exchange, 2022. URL https://math.stackexchange.com/q/4500138. (version: 2022-08-27). 80

Sanket Kamthe, So Takao, Shakir Mohamed, and Marc Peter Deisenroth. Iterative State Estimation in Non-linear Dynamical Systems Using Approximate Expectation Propagation. *Transactions on Machine Research (TMLR)*, 2022. URL https://openreview.net/pdf?id=xyt4wfdo4J. 51

George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 2021. ISSN 2522-5820. doi: 10.1038/s42254-021-00314-5. 51

K Kashinath, M Mustafa, A Albert, J-L Wu, C Jiang, S Esmaeilzadeh, K Azizzadenesheli, R Wang, A Chattopadhyay, A Singh, A Manepalli, D Chirila, R Yu, R Walters, B White, H Xiao, H A Tchelepi, P Marcus, A Anandkumar, P Hassanzadeh, and P Prabhat. Physics-informed machine learning: Case studies for weather and climate modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 2021. doi: 10.1098/rsta.2020.0093. 51

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015. URL https://arxiv.org/abs/1412.6980. 12

Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research (JMLR)*, 2008. URL http://jmlr.org/papers/v9/krause08a.html. 154

Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/df438e5206f31600e6ae4af72f2725f1-Abstract.html. 51

Vidhi Lalchand, Wessel Bruinsma, David Burt, and Carl Edward Rasmussen. Sparse Gaussian process hyperparameters: Optimize or integrate? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022a. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/69c49f75ca31620f1f0d38093d9f3d9b-Abstract-Conference.html. 47

Vidhi Lalchand, Kenza Tazi, Talay M Cheema, Richard E Turner, and Scott Hosking. Kernel learning for explainable climate science. In *Proceedings of the Bayesian Modelling Applications Workshop (BMAW) at the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2022b. URL https://arxiv.org/abs/2209.04947. 137

Miguel Lázaro-Gredilla. Bayesian warped Gaussian processes. *Advances in Neural Information Processing Systems (NeurIPS)*, 2012. URL https://papers.nips.cc/paper_files/paper/2012/hash/d840cc5d906c3e9c84374c8919d2074e-Abstract.html. 135

Miguel Lázaro-Gredilla and Aníbal Figueiras-Vidal. Inter-domain Gaussian processes for sparse inference using inducing features. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2009. URL https://papers.nips.cc/paper_files/paper/2009/hash/5ea1649a31336092c05438df996a3e59-Abstract.html. 61, 63, 76, 77

Miguel Lázaro-Gredilla, Joaquin Quinonero-Candela, Carl Edward Rasmussen, and Aníbal R Figueiras-Vidal. Sparse spectrum Gaussian process regression. *Journal of Machine Learning Research (JMLR)*, 2010. URL https://jmlr.org/beta/papers/v11/lazaro-gredilla10a.html. 132

Jacob Leander, Torbjörn Lundh, and Mats Jirstrand. Stochastic differential equations as a tool to regularize the parameter estimation problem for continuous time dynamical systems given discrete time measurements. *Mathematical Biosciences*, 2014. doi: 10.1016/j.mbs.2014.03.001. 171

Mikhail A Lifshits. *Lectures on Gaussian Processes*. Springer, 2012. ISBN 978-3-642-24938-9. 54, 55, 57, 76

Jarno Lintusaari, Henri Vuollekoski, Antti Kangasrääsiö, Kusti Skytén, Marko Järvenpää, Pekka Marttinen, Michael U. Gutmann, Aki Vehtari, Jukka Corander, and Samuel Kaski. ELFI: Engine for likelihood-free inference. *Journal of Machine Learning Research (JMLR)*, 2018. URL http://jmlr.org/papers/v19/17-374.html. 7

Haitao Liu, Jianfei Cai, Yi Wang, and Yew Soon Ong. Generalized robust Bayesian committee machine for large-scale Gaussian process regression. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018. URL https://proceedings.mlr.press/v80/liu18a.html. 48

Da Long, Zheng Wang, Aditi Krishnapriyan, Robert Kirby, Shandian Zhe, and Michael Mahoney. AutoIP: A united framework to integrate physics into Gaussian processes. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2022. URL https://proceedings.mlr.press/v162/long22a.html. 51

G G Lorentz. Approximation of smooth functions. *Bulletin of the American Mathematical Society*, 1960. URL https://projecteuclid.org/journals/bulletin-of-the-american-mathematical-society/volume-66/issue-2/Approximation-of-smooth-functions/bams/1183523468.full. 110

V Losert and Ethen Akin. Dynamics of games and genes: Discrete versus continuous time. *Journal of Mathematical Biology*, 1983. doi: 10.1007/BF00305762. 166

D J C MacKay. Introduction to Gaussian processes. In *Neural networks and machine learning*. 1998. URL https://www.inference.org.uk/mackay/gpB.pdf. 11

Stéphane Mallat. *A Wavelet Tour of Signal Processing*. 2009. ISBN 978-0-12-374370-1. 141, 142, 143

Jan Mandel. Efficient implementation of the ensemble Kalman filter. Technical report, University of Colorado, 2006. URL https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=6f1d17ff23ee9f144ffeb66f94e106da6f950e58. 51

Alexander G de G Matthews, James Hensman, Richard E Turner, and Zoubin Ghahramani. On sparse variational methods and the Kullback-Leibler divergence between stochastic processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016. URL https://proceedings.mlr.press/v51/matthews16.html. 173, 174

Alexander G de G Matthews, Mark van der Wilk, Tom Nickson, Keisuke Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman. GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research (JMLR)*, 2017. URL https://jmlr.org/papers/v18/16-537.html. 241

César Lincoln C Mattos and Guilherme A Barreto. A stochastic variational framework for Recurrent Gaussian Processes models. *Neural Networks*, 2019. doi: 10.1016/j.neunet.2019.01.005. 177, 194

Charles A Micchelli, Yuesheng Xu, and Haizhang Zhang. Universal kernels. *Journal of Machine Learning Research (JMLR)*, 2006. URL https://jmlr.org/papers/v7/micchelli06a.html. 135

Tom Minka. *A Family of Algorithms for Approximate Bayesian Inference*. PhD thesis, MIT, 2001. URL https://hd.media.mit.edu/tech-reports/TR-533.pdf. 34

Tom Minka. Power EP. Technical report, Microsoft Research, 2004. URL https://www.microsoft.com/en-us/research/publication/power-ep/. 34

Jacob D Moss, Felix L Opolka, Bianca Dumitrascu, and Pietro Lió. Approximate latent force model inference. In *Workshop on Machine Learning for the Physical Sciences (ML4PS) at Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://ml4physicalsciences.github.io/2021/files/NeurIPS_ML4PS_2021_78.pdf. 50

Kevin P Murphy. *Probabilistic Machine Learning: Advanced Topics*. MIT Press, 2023. ISBN 978-0-262-04843-9. URL http://probml.github.io/book2. 7, 9, 14, 33

Iain Murray. Differentiation of the Cholesky decomposition, 2016. URL https://arxiv.org/abs/1602.07527. 147

Sebastian W Ober and Laurence Aitchison. Global inducing point variational posteriors for bayesian neural networks and deep gaussian processes. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2021. URL https://proceedings.mlr.press/v139/ober21a.html. 136

Bernt Øksendal. *Stochastic differential equations: An introduction with applications*. Springer, 2013. ISBN 978-3-540-04758-2. 164, 173

Roland Opfer. Multiscale kernels. *Advances in Computational Mathematics*, 2006. doi: 10.1007/s10444-004-7622-3. 159

Christopher Paciorek and Mark Schervish. Nonstationary covariance functions for Gaussian process regression. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2003. URL https://papers.nips.cc/paper_files/paper/2003/hash/326a8c055c0d04f5b06544665d8bb3ea-Abstract.html. 137

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research (JMLR)*, 2021. URL http://jmlr.org/papers/v22/19-1028.html. 166

Tobias Pfingsten, Malte Kuss, and Carl Edward Rasmussen. Nonstationary Gaussian process regression using a latent extension of the input space. In *World Meeting of the International Society for Bayesian Analysis (ISBA)*, 2006. URL https://is.mpg.de/fileadmin/user_upload/files/publications/Nonstationary_GP_3985[1].pdf. 135

Marvin Pförtner, Ingo Steinwart, Philipp Hennig, and Jonathan Wenger. Physics-informed Gaussian process regression generalizes linear PDE solvers, 2023. URL https://arxiv.org/abs/2212.12474. 51

Roger A Pielke Sr and Robert L Wilby. Regional climate downscaling: What's the point? *Eos, Transactions American Geophysical Union*, 2012. doi: 10.1029/2012EO050008. 16

Geoff Pleiss, Jacob Gardner, Kilian Weinberger, and Andrew Gordon Wilson. Constant-time predictive distributions for Gaussian processes. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018. URL https://proceedings.mlr.press/v80/pleiss18a.html. 49

Rachel Prudden, Niall Robinson, Peter Challenor, and Richard Everson. Stochastic downscaling to chaotic weather regimes using spatially conditioned Gaussian random fields with adaptive covariance. *Weather and Forecasting*, 2021. doi: 10.1175/WAF-D-20-0217.1. 135

Florence Rabier and Zhiquan Liu. Variational data assimilation theory and overview. Technical report, European Centre for Medium-Range Weather Forecasts (ECMWF), 2003. URL https://www.ecmwf.int/sites/default/files/elibrary/2003/11805-variational-data-assimiltion-theory-and-overview.pdf. 51

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2007. URL https://papers.nips.cc/paper_files/paper/2007/hash/013a006f03dbc5392effeb8f18fda755-Abstract.html. 48, 132

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Inferring solutions of differential equations using noisy multi-fidelity data. *Journal of Computational Physics*, 2017. doi: 10.1016/j.jcp.2017.01.060. 51

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Numerical Gaussian processes for time-dependent and nonlinear partial differential equations. *SIAM Journal on Scientific Computing*, 2018. doi: 10.1137/17M1120762. 51

Maziar Raissi, Paris Perdikaris, and Georg Em Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 2019. doi: 10.1016/j.jcp.2018.10.045. 50

Carl Edward Rasmussen and Christopher K I Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. ISBN 978-0-262-18253-9. 15, 18, 21, 47, 57, 59, 135, 160

David Reeb, Andreas Doerr, Sebastian Gerwinn, and Barbara Rakitsch. Learning Gaussian processes by minimizing PAC-Bayesian generalization bounds. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/d43ab110ab2489d6b9b2caa394bf920f-Abstract.html. 49

Steven Reece and Stephen Roberts. An introduction to Gaussian processes for the Kalman filter expert. In *Proceedings of the International Conference on Information Fusion*, 2010. doi: 10.1109/ICIF.2010.5711863. 49

Sami Remes, Markus Heinonen, and Samuel Kaski. Non-stationary spectral kernels. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. URL https://papers.nips.cc/paper_files/paper/2017/hash/c65d7bd70fe3e5e3a2f3de681edc193d-Abstract.html. 139

Walter Rudin. *Real and Complex Analysis*. McGraw-Hill, 3rd edition, 1987. ISBN 978-0-07-100276-9 978-0-07-054234-1. 55

Yunus Saatçi. *Scalable Inference for Structured Gaussian Process Models*. PhD thesis, University of Cambridge, 2011. URL https://mlg.eng.cam.ac.uk/pub/pdf/Saa11.pdf. 48, 72

Hugh Salimbeni and Marc P Deisenroth. Doubly stochastic variational inference for deep Gaussian processes. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/8208974663db80265e9bfe7b222dcb18-Abstract.html. 136

Hugh Salimbeni, Stefanos Eleftheriadis, and James Hensman. Natural gradients in practice: Non-conjugate variational inference in Gaussian process models. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018. URL https://proceedings.mlr.press/v84/salimbeni18a.html. 47

Paul D Sampson and Peter Guttorp. Nonparametric estimation of nonstationary spatial covariance structure. *Journal of the American Statistical Association*, 1992. doi: 10.2307/2290458. 135

Simo Särkkä and Ángel F García-Fernández. Temporal parallelization of Bayesian smoothers. *IEEE Transactions on Automatic Control*, 2020. doi: 10.1109/TAC.2020.2976316. 51, 177

Simo Särkkä and Arno Solin. *Applied Stochastic Differential Equations*. Cambridge University Press, 2019. ISBN 978-1-316-51008-7. 51, 164, 173, 176

