

Comparing Correlation Functions and Exploring Efficiency and Identifiability Issues for the Gaussian Process

Ivor Walker

Supervised by Dr Michail Papathomas

Abstract

Gaussian processes (GPs) provide a flexible, non-parametric Bayesian framework for statistical modelling and machine learning, enabling principled uncertainty quantification in regression, classification, and beyond. This dissertation undertakes a systematic investigation of GP models, beginning with a conceptual bridge between the weight-space view of Bayesian linear models and the function-space formulation of GPs.

Using both simulated and real datasets, we compare a range of widely used covariance (correlation) functions, assessing their implications for smoothness, differentiability, and spectral properties. A unified notion of smoothness is developed, linking length-scale and mean-square differentiability through Taylor expansion, upcrossing theory, and spectral decay rates.

Alongside theoretical exploration, the work addresses key practical challenges in GP modelling, notably the cubic computational cost of large-scale inference and identifiability issues in residual modelling after linear fits. Two distinct computational strategies dominate: stochastic variational Gaussian processes (SVGP), which provide slower and less accurate approximations for any kernel, and structured kernel methods such as celerite, which exploit algebraic properties of specific kernels highly accurate and fast inversion. Benchmark experiments on galaxy spectral energy distribution residuals highlight a trade-off between identifiability and computational efficiency: the squared exponential kernel performs well but is computationally intractable for large datasets, while celerite is efficient but performed poorly. The findings underscore that the choice of covariance function and computational strategy must be guided jointly by the data's underlying structure and the end goal. Recommendations are provided for future work, including GPU-accelerated SVGP, modified Matern kernels to emulate squared exponentials, and other structured, low-dimensional GP approximations.

Contents

1	Introduction	4
2	Defining the Gaussian Process	5
2.1	Weight-space view	5
2.1.1	Standard linear model	5
2.1.2	Determining weights	5
2.1.3	Predictive distribution	7
2.1.4	Projections of inputs into feature space	8
2.1.5	Computational issues	9
2.2	Function-space view	10
2.2.1	Gaussian processes (GP)	10
2.2.2	Predictive distributions with noise-free observations	10
2.2.3	Predictive distributions with noisy observations	11
2.2.4	Marginal likelihood	11
2.2.5	Algorithm for Gaussian processes	12
3	Exploring Covariance Functions	13
3.1	Characteristics of covariance functions and matrices	13
3.1.1	Gram matrices	13
3.1.2	Eigenvalue and eigenfunctions of covariance matrices	13
3.1.3	Choosing the length scale and other hyperparameters	14
3.1.4	Mean square continuity and differentiability	15
3.2	Stationary covariance functions	19
3.2.1	Stationarity and isotropicism	19
3.2.2	Spectral density	19
3.2.3	GPs from stationary covariance functions in MS space	19
3.3	Stationary covariance functions	20
3.3.1	Squared exponential (SE)	20
3.3.2	Rational quadratic (RQ)	24
3.3.3	γ -exponential and exponential	27
3.3.4	Matern-class	30
4	Computational Issues	35
4.1	Cholesky decomposition	35
4.2	Subset-of-data (SoD)	35
4.3	Eigenfunction and eigenvalue approximation of covariance matrices	36
4.3.1	Approximating covariance matrices with Nystrom	36
4.4	Sparse approximations	37
4.4.1	Prior approximations	38
4.4.2	Posterior approximations	39
4.5	Celerite kernels	41
4.5.1	The celerite model	41
4.5.2	Building Matern 3/2 from celerite	41
4.5.3	Cholesky factorisation and inversion of celerite kernels	42
5	Applying Gaussian Processes to Astrophysics	44
5.1	Introduction	44
5.1.1	Astronomical background	44
5.1.2	Low-frequency wobbles	45
5.2	Methodology	46
5.2.1	Using GPs for SED residual modelling	46
5.2.2	Approximations and kernels considered	46
5.3	Results	47
5.4	Discussion	47
5.4.1	Computational cost	47
5.4.2	Identifiability	48
5.5	Conclusion	48
6	Conclusion	49

Introduction

Defining Gaussian processes Most statistical methods describe relationships between data by estimating weights that linearly combine the data to produce an output. This is fundamentally different to Gaussian processes (GPs), which describe a distribution over possible functions that describe the data. We aim to describe what Gaussian processes are in terms of familiar weight-based statistical tools, namely Bayesian regression. We start with an in-depth look into Bayesian linear regression, relating it to frequentist approaches and deriving important results such as the weight and predictive distributions. By applying the linear model to some non-linear flexible function of the data $\phi(X)$, we make the model more flexible. Then, we generalise our Bayesian linear regression in terms of a distribution of possible functions that ϕ could be, and translate our Bayesian linear regression into a Gaussian process. We finish by investigating general properties of Gaussian processes that are analogous to properties of Bayesian linear regression, such as predictive distributions and log-likelihoods, and describe some practical considerations involved in their usage.

Covariance functions The behaviour of the covariance function and its resulting covariance matrix is crucial to understanding the behaviour of a Gaussian process - the covariance matrix appears in every term of the predictive distribution and the marginal likelihood. In particular, the choice of covariance function affects the smoothness of the overall GP. We describe the mechanisms that the covariance function has at its disposal to vary smoothness, namely its degree of differentiability and the choice of length scale. We investigate in detail the stationary family of covariance functions, and develop a new tool to unify differentiability and length scale into a single notion of smoothness. Finally, we introduce the most widespread stationary covariance functions, their properties, and apply them to toy data to illustrate their behaviour.

Computational concerns Training and performing inference on a Gaussian processes requires inverting an $n \times n$ matrix $[K(X, X) + \sigma_n^2]$. Naively inverting this matrix X has a computational complexity of $O(n^3)$. Unfortunately, this cubic complexity is shared by most covariance functions and renders Gaussian processes unusable in practice - a covariance matrix with $n = 1000$ would require over one billion operations to invert naively, and this cost would need to be paid for each hyperparameter optimisation step (epoch) and each prediction. One approach is to approximate this inversion for any covariance matrix. These strategies for any covariance function fall into two categories: those that produce a single approximation for the entire dataset, or those that produce several approximations that are "experts" in a particular region of the dataset and combine these local approximations to form a global approximation. We focus on global approximations because the local-expert methods are much newer and lack robust open-source implementations. Another approach is to design a specific covariance function that produces a covariance matrix which has certain exploitable properties that allow for much quicker and more accurate inversions, at the cost of flexibility. We introduce one structured approach, the celerite model, and discuss the origins of these tradeoffs in detail.

Case study: SED residuals in astrophysics The case study applies these ideas to SED residual modelling in astrophysics. Contemporary SED models can exhibit "low-frequency wobbles" in their residuals that violate the assumption of zero-mean noise used for inference on physical parameters. Because the source of these wobbles is not well understood and their scale varies across galaxies, we need a flexible, probabilistic filter that can remove such correlated noise without causing identifiability issues by touching the regular noise that downstream inference relies on. This is a task that GPs are well suited for. We evaluate the two most performant approximation methods on SED residuals modelling: a Stochastic Variational GP (SVGP) that trades exactness and completeness for the ability to use the preferred SE kernel, and a celerite model that is faster and more accurate at the price of being tied to an inferior Matern 3/2 kernel. The central question is whether the dramatic speed gains from celerite outweigh the fidelity loss when the rougher Matern 3/2 kernel replaces SE in a task that ideally prefers SE-like smoothness. Wall-clock results demonstrate that SVGP required orders-of-magnitude more computation to converge than the celerite setup, and is generally infeasible for the size of datasets being analysed. At the same time, we show that Matern 3/2 overfits and causes identifiability issues by tracking the regular noise too closely, while SE avoids modelling them and captures broad trends. We conclude by developing and reviewing several strategies for addressing SVGP's computational shortcomings, adapting Matern 3/2 to preserve identifiability whilst retaining celerite's speed, and alternative structured approximation methods.