Simo Särkkä, Arno Solin, and Jouni Hartikainen. Spatiotemporal learning via infinite-dimensional Bayesian filtering and smoothing: A look at Gaussian process regression through Kalman filtering. *IEEE Signal Processing Magazine*, 2013. doi: 10.1109/MSP.2013.2246292. 28, 39, 42, 43, 51

Simo Särkkä, Mauricio A Álvarez, and Neil D Lawrence. Gaussian process latent force models for learning and stochastic control of physical systems. *IEEE Transactions on Automatic Control*, 2019. doi: 10.1109/TAC.2018.2874749. 50

Alexandra M Schmidt and Anthony O'Hagan. Bayesian inference for non-stationary spatial covariance structure via spatial deformations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2003. doi: 10.1111/1467-9868.00413. 135

Matthias W Seeger, Sham M Kakade, and Dean P Foster. Information consistency of nonparametric Gaussian process methods. *IEEE Transactions on Information Theory*, 2008. doi: 10.1109/TIT.2007.915707. 58

D F Shanno. Conditioning of quasi-Newton methods for function minimization. *Mathematics of Computation*, 1970. doi: 10.1090/S0025-5718-1970-0274029-X. 12

John Shawe-Taylor, Christopher K I Williams, Nello Cristianini, and Jaz Kandola. On the eigenspectrum of the gram matrix and the generalization error of kernel-PCA. *IEEE Transactions on Information Theory*, 2005. doi: 10.1109/TIT.2005.850052. 67

Jasper Snoek, Kevin Swersky, Rich Zemel, and Ryan Adams. Input warping for Bayesian optimization of non-stationary functions. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2014. URL https://proceedings.mlr.press/v32/snoek14.html. 135

Arno Solin and Simo Särkkä. Hilbert space methods for reduced-rank Gaussian process regression. *Statistics and Computing*, 2020. doi: 10.1007/s11222-019-09886-w. 45, 48, 58

Ingo Steinwart and Clint Scovel. Mercer's theorem on general domains: On the interaction between measures, kernels, and RKHSs. *Constructive Approximation*, 2012. doi: 10.1007/s00365-012-9153-3. 56

Andreas Svensson, Arno Solin, Simo Särkkä, and Thomas Schön. Computationally efficient Bayesian learning of Gaussian process state space models. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016. 193

Will Tebbutt, Arno Solin, and Richard E Turner. Combining pseudo-point and state space approximations for sum-separable Gaussian processes. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2021. URL https://proceedings.mlr.press/v161/tebbutt21a.html. 49

Michalis K Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2009. URL https://proceedings.mlr.press/v5/titsias09a.html. 23, 47, 63, 219

Gia-Lac Tran, Dimitrios Milios, Pietro Michiardi, and Maurizio Filippone. Sparse within sparse Gaussian processes using neighbor information. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2021. URL https://proceedings.mlr.press/v139/tran21a.html. 48

Ryan Darby Turner. *Gaussian Processes for State Space Models and Change Point Detection*. PhD thesis, University of Cambridge, 2011. 51

Richard E Turner and Maneesh Sahani. Two problems with variational expectation maximisation for time series models. In *Bayesian Time Series Models*. 2011. ISBN 978-0-521-19676-5. URL https://www.gatsby.ucl.ac.uk/~maneesh/papers/turner-sahani-2010-ildn.pdf. 161, 172

Mark van der Wilk, Vincent Dutordoir, ST John, Artem Artemev, Vincent Adam, and James Hensman. A framework for interdomain and multioutput Gaussian processes, 2020. URL https://arxiv.org/abs/2003.01115. 47

Martin Vetterli, Jelena Kovačević, and Vivek K Goyal. *Foundations of Signal Processing*. Cambridge University Press, 2012. ISBN 978-1-139-83909-9. 104, 111, 238

Glenn Vinnicombe. Sparse models of continuous time reaction networks from discrete time data: A conundrum (unpublished talk). 2015. 194

Christian Walder and Bernhard Schölkopf. Diffeomorphic dimensionality reduction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2008. URL https://papers.nips.cc/paper_files/paper/2008/hash/647bba344396e7c8170902bcf2e15551-Abstract.html. 166

Junyang Wang, Jon Cockayne, Oksana Chkrebtii, T J Sullivan, and Chris J Oates. Bayesian numerical methods for nonlinear partial differential equations. *Statistics and Computing*, 2021. doi: 10.1007/s11222-021-10030-w. 198

Jack M Wang, David J Fleet, and Aaron Hertzmann. Gaussian process dynamical models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2005. URL https://papers.nips.cc/paper_files/paper/2005/hash/ccd45007df44dd0f12098f486e7e8a0f-Abstract.html. 193

Wil Ward, Tom Ryder, Dennis Prangle, and Mauricio A Álvarez. Black-box inference for non-linear latent force models. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020. URL https://proceedings.mlr.press/v108/ward20a.html. 50

William Wilkinson, Paul Chang, Michael Andersen, and Arno Solin. State space expectation propagation: Efficient inference schemes for temporal Gaussian processes. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020. URL https://proceedings.mlr.press/v119/wilkinson20a.html. 40, 47, 49

Andrew Gordon Wilson and Ryan Adams. Gaussian process kernels for pattern discovery and extrapolation. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2013. URL https://proceedings.mlr.press/v28/wilson13.html. 21

Andrew Gordon Wilson and Hannes Nickisch. Kernel interpolation for scalable structured Gaussian processes (KISS-GP). In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015. URL http://proceedings.mlr.press/v37/wilson15.html. 48

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016. URL https://proceedings.mlr.press/v51/wilson16.html. 121, 135

James T Wilson, Viacheslav Borovitskiy, Alexander Terenin, Peter Mostowsky, and Marc P Deisenroth. Efficiently sampling functions from Gaussian process posteriors. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020. URL https://proceedings.mlr.press/v119/wilson20a.html. 22, 48, 63

Luhuan Wu, Geoff Pleiss, and John P Cunningham. Variational nearest neighbor Gaussian process. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2022. URL https://proceedings.mlr.press/v162/wu22h.html. 48

Fatemeh Yaghoobi, Adrien Corenflos, Sakira Hassan, and Simo Särkkä. Parallel iterated extended and sigma-point Kalman smoothers. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021. doi: 10.1109/ICASSP39728.2021.9413364. 51, 177, 192

Liu Yang, Xuhui Meng, and George Em Karniadakis. B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data. *Journal of Computational Physics*, 2021. doi: 10.1016/j.jcp.2020.109913. 51

Appendix A

Calculations

A.0.1 The Woodbury, or matrix inversion, identity is

$$(\mathsf{A}+\mathsf{UBV})^{-1} = \mathsf{A}^{-1} - \mathsf{A}^{-1}\mathsf{U}(\mathsf{B}^{-1}+\mathsf{VA}^{-1}\mathsf{U})^{-1}\mathsf{VA}^{-1} \tag{A.1}$$

assuming A, B are both invertible. For the special case where $\mathsf{U} = \mathsf{V} = \mathsf{I}$,

$$(\mathsf{A}+\mathsf{B})^{-1} = \mathsf{A}^{-1} - \mathsf{A}^{-1}(\mathsf{A}^{-1}+\mathsf{B}^{-1})^{-1}\mathsf{A}^{-1} = \mathsf{B}^{-1} - \mathsf{B}^{-1}(\mathsf{A}^{-1}+\mathsf{B}^{-1})^{-1}\mathsf{B}^{-1}. \tag{A.2}$$

A.0.2 For determinants, there is a similar identity.

$$|\mathsf{A}+\mathsf{UBV}| = |\mathsf{A}||\mathsf{B}||\mathsf{B}^{-1}+\mathsf{VA}^{-1}\mathsf{U}| \tag{A.3}$$

A.1 Manipulating Gaussians

A.1.1 Consider two Gaussian densities $\mathcal{N}(x|\mu_1, \Sigma_1)$, $\mathcal{N}(x|\mu_2, \Sigma_2)$. By straightforward algebraic manipulations in the exponent, one can show that the product can be rearranged as

$$\mathcal{N}(x|\mu_1, \Sigma_1)\mathcal{N}(x|\mu_2, \Sigma_2) = \mathcal{N}(\mu_2|\mu_1, \Sigma_1+\Sigma_2)\mathcal{N}(x|\mathsf{S}(\Sigma_1^{-1}\mu_1+\Sigma_2^{-1}\mu_2), \mathsf{S}) \tag{A.4}$$

$$\text{where} \quad \mathsf{S} \stackrel{\text{def}}{=} (\Sigma_1^{-1}+\Sigma^{-1})^{-1} \tag{A.5}$$

$$= \Sigma_1 - \Sigma_1(\Sigma_1+\Sigma_2)^{-1}\Sigma_1 = \Sigma_2 - \Sigma_2(\Sigma_1+\Sigma_2)^{-1}\Sigma_2. \tag{A.6}$$

A.1.2 The last line follows by applying Equation (A.2).

A.1.3 A modest variation on this result, obtained by a similar method, is

$$\mathcal{N}(x_2|\mathsf{A}x_1+b,\Sigma_{2|1})\mathcal{N}(x_1|\mu_1,\Sigma_1)=\mathcal{N}(x_2|\mu_2,\Sigma_2) \tag{A.7}$$
$$\times\,\mathcal{N}(x_1|\mu_1+\Sigma_1\mathsf{A}^\top\Sigma_2^{-1}(x_2-\mu_2),(\mathsf{A}^\top\Sigma_{2|1}^{-1}\mathsf{A}+\Sigma_1^{-1})^{-1})$$
$$\text{where}\quad \mu_2 \stackrel{\text{def}}{=} \mathsf{A}\mu_1+b \tag{A.8}$$
$$\Sigma_2 \stackrel{\text{def}}{=} \mathsf{A}\Sigma_1\mathsf{A}^\top+\Sigma_{2|1} \tag{A.9}$$
$$\mu_1+\Sigma_1\mathsf{A}^\top\Sigma_2^{-1}(x_2-\mu_2)=(\mathsf{A}^\top\Sigma_{2|1}^{-1}\mathsf{A}+\Sigma_1^{-1})^{-1}(\mathsf{A}^\top\Sigma_{2|1}^{-1}(x_2-b)+\Sigma_1^{-1}\mu_1). \tag{A.10}$$

A.1.4 The conditional $\mathcal{N}(x_2|\mathsf{A}x_1+b,\Sigma_{2|1})$ is equivalent to the transition model

$$x_2=\mathsf{A}x_1+b+\kappa,\qquad \kappa\sim\mathcal{N}(0,\Sigma_{2|1}) \tag{A.11}$$

from which it follows that the covariance is $\Sigma_{21}=\mathbb{E}[x_2x_1^\top]=\mathsf{A}\Sigma_1=\Sigma_{21}^\top$. In particular, suppose that x_1,x_2 are jointly Gaussian with the pairwise marginal

$$\begin{bmatrix}x_1\\x_2\end{bmatrix}\sim\mathcal{N}\left(\begin{bmatrix}\mu_1\\\mu_2\end{bmatrix},\begin{bmatrix}\Sigma_1&\Sigma_{12}\\\Sigma_{21}&\Sigma_2\end{bmatrix}\right) \tag{A.12}$$

then the standard Gaussian conditionals can be extracted using Equation (A.7) by setting $\mathsf{A}=\Sigma_{12}\Sigma_1^{-1}$ and $\Sigma_{2|1}=\Sigma_2-\mathsf{A}\Sigma_1\mathsf{A}^\top$ and performing some minor rearrangements.

$$p(x_1|x_2)=\mathcal{N}(x_1|\mu_1+\Sigma_{12}\Sigma_2^{-1}(x_1-\mu_2),\ \Sigma_1-\Sigma_{12}\Sigma_2^{-1}\Sigma_{21}) \tag{A.13}$$

A.1.5 It is often useful to express the exponent of a multivariate Gaussian in terms of a trace.

$$\log\mathcal{N}(x|\mu,\Sigma)=-\frac{1}{2}\log|2\pi\Sigma|-\frac{1}{2}\operatorname{tr}\left(\Sigma^{-1}((x-\mu)(x-\mu)^\top)\right). \tag{A.14}$$

A.1.6 **KL divergence** The KL divergence between two Gaussian distributions, using the natural logarithm, is

$$D_{KL}(\mathcal{N}(\mu_1,\Sigma_1)||\mathcal{N}(\mu_2,\Sigma_2))=\frac{1}{2}\log\frac{|\Sigma_2|}{|e\Sigma_1|}+\frac{1}{2}\operatorname{tr}\left(\Sigma_2^{-1}(\Sigma_1+(\mu_1-\mu_0)(\mu_1-\mu_0)^\top)\right). \tag{A.15}$$

A.1.7 **Bonnet's and Price's theorem** The following fact is useful for optimising.

$$\frac{d}{d\mu}\int f(x)\mathcal{N}(x|\mu,\Sigma)\,dx = \int \frac{df}{dx}\mathcal{N}(x|\mu,\Sigma)\,dx \tag{A.16}$$

A.2 Supplementary details for Chapter 1

A.2.1 Regression equations

A.2.1.1 For parametric regression, the log likelihood is

$$\log p(y|w,x) = \log \mathcal{N}(y|w^\top \Phi, \sigma^2 \mathsf{I}) \tag{A.17}$$

$$= -\frac{1}{2}N\log 2\pi\sigma^2 - \frac{1}{2\sigma^2}||y - w^\top\Phi||_2^2 \tag{A.18}$$

for which the maximiser is the classical least squares solution $\hat{w}_{ML} = (\Phi\Phi^\top)^{-1}\Phi y$ which yields $\hat{w}_{ML}^\top\Phi$ as the orthogonal projection of y onto the column space of Φ.

A.2.1.2 For the posterior and marginal likelihood, I use Equation (A.7). Then the MAP estimate is just the posterior mean.

$$p(y|\theta)p(w|y,\theta) = p(y,w|\theta) = p(y|w,\theta)p(w|\theta) \tag{A.19}$$

$$= \mathcal{N}(y|\Phi^\top w, \sigma^2\mathsf{I})\mathcal{N}(w|\mu_w,\Sigma_w) \tag{A.20}$$

$$= \mathcal{N}(y|\Phi^\top\mu_w, \Phi^\top\Sigma_w\Phi + \sigma^2\mathsf{I})\mathcal{N}(w|\mathsf{S}(\Sigma_w^{-1}\mu_w + \sigma^{-2}\Phi y), \mathsf{S}) \tag{A.21}$$

$$\mathsf{S} \stackrel{\text{def}}{=} (\sigma^{-2}\Phi\Phi^\top + \Sigma_w^{-1})^{-1} \tag{A.22}$$

A.2.1.3 **Collapsed sparse GP regression** This follows from Titsias (2009). Considering a single log conditional likelihood factor, and using Equation (1.42),

$$\int q(f)\log p(y_n|f,x_n)\,df = \int q(u)\int q(f|u)\log\mathcal{N}(y_n|f(x_n),\sigma^2)\,df\,du \tag{A.23}$$

$$= \int q(u)\left[-\frac{1}{2}\log 2\pi\sigma^2 - \frac{1}{2}\sigma^{-2}(y_n^2 + u^\top\mathsf{K}_{uu}^{-1}\mathsf{K}_{uf_n}\mathsf{K}_{uf_n}^\top\mathsf{K}_{uu}^{-1}u \right.$$

$$\left. + k(x_n,x_n) - \mathsf{K}_{uf_n}\mathsf{K}_{uu}^{-1}\mathsf{K}_{uf_n}^\top - 2y_n\mathsf{K}_{uf_n}^\top\mathsf{K}_{uu}^{-1}u)\right]du \tag{A.24}$$

so summing over n, we get

$$\int q(f)\log p(y|f,x)\,df = \int q(u)\left[-\frac{1}{2}N\log 2\pi\sigma^2 - \frac{1}{2}\sigma^{-2}(||y||_2^2 + u^\top \mathsf{K}_{uu}^{-1}\mathsf{K}_{u\mathsf{f}}\mathsf{K}_{u\mathsf{f}}^\top\mathsf{K}_{uu}^{-1}u + \operatorname{tr}(\mathsf{K}_{\mathsf{ff}} - \mathsf{K}_{u\mathsf{f}}\mathsf{K}_{uu}^{-1}\mathsf{K}_{u\mathsf{f}}^\top) - 2y^\top \mathsf{K}_{u\mathsf{f}_n}^\top \mathsf{K}_{uu}^{-1}u)\right] du \tag{A.25}$$

which is quadratic in u. At the same time,

$$\int q(f)\log\frac{q(f)}{p(f)}\,df = D_{KL}(q(f)||p(f)) = D_{KL}(q(u)||p(u)) \tag{A.26}$$

which leads to

$$\mathcal{L}' = \int q(f)\log\frac{p(y|f,x)p(f)}{q(f)}\,df = \int q(f)\log p(y|f,x)\,df - D_{KL}(q(u)||p(u)) \tag{A.27}$$

$$= \int q(u)\log\frac{p(u)e^{\int q(f)\log p(y|f,x)}}{q(u)}\,du. \tag{A.28}$$

A.2.1.4 This is almost a negative KL divergence, except that the numerator is not a proper probability distribution. The maximum is attained when $q(u)$ is proportional to the numerator, and the maximum value is the log normaliser of the numerator. With a few straightforward calculations, one can arrive at Equations (1.44) and (1.46).

A.2.1.5 **Rearranging the quadratic and log determinant** The key step to Equation (1.46) is to rearrange the inverse and log determinant. We have an $N \times N$ matrix $\mathsf{K}_{u\mathsf{f}}^\top\mathsf{K}_{uu}^{-1}\mathsf{K}_{u\mathsf{f}} + \sigma^2\mathsf{I}$, which appears as the covariance matrix. Then its inverse and determinant are, by direct application of Equations (A.1) and (A.3),

$$(\mathsf{K}_{u\mathsf{f}}^\top\mathsf{K}_{uu}^{-1}\mathsf{K}_{u\mathsf{f}} + \sigma^2\mathsf{I})^{-1} = \sigma^{-2}\mathsf{I} - \sigma^{-4}\mathsf{K}_{u\mathsf{f}}^\top(\mathsf{K}_{uu} + \sigma^{-2}\mathsf{K}_{u\mathsf{f}}\mathsf{K}_{u\mathsf{f}}^\top)^{-1}\mathsf{K}_{u\mathsf{f}} \tag{A.29}$$

$$|\mathsf{K}_{u\mathsf{f}}^\top\mathsf{K}_{uu}^{-1}\mathsf{K}_{u\mathsf{f}} + \sigma^2\mathsf{I}| = |\sigma^2\mathsf{I}||\mathsf{K}_{uu}^{-1}||\mathsf{K}_{uu} + \sigma^{-2}\mathsf{K}_{u\mathsf{f}}\mathsf{K}_{u\mathsf{f}}^\top| \tag{A.30}$$

and $|\mathsf{K}_{uu}^{-1}| = |\mathsf{K}_{uu}|^{-1}$, $|\sigma^{-2}\mathsf{I}| = \sigma^{-2N}$.

A.2.1.6 Of course, replacing $\mathsf{K}_{u\mathsf{f}}^\top$ with $\mathsf{K}_{u\mathsf{f}}^*$ changes nothing.

A.2.2 Message passing for SSMs

A.2.2.1 **The general case** Following from the definition of the collapsed messages in Figure 1.5 and the general updates in Equation (1.65), the messages are calculated as follows.

$$m_{ft}(x_t) = \int p(x_t|x_{t-1}) m_{f,t-1}(x_{t-1}) m_{o,t-1}(x_{t-1})\, dx_{t-1} \tag{A.31}$$

$$m_{bt}(x_t) = \int p(x_{t+1}|x_t) m_{b,t+1}(x_{t+1}) m_{o,t+1}(x_{t+1})\, dx_{t+1} \tag{A.32}$$

$$m_{ot}(x_t) = \begin{cases} p(y_t|x_t) & y_t \text{ observed} \\ \int p(y_t|x_t)\, dy_t = 1 & \text{otherwise} \end{cases} \tag{A.33}$$

A.2.2.2 The marginal smoothing posteriors are proportional to the intrinsic variable messages. The filtered posterior $p(x_t|y_{1:t})$ is given by excluding the backward message. We may also need pairwise marginals, which are easily extracted by combining adjacent nodes into a super-node. The factor between the variables then also has to be pulled out into the preceding factor to get the right joint distribution. Hence we get

$$p(x_t|y_{1:t}) = \frac{1}{p(y_{1:t})} m_{ft} m_{ot} \tag{A.34}$$

$$p(x_t|y_{1:T}) = \frac{1}{p(y_{1:T})} m_{ft} m_{ot} m_{bt} \tag{A.35}$$

$$p(x_t, x_{t+1}|y_{1:T}) = \frac{1}{p(y_{1:T})} p(x_{t+1}|x_t) m_{ft} m_{ot} m_{b,t+1} m_{o,t+1}. \tag{A.36}$$

A.2.2.3 The backward message at the end is $m_{bT} = 1$. The forward message at the start is just $m_{f1} = \int p(x_t)\, dx_{t-1} = p(x_1)$, the prior. By computing the forward and backward messages in parallel, in T steps we get all the information we need to compute the state posteriors.

A.2.2.4 **Predictions** In the predictive part of the graph, all the backward messages are 1, so the Bayes optimal predictor is just the filter

$$p(x_{T+k}|y_{1:T}) = \frac{1}{p(y_{1:T})} \int p(x_{T+k}|x_{T+k-1}) m_{f,T+k-1}\, dx_{T+k-1} \tag{A.37}$$

$$p(y_{T+k}|y_{1:T}) = \frac{1}{p(y_{1:T})} \int p(y_{T+k}|x_{T+k}) m_{f,T+k}\, dx_{T+k}. \tag{A.38}$$

A.2.2.5 **Missing observations** If there are intermediate missing observations, we can follow the same procedure, but the predictive for these observations will include the appropriate backward message, that is $p(y_t|y_{\text{observed}}) \propto \int p(y_t|x_t) m_{ft} m_{bt}\, dx_t$.

A.2.2.6 **Marginal likelihood** The marginal likelihood is the normaliser of the smoothing marginals, which is also the normaliser of the filtering distribution for the last state.

A.2.2.7 **Smoothing** If we are seeking the smoothing distribution $p(x_t|y_{1:T})$, we can directly form a recursion for it.

$$m_{ft} m_{ot} m_{bt} = m_{ft} m_{ot} \int p(x_{t+1}|x_t) m_{o,t+1} m_{b,t+1}\, dx_{t+1} \tag{A.39}$$

$$= \int \frac{p(x_{t+1}|x_t) m_{ft} m_{ot}}{m_{f,t+1}} m_{f,t+1} m_{o,t+1} m_{b,t+1}\, dx_{t+1} \tag{A.40}$$

A.2.2.8 Note that $m_{f,t+1}$ is just the normaliser of the numerator, so that the first term is $p(x_t|x_{t+1}, y_{1:t})$. We can initialise this recursion with $m_{fT} m_{oT}$ after having calculated all the forward messages.