Defining the Gaussian Process

2.1 Weight-space view

2.1.1 Standard linear model

The standard linear model summarises that we have some data-generating function $f(\cdot)$ that linearly combines training data X , parameters of some model W , and some irreducible noise ϵ (because y is rarely a perfect observation of $f(X)$, e.g. if y is measured inaccurately) to produce an output y :

$$\begin{aligned} f(X) &= X^T W \\ y &= f(X) + \epsilon \end{aligned} \tag{1}$$

Errors The standard linear model assumes that our irreducible noise, or errors, ϵ are drawn from a Gaussian distribution:

$$\epsilon \sim N(0, \sigma_n^2 I) \tag{2}$$

We assume our model has zero mean because all the information we have about the data-generating function $f(X)$ is contained in the weights W and the data X . We add a covariance matrix I to describe how the noise for one observation is related to the noise of another observation. In this case, we assume that the noise is independent and identically distributed (i.i.d.) across all observations, so I is an "identity matrix" where each diagonal element is 1 and all off-diagonal elements are 0. The variance σ_n^2 describes how much noise we expect in our observations.

We can combine our expressions for the data-generating function and our assumptions about our noise to produce a conditional distribution that fully describes y , also known as the likelihood of y given X and W :

$$p(y|X, W) = \mathcal{N}(X^T W, \sigma_n^2 I)$$

2.1.2 Determining weights

Our first task is to estimate the weights for our data-generating function W , as we typically do not know these in advance. Frequentist approaches focus on arriving at a single estimate of these weights (\hat{W}) via "maximum likelihood estimation" (MLE), whereas Bayesian approaches treat W as a random variable for which a distribution is specified.

Frequentist regression Because our likelihood $p(y|X, W)$ is Gaussian, it is at its highest density around the expected value $X^T W$ and we can use iterative optimisation routines to find a single estimate for W , \hat{W} , that maximises our likelihood. Because we assume $E[\epsilon] = 0$, the values of W at the maximum of $p(y|X, W)$ are also the values of W that minimise the squared error $\|y - X^T W\|^2$. We can communicate uncertainty surrounding our estimations of W by computing "standard errors", or the ratio between the variance of errors and the variance in X . A high variance in errors shows that the variation in y is more noise than signal captured by X , but a broader range of X makes estimating W easier.

Bayesian regression Instead of producing single point estimates for weights and uncertainty in estimating them, Bayesian statistics treats W as a random variable and specifies an expected value and a variance. Placing W in a probabilistic framework allows us to propagate uncertainty throughout the model and to encode beliefs (e.g. from domain experts) about the weights before observing the data.

We start with a "prior" distribution of W , which the Bayesian linear model assumes:

$$p(W) = N(0, \Sigma_p) \tag{3}$$

Then, we observe the data and update our beliefs about the weights using Bayes' theorem to produce a "posterior" distribution $p(W|X, y)$:

$$p(W|X, y) = \frac{p(y|X, W)p(W)}{p(y|X)}$$

$p(y|X, W)$ is the density of the residuals after applying $p(W)$ to X, W under our assumed noise model ϵ , and $p(y|X)$ is our likelihood.

Deriving our posterior To understand the relationship between $p(W|X, y)$ and W , we can ignore terms that do not vary with W (e.g. our marginal likelihood) by absorbing them into a proportionality constant:

$$p(W|X, y) \propto p(y|X, W)p(W) \tag{4}$$

The probability density function (PDF) measures probability per unit length. Integrating the PDF over a range gives us the probability of observing a value in that range. The PDF of a Gaussian distribution is given by:

$$N(x|\mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu)\right) \tag{5}$$

We can get a PDF for our likelihood by representing our errors in squared error form and substituting it into the Gaussian PDF 5:

$$p(y|X, W) = \exp\left(-\frac{1}{2\sigma_n^2}\|y - X^T W\|^2\right) \quad (6)$$

Reframing $p(W)$ as a PDF:

$$p(W) = \frac{1}{[\sqrt{\sigma_p}] \sqrt{2\pi}} \exp\left(-\frac{1}{2} \frac{([W] - [0])}{[\Sigma_p]}\right)$$

The first term can be absorbed into the proportionality constant. Rewriting the second term as a negative exponential:

$$p(W) \propto \exp\left(-\frac{1}{2} W^T \Sigma_p^{-1} W\right) \quad (7)$$

Putting both expressions for 6 and 7 into 4:

$$p(W|X, y) \propto \exp\left(-\frac{1}{2\sigma_n^2}\|y - X^T W\|^2\right) \exp\left(-\frac{1}{2} W^T \Sigma_p^{-1} W\right)$$

Expanding $\|y - X^T W\|^2$ to $y^T y - 2y^T X W + W^T X^T X W$:

$$p(W|X, y) \propto \exp\left(-\frac{1}{2\sigma_n^2}(y^T y - 2y^T X W + W^T X^T X W)\right) \exp\left(-\frac{1}{2} W^T \Sigma_p^{-1} W\right)$$

Putting both exponentials together by adding their powers:

$$p(W|X, y) \propto \exp\left(\frac{1}{\sigma_n^2}(y^T y - 2y^T X W + W^T X^T X W) + \left(-\frac{1}{2} W^T \Sigma_p^{-1} W\right)\right)$$

Rearranging the inside term to be a quadratic, linear and constant term in W :

$$p(W|X, y) \propto \exp\left(\frac{1}{2} W^T \left(\frac{1}{\sigma_n^2} X^T X + \Sigma_p^{-1}\right) W - \left(\frac{1}{\sigma_n^2} y^T X\right) W + \frac{1}{2} y^T y\right)$$

We can ignore the constant final term. Introducing these terms to simplify this result:

$$\begin{aligned} A &= \Sigma_p^{-1} + \frac{1}{\sigma_n^2} X^T X \\ b &= \frac{1}{\sigma_n^2} y^T X \\ p(W|X, y) &\propto \exp\left(-\frac{1}{2} W^T A W + b^T W\right) \end{aligned} \quad (8)$$

Deriving the properties of the posterior by completing the square Now we have a simplified form for the posterior's PDF, we need to get it into a Gaussian form to recover the properties of the posterior distribution.

Bringing all terms inside the exponential to a single term:

$$-\frac{1}{2} W^T A W + b^T W = \frac{1}{2} (-W^T A W + 2b^T W)$$

Completing the square on our new inner term $W^T A W - 2b^T W$

$$W^T A W - 2b^T W = (W - A^{-1}b)^T A (W - A^{-1}b) - b^T A^{-1}b \quad p(W|X, y) \propto \exp\left(-\frac{1}{2} ((W - A^{-1}b)^T A (W - A^{-1}b) - b^T A^{-1}b)\right) \quad (9)$$

Our expression lines up with the Gaussian PDF's "kernel" term $\exp(-\frac{1}{2}(W - \mu)^T \Sigma^{-1}(W - \mu))$, where $\mu = A^{-1}b$ and $\Sigma = A^{-1}$ ($\Sigma^{-1} = A$). Therefore, 9 can be represented as a Gaussian distribution:

$$p(W|X, y) \sim N(A^{-1}b, A^{-1}) \quad (10)$$

Inside our definition of A at 8, we are missing an expression for Σ_p . Assuming independence of noise under the linear model 2, our weight variance Σ_p under the Bayesian linear model is "isotropic", meaning it is the same in all directions.

$$\Sigma_p = \tau^2 I$$

I is our familiar identity matrix and τ^2 is a scalar variance term, chosen as a prior.