A.2.2.9 **Kalman filtering and smoothing** We can briefly recover the Kalman updates as follows. First we get a recursion for $m_{ft} m_{ot}$ which is poportional to the filtering distribution $p(x_t|y_{1:t})$. Assuming $m_{f,t-1} m_{o,t-1} = Z_{t-1} \mathcal{N}(x_{t-1}|\vec{\mu}_{t-1}, \vec{\Sigma}_{t-1})$, from Equation (A.31) we see that

$$m_{ft} = Z_{t-1} \int \mathcal{N}(x_t|\mathsf{A}_{t-1} x_{t-1}, \mathsf{Q}_{t-1}) m_{f,t-1} m_{o,t-1}\, dx_{t-1} \tag{A.41}$$

$$= Z_{t-1} \int \mathcal{N}(x_t|\mathsf{A}_{t-1} x_{t-1}, \mathsf{Q}_{t-1}) \mathcal{N}(x_{t-1}|\vec{\mu}_{t-1}, \vec{\Sigma}_{t-1})\, dx_{t-1} \tag{A.42}$$

$$= Z_{t-1} \mathcal{N}(x_t| \underbrace{\mathsf{A}_{t-1} \vec{\mu}_{t-1}}_{\mu_{t+}}, \underbrace{\mathsf{A}_{t-1} \vec{\Sigma}_{t-1} \mathsf{A}_{t-1}^\top + \mathsf{Q}_{t-1}}_{\Sigma_{t+}}) \tag{A.43}$$

as reported in Equation (1.71). Incorporating the observation and applying Equation (A.7) yields the recursion of Equation (1.74).

$$m_{ft} m_{ot} = m_{ft} \mathcal{N}(y_t|\mathsf{C}_t x_t + d_t, \mathsf{R}_t) \tag{A.44}$$

$$= Z_{t-1}\mathcal{N}(x_t|\mu_t^+, \Sigma_t^+)\mathcal{N}(y_t|\mathsf{C}_t x_t + d_t, \mathsf{R}_t) \tag{A.45}$$

$$= \underbrace{Z_{t-1}\mathcal{N}(y_t|c_t^+, \mathsf{S}_t^+)}_{Z_t}\,\mathcal{N}(x_t|\vec{\mu}_t, \vec{\Sigma}_t) \tag{A.46}$$

where
$$c_t^+ \stackrel{\text{def}}{=} \mathsf{C}_t\mu_t^+ + d_t \tag{A.47}$$

$$\mathsf{S}_t^+ \stackrel{\text{def}}{=} \mathsf{C}_t\Sigma_t^+ C_t^\top + \mathsf{R}_t \tag{A.48}$$

$$\vec{\mu}_t \stackrel{\text{def}}{=} \mu_t^+ + \Sigma_t^+\mathsf{C}_t^\top \mathsf{S}_t^{+^{-1}}(y_t - c_t^+) \tag{A.49}$$

$$\vec{\Sigma}_t \stackrel{\text{def}}{=} \Sigma_t^+ - \Sigma_t^+\mathsf{C}_t^\top S_t^{+^{-1}}\mathsf{C}_t\Sigma_t^+ \tag{A.50}$$

A.2.2.10 The update for the log marginal likelihood $\log Z_t$ has also emerged, matching Equations (1.72) and (1.73).

A.2.2.11 Now for sequential smoothing we need $p(x_t|x_{t+1}, y_{1:t-1})$ (Equation (A.40)) which we can extract from the appropriate joint distribution.

$$\begin{bmatrix} x_t \\ x_{t+1} \end{bmatrix} \Bigg| y_{1:t} \sim \mathcal{N}\left(\begin{bmatrix} \mu_t \\ \mu_{t+1}^+ \end{bmatrix}, \begin{bmatrix} \Sigma_t & \Sigma_t\mathsf{A}_t^\top \\ \mathsf{A}_t\Sigma_t & \Sigma_{t+1}^+ \end{bmatrix} \right) \tag{A.51}$$

A.2.2.12 So, the conditional we need is (using Equation (A.13))

$$x_t|x_{t+1}, y_{1:t} \sim \mathcal{N}(\mu_t + \Sigma_t\mathsf{A}_t^\top \Sigma_{t+1}^{+^{-1}}(x_{t+1} - \mu_{t+1}^+),\ \Sigma_t - \Sigma_t\mathsf{A}_t^\top \Sigma_{t+1}^{+^{-1}}\mathsf{A}_t\Sigma_t). \tag{A.52}$$

A.2.2.13 Assuming $m_{f,t+1}m_{o,t+1}m_{b,t+1} \propto \mathcal{N}(x_{t+1}|\mu_{t+1}, \Sigma_{t+1})$, we can now simplify Equation (A.40).

$$m_{ft}m_{bt}m_{ot} = \int p(x_t|x_{t+1}, y_{1:t})m_{f,t+1}m_{o,t+1}m_{b,t+1}\,dx_{t+1} \tag{A.53}$$

$$\propto \int p(x_t|x_{t+1}, y_{1:t})\mathcal{N}(x_{t+1}|\mu_{t+1}, \Sigma_{t+1})\,dx_{t+1} \tag{A.54}$$

$$= \int \mathcal{N}(x_t|\mu_t + \Sigma_t\mathsf{A}_t^\top \Sigma_{t+1}^{+^{-1}}(x_{t+1} - \mu_{t+1}^+),\ \Sigma_t - \Sigma_t\mathsf{A}_t^\top \Sigma_{t+1}^{+^{-1}}\mathsf{A}_t\Sigma_t) \tag{A.55}$$

$$\times\, \mathcal{N}(x_{t+1}|\mu_{t+1}, \Sigma_{t+1})\,dx_{t+1}$$

$$= \mathcal{N}(x_t|\mu_t, \Sigma_t) \tag{A.56}$$

where
$$\mu_t \stackrel{\text{def}}{=} \mu_t + \Sigma_t\mathsf{A}_t^\top \Sigma_{t+1}^{+^{-1}}(\mu_{t+1} - \mu_{t+1}^+) \tag{A.57}$$

$$\Sigma_t \stackrel{\text{def}}{=} \Sigma_t - \Sigma_t\mathsf{A}_t^\top \Sigma_{t+1}^{+^{-1}}\mathsf{A}_t\Sigma_t + \Sigma_t\mathsf{A}_t^\top \Sigma_{t+1}^{+^{-1}}\Sigma_{t+1}\Sigma_{t+1}^{+^{-1}}\mathsf{A}_t\Sigma_t \tag{A.58}$$

$$= \Sigma_t - \Sigma_{t+1}^{+^{-1}}(\Sigma_{t+1}^+ - \Sigma_{t+1})\Sigma_{t+1}^{+^{-1}}\mathsf{A}_t\Sigma_t \tag{A.59}$$

A.2.2.14 **Moment matching** For completeness, I give a brief overview of the moment matching updates. Let the approximate filtering distribution at time $t-1$ be $q(x_{t-1})$. Then the target distribution for the next step is

$$m_{ot}m_{ft} = m_{ot}\int p(x_t|x_{t-1})\underbrace{m_{f,t-1}m_{o,t-1}}_{Z_{t-1}q(x_{t-1})}dx_{t-1} \tag{A.60}$$

$$= Z_{t-1}\int p(y_t|x_t)p(x_t|x_{t-1})q(x_{t-1})\,dx_{t-1} \tag{A.61}$$

so we match the moments of $q(x_t)$ to this, or equivalently compute

$$q(x_t) = \arg\min_{q(x_t)} D_{KL}\left(Z_{t-1}^{-1}\int p(y_t|x_t)p(x_t|x_{t-1})q(x_{t-1})\,dx_{t-1}\middle\|\, q(x_t)\right). \tag{A.62}$$

A.2.2.15 In particular, for prediction we lose the observation model term so we just take expectactions over $q(x_{t-1})$ of the model for $x_t = f(x_{t-1}) + \kappa_t$, yielding the update (for the nonlinear Gaussian case)

$$q(x_t) = \mathcal{N}\left(x_t\middle|\underbrace{\int f(x_{t-1})q(x_{t-1})\,dx_{t-1}}_{\vec{\mu}_t}, \int (f(x_{t-1})-\vec{\mu}_t)(f(x_{t-1})-\vec{\mu}_t)^\top q(x_{t-1})\,dx_{t-1}\right) \tag{A.63}$$

which agrees with statistical linearisation in the mean after one step, but propagates variances, which should improve long term predictions.

A.2.2.16 EP can be viewed as assumed density smoothing. Writing $q_{ft} = m_{ft}m_{ot}$, $q_{bt} = m_{bt}$, then we do iteratively (from Equation (1.69))

$$q_{ft}(x_t) = \arg\min_{q_{ft}} D_{KL}\left(\frac{\int p(y_t|x_t)p(x_t|x_{t-1})q_{f,t-1}(x_{t-1})dx_{t-1}}{Z_t}q_{bt}\middle\|\,\frac{q_{ft}q_{bt}}{Z_t'}\right) \tag{A.64}$$

$$q_{bt}(x_t) = \arg\min_{q_{bt}} D_{KL}\left(\frac{\int p(y_{t+1}|x_{t+1})p(x_{t+1}|x_t)q_{b,t+1}(x_{t+1})dx_{t+1}}{Z_{t+1}}q_{ft}\middle\|\,\frac{q_{ft}q_{bt}}{Z_t'}\right) \tag{A.65}$$

where the left hand equations are the message passing smoother $\propto m_{ft}m_{ot}m_{bt}$, with the appropriate factor recalculated from the message passing recursion. Note that

$$\int p(y_{t+1}|x_{t+1})p(x_{t+1}|x_t)q_{b,t+1}(x_{t+1})\,dx_{t+1}q_{ft} \tag{A.66}$$

$$= \int \underbrace{p(y_{t+1}|x_{t+1})}_{m_{o,t+1}} p(x_{t+1}|x_t) m_{b,t+1}(x_{t+1}) m_{ft} m_{ot} \, dx_{t+1} \tag{A.67}$$

$$\propto \int p(x_t, x_{t+1}|y_{1:T}) \, dx_{t+1} \tag{A.68}$$

with the last line ignoring the approximations. So, the the iterative process can be viewed as trying to reconcile the approximate marginals with the true conditional. The intuition is that this should help deal with nonlinearity.

A.2.2.17 **EP vs VI** Glossing over the fact that EP's objective is only local, the two methods optimise slightly different objective functions on the posterior. The KL in VI, $D_{KL}(q||p)$ averages over q; if q is simpler than p it can get away with fitting only part of p. Typically this would underestimate the uncertainty in the posterior, giving a more compact distribution. On the other hand EP matches moments in each step, which might be desirable.

A.3 The integrated variational Fourier feature perspective

A.3.0.1 Originally (Cheema & Rasmussen, 2024a) I presented AFS a numerical approximation to local averages of the Fourier transform of the function. This allows for more flexibility, such as placing inducing frequencies irregularly, but is a little more difficult to analyse. I provide a brief sketch of the main ideas here.

A.3.0.2 We are not able, in general, to integrate out the Dirac delta in Equation (2.73) and retain the desirable computational properties without introducing further approximations. One option is to average Fourier features over *disjoint* intervals of width W^{-1}, and approximate the spectral density as constant over the integration width. I focus on $D = 1$ to lighten the presentation here; to enforce that the intervals are disjoint we require $|z_m - z_{m'}| \geq \varepsilon$ for any $m \neq m'$.

$$\langle f, \varphi_m \rangle \overset{\text{def}}{=} W \int_{z_m - \frac{W^{-1}}{2}}^{z_m + \frac{W^{-1}}{2}} \int f(x) e^{-i2\pi\xi x} \, dx \frac{1}{s(2\pi\xi)} \, d\xi \tag{A.69}$$

$$c_m(x) = W \int_{z_m - \frac{W^{-1}}{2}}^{z_m + \frac{W^{-1}}{2}} e^{-i2\pi\xi x} \, d\xi \approx e^{-i2\pi z_m x} \tag{A.70}$$

$$\bar{k}_{m,m'} = W^2 \int_{z_m - \frac{W^{-1}}{2}}^{z_m + \frac{W^{-1}}{2}} \int_{z_{m'} - \frac{W^{-1}}{2}}^{z_{m'} + \frac{W^{-1}}{2}} \frac{1}{s(2\pi\xi)} \, d\xi' \, d\xi \approx W \frac{\delta_{m-m'}}{s(2\pi z_m)} \tag{A.71}$$

A.3.0.3 Now, δ is the Kronecker delta. Note that if the intervals were not chosen to be disjoint then only the last line would change. In the context of Chapter 3, the natural question to ask is what is the approximating covariance function? It is one which has a piecewise constant spectral density.

$$\hat{k}(r) = W^{-1} \operatorname{sinc}(\pi W^{-1} r) \sum_m s(2\pi z_m) e^{i 2\pi z_m r} \tag{A.72}$$

A.3.0.4 This is like k_P with a sinc window applied. We could place the z_m irregularly, and modify W to be dependent on m. As long as the averaging intervals $[z_m - \frac{W_m^{-1}}{2}, z_m + \frac{W_m^{-1}}{2}]$ remain disjoint, these changes can be carried through without much impact.

A.3.0.5 I originally provided a convergence proof whose rate was controlled by a midpoint error bound for the integral of s. This leads to $M \in \mathcal{O}(\sqrt{N})$. Generally, the basic difficulty in the analysis is the presence of sinc multiplier.