Substituting the isotropic prior Σ_p into A :

$$A = \Sigma_p^{-1} + \frac{1}{\sigma_n^2} X^T X = [\tau^2 I]^{-1} + \frac{1}{\sigma_n^2} X^T X = \frac{1}{\tau^2} I + \frac{1}{\sigma_n^2} X^T X$$

Simplifying:

$$A = \frac{1}{\sigma_n^2} \left(X^T X + \frac{\sigma_n^2}{\tau^2} I \right) \quad (11)$$

Gaussian posteriors and ridge regression So far we have worked exclusively within the Bayesian paradigm, but we can draw some value of W from our posterior distribution to relate it to a frequentist framework. For Gaussian posteriors, our expected value of $W A^{-1}b$ is also its mode. This is called the maximum a posteriori (MAP) estimate of W , and is due to symmetries in linear model and posterior and is not the case in general. Our MAP estimate does not matter within the Bayesian framework but is equivalent to our frequentist \hat{W} .

Substituting our full expressions for A 11 and b 8 into our MAP estimation:

$$W_{\text{MAP}} = A^{-1}b = \left[\frac{1}{\sigma_n^2} (X^T X + \frac{\sigma_n^2}{\tau^2} I) \right]^{-1} \cdot \left[\frac{1}{\sigma_n^2} y^T X \right]$$

Inverting LHS term of A :

$$A^{-1} = \frac{\sigma_n^2}{X^T X + \frac{\sigma_n^2}{\tau^2} I} = \sigma_n^2 \left(X^T X + \frac{\sigma_n^2}{\tau^2} I \right)^{-1}$$

Substituting this back into W_{MAP} cancels out the σ_n^2 term in A with the $\frac{1}{\sigma_n^2}$ term in B :

$$W_{\text{MAP}} = \sigma_n^2 \left(X^T X + \frac{\sigma_n^2}{\tau^2} I \right)^{-1} \cdot \frac{1}{\sigma_n^2} y^T X = \left(X^T X + \frac{\sigma_n^2}{\tau^2} I \right)^{-1} \cdot y^T X$$

This is equivalent to the solution to ridge regression, where $\lambda = \frac{\sigma_n^2}{\tau^2}$.

$$W_{\text{ridge}} = (X^T X + \lambda I)^{-1} X^T y$$

Ridge regression introduces some bias to lower the variance in a frequentist linear model by shrinking weights, where the λ term controls the amount of shrinkage applied to the weights. This is traditionally useful where variance is particularly high (e.g. multicollinearity) and can be reduced at the cost of little bias.

Our MAP estimation in Bayesian linear regression with isotropic priors is equivalent to ridge regression, where the amount of bias we introduce depends on our confidence in our priors. The more we trust our prior, the higher our λ and the more we shrink our weights towards zero. A lower τ means we are more confident in $p(W)$ and have better priors, whereas a higher σ_n means lower confidence in $p(y|X, W)$ and worse weights that should be shrunk closer to zero.

2.1.3 Predictive distribution

Deriving the predictive distribution Our second task is to make predictions y_* using new input data X_* and our previously learned weights W . Frequentist methods simply multiply \hat{W} by X_* , but this does not propagate uncertainty in W . In this Bayesian framework, we form a "predictive distribution" which we sample from to get our noise-free function evaluations $f(X_*)$ (denoted f_*) and add ϵ to get our noisy predictions y_* .

$$p(f_*|X_*, X, y) = \int p(f_*|X_*, W) \cdot p(W|X, y) dW$$

$p(f_*|X_*, W)$ is what we think the function looks like after producing a prediction using X_* and perfect knowledge of W . $p(W|X, y)$ is our familiar 10 posterior distribution of weights. $p(f_*|X_*, W) \cdot p(W|X, y)$ is the joint distribution of our predictions and our posterior weights, which gets us the conditional distribution $p(f_*, W|X_*, X, y)$ by definition of conditional probability. Because $p(f_*, W|X_*, X, y)$ relies on our perfect knowledge of W , which we lack, we integrate over all possible W to get the final predictive distribution $p(f_*|X_*, X, y)$

$p(f_*|X_*, W)$ is our error distribution, which we assume to be distributed normally and independently with our I identity matrix:

$$p(f_*|X_*, W) = \mathcal{N}(f_*|W^T X_*, \sigma_n^2 I)$$

Substituting into 5 and absorbing the LHS term into the proportionality constant:

$$p(f_*|X_*, w) \propto \exp \left(-\frac{1}{2} \frac{1}{\sigma_n^2} (f_* - W^T X_*)^2 \right)$$

Multiplying $P(f_*|X_*, W)$ and $p(W|X, y)$ to get our conditional $p(f_*, W|X_*, X, y)$, and add the exponents:

$$p(f_*, W|X_*, X, y) \propto \exp \left(\frac{1}{2} (-W^T A W + 2b^T W) + \left(-\frac{1}{2} \frac{1}{\sigma_n^2} (f_* - W^T X_*)^2 \right) \right)$$

Combining the terms inside the exponent:

$$p(f_*, W|X_*, X, y) \propto \exp \left(-\frac{1}{2} \left(W^T A W - 2b^T W + \frac{1}{\sigma_n^2} (f_* - W^T X_*)^2 \right) \right)$$

Expanding the squared term:

$$p(f_*, W|X_*, X, y) \propto \exp \left(-\frac{1}{2} \left(W^T A W - 2b^T W + \frac{1}{\sigma_n^2} (f_*^2 - 2f_* W^T X_* + W^T X_* X_*^T X_*) \right) \right)$$

Similar to our posterior, we can rearrange this to be a quadratic, linear and constant term in W :

$$p(f_*, W|X_*, X, y) \propto \exp \left(-\frac{1}{2} \left(W^T \left(A + \frac{1}{\sigma_n^2} X_* X_*^T \right) W - 2 \left(b + \frac{1}{\sigma_n^2} f_* X_* \right)^T W + \frac{1}{\sigma_n^2} f_*^2 \right) \right) \quad (12)$$

We can define new terms A_* and b_* to simplify this expression:

$$\begin{aligned} A_* &= A + \frac{1}{\sigma_n^2} X_* X_*^T \\ b_* &= b + \frac{1}{\sigma_n^2} f_* X_* \end{aligned}$$

Substituting into 12:

$$p(f_*, W|X_*, X, y) \propto \exp \left(-\frac{1}{2} \left(W^T A_* W - 2b_*^T W + \frac{1}{\sigma_n^2} f_*^2 \right) \right)$$

Integrating out W to get our predictive distribution:

$$p(f_*|X_*, X, y) = \int p(f_*, W|X_*, X, y) dW \propto \int \exp \left(-\frac{1}{2} \left(W^T A_* W - 2b_*^T W + \frac{1}{\sigma_n^2} f_*^2 \right) \right) dW \quad (13)$$

Factoring out $\frac{1}{\sigma_n^2} f_*^2$ as it does not depend on W (since $\int \exp(X) dX = \exp(X)$):

$$= \exp \left(-\frac{1}{2} \frac{1}{\sigma_n^2} f_*^2 \right) \times \int \exp \left(-\frac{1}{2} (W^T A_* W - 2b_*^T W) \right) dW$$

Evaluating the RHS multivariate Gaussian integral:

$$\int \exp \left(-\frac{1}{2} (W^T A_* W - 2b_*^T W) \right) dW = \frac{(2\pi)^{D/2}}{\sqrt{|A_*|}} \exp \left(\frac{1}{2} b_*^T A_*^{-1} b_* \right)$$

Substituting back into 13:

$$p(f_*|X_*, X, y) \propto \exp \left(-\frac{1}{2} \frac{1}{\sigma_n^2} f_*^2 \right) + \frac{(2\pi)^{D/2}}{\sqrt{|A_*|}} \cdot \exp \left(\frac{1}{2} b_*^T A_*^{-1} b_* \right)$$

Now that no part of our expression is dependent on W , we need an expression for everything that depends on f_* .