A.4 Details of the multiresolution covariance functions

A.4.0.1 Recall Equations (4.24) and (4.25).

$$\alpha_t = 2^{-1} \tag{A.73}$$

$$\beta_{jt} = 2^{-1} \frac{2^{1+a_0} - 1}{2^{-(1+a_0)}} 2^{-(1+a_0)(j-\iota+2)} \tag{A.74}$$

To verify the constant term, note

$$\sum_{j=\iota}^{\infty} 2^{-(1+a_0)(j-\iota+2)} = \sum_{j=0}^{\infty} 2^{-(1+a_0)(j+2)} = \frac{2^{-2(1+a_0)}}{1 - 2^{-(1+a_0)}} = \frac{2^{-(1+a_0)}}{2^{1+a_0} - 1}. \tag{A.75}$$

A.4.0.2 For the second case, we want to pick Z_{jt} is a suitable way. One choice is to let $t(j) \approx 2^j x$ for some x, so that we are examining only wavelets which are shifted to near a certain location x, and the approximation is due to the need for t to be an

integer. If they were exactly equal, then we would have (setting $Z_{jt} = 1$ for now)

$$\sum_j \beta_{jt(j)} = 1 + \sum_{q=1}^{Q} b_q 2^{-\left(\frac{x-c_q}{s_q}\right)^2} \tag{A.76}$$

so setting the normaliser as two times this would keep $\alpha_{jt} + \sum_j \beta_{jt(j)}$ constant, which should keep the marginal variance fairly close to constant also (dependent on the choice of wavelet). But since $t2^{-j}$ is not exactly x, I vary the normaliser as a function of j, t to try to compensate.

A.4.0.3 **Calculating the adjoints** For the log determinant, it is immediate that since $\frac{d}{d\mathsf{B}} \log|\mathsf{B}| = \mathsf{B}^{-\top}$, and B is symmetric, the adjoint for B should include a component $\mathsf{B}\overleftarrow{\delta}$, where $\delta = \log|\mathsf{B}|$. For the linear solve $\gamma = \mathsf{B}^{-1}b$, we have the corresponding updates $\overleftarrow{b} = \mathsf{B}^{-\top}\overleftarrow{\gamma}$ and $\overleftarrow{\mathsf{B}} = -\overleftarrow{b}\gamma^\top$. Using the fact that B is symmetric yields the result.

A.5 Supplementary details for Chapter 5

A.5.0.1 **Solving Equation (5.22)** In order to in the first instance simplify the problem, take $\zeta = 0$ and work with the discrete time case. Then also define

$$\tilde{\mathcal{E}}_t = \int q(x_t, f|u)(f(x_t) - \hat{f}(x_t))(f(x_t) - \hat{f}(x_t))^\top \, dx_t \, df + \hat{\mathsf{Q}}_t \tag{A.77}$$

$$\tilde{\mathcal{G}}_t = \log \frac{|\mathsf{Q}|}{|e\hat{\mathsf{Q}}|} + \mathrm{tr}\left(\mathsf{Q}^{-1}\tilde{\mathcal{E}}_t\right) \tag{A.78}$$

so that we can write down the solution to Equation (5.22) as

$$\log Z + \log q(u) = \log p(u) - \frac{1}{2}\sum_{t=1}^{t_T-1} \tilde{\mathcal{G}}_t. \tag{A.79}$$

A.5.0.2 Expanding out the quadratic, and using $\mathbb{E}$ for the expectation over $q(x_t)$

$$\tilde{\mathcal{E}}_t = \hat{\mathsf{Q}}_t + \int q(f, x_t|u)(f(x)f(x)^\top + \hat{f}_t(x_t)\hat{f}_t(x_t)^\top - f(x)\hat{f}_t(x_t)^\top - \hat{f}_t(x_t)f(x_t)^\top) \quad \text{(A.80)}$$

$$df\, dx_t$$

$$= \hat{\mathsf{Q}}_t + \mathbb{E}[\mathsf{K}_{\mathsf{f}_t u}\mathsf{K}_{uu}^{-1}uu^\top\mathsf{K}_{uu}^{-1}\mathsf{K}_{\mathsf{f}_t u} + \mathsf{K}_{\mathsf{f}_t\mathsf{f}_t} - \mathsf{K}_{\mathsf{f}_t u}\mathsf{K}_{uu}^{-1}\mathsf{K}_{\mathsf{f}_t u} \quad \text{(A.81)}$$
$$- \hat{f}_t(x)u^\top K_{uu}^{-1}\mathsf{K}_{u\mathsf{f}_t} - \mathsf{K}_{u\mathsf{f}_t}^\top\mathsf{K}_{uu}^{-1}u\hat{f}_t(x_t)^\top].$$

A.5.0.3 Now, since f is independent across dimensions, $\mathsf{K}_{u\mathsf{f}_t}$ and K_{uu} are compatibly block diagonal with $M \times 1$ and $M \times M$ blocks respectively. Since also Q is diagonal, $\mathsf{Q}^{-1}\mathsf{K}_{u\mathsf{f}}^\top = \mathsf{K}_{u\mathsf{f}}^\top(\mathsf{Q}^{-1} \otimes \mathsf{I}_M)$ and $\mathsf{Q}^{-1}\mathsf{K}_{u\mathsf{f}_t}^\top\mathsf{K}_{uu}^{-1} = \mathsf{K}_{u\mathsf{f}_t}^\top\mathsf{K}_{uu}^{-1}(\mathsf{Q}^{-1} \otimes \mathsf{I}_M)$; in fact, $(\mathsf{Q}^{-1} \otimes \mathsf{I}_M)$ can be freely interchanged with the block diagonal matrices which have $M \times M$ blocks. Combining this with the invariance of the trace to cyclic permutations allows to simplify the terms in $\operatorname{tr}(\mathsf{Q}^{-1}\mathcal{E}_t)$.

$$\operatorname{tr}(\mathsf{Q}^{-1}\mathbb{E}[\mathsf{K}_{u\mathsf{f}_t}^\top\mathsf{K}_{uu}^{-1}uu^\top\mathsf{K}_{uu}^{-1}\mathsf{K}_{u\mathsf{f}_t}]) = \operatorname{tr}((\mathsf{Q}^{-1} \otimes \mathsf{I}_M)\mathsf{K}_{uu}^{-1}\mathbb{E}[\mathsf{K}_{u\mathsf{f}_t}\mathsf{K}_{u\mathsf{f}_t}^\top]\mathsf{K}_{uu}^{-1}uu^\top) \quad \text{(A.82)}$$
$$= \operatorname{tr}((\mathsf{Q}^{-1} \otimes \mathsf{I}_M)\mathsf{K}_{uu}^{-1}\tilde{\Psi}_{ff,t}\mathsf{K}_{uu}^{-1}uu^\top) \quad \text{(A.83)}$$
$$\operatorname{tr}(\mathsf{Q}^{-1}\mathbb{E}[\mathsf{K}_{\mathsf{f}_t\mathsf{f}_t}]) = \operatorname{tr}(\mathsf{Q}^{-1}\Psi_{ff,t}) \quad \text{(A.84)}$$
$$-\operatorname{tr}(\mathsf{Q}^{-1}\mathsf{K}_{u\mathsf{f}_t}^\top\mathsf{K}_{uu}^{-1}\mathsf{K}_{u\mathsf{f}_t}) = \operatorname{tr}((\mathsf{Q}^{-1} \otimes \mathsf{I}_M)\mathsf{K}_{uu}^{-1}\mathbb{E}[\mathsf{K}_{u\mathsf{f}_t}\mathsf{K}_{u\mathsf{f}_t}^\top]) \quad \text{(A.85)}$$
$$= \operatorname{tr}((\mathsf{Q}^{-1} \otimes \mathsf{I}_M)\mathsf{K}_{uu}^{-1}\tilde{\Psi}_{ff,t}) \quad \text{(A.86)}$$
$$-\operatorname{tr}(\mathsf{Q}^{-1}\mathbb{E}[\hat{f}_t(x)u^\top\mathsf{K}_{uu}^{-1}\mathsf{K}_{u\mathsf{f}_t}]) = -\operatorname{tr}((\mathsf{Q}^{-1} \otimes \mathsf{I}_M)\mathsf{K}_{uu}^{-1}\mathbb{E}[\mathsf{K}_{u\mathsf{f}_t}\hat{f}_t(x_t)]u^\top) \quad \text{(A.87)}$$
$$= -\operatorname{tr}((\mathsf{Q}^{-1} \otimes \mathsf{I}_M)\mathsf{K}_{uu}^{-1}\Psi_{f\hat{f},t}u^\top) \quad \text{(A.88)}$$
$$= -\operatorname{tr}(\mathsf{Q}^{-1}\mathbb{E}[\mathsf{K}_{u\mathsf{f}_t}^\top\mathsf{K}_{uu}^{-1}u\hat{f}_t(x_t)^\top]) \quad \text{(A.89)}$$

A.5.0.4 Now, recall $\log p(u) = \log \mathcal{N}(u|0, \mathsf{K}_{uu})$. Then collecting the u terms only of $\log p(u) - \frac{1}{2}\sum_t \tilde{\mathcal{G}}_t$ we get

$$\operatorname{tr}\left(-\frac{1}{2}(\mathsf{K}_{uu}^{-1} + (\mathsf{Q}^{-1} \otimes \mathsf{I}_M)\mathsf{K}_{uu}^{-1}\tilde{\Psi}_{ff}\mathsf{K}_{uu}^{-1})uu^\top + (\mathsf{Q}^{-1} \otimes \mathsf{I}_M)\mathsf{K}_{uu}^{-1}\Psi_{f\hat{f},t}u^\top\right) \quad \text{(A.90)}$$

A.5.0.5 which shows that $q(u)$ is Gaussian.

$$\Sigma_u^{-1} = (\mathsf{K}_{uu}^{-1} + (\mathsf{Q}^{-1} \otimes \mathsf{I}_M)\mathsf{K}_{uu}^{-1}\tilde{\Psi}_{ff}\mathsf{K}_{uu}^{-1}) \quad \text{(A.91)}$$
$$= (\mathsf{Q}^{-1} \otimes \mathsf{I}_M)\mathsf{K}_{uu}^{-1}((\mathsf{Q} \otimes \mathsf{I}_M)\mathsf{K}_{uu} + \tilde{\Psi}_{ff})\mathsf{K}_{uu} \quad \text{(A.92)}$$
$$\Rightarrow \Sigma_u = \mathsf{K}_{uu}((\mathsf{Q} \otimes \mathsf{I}_M)\mathsf{K}_{uu} + \tilde{\Psi}_{ff})^{-1}\mathsf{K}_{uu}(\mathsf{Q} \otimes \mathsf{I}_M) \quad \text{(A.93)}$$

$$= (\mathsf{Q} \otimes \mathsf{I}_M)\mathsf{K}_{uu}((\mathsf{Q} \otimes \mathsf{I}_M) + \tilde{\Psi}_{ff})^{-1}\mathsf{K}_{uu} \tag{A.94}$$

$$\mu_u = \Sigma_u(\mathsf{Q}^{-1} \otimes \mathsf{I}_M)\mathsf{K}_{uu}^{-1}\Psi_{f\hat{f},t} \tag{A.95}$$

$$= \mathsf{K}_{uu}((\mathsf{Q} \otimes \mathsf{I}_M)\mathsf{K}_{uu} + \tilde{\Psi}_{ff})^{-1}\Psi_{f\hat{f}} \tag{A.96}$$

It is prudent to also define

$$\mathsf{G} = ((\mathsf{Q} \otimes \mathsf{I}_M)\mathsf{K}_{uu} + \tilde{\Psi}_{ff}). \tag{A.97}$$

A.5.0.6 These terms can be collected and rearranged to compute the residual contribution to the loss ($\log Z$, which shall be computed shortly), or μ_u and Σ_u can be used to calculated $\mathcal{E}_t$ and the function KL, as presented in the main text.

A.5.0.7 For the directly parameterised case, we can make a forward pass through the data, summing Ψ terms and the log conditional likelihood terms. At the end, we update the values of μ_u, Σ_u, compute $\mathcal{L}'$ and take a gradient step in the parameters. In the correlated case, or for alternating between setting $q(u)$ is closed form and Kalman smoothing, we accumulate the KL directly.