Absorbing the second term (since it does not depend on f_*) into the proportionality constant, and combining the remaining exponential terms by adding their powers:

$$p(f_*|X_*, X, y) \propto \exp \left(-\frac{1}{2} \frac{1}{\sigma_n^2} f_*^2 + \frac{1}{2} b_*^T A_*^{-1} b_* \right)$$

Similar to deriving properties from our posterior, we can rearrange this expression and complete the square to derive the properties of our predictive distribution:

$$p(f_*|X_*, W) \sim N(X_*^T A^{-1} b, X_*^T A^{-1} X_*) \quad (14)$$

The variance is quadratic in X_* with A^{-1} , showing that predictive uncertainties grow with size of X_* .

2.1.4 Projections of inputs into feature space

One problem with this model is that it assumes a linear relationship between X and y . We can project our inputs into a higher dimensional feature space and apply a linear model in this space to express non-linear relationships between X and y .

Defining $\phi(X)$ as a function that maps a D -dimensional input vector X into an N dimensional feature space, our standard linear model becomes:

$$f(X) = \phi(X)^T W$$

For example, a scalar x could be projected into the space of powers of x : $\phi(x) = [1, x, x^2, \dots, x^d]^T$ for a polynomial basis expansion of degree d to represent a d -power relationship between x and y . Substituting $\phi(X)$ for X in 14:

$$p(f_*|X_*, X, y) = N(\phi(X_*)^T A_\phi^{-1} b_\phi, \phi(X_*)^T A_\phi^{-1} \phi(X_*)) \quad (15)$$

Where A_ϕ and b_ϕ are now:

$$A_\phi = \Sigma_p^{-1} + \frac{1}{\sigma_n^2} \phi(X)^T \phi(X) b_\phi = \frac{1}{\sigma_n^2} \phi(X)^T y$$

2.1.5 Computational issues

Avoiding inversion of A_ϕ 15 requires inverting the $N \times N$ matrix A_ϕ , where N is dimension of feature space, to get the expected value and variance.

Typically, matrices are inverted using Gaussian elimination. We need to perform a "forward" pass which requires N pivots on every row and column, N eliminations per pivot, and up to $2N$ columns to update, resulting in an $O(N^3)$ time complexity. Then, we need to perform a backwards pass in the opposite direction which is another $O(N^3)$ operation. Finally, we need to multiply the inverse by the RHS vector b_ϕ , which is an $O(N^2)$ operation but appears trivial next to these two cubic steps.

We can mitigate this for a particular class of high-dimensional $N > n$ problems by restating the predictive distribution in terms of the number of training data points n which would require inverting an $n \times n$ matrix instead. For polynomial basis expansions, N is degree D multiplied by number of features, so N can be very large or even infinite (e.g. SE).

Substituting b_ϕ into our predictive distribution mean:

$$\mathbb{E}_{p(f_*|X_*,X,y)}[f_*] = \phi(X_*)^T \cdot A_\phi^{-1} \cdot \left[\frac{1}{\sigma_n^2} \phi(X)^T y \right]$$

Rearranging to isolate $A_\phi^{-1} \phi(X)$

$$= \frac{1}{\sigma_n^2} \left[A_\phi^{-1} \phi(X) \right]^T y$$

We can use the Sherman-Morrison-Woodbury identity (SMW) to get an expression for A_ϕ^{-1} directly, where $K = \phi(X)^T \Sigma_p \phi(X)$:

$$A_\phi^{-1} = \Sigma_p - \Sigma_p \phi(X) (K + \sigma_n^2 I)^{-1} \phi(X)^T \Sigma_p$$

For the mean, we can use the SMW identity again to get an expression for $A_\phi^{-1} \phi(X)$

$$A_\phi^{-1} \phi(X) = \sigma_n^2 \Sigma_p \phi(X) (K + \sigma_n^2 I)^{-1} \quad (16)$$

Substitute in 16 into our 15:

$$\mathbb{E}_{p(f_*|X_*,X,y)}[f_*] = \phi(X_*)^T \frac{1}{\sigma_n^2} \left[\sigma_n^2 \Sigma_p \phi(X) (K + \sigma_n^2 I)^{-1} \right]^T y$$

$\frac{1}{\sigma_n^2}$ and σ_n^2 cancel out, leaving us with this final expression for the mean:

$$\mathbb{E}_{p(f_*|X_*,X,y)}[f_*] = \phi(X_*)^T \cdot \Sigma_p \phi(X) (K + \sigma_n^2 I)^{-1} y \quad (17)$$

For the variance, we cannot use the Sherman-Morrison identity to arrive at an expression for $A_\phi^{-1} \phi(X_*)$ because $\phi(X_*)$ is an arbitrary N -vector, not one of the columns of $\phi(X)$. Instead, we use the A_ϕ^{-1} expression we derived earlier to get an expression for $A_\phi^{-1} \phi(X_*)$:

$$A_\phi^{-1} \phi(X_*) = \Sigma_p \cdot \phi(X_*) - \Sigma_p \phi(X) (K + \sigma_n^2 I)^{-1} \phi(X)^T \Sigma_p \cdot \phi(X_*)$$

Substituting this into 15:

$$\text{Var}_{p(f_*|X_*,X,y)}[f_*] = \phi(X_*)^T \Sigma_p \phi(X_*) - \phi(X_*)^T \Sigma_p \phi(X) (K + \sigma_n^2 I)^{-1} \phi(X)^T \Sigma_p \phi(X_*) \quad (18)$$

With our alternative mean 17 and variance 18, we can form an alternative expression for our predictive distribution:

$$p(f_*|X_*,X,y) = \mathcal{N} \left(\begin{aligned} &\phi(X_*)^T \Sigma_p \phi(X) (K + \sigma_n^2 I)^{-1} y, \\ &\phi(X_*)^T \Sigma_p \phi(X_*) - \phi(X_*)^T \Sigma_p \phi(X) (K + \sigma_n^2 I)^{-1} \phi(X)^T \Sigma_p \phi(X_*) \end{aligned} \right) \quad (19)$$

With this alternative formulation, we need to invert the $n \times n$ matrix $K + \sigma_n^2 I$ only. Geometrically, n datapoints can span at most n dimensions in the feature space - if $N > n$, the data forms a subspace of the feature space.

Kernels and the kernel trick In 19, $\phi(\cdot)$ is always an inner product of a positive definite correlation matrix Σ_p , but with different arrangements of $\phi(X)$ and $\phi(X_*)$. We can define $k(X, X') = \phi(X)^T \Sigma_p \phi(X')$ as a covariance function or kernel, where X and X' are either X or X_* . For example, in 19 the definition of $K = \phi(X)^T \Sigma_p \phi(X)$ becomes $K = k(X, X)$.

Introducing $\psi(X)$ to better represent $k(X, X')$ as an inner product:

$$\begin{aligned} \psi(X) &= \phi(X) \Sigma_p^{1/2} \\ k(X, X') &= \psi(X)^T \psi(X') \end{aligned}$$

These inner product representations require us to compute $\phi(X)$ and $\phi(X')$ in the feature space. A higher-dimensional feature space requires more compute to evaluate $\phi(X)$ and more memory to store $\phi(X)$ and $\phi(X')$.

Instead, the representer theorem guarantees that we can find an equivalent kernel that does not require us to explicitly compute $\phi(X)$ or $\phi(X')$ in the feature space. With this "kernel trick" we avoid the associated memory and computational costs of explicitly computing $\phi(X)$ and $\phi(X')$. Since computing the kernel directly is more convenient than the feature vectors themselves, these kernels become the object of primary interest.

For example, if we had some polynomial transformation $\phi(X) = [1, x^1, \dots, x^D]^T$ and Σ_p as an identity matrix, we could define $k(X, X')$ as inner products:

$$\psi(X) = [1, x^1, \dots, x^D]^T k(X, X') = \psi(X)^T \psi(X')$$

This approach requires arranging ϕ and $\phi(X')$ into a D sized vector, then taking the dot product. This is trivial for small D , but as D becomes infinite (e.g. RBF kernel), arranging a D sized vector requires too much memory and the dot product becomes computationally expensive.