A.5.0.8 **The residual term** $\log Z$ All the u terms have cancelled, and we have just to collect the non-u terms from the previous calculations.

$$\log Z = \log p(u) - \frac{1}{2}\sum_{t=1}^{t_T-1}\mathcal{G}_t - \log q(u) \tag{A.98}$$

$$= -\frac{1}{2}\log|2\pi\mathsf{K}_{uu}| + \frac{1}{2}\log|2\pi\Sigma_u| + \frac{1}{2}\mu_u^\top\Sigma_u^{-1}\mu_u \tag{A.99}$$
$$- \frac{1}{2}\sum_t\left[\log\frac{|\mathsf{Q}|}{|\mathsf{Q}_t|} + \operatorname{tr}(\mathsf{Q}^{-1}(\hat{\mathsf{Q}}_t + \Psi_{ff,t})) - \operatorname{tr}((\mathsf{Q}^{-1} \otimes \mathsf{I})\mathsf{K}_{uu}^{-1}\tilde{\Psi}_{ff,t})\right]$$

$$= \frac{1}{2}\log\frac{|\mathsf{K}_{uu}|}{|(\mathsf{Q}^{-1} \otimes \mathsf{I}_M)\mathsf{G}|} + \frac{1}{2}\Psi_{f\hat{f}}^\top(\mathsf{Q}^{-1} \otimes \mathsf{I}_M)\mathsf{G}^{-1}\Psi_{f\hat{f}} - \frac{1}{2}\operatorname{tr}(\mathsf{Q}^{-1}\Psi_{ff}) \tag{A.100}$$
$$+ \operatorname{tr}((\mathsf{Q}^{-1} \otimes \mathsf{I}_M)\mathsf{K}_{uu}^{-1}\tilde{\Psi}_{ff}) - \frac{1}{2}\sum_{t=1}^{t_T-1}\left[\log\frac{|\mathsf{Q}|}{e\hat{\mathsf{Q}}} + \operatorname{tr}(\mathsf{Q}^{-1}\hat{\mathsf{Q}}_t)\right]$$

A.5.0.9 The summand is zero if $\mathsf{Q} = \hat{\mathsf{Q}}_t$.

A.5.0.10 In continuous time, the results are essentially the same, but the sums are exchanged for integrals, which corresponds to the Ψ definitions given in the main text.

A.5.0.11 For a non-zero mean function, just work with $f - \zeta$ (which is zero mean) and $\hat{f} - \zeta$. The difference between these is the same as between f and $\hat{f}$. With suitable modifications to the definitions (as given in the main text), this leads to no change in the calculations.

A.5.1 Variational filtering and smoothing

A.5.1.1 An alternative to Kalman filtering/smoothing is to fix $q(x)$ jointly Gaussian and optimise the variational objective $\mathcal{L}'$. For variational *filtering*, we consider optimising one factor of $q(x)$ at a time: given $q(x_t)$, we maximise $\mathcal{L}$ with respect to $\hat{\mathsf{A}}'_t, \hat{b}'_t, \hat{\mathsf{Q}}'_t$.

A.5.1.2 An iterative variational *smoother* (VS) was presented for the CT case by Archambeau et al (2007). This calculates $\hat{\mathsf{A}}'_t, \hat{b}'_t$ and $\hat{\mathsf{Q}}'_t$ on the backward pass, then updates the marginal statistics μ_t, Σ_t on the forward pass, iterating until convergence. This routine optimises our objective function $\mathcal{L}'$ but scales poorly to longer sequences. The connection between these optimal approaches and statistical linearisation motivates using the SLF/S instead.

A.5.1.3 This presentation holds for the mean field case; for the correlated case we just apply this method to each function. For example, if we condition on the inducing points only, we replace μ_t, Σ_t with $\mu_{t|u}, \Sigma_{t|u}$, and have transition parameters $\hat{\mathsf{A}}'_{t|u}, \hat{b}'_{t|u}, \hat{\mathsf{Q}}'_{t|u}$ for each u.

A.5.1.4 **Filtering** For direct variational filtering, I consider adding one latent and observation to the KL. The new term is

$$-\frac{1}{2}|2\pi\mathsf{R}| - \frac{1}{2}\operatorname{tr}(\mathsf{R}^{-1}(\mathsf{C}(\hat{\mathsf{A}}'_t\vec{\mu}_t + \hat{b}'_t))(\mathsf{C}(\hat{\mathsf{A}}'_t\vec{\mu}_{t_i} + \hat{b}'_t))^\top + \mathsf{C}(\hat{\mathsf{A}}'_t\vec{\Sigma}_t\hat{\mathsf{A}}'^\top_t + \hat{\mathsf{Q}}'_t)\mathsf{C}^\top) + \mathcal{G}_t \quad \text{(A.101)}$$

which, when maximised with respect to the transition parameters $(\hat{\mathsf{A}}'_t, \hat{b}'_t, \hat{\mathsf{Q}}'_t)$, yields the following.

$$\hat{\mathsf{Q}}'_t = (\mathsf{Q}^{-1} - \mathsf{C}^\top\mathsf{R}^{-1}\mathsf{C})^{-1} \qquad \hat{\mathsf{A}}'_t = \hat{\mathsf{Q}}'_t\mathsf{Q}^{-1}\hat{\mathsf{A}}_t \quad \text{(A.102)}$$

$$\hat{b}'_t = (\mathsf{I} - \hat{\mathsf{Q}}'_t\mathsf{C}^\top\mathsf{R}^{-1}\mathsf{C})(\hat{\mathsf{b}}_t + \hat{\mathsf{A}}_t\vec{\mu}_t) - (\hat{\mathsf{A}}'_t\vec{\mu}_t + \hat{\mathsf{Q}}'_t\mathsf{C}^\top\mathsf{R}^{-1}y_{t+1}) \quad \text{(A.103)}$$

A.5.1.5 Here, $\hat{\mathsf{A}}_t, \hat{\mathsf{b}}_t$ are the SLF linearisation parameters. Both the SLF and the variational filter compute $\hat{\mathsf{A}}_t, \hat{b}_t$ and modify them slightly based on the observations. Indeed, if

we ignored the observation term, found the VI-optimal transition, then the maximum likelihood update using the observation, we would recover the SLF.

A.5.1.6 **Smoothing** The continuous time method was derived by Archambeau et al (2007). I provide the discrete time updates for reference. We want to maximise the Lagrangian

$$\mathfrak{L} = \mathcal{L}' + \sum_t \lambda_t^\top (\hat{\mathsf{A}}'_t \mu_t + \hat{b}'_t) + \frac{1}{2} \sum_t \mathrm{tr}(\mathsf{L}_t (\hat{\mathsf{A}}'_t \Sigma_t \hat{\mathsf{A}}'_t + \hat{\mathsf{Q}}'_t)) \tag{A.104}$$

which enforces that the transition parameters are consistent with the marginals; L_t is symmetric. I write S_0 for the structure matrix of a symmetric matrix (1 on the diagonal, 2 elsewhere), S_1 for its element-wise reciprocal (1s on the diagonal, half elsewhere), and $\odot$ for the element-wise product. Optimising with respect to each parameter in turn:

$$\mu_t : \lambda_t - \hat{\mathsf{A}}'^\top_t \lambda_{t+1} + \frac{\partial \mathcal{L}'}{\partial \mu_t} = 0 \implies \lambda_t = \hat{\mathsf{A}}'^\top_t \lambda_{t+1} - \frac{\partial \mathcal{L}'}{\partial \mu_t} \tag{A.105}$$

$$\Sigma_t : (\mathsf{L}_t - \hat{\mathsf{A}}'_t \mathsf{L}_{t+1} \hat{\mathsf{A}}'^\top_t) \odot S_0 + 2\frac{\partial \mathcal{L}'}{\partial \Sigma_t} = 0 \implies \mathsf{L}_t = \hat{\mathsf{A}}'_t \mathsf{L}_{t+1} \hat{\mathsf{A}}'^\top_t - 2\frac{\partial \mathcal{L}'}{\partial \Sigma_t} \odot S_1 \tag{A.106}$$

$$\hat{\mathsf{Q}}'_t : -\mathsf{L}_{t+1} \odot S_0 + 2\frac{\partial \mathcal{L}'}{\partial \hat{\mathsf{Q}}'_t} = 0 \implies \mathsf{L}_{t+1} = -2\frac{\partial \mathcal{L}'}{\partial \hat{\mathsf{Q}}'_t} \odot S_1 \tag{A.107}$$

$$\hat{b}'_t : -\lambda_{t+1} + \frac{\partial \mathcal{L}'}{\partial \hat{b}'_t} = 0 \implies \lambda_{t+1} = \frac{\partial \mathcal{L}'}{\partial b_t} \tag{A.108}$$

$$\hat{\mathsf{A}}'_t : -\lambda_{t+1}\mu_t^\top - \mathsf{L}_{t+1} \hat{\mathsf{A}}'_t \Sigma_t + \frac{\partial \mathcal{L}'}{\partial \hat{\mathsf{A}}'_t} = 0 \implies \mathsf{L}_{t+1} \hat{\mathsf{A}}'_t \Sigma_t = \frac{\partial \mathcal{L}'}{\partial \hat{\mathsf{A}}'_t} - \lambda_{t+1}\mu_t^\top. \tag{A.109}$$

A.5.1.7 The dependence of $\mathcal{L}'$ on $(\hat{\mathsf{A}}'_t, \hat{b}'_t, \hat{\mathsf{Q}}'_t)$ is limited to $\mathcal{G}_t$. In particular, using E_t for the expectation over $q(x_t)$,

$$\frac{\partial \mathcal{L}'}{\partial \hat{\mathsf{Q}}'_t} = -\frac{1}{2}\frac{\partial \mathcal{G}_t}{\partial \hat{\mathsf{Q}}'_t} = -\frac{1}{2}(\mathsf{Q}^{-1} - \hat{\mathsf{Q}}'^{-1}_t) \odot S_0 \tag{A.110}$$

$$\frac{\partial \mathcal{L}'}{\partial \hat{b}'_t} = -\frac{1}{2}\frac{\partial \mathcal{G}_t}{\partial \hat{b}'_t} = \mathsf{Q}^{-1}(\mathbb{E}_t[f(x)] - \hat{\mathsf{A}}'_t \mu_t - \hat{b}'_t) \tag{A.111}$$

$$\frac{\partial \mathcal{L}'}{\partial \hat{\mathsf{A}}'_t} = -\frac{1}{2}\frac{\partial \mathcal{G}_t}{\partial \hat{\mathsf{A}}'_t} = \mathsf{Q}^{-1}(\mathbb{E}_t[f(x)x^\top] - (\hat{\mathsf{A}}'_t \mu_t + \hat{b}'_t)\mu_t^\top - \hat{\mathsf{A}}'_t \Sigma_t). \tag{A.112}$$

A.5.1.8 Combining Equations (A.110) to (A.112) with Equations (A.107) to (A.109) we can write down the optimal updates for the transition parameters.

$$\hat{\mathsf{Q}}'_t = (\mathsf{L}_{t+1} + \mathsf{Q}^{-1})^{-1}, \quad \hat{b}'_t = (\mathbb{E}_t[f(x)] - \hat{\mathsf{A}}'_t\mu_t) - Q\lambda_{t+1}, \quad \hat{\mathsf{A}}'_t = \hat{\mathsf{Q}}'_t\mathsf{Q}^{-1}\mathbb{E}_t\left[\nabla f\right] \tag{A.113}$$

A.5.1.9 For $\hat{\mathsf{A}}'_t$, I use the fact that for $x \sim \mathcal{N}(\mu, \Sigma)$, $E[f(x)(x-\mu)]\Sigma^{-1} = \mathbb{E}[\nabla f]$. Combined with Equations (A.105) and (A.106), this gives a backward recursion to set the transition parameters. We can then recompute the marginals in a forward recursion enforcing the constraints

$$\mu_{t+1} = \hat{\mathsf{A}}'_t\mu_t + \hat{b}'_t, \quad \Sigma_{t+1} = \hat{\mathsf{A}}'_t\Sigma_t\hat{\mathsf{A}}'^{\top}_t + \hat{\mathsf{Q}}'_t \tag{A.114}$$

and iterate until convergence. This will yield a local optimum of $\mathcal{L}'$ with transition parameters consistent with the marginals. However, since we are performing co-ordinate ascent on the Lagrangian, we may need many iterations to close in on an optimum.

A.5.1.10 I have not addressed the calculation of $\frac{\partial \mathcal{L}'}{\partial \mu_t}, \frac{\partial \mathcal{L}'}{\partial \Sigma_t}$. The component of this which comes from differentiation $\mathcal{G}$ is challenging, since it involves differentiation an expectation over $\mathcal{N}(\mu_t, \Sigma_t)$. We can do this in various different ways.