Instead, we can define $k(X, X')$ as an equivalent function of X and X' directly:

$$k(X, X') = (1 + X \cdot X')^D$$

This is the polynomial kernel, which is equivalent to our original polynomial basis expansion $\phi(X)$ without explicitly computing $\phi(X)$.

2.2 Function-space view

2.2.1 Gaussian processes (GP)

Bayesian linear model We can define our Bayesian linear model of a real process $f(X)$ entirely in terms of mean function $m(X)$ and covariance function $k(X, X')$:

$$\begin{aligned} m(X) &= \phi(X)^T \mathbb{E}[W] = \phi(X)^T [0] = 0 \\ k(X, X') &= \phi(X)^T \mathbb{E}[WW^T] \phi(X') = \phi(X)^T \Sigma_P \phi(X') \end{aligned} \quad (20)$$

Our covariance function here is in inner product form. The kernel trick here uses the squared exponential (SE) covariance function, also known as the radial basis function (RBF) or Gaussian kernel:

$$k(f(X), f(X')) = \exp\left(-\frac{1}{2} \frac{|X - X'|^2}{l^2}\right)$$

It can be shown that SE corresponds to a Bayesian linear regression model with infinite basis functions.

Function evaluations to a random function We can choose a subset X_{*1} from our test data X_* and apply it to our model get some function evaluations $f(X_{*1})$. $f(X_{*1})$ can be described as a multivariate Gaussian distribution, e.g. in the Bayesian linear model $f(X_{*1}) \sim N(0, k(X_{*1}, X_{*1}))$. Each output $f(X_{\theta*1})$ in our $f(X_{*1})$ vector is a random variable with mean 0 and covariance with each other $K_{\theta\theta'} = k(X_{\theta*}, X_{\theta'*})$. There exists some random function $g(X_{*1})$ for our subsets such that $f(X_{*1}) = g(X_{*1})$. We only know the value of $g(X_{*1})$ at the points X_{*1} , so $g(X_{*1}) = X_{*1} : f(X_{*1})$. Because $g(X)$ entirely consists of random points, we can think of $g(X_{*1})$ as a random function and our distribution $f(X)$ can be seen as a distribution of these random $g(X)$ functions. We can recover our individual $g(X_{*1})$ thanks to consistency - if we marginalised out our subset from the entire distribution $f(X_*)$, we would recover the subset distribution $N(0, K_*(X_{*1}, X_{*1}))$ that describes our random function $g(X_{*1})$.

Definition of a GP A GP is a collection of random variables, any finite number of which have a joint Gaussian distribution. Ultimately, GPs describe a distribution of random functions where each drawn function is a $g(X)$ sample from the GP.

$$\begin{aligned} f(X) &\sim \mathcal{GP}(m(X), k(X, X')), \\ m(X) &= \mathbb{E}[f(X)], \\ k(X, X') &= \text{Cov}(f(X), f(X')) = \mathbb{E}[(f(X) - m(X))(f(X') - m(X'))] \end{aligned}$$

Consistency requirement This definition implies a consistency requirement - any group of functions drawn from our GP can be described by the same distribution as our GP. For example, if our GP implies that $(f(X_1), f(X_2)) \sim \mathcal{N}(\mu, \Sigma)$, then $(f(X_1) \sim \mathcal{N}(\mu_1, \Sigma) \text{ and } f(X_2) \sim \mathcal{N}(\mu_2, \Sigma))$ where $\mu_\theta = m(X_\theta)$ and $\Sigma_{\theta\theta} = k(X_\theta, X_\theta)$. This requirement is also called the marginalisation property, because to get the smaller distribution of $f(X_1)$ we marginalise out the larger distribution of $f(X_1), f(X_2)$ by integrating the larger distribution wrt $f(X_2)$. Consistency is automatically gained if our covariance function specifies entries in a covariance matrix.

2.2.2 Predictive distributions with noise-free observations

Prior distribution over functions $f(X)$ and $f(X_*)$ are jointly distributed according to the prior:

$$\begin{pmatrix} f(X) \\ f(X_*) \end{pmatrix} \sim N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} K(X, X) & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{pmatrix}\right) \quad (21)$$

Posterior distribution of functions To get the posterior distribution of functions given the training data and our prior, we can condition the joint prior distribution on the training data. Intuitively, this is like generating random functions $g(X)$ and rejecting those that do not pass through the training data. Probabilistically, we condition our joint Gaussian prior distribution on the observations to produce a predictive distribution for some point we want to predict for X_* .

Substituting our prior and our conditioning X into the Gaussian multivariate conditioning identity:

$$p(f(X_*)|X_*, X, f(X)) \sim N([0] + [K(X_*, X)][K(X, X)]^{-1}([f(X)] - [0]), [K(X_*, X_*)] - [K(X_*, X)][K(X, X)]^{-1}[K(X, X_*)])$$

Although we condition on X_* , X , and $f(X)$, we only substitute $f(X)$ because X_* and X are known constants, but $f(X)$ is random because it is a sample from the prior. We also swap $f(X_*)$ and $f(X)$ in our prior to match the conditioning identity, such that our input vector into the conditioning identity is $(f(X_*), f(X))^T$.

Simplifying the last term in the mean:

$$p(f(X_*)|X_*, X, f(X)) \sim N(K(X, X_*)K(X, X)^{-1}f(X), K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*)) \quad (22)$$

2.2.3 Predictive distributions with noisy observations

Noisy observations prior It is typical to not have the noise-free function evaluations $f(X)$ as our training data, but instead our noisy observations y . We can simply add ϵ :

$$\text{Cov}(y_p, y_q) = K(X_p, X_q) + \sigma_n^2 \delta_{pq}$$

δ_{pq} represents our independence condition in 1D. This is the Kronecker delta, which returns 1 if indices (p, q) are equal and 0 otherwise. σ_n^2 is the noise variance, which is a constant for all observations.

In matrix form:

$$\text{Cov}(Y) = k(X, X) + \sigma_n^2 I$$

This gives us this prior:

$$\begin{pmatrix} Y \\ f(X_*) \end{pmatrix} \sim N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} K(X, X) + \sigma_n^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{pmatrix}\right) \quad (23)$$

Noisy observations posterior As before, we can form a predictive distribution using the Gaussian multivariate conditioning identity:

$$p(f(X_*)|X_*, X, Y) \sim N(K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1}Y, K(X_*, X_*) - K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1}K(X, X_*)) \quad (24)$$

Substituting $k(X, X') = \phi(X)^T \Sigma_p \phi(X')$ into here gives us the exact same result as 19.

Our variance is independent of the targets y and only depends on our inputs X and X_* . Our variance is two terms: our prior covariance $K(X_*, X_*)$, a term representing the information the observations give us about the function. As before, we can compute the predictive distribution of y_* by adding the noise term $\sigma_n^2 I$ to the variance.

2.2.4 Marginal likelihood

Even though we are working within the Bayesian paradigm, certain practical subroutines e.g. kernel hyperparameter optimisation algorithms need a likelihood to maximise. The marginal likelihood $p(Y|X)$ is a measure of how well our GP fits the data:

$$p(Y|X) = \int p(Y|f, X)p(f|X)df \quad (25)$$

$p(f|X)$ is our familiar prior distribution over weights $\sim N(0, K)$ which we use here to represent the complexity of f . $p(y|f, X)$ is the likelihood of our observations given $p(y|f, X) \sim N(f, \sigma_n^2 I)$, and represents how well f maps X to y .

Our weight's mean will always be the same under our prior, but our $K(X, X')$ tells us how "wiggly" our function is. The closer our $p(f|X)$ distribution is to the true complexity of the function, the higher our marginal likelihood. For example, for SE if our data is close together, then $|X - X'|$ becomes small and our covariance $k(X, X')$ on our function distribution prior is large. Therefore, we get a high variety of functions and a higher probability of sampling a more complex function.

We can express $p(Y|X)$ as a Gaussian integral over the joint distribution of f and Y ($p(Y, f|X)$), and marginalise out f to get this PDF:

$$\log p(Y|X) = -\frac{1}{2}Y^T(K(X, X) + \sigma_n^2 I)^{-1}Y - \frac{1}{2}\log|K(X, X) + \sigma_n^2 I| - \frac{n}{2}\log(2\pi) \quad (26)$$

Alternatively, from 1 we know that y is Gaussian. Since both y and f are Gaussian, we can simply add their means and variances:

$$p(Y|X) = N(0, K(X, X) + \sigma_n^2 I)$$

We can plug these mean and variances into the Gaussian PDF 5 to get 26. Assembling the likelihood is necessary for training to select hyperparameters, but the $K(X, X) + \sigma_n^2 I$ inversion costs $O(n^3)$ to compute.

2.2.5 Algorithm for Gaussian processes

Here is an algorithm for computing the predictive distribution for predictions and log-likelihood for training that uses Cholesky decomposition 48 to address the matrix inversion requirement.

1. Take in inputs X , outputs y , covariance function k , noise level σ_n^2 , and test input X_*
2. $L = \text{cholesky}(K(X, X) + \sigma_n^2 I)$
 - Compute the Cholesky factor 48 for the $[K(X, X) + \sigma_n^2 I]$ matrix that needs to be inverted for training and inference
3. $\alpha = L^T \backslash (L \backslash y)$
 - Find $[K(X, X) + \sigma_n^2 I]^{-1}y$ using Cholesky decomposition 48, which is equivalent to solving the linear system $L^T L \cdot \alpha = y$.
4. $\mu = K(X_*, X)^T \cdot \alpha$
 - Compute the mean
5. $v = L \backslash K(X_*, X)^T$
 - Compute $v = L^{-1}K(X_*, X)$
6. $\text{var} = K(X_*, X_*) - v^T v$
 - Plug v into the variance restated in terms of the Cholesky factors, where $v^T v = (L^{-1}K(X_*, X))^T (L^{-1}K(X_*, X)) = K(X_*, X)^T [K(X, X) + \sigma_n^2 I]^{-1} K(X_*, X)$
7. $\log p(Y|X) = -\frac{1}{2}y^T \cdot \alpha - \frac{1}{2}\log|K(X, X) + \sigma_n^2 I| - \frac{n}{2}\log(2\pi)$
 - Compute the log marginal likelihood
8. Return the mean μ , variance var , and log marginal likelihood

\backslash denotes a linear system solver, i.e. $L \backslash y$ solves the linear system $Lx = y$ for x .

Exploring Covariance Functions

3.1 Characteristics of covariance functions and matrices

3.1.1 Gram matrices

The inner product representations of the covariance for our Bayesian model in function-space 20, weight-space 19, and valid covariance matrices in general can be represented as Gram matrices. [21] A Gram matrix is the matrix of all pairwise inner products of some set of vectors in an inner-product space. Formally, let V be an inner-product space with an inner product $\langle \cdot, \cdot \rangle$, and let the vectors in our inner product space be $v_1, v_2, \dots, v_n \in V$. The Gram matrix G is defined entry-wise by:

$$G_{ij} = \langle v_i, v_j \rangle$$

$i, j = 1, \dots, n$.

The Gram matrix has two properties of interest; symmetry and positive semidefiniteness:

$$\begin{aligned} G_{ij} &= G_{ji} \\ X^T G X &\geq 0 \end{aligned}$$

Symmetry is inherent from the inner product property since inner products are symmetric, i.e. $X_i^T X_j = X_j^T X_i$ and $\langle v_i, v_j \rangle = \langle v_j, v_i \rangle$. The Gram matrix is symmetric if the inner product space is real-valued, and Hermitian if it is complex-valued.

G satisfies positive semidefiniteness (PSD) if, for any coefficient vector $c = (c_1, \dots, c_n)^T$ to act as coefficients on the vectors v_i in our Gram matrix, the quadratic form $c^T G c$ is non-negative. Assembling the quadratic form:

$$c^T G c = \sum_{i=1}^n \sum_{j=1}^n c_i c_j \langle v_i, v_j \rangle$$

We can rewrite the double sum as the inner product of the coefficient vector applied to the constituent vectors with itself:

$$c^T G c = \left\langle \sum_{i=1}^n c_i v_i, \sum_{j=1}^n c_j v_j \right\rangle$$

Representing this self-multiplication as a squared norm:

$$c^T G c = \left\| \sum_{i=1}^n c_i v_i \right\|^2$$

Since norms are always ≥ 0 , the quadratic application of any coefficient vector c to the Gram matrix G must be non-negative.

Any proposed covariance matrix K needs to exhibit these properties to be a valid covariance matrix, and these existence of these properties have consequences on the composition of the covariance matrix and the GP as a whole. The symmetry property requires that $K(X, X') = K(X', X)$, whilst PSD requires the eigenvalues of K to be non-negative. The latter requirement guarantees the existence of the inverse and the positivity of the determinant of K , which are required for the GP to be well-defined and represent a distribution over functions respectively.

3.1.2 Eigenvalue and eigenfunctions of covariance matrices

Integral operators If we wanted to multiply a vector Φ by a matrix $K(X, X')$, we would use the following matrix-vector multiplication to produce a discrete vector $[K\phi]$:

$$[K\phi]_i = \sum_{j=1}^n K_{ij} \Phi_j$$

If we wanted to multiply a function $\phi(x)$ by our matrix $K(X, X')$, we could use an "integral operator" K to produce a new continuous function $[K\phi](x')$:

$$[K\phi](x') = \int K(x, x') \phi(x) dx$$

This specific usage of $K(X, X')$ is where covariance matrices are known as kernels, but the terms "kernel" and "covariance matrix" are used interchangeably outside of this context. For example, using the linear kernel $K(x, x') = x \cdot x'$, a function $\phi(x) = x^2$, and the limits of our domain $a = 0$ and $b = 1$:

$$[K\phi](x') = \int x \cdot x' x^2 dx = x' \int_0^1 x^3 dx = x' \left[\frac{x^4}{4} \right]_0^1 = x' \cdot \frac{1}{4}$$

$[K\phi](x')$ becomes x' scaled by the RHS integral - the first term depends on our choice of kernel, and the size of the scale depends on $\phi(x)$ and a, b .

We can refer to some vector Φ as the eigenvector and λ as the eigenvalue of K if it obeys this equation:

$$[K\Phi] = \lambda\Phi$$

Since our integral operator K is like an infinite-dimensional matrix whose entries are $k(x, x')$, we can refer to a function $\phi(x)$ as the eigenfunction and λ as the eigenvalue of our integral operator K if:

$$\int K(x, x')\phi(x)d\mu(x) = \lambda\phi(x')$$

where μ is some measure over the domain of x upon which the integration will occur thus defining the limits of integration a, b .

Mercer's theorem If our kernel satisfies the conditions of a Gram matrix and our measure μ is any compact subset $C \subset \mathbb{R}^n$ (e.g. the probability space $[0, 1]$), then it conforms to Mercer's theorem. Mercer's theorem states that these eigenvalues $[\lambda_i]_{i=1}^{\infty}$ are non-negative and decay to zero, and our kernel itself can be decomposed into a sum of eigenfunctions ϕ_i and eigenvalues λ_i :

$$K(X, X') = \sum_{i=1}^{\infty} \lambda_i \phi_i(X) \phi_i(X') \quad (27)$$

A kernel with infinite rank, or a covariance function that has infinite basis functions (e.g. SE), is called a nondegenerate kernel, else it is degenerate. For example, our N -dimensional linear kernel $K(X, X') = X \cdot X'$ results in a degenerate kernel with at most N non-zero eigenvalues.