- We can compute the gradient function of $\mathcal{G}_t$ using automatic differentiation (which is computationally intensive);
- We can differentiate under the expectation yielding epxectations such as $\mathbb{E}_t[f(x)(x-\mu_t)], \mathbb{E}_t[f_d(x)(x-\mu_t)(x-\mu_t)^{\top}]$ which can be approximated using numerical integration routines, e.g. σ-point methods.
- We can convert the above expectations into expectations of derivatives such as $\mathbb{E}_t[\nabla f(x)], \mathbb{E}[\nabla\nabla^{\top} f(x)]$ using Equation (A.16) and simplify. In the case of an SE kernel, this yields closed form results.

A.5.2 Kernel expectations for a squared exponential covariance functions

A.5.2.1 In the experiments, I use an SE kernel of the form

$$k(x, z|\Lambda) = \exp\left(-\frac{1}{2}(x-z)^{\top}\Lambda^{-1}(x-z)\right) = \sqrt{|2\pi\Lambda|}\mathcal{N}(x|z, \Lambda). \tag{A.115}$$

A.5.2.2 If σ_f^2 is not 1, we just need to scale all results accordingly. For kernel expectations, I note the following, using standard rearrangements of Gaussian distributions.

$$\int h(x)k(x,z|\Lambda)\mathcal{N}(x|\mu,\Sigma)\,dx = \sqrt{|2\pi\Lambda|}\int h(x)\mathcal{N}(z|x,\Lambda)\mathcal{N}(x|\mu,\Sigma)dx \tag{A.116}$$

$$= \sqrt{|2\pi\Lambda|}\int h(x)\mathcal{N}(x|m,\mathsf{S})\,dx\mathcal{N}(z|\mu,\Lambda+\Sigma) \tag{A.117}$$

$$= \frac{k(\mu,z|\Lambda+\Sigma)}{\sqrt{|\mathsf{I}+\Sigma\Lambda^{-1}|}}\int f(x)\mathcal{N}(x|m,\mathsf{S})\,dx \tag{A.118}$$

$$\text{where}\quad m = \mu+\Sigma(\Lambda+\Sigma)^{-1}(z-\mu) \tag{A.119}$$

$$\mathsf{S} = (\Lambda^{-1}+\Sigma^{-1})^{-1} \tag{A.120}$$

$$\int h(x)k(x,z_1|\Lambda)k(x,z_2|\Lambda)\mathcal{N}(x|\mu,\Sigma)dx \tag{A.121}$$

$$= |2\pi\Lambda|\int h(x)\mathcal{N}(x|z_2,\Lambda)\mathcal{N}(x|m,\mathsf{S})\,dx\mathcal{N}(z_1|\mu,\Sigma+\Lambda) \tag{A.122}$$

$$= \frac{k(\mu,z_1|\Lambda+\Sigma)}{\sqrt{|\mathsf{I}+\Sigma\Lambda^{-1}|}}\sqrt{|2\pi\Lambda|}\int h(x)\mathcal{N}(x|m_2,\mathsf{S}_2)\,dx\mathcal{N}(z_2|m,\Lambda+\mathsf{S})\mathcal{N}(z_2|m,\Lambda+\mathsf{S}) \tag{A.123}$$

$$= \frac{k(\mu,z_1|\Lambda+\Sigma)k(m,z_2|\Lambda+S)}{\sqrt{|I+\Sigma\Lambda^{-1}||I+S\Lambda^{-1}|}}\int h(x)\mathcal{N}(x|m_2,\mathsf{S}_2)\,dx \tag{A.124}$$

$$= \frac{k(\mu,\frac{z_1+z_2}{2}|\frac{1}{2}\Lambda+\Sigma)k(z_1,z_2|2\Lambda)}{\sqrt{|I+2\Sigma\Lambda^{-1}|}}\int h(x)\mathcal{N}(x|m_2,\mathsf{S}_2)\,dx \tag{A.125}$$

$$m_2 = (\Lambda^{-1}+\mathsf{S}^{-1})^{-1}(\Lambda^{-1}z_2+\mathsf{S}^{-1}m) \tag{A.126}$$

$$\mathsf{S}_2 = (\Lambda^{-1}+\mathsf{S}^{-1})^{-1} \tag{A.127}$$

A.5.2.3 Now, for $\mathrm{Z}=0$, for $\Psi_{f\hat{f},t}$, $h(x)=\hat{\mathsf{A}}_t'x+\hat{b}_t'$, and for $\tilde{\Psi}_{ff,t}$, $h(x)=1$, so indexing into the block diagonal matrices' blocks with d, and within the block with $i,j\in\{1{:}M\}$, and writing m as a function of z,μ,

$$\Psi_{ff,t} = \mathsf{I} \tag{A.128}$$

$$\Psi_{f\hat{f}t,di} = \frac{k_d(\mu_t,z_{di}|\Lambda_d+\Sigma_t)}{\sqrt{|\mathsf{I}+\Sigma_t\Lambda_d^{-1}|}}\hat{f}(m(\mu_t,z_{di}))_d \tag{A.129}$$

$$\Psi_{ff,tdij} = \frac{k_d(\mu_t,\frac{z_{di}+z_{dj}}{2}|\frac{1}{2}\Lambda_d+\Sigma_t)k(z_{di},z_{dj}|2\Lambda_d)}{\sqrt{|I+2\Sigma_t\Lambda_d^{-1}|}}. \tag{A.130}$$

A.5.2.4 **Deterministic augmentation** Often, we need to augment the state x_t with deterministic control inputs $u_t \in \mathbb{R}^{D_c}$, for example in our real-world datasets. This is easily incorporated in the framework: let $D' = D + D_c$, then $f : \mathbb{R}^{D'} \to \mathbb{R}^D$, and $\tilde{x} = [x\, u]$. Then $k(\tilde{x}, \tilde{x}') = k(x, x')k(u, u')$. The inducing inputs $z \in \mathbb{R}^{D'}$ also. Then, since u is deterministic, all that changes in the expressions is that

- each kernel evaluation and kernel expectation involving x, x' is scaled by $k(u, u')$, and
- each f takes u as an argument also.

A.5.2.5 Note that there is no need to make any explicit change to $q(x|f)$.

A.5.2.6 Similarly, control inputs can be incorporated in the likelihood: $y_t = \mathsf{C}x_t + \mathsf{D}u_t + \rho_t$, which slightly modifies the likelihood terms. Note that by setting the first co-ordinate of u_t to 1 for all t, we can incorporate a bias.

A.5.3 Linearisation parameters for squared exponential covariance functions

A.5.3.1 For the EKF parameters, we only need

$$\left[\frac{\partial k(x, z|\Lambda)}{\partial x}\right]_{\vec{\mu}_t} = -\Lambda^{-1}(\vec{\mu}_t - z)k(\vec{\mu}_t, z|\Lambda) \tag{A.131}$$

and for the SLF parameters, we can use the results for expectations to show

$$\mathbb{E}_t\left[\frac{\partial k(x, z|\Lambda)}{\partial x}\right] = -\Lambda^{-1}(\vec{\mu}_t - z)\frac{k(\vec{\mu}_t, z|\Lambda + \vec{\Sigma}_t)}{\sqrt{|I + \vec{\Sigma}_t\Lambda^{-1}|}} \tag{A.132}$$

$$\mathbb{E}_t[k(x, z|\Lambda)] = \frac{k(\vec{\mu}_t, z|\Lambda + \vec{\Sigma}_t)}{\sqrt{|I + \vec{\Sigma}_t\Lambda^{-1}|}}. \tag{A.133}$$

Appendix B

Mathematical background

B.1 Probability and measure

B.1.1 Roughly speaking, a *measure space* $(\Omega, \Sigma, \mathbb{P})$ is a set of values Ω, a set of reasonable ("measurable") subsets of Ω collected in Σ, and a function which measures the size of any subset ($\mathbb{P}$). Usually, there isn't any need to think about the measure space in any detail.

B.1.2 Measure gives a good rigorous foundation to integration. The integral

$$\int g(x)\, d\mathbb{P}(x) \tag{B.1}$$

means integrate g with respect to x, with the length of of subsets of the domain being given by $\mathbb{P}$. Various other notations are used, such as $\int g(x)\, \mathbb{P}(dx)$.

B.1.3 Σ barely ever needs mentioning. It is introduced to deal with technicalities in continuous spaces. In particular, the Borel subsets of $\mathbb{R}$ are a collection of well behaved subsets that avoid exotic contradictions.

B.1.4 **Countably additive Borel measure** A countably additive Borel measure is one for which every Borel set is measurable. It is finite if the measure of Ω is finite.

B.1.5 **Lebesgue measure and densities** The usual measure of length for functions of a real variable is called Lebesgue measure. A measure $\mathbb{P}$ is said to be absolutely

continuous with respect to another measure $\mathbb{Q}$ if $\mathbb{P}[A] = 0$ whenever $\mathbb{Q}[A] = 0$ for any $A \in \Sigma$. Then there is a change of variables formula

$$\int g(x) d\mathbb{P}(x) = \int g(x) \frac{d\mathbb{P}}{d\mathbb{Q}}(x)\, d\mathbb{Q}(x) \tag{B.2}$$

where the function $\frac{d\mathbb{P}}{d\mathbb{Q}}(x)$ is called the Radon-Nikodym derivative.

B.1.6 **Probability measures** A probability measure is a measure for which $\mathbb{P}[\Omega] = 1$. If such a measure has density with respect to Lebesgue measure, then that density is its probability density function. Then the $\mathbb{E}[g(x)]$ is exactly the integral in Equation (B.1).

B.1.7 **Infinite dimensions** There isn't any Lebesgue measure in infinite dimensional spaces. Hence there is no probability density function. Throughout the thesis, I use the informal notation

$$\int g(x) p(f)\, df = \int g(x)\, d\mathbb{P}(f) \tag{B.3}$$

where $\mathbb{P}$ is the measure of f. (It is noted in the introduction that I do not normally distinguish between measures and densities in notation.)

B.1.8 **The infinite dimensional KL** The suitable generalisation for the KL divergence when either one or the other measures has no density with respect to Lebesgue measure is

$$D_{KL}(\mathbb{Q}||\mathbb{P}) = \int \log \frac{d\mathbb{Q}}{d\mathbb{P}}\, d\mathbb{Q}. \tag{B.4}$$

B.1.9 For details on probability and measure, see the excellent introduction of Fremlin (2010).

B.2 Function spaces and linear operators

B.2.1 A function space is a vector space (a linear space). That is, it is closed under addition and scalar multiplication.

B.2.2 Usually, we are interested in using function spaces with some additional structure, like an inner product space. This is nothing but a function space with an inner product which exists for any pair of functions in the space.

B.2.3 **Hilbert spaces** If the inner product space is complete (roughly, closed under taking limits) then it is called a Hilbert space.

B.2.4 **L^2 spaces** The most usual example of Hilbert spaces are $L^2(\Omega, \mathbb{P})$ spaces. These are spaces where the inner product is

$$\langle h, g \rangle = \int_\Omega g(x)^* h(x) d\mathbb{P}(x). \tag{B.5}$$

B.2.5 When L^2 is used without qualification, it usually refers to the case where the domain is Euclidean and the measure is Lebesgue.

B.2.6 **Linear operators** A linear operator maps one function space to another. Sometimes, there are minor technicalities to deal with around edge cases: for instance, we think of the Fourier transform as a linear operator on L^2 spaces, but we extend it functions which are not square integrable, like sinc or even sin. In the latter case, we get a Fourier transform which is made of Dirac δs, which are not functions in the ordinary sense. As with the treatment of the measure of functions as densities, we lose nothing by thinking of δ as a density.

B.2.7 **Commuting self-adjoint operators share an eigenbasis** The adjoint of an operator $\mathcal{A}$ is denoted $\mathcal{A}^*$, and is such that

$$\langle \mathcal{A} g, h \rangle = \langle g, \mathcal{A}^* h \rangle, \tag{B.6}$$

comparably to finite dimensional case (matrices). An operator which is self-adjoint has an orthonormal eigenbasis. Suppose that two self-adjoint operators commute. Then if v is an eigenfunction of $\mathcal{B}$ with eigenvalue λ,

$$\mathcal{A}\mathcal{B} v = \mathcal{A} \lambda v \tag{B.7}$$

which means that $\mathcal{B}v$ is an eigenfunction of $\mathcal{A}$, but $\mathcal{B}v \propto v$ so v is also an eigenfunction of $\mathcal{A}$.