3.1.3 Choosing the length scale and other hyperparameters

Our covariance functions have some hyperparameters that alter their behaviour based on the encountered data. For example, the full form of SE in one dimension contains some free parameters σ_f^2 , σ_n^2 , and l that are estimated via MLE:

$$k_y(x, x') = \sigma_f^2 \exp\left(-\frac{1}{2} \frac{|x - x'|^2}{l^2}\right) + \sigma_n^2 \delta_{x, x'}$$

σ_f^2 is the signal variance, which controls the overall scale of the function. σ_n^2 is the noise variance, which controls the amount of noise in the observations, and $\delta_{X, X'}$ is the Kronecker delta which represents our independence of noise assumption.

Analysing the length scale Although we estimate l via MLE in practice, we can obtain an analytical expression for l using a Taylor expansion to understand its role in the covariance function. Taylor expanding a stationary covariance function $k(X, X')$:

$$k(X, X') = k(X, X') + k'(X, X')(X - X') + \frac{1}{2}k''(X, X')(X - X')^2 + \dots$$

We choose a stationary covariance function for convenience, but this expansion can be applied to any covariance function. By symmetry, $k'(X, X') = 0$. Our second order Taylor expansion becomes:

$$k(X, X') \approx k(X, X') + [0] + \frac{1}{2}k''(X, X')$$

Defining l^2 :

$$l^2 = \frac{k(X, X')}{-k''(X, X')} \quad (28)$$

l is our length scale; it is the ratio between the covariance function $k(X, X')$ and its second derivative $k''(X, X')$, a measure of wiggleness. As our covariance function becomes less wiggly, our length scale increases in size. The length scale effectively measures "distance per wiggle" - a less wiggly covariance function needs a larger distance to produce the same wiggle as a more wiggly covariance function. We introduce the assumption that the chosen covariance function has a second derivative that is negative, i.e. $k''(X, X') < 0$, to guarantee $l^2 > 0$. This assumption is violated for certain kernels (i.e. the exponential kernel), so these violating kernels require a different notion of length scale.

Substituting l^2 into the above:

$$k(X, X') \approx k(X, X') \left(1 - \frac{(X - X')^2}{2l^2}\right)$$

l^2 represents the quadratic distance over which the quadratic distance term $(X - X')^2$ in the Taylor expansion becomes comparable in size to the original $k(X, X')$. For example, at $X - X' = l$, our second-order approximation is half the size of the actual covariance function:

$$k(X, X') \approx k(X, X') \left(1 - \frac{[l]^2}{2l^2}\right) = k(X, X') \frac{1}{2}$$

Our approximation will always be less than $k(X, X')$ thanks to the lost Taylor remainder, but there are two factors that affect its accuracy. The first is the difference between the two points in input space being evaluated - as the gap $(X - X')^2$ grows, the second-order approximation becomes less accurate. Secondly, the always-positive squared length scale l^2 - as $l \rightarrow \infty$ and our function gets less wiggly, the second-order approximation approaches $k(X, X')$.

Length scale and upcrossings The effect of this measure of wiggleness l is more obvious when contextualised in terms of the number of "upcrossings" our covariance function makes at a given level u . A function performs an upcrossing at u when $u = f(x)$ and $dy/dx > 0$. For example, with $u = 2$ and $y = x^2$, there exists one upcrossing at $(2, \sqrt{2})$ where $dy/dx = 4$, and a downcrossing at $(2, -\sqrt{2})$ where $dy/dx = -4$.

Given a zero-mean GP $p(f(X)) = N(0, k)$ produced by a given covariance function $k(X, X')$, the expected number of upcrossings $\mathbb{E}[N_u]$ at a given level $0 < u < 1$ for a typical sample function drawn from this GP $f(X) \sim N(0, k)$ is: [21]

$$\mathbb{E}[N_u] = \frac{1}{2\pi} \sqrt{\frac{-k''(0)}{k(0)}} \exp\left(-\frac{u^2}{2k(0)}\right) \quad (29)$$

Substituting our expression for l^2 from our second-order Taylor expansion 28:

$$\mathbb{E}[N_u] = \frac{1}{2\pi} \sqrt{[l^2]} \exp\left(-\frac{u^2}{2k(0)}\right)$$

Analysing $u = 0$ reduces the RHS exponential term to 1:

$$\mathbb{E}[N_0] = \frac{1}{2\pi} l[1]$$

Moving the inverse to the LHS:

$$\frac{1}{\mathbb{E}[N_0]} = 2\pi l$$

This expression suggests that the number of upcrossings and the length scale l are inversely proportional - the wigglier the covariance function, the smaller its length scale is and the more upcrossings the resulting functions drawn from the GP will have.

3.1.4 Mean square continuity and differentiability

Our length-scale based understanding of the smoothness of a sample function from our GP incorporates information about the second derivative of our covariance function. Other measures of smoothness include differentiability and continuity - more differentiable function implies that the function contains higher order polynomials which makes it smoother, and a continuous function avoids any reductions in smoothness produced by discontinuities.

Mean square space Because our GP is a continuous distribution of functions, we have infinitely many possible sample functions we can draw and determining if each sample function is continuous or differentiable is impossible. When determining smoothness by length scale, our upcrossing theorem 29 required analysing our covariance function alone.

We can achieve similar results by placing our GP in mean-square (MS) space. MS space is the vector space where vectors are random variables like our GP. In MS space, continuity and differentiability become statements about the mean-square distance:

$$\|f(x) - f(x_*)\|_2 = \sqrt{\mathbb{E}[(f(x) - f(x_*))^2]}$$

By using MS definitions of continuity and differentiability, we can derive conditions for almost-surely continuity and differentiability of any sample function based on our covariance function.

MS continuity Let x be an infinite-length vector of inputs x_1, x_2, \dots whose values linearly approach some stable fixed point x_* as the sequence progresses. Formally:

$$\lim_{k \rightarrow \infty} |x_k - x_*| = 0$$

A regular function f is continuous at x_* if three conditions are met. x_* must exist in the domain of f , e.g. $f(X) = 1/x$ is not continuous at $f(0)$ because 0 is not in the domain of f . There must be a single limit the function approaches.

$$f(X) = \begin{cases} 0 & \text{if } X < 0 \\ 1 & \text{if } X > 0 \\ 2 & \text{if } X = 0 \end{cases}$$

In this example, at $f(0)$ x is not continuous at $x = 0$ since f approaches both 0 and 1. This single limit needs to be the same as the function evaluation at the limit. If we removed the 0 if $X < 0$ case to produce a single limit, it would still not be continuous at $x = 0$ since the limit would be 1 but $f(0) = 2$. Formally summarising these three ideas:

$$\lim_{x \rightarrow x_*} |f(x) - f(x_*)| = 0$$

A Gaussian process f is MS continuous at x_* if the expected function evaluations $E[f(x)]$ quadratically approach $f(x_*)$ as $x \rightarrow x_*$. [21] Formally:

$$\lim_{x \rightarrow x_*} \mathbb{E}[(f(x) - f(x_*))^2] = 0$$

Thanks to the quadratic term, we can expand our error term directly into kernel terms. The covariance function k is defined as the expected value of the product of two function evaluations:

$$\begin{aligned} \mathbb{E}[(f(x) - f(x_*))^2] &= \mathbb{E}[f(x)^2] + \mathbb{E}[f(x_*)^2] - 2\mathbb{E}[f(x)f(x_*)] \\ &= k(x, x) + k(x_*, x_*) - 2k(x, x_*) \end{aligned}$$

Using this expansion in our definition of MS continuity:

$$\lim_{x \rightarrow x_*} |k(x, x) - 2k(x, x_*) + k(x_*, x_*)| = 0 \quad (30)$$

Because x_* is given, $k(x_*, x_*)$ is a constant. This limit requires that both $k(x, x)$ and $k(x, x_*) \rightarrow k(x_*, x_*)$ as $x \rightarrow x_*$, which is the very definition of continuity of k at x_* - they guarantee that x_* is in the domain of k , that it has a single limit, and this limit of $k(x, x_*)$ is $k(x_*, x_*)$ as $x \rightarrow x_*$. Therefore, a Gaussian process is MS continuous at x_* if and only if the covariance function k is continuous at x_* .