B.2.8 Further relevant details on function spaces and the Fourier transform can be found in Vetterli et al (2012).

B.2.1 O notation

B.2.1 $g(N) \in \mathcal{O}(h(N))$ means there are constants $c, N_0 \in \mathbb{R}$ such that

$$\frac{g(N)}{h(N)} \leq c \quad \forall N \geq N_0. \tag{B.8}$$

B.2.2 Comparably, $g(N) \in o(h(N))$, assuming $h(N)$ becomes nonzero for sufficiently large N, means

$$\lim_{N \to \infty} \frac{g(N)}{h(N)} = 0. \tag{B.9}$$

Appendix C

Dataset and experimental information

C.1 Experimental details for Chapters 2 and 3

C.1.1 **Figure 2.5** For each case, I used three 80/20 train/test splits. For the full batch results, the inducing points were optimised using the EM-like method of Burt et al (2020b), and the parameters were optimised using LBFGS (Paragraph 1.2.2.7). A range of values of M were used: up to 3 000 for the house price dataset, and up to 3 600 otherwise.

C.1.2 For the batched data, I used the Adam optimiser (Paragraph 1.2.2.6) with a learning rate of 10^{-3}, which was selected as the best performing of a few different learning rates for some example choices of M and the batch size. I ran between 10^4 and 2×10^4 iterations (fewer for larger M or N), and evaluated the test performance every 10^3 iterations, which is what is plotted. The time to run the test evaluation is not included.

C.1.3 **Figure 3.6** I considered target data widths $\tilde{W}_x$ as indicated by the x-axis labels, and sampled $N = 1\,000$ inputs from either a uniform distribution on $[-\tilde{W}_x/2, \tilde{W}_x/2]$, or a zero mean Gaussian distribution with standard deviation $\tilde{W}_x/6$. In either case, I found W_x from the actual sampled data.

C.1.4 I used a randomly sampled signal standard deviation $\sigma_f \approx 0.5422$ and a signal to noise ratio (SNR) $\sigma_f/\sigma \approx 1.255$, and used a unit lengthscale ($\ell = 1$). The outputs were sampled from the zero mean GP indicated. In either case, I set M as the next larger even number than MW^{-1}, and evaluated both the log marginal likelihood and variational objective for gridded values of MW^{-1} and W/W_x. The MW^{-1} plotted is the design value; it does not take into account slight changes in the value due to enforcing M is even.

C.1.5 **Figures 3.9 and 3.10** For the synthetic experiment, I generated $N = 10\,000$ data points in 1 and 2 dimensions by sampling from a GP with a Gaussian or Matérn-5/2 covariance function, with unit lengthscales, unit variance, and set the SNR to 0.774 (arbitrarily chosen, poor signal to noise ratio for a challenging dataset). In 1D I sample the training inputs uniformly on a width $6\sqrt{N/2}$ centred interval. In 2D I did the same in each dimension, but with width 5. I then fit each model plotted, training using LBFGS and using the same initialisation in each case, other than necessary restrictions on the choice of covariance functions as described in the main text. The initial values were lengthscales of 0.2, and unit signal and noise variances. I did five random trials, and plot the 2 standard deviation error bars; this is for error bars in timing, but for inducing points we also have uncertainty due to randomness in the inducing point (re)initialisation method.

C.1.6 Inducing points with inducing inputs optimised takes far longer than the other methods, so was excluded. Here AFS uses $W = W_x/0.95$ as discussed in the text, and the approximation interval for VFF is set 0.1 wider than the data in each direction. I found unresolved issues in the implementation of the B-splines method which prevented getting comparable results in this setting.

C.1.7 **Figure 3.13** For the gridded setting, I repeat the same experiment, but constrain the inputs to be gridded. Now $W = W_x$ for AFS.

C.1.8 **Figures 3.11 and 3.12** For the real-world experiments, I used a similar setup as the synthetic experiments in terms of initialisations. Guided by the synthetic results, I use the full cubic grid of frequencies for RKHS-FS, but use a spherical mask for AFS. I set the approximation window for B-splines as with RKHS-FS, and I use fourth order B-splines. The experiments were generally run on CPU to avoid memory-related distortion of the results, with the exception of SKI, which was run

on GPU since it depends on GPU execution for faster MVMs. The grid size setting for SKI is discussed in the main text.

C.1.9 I implemented AFS and RKHS-FS using `gpflow` (Matthews et al, 2017), which I also used for inducing points, reusing some of the code of Burt et al (2020b) for the k-DPP (re)initialisation method. I used the `gpytorch` implementation of SKI (Gardner et al, 2018a), and the publicly available Tensorflow 2 implementation for B-splines[1], in both cases without any modification.

C.2 Experimental details for Chapter 4

C.2.1 **Figure 4.2** The training data is sampled with $N = 162$. For the stationary baseline, I used exact GP regression, with a squared exponential covariance function, and the lengthscale and noise variance initialised low (0.02 and 0.05 respectively).

C.2.2 For the multiresolution approximations, I set $\iota = -2, Q = 1$, used features up to scale 3, and set the maximum width such that shifts to ± 20 were included. I kept the component weight fixed (at 1), which seemed to lead to better behaviour. The decay rates and noise variance were initialised at 0.5, the centre at 0, and the weight and scale at 1. In all of these cases, I optimised the objective using LBFGS (Paragraph 1.2.2.7). For Gibbs' covariance function, I used the sparse formulation described in the main text, $M = N$ inducing points initialised using k-means, and set the prior mean for the log lengthscale to $\log 0.3$, and the hyperprior variance and lengthscale to 1, optimising using 10^4 iterations of Adam with a learning rate of 10^{-2}.

C.2.3 For the precipitation dataset, I used a similar setup. The differences were as follows. For the multiresolution covariance functions, I used $Q = 3$. I used a maximum scale of 3 for db4 and 4 for Haar. I used $\iota = -1$ for db4. I set the width equal to that of the data in each dimension for Haar, but 5 wider than the data in each direction for db4, since those wavelets have larger support size. For the stationary covariance function, I used an odd frequency AFS approximation with $M = 5\,700$. For Gibbs' covariance function, I used $M = 1\,000$, and found unstable performance for larger M. I tried various values of the hyperprior settings and reported the best (mean of

[1] https://github.com/HJakeCunningham/ASVGP

$\log 0.3$, lengthscale 1.5, variance 1). I used 10^3 iterations, still with a learning rate of 10^{-2}, which I found to be sufficient.

C.2.4 For the synthetic active learning experiment, the models were set up as above, but pretrained on data which was evenly spread across the whole dataset (no censoring; $N = 180$). For the precipitation dataset, the pretrained models are from the thinned dataset. The initial data for the synthetic active learning was every 18th point starting with the 9th.

C.2.5 **Implementation details** I implemented Gibbs' covariance function in `gpytorch` (Gardner et al, 2018a).

C.2.6 As with AFS and RKHS-FS, I implement the multiresolution methods using `gpflow`, while the sparse Gibbs' implementation is based on `gpytorch`. To leverage sparsity as described in Section 4.3, I place the code snippet shown below in the precomputation phase code, with `model` being the model class, and `model.B` contains $\mathsf{B}' = \mathsf{K}_{u\mathsf{f}}\mathsf{K}_{u\mathsf{f}}^\top$. The listing has `tf` as tensorflow, and uses the `lab`[2].

[2] https://github.com/wesselb/lab

```
    # Store a fill-reducing permutation in the model
    placeholder = model.B + scipy.sparse.identity(M)
    model.factor = sksparse.cholmod.analyze(placeholder)

    @tf.custom_gradient
    def sparse_logdet_and_solve(B_vals, b, B_idx):
        # Convert to scipy's CSC matrix format -- B has the same
                                            sparsity and shape as
                                            placeholder
        # But due to reordering for tf.sparse ops, need to explicitly
                                             use B_idx to get the
                                            alignment right
        # Then need to provide a dummy adjoint for B_idx
        v = lab.to_numpy(B_vals)
        row = lab.to_numpy(B_idx[:, 0])
        col = lab.to_numpy(B_idx[:, 1])

        B_scipy = scipy.sparse.coo_matrix((v, (row, col)), shape=
                                            placeholder.shape).tocsc()

        model.factor.cholesky_inplace(B_scipy)
        # Log determinant
        out1 = model.factor.logdet()
        # Linear solve
        out2 = model.factor(lab.to_numpy(b))

        def grad(upstream1, upstream2): # adjoint
            b_back = model.factor(upstream2)
            A_back = - b_back[row] * out2[col]
            A_back = A_back + upstream1*lab.squeeze(model.factor.inv
                                                ()[row, col].A)
            return A_back, b_back, lab.zeros(*A_idx.shape)

        return (out1, out2), grad
    model.sparse_logdet_and_solve = sparse_logdet_and_solve
```

C.3 Experimental details for Chapter 5

C.3.1 **Comparing continuous and discrete time models on van der Pol data** The plotted groundtruth trajectory was sampled with a time spacing of 0.1, and the plotted training data was sampled with the stated observation noise with a spacing of 0.3. For both the continuous and discrete time models, I used $M = 25$, initialised $\hat{\mathsf{A}}, \hat{\mathsf{Q}}$ randomly with small values, and initialised $\hat{b}$ such that the initial latent means interpolated the training data.

C.3.2 In continuous time, I learnt the parameters on the same 0.1 spaced grid as the groundtruth latents are plotted. These are nominally interpolated with a zero order hold, but I used a fixed step size fourth order Runge-Kutta integrator with a step size of 0.1, so there was no need to interpolate.

C.3.3 In both cases, I initialised the initial mean to be the first observation and the initial covariance to a small scaled identity. I kept C and R fixed at the groundtruth, and kept L fixed as stated in the main text.

C.3.4 I chose the learning rate to lead to well behaved progress in the objective, and ran enough iterations to be close to convergence: for discrete time, I used a learning rate of 0.1 and used 500 iterations of Adam with the covariance function parameters fixed for the first 400; for continuous time I used a learning rate of 0.01, 1 000 iterations, with covariance function parameters fixed for the first 500.

C.3.5 **Noise bias experiments** Further to the details in the main text, to generate the data I generated the initial state randomly (from $\mathcal{N}(0, 4)$ for the linear case, and uniformly on $[0, 1]$ in the logistic case) and use three random seeds. I initialised the covariance function parameters at default values (unit lengthscale and variance) and the noise standard deviations at 0.1. For the uncorrelated models, I initialised the inducing inputs linearly spaced between the smallest and largest measurement.

C.3.6 For the correlated model (Correlated SLS), I initialised the inducing inputs, μ_u and Σ_u at the optimum for the the SLS case, since a good initialisation of μ_u, Σ_u is crucial to make progress. However, I still initialise $\mathsf{C} = \mathsf{I}$, so I rescale z, μ_u, Σ_u using

Table C.1 Learning rates and number of steps for Section 5.2.2

LINEAR	LEARNING RATE	ITERATIONS (1)	ITERATIONS (2)
DP	0.01	100	50
DP, Q MATCHED	0.01	100	50
EKS-SG	0.01	30	30
SLS-SG	0.01	30	30
SLS	0.01	30	30
CORRELATED SLS	0.001	100	250
LOGISTIC			
DP	0.001	125	75
DP, Q MATCHED	0.01	125	75
EKS-SG	0.001	30	70
SLS-SG	0.001	30	70
SLS	0.001	30	70
CORRELATED SLS	0.001	100	250

the SLS optimised C_{opt}. That is

$$z_{\text{init}} = z_{\text{opt}}/\mathsf{C}_{\text{opt}} \tag{C.1}$$

$$\mu_{u,\text{init}} = \mu_{u,\text{opt}}/\mathsf{C}_{\text{opt}} \tag{C.2}$$

$$\Sigma_{u,\text{init}} = \Sigma_{u,\text{opt}}/\mathsf{C}^2_{\text{opt}}. \tag{C.3}$$

C.3.7 I checked a few settings of the learning rate number of iterations for a few values of $\mathsf{P}_{gt}, \mathsf{L}_{gt}$ for each method and each dataset to determine a suitable scheme for each. The learning rates and number of iterations are tabulated in Table C.1.