MS differentiability f is MS differentiable at x_* in the i th direction of our input vector X if: [21]

$$\lim_{h \rightarrow 0} \mathbb{E} \left[\left| \frac{(f(x_* + he_i) - f(x_*))}{h} - \frac{\partial f}{\partial x_{*i}}(x_*) \right|^2 \right] = 0$$

$e_i = (0_0, \dots, 0_{i-1}, 1_i, 0_{i+1}, \dots, 0_n)$ is a unit vector that constrains the change in x to only the i th dimension. We assume $m(x_*) = 0$ for simplicity, but a non-zero mean does not change the structure of this proof and leads to the same result.

Expanding and creating simplifying A and B terms to represent the limit and the proposed derivative respectively:

$$\begin{aligned} \lim_{h \rightarrow 0} \mathbb{E}[(A_h - B)^2] &= \mathbb{E}[A_h^2] + \mathbb{E}[B^2] - 2\mathbb{E}[A_h B] = 0 \\ A_h &= \frac{f(x_* + he_i) - f(x_*)}{h} \\ B &= \frac{\partial f}{\partial x_{*i}}(x_*) \end{aligned}$$

Similar to MS continuity, we expand A_h^2 into kernel terms:

$$\mathbb{E}[A_h^2] = \frac{1}{h^2} (k(x_* + he_i, x_* + he_i) + k(x_*, x_*) - 2k(x_* + he_i, x_*)) \quad (31)$$

$k(x_*, x_*)$ is a constant, but the other terms vary with h . We can use a second-order Taylor series to approximate these terms if k is twice continuously differentiable at x_* . Starting with the univariate $k(x_* + he_i, x_*)$:

$$\begin{aligned} k(x_* + he_i, x_*) &= \\ & k(x_*, x_*) \\ & + h[\partial_{u_i} k](x_*, x_*) \\ & + \frac{h^2}{2} [\partial_{u_i u_i}^2 k](x_*, x_*) \\ & + O(h^3) \end{aligned} \quad (32)$$

u_i and v_i denote the first $x_* + he_i$ and second x_* arguments respectively in the i th direction. O represents the remainder of the Taylor series. Since we are using a second-order Taylor series, the remainder is of order h^3 and higher.

Then, the multivariate $k(x_* + he_i, x_* + he_i)$:

$$\begin{aligned} k(x_* + he_i, x_* + he_i) &= \\ & k(x_*, x_*) \\ & + h[\partial_{u_i} k + \partial_{v_i} k](x_*, x_*) \\ & + \frac{h^2}{2} [\partial_{u_i u_i}^2 k + \partial_{u_i v_i}^2 k + \partial_{v_i v_i}^2 k](x_*, x_*) \\ & + O(h^3) \end{aligned} \quad (33)$$

Substituting these approximations into 31:

$$\begin{aligned} \mathbb{E}[A_h^2] &= \frac{1}{h^2} \\ & k(x_*, x_*) + k(x_*, x_*) - 2k(x_*, x_*) \\ & + h[\partial_{u_i} k + \partial_{v_i} k](x_*, x_*) - 2h[\partial_{u_i} k](x_*, x_*) \\ & + \frac{h^2}{2} [\partial_{u_i u_i}^2 k + \partial_{u_i v_i}^2 k + \partial_{v_i v_i}^2 k](x_*, x_*) - 2\frac{h^2}{2} [\partial_{u_i u_i}^2 k](x_*, x_*) \\ & + O(h^3) \end{aligned}$$

The constant $k(x_*, x_*)$ terms cancel and the $O(h^3)$ remainders combine but remain the same order.

We can combine the linear partial derivatives to form a single function $[\partial_{u_i} k + \partial_{v_i} k - 2\partial_{v_i} k]$. Thanks to the symmetry of the covariance matrix inherited from the Gram matrix, $k(u, v) = k(v, u)$ and $\partial_{u_i} k = \partial_{v_i} k$ and substituting in either into our linear term cancels it completely.

Similarly, we can combine the quadratic derivatives:

$$E[A^2] = \frac{1}{h^2} \frac{h^2}{2} [\partial_{u_i u_i}^2 k + \partial_{u_i u_i}^2 k + \partial_{v_i v_i}^2 k](x_*, x_*) - h^2 [\partial_{u_i v_i}^2 k](x_*, x_*)$$

Cancelling $\frac{1}{h^2}$ and $\frac{h^2}{2}$ and simplifying:

$$\begin{aligned} &= [\partial_{u_i v_i}^2 k + \partial_{u_i u_i}^2 k + \partial_{v_i v_i}^2 k - \partial_{u_i v_i}^2 k](x_*, x_*) \\ &= [\partial_{u_i u_i}^2 k + \partial_{u_i v_i}^2 k - \partial_{v_i v_i}^2 k](x_*, x_*) \end{aligned}$$

Similarly to the linear terms, using the symmetry of k to cancel $\partial_{u_i u_i}^2$ and $\partial_{v_i v_i}^2$ yields the final expression for $\mathbb{E}[A_h^2]$:

$$\mathbb{E}[A_h^2] = \partial_{u_i v_i}^2 k(x_*, x_*) + O(h^3) \quad (34)$$

Because B is Gaussian, its square is simply its variance:

$$\begin{aligned} \mathbb{E}[B^2] &= \text{Var} \left(\frac{\partial f}{\partial x_{*i}}(x_*) \right) \\ &= \text{Cov} \left(\frac{\partial f}{\partial x_{*i}}(x_*), \frac{\partial f}{\partial x_{*i}}(x_*) \right) \end{aligned}$$

We can substitute our regular limit definition of differentiability outside of MS space, with different small increments h and h' :

$$= \text{Cov} \left(\frac{f(x_* + h e_i) - f(x_*)}{h}, \frac{f(x_* + h' e_i) - f(x_*)}{h'} \right)$$

The h and h' denominators merely scale the covariance, so we can take them out of the covariance expression:

$$= \frac{1}{hh'} \text{Cov}(f(x_* + h e_i) - f(x_*), f(x_* + h' e_i) - f(x_*))$$

Using additivity of covariance:

$$\mathbb{E}[B^2] = \frac{1}{hh'} (k(x_* + h e_i, x_* + h' e_i) - k(x_*, x_* + h' e_i) - k(x_* + h e_i, x_*) + k(x_*, x_*))$$

We already have $k(x_* + h e_i, x_*)$ from 32 directly, and $k(x_*, x_* + h e_i)$ by symmetry and substituting h' for h . Our multivariate Taylor series $k(x_* + h e_i, x_* + h' e_i)$ looks similar to 33:

$$\begin{aligned} k(x_* + h e_i, x_* + h' e_i) &= \\ & k(x_*, x_*) \\ & + h[\delta_{u_i} k](x_*, x_*) + h'[\delta_{v_i} k](x_*, x_*) \\ & + \frac{h^2}{2} [\partial_{u_i u_i}^2 k](x_*, x_*) \\ & + hh' [\partial_{u_i v_i}^2 k](x_*, x_*) \\ & + \frac{h'^2}{2} [\partial_{v_i v_i}^2 k](x_*, x_*) \\ & + O(\|(h, h')\|^3) \end{aligned}$$

Substituting these results into $\mathbb{E}[B^2]$:

$$\begin{aligned} \mathbb{E}[B^2] &= \frac{1}{hh'} \\ & k(x_*, x_*) \\ & + k(x_*, x_*) + k(x_*, x_*) - k(x_*, x_*) \\ & + h[\delta_{u_i} k](x_*, x_*) - h[\delta_{u_i} k](x_*, x_*) \\ & + h'[\delta_{v_i} k](x_*, x_*) - h'[\delta_{v_i} k](x_*, x_*) \\ & + \frac{h^2}{2} [\partial_{u_i u_i}^2 k](x_*, x_*) - \frac{h^2}{2} [\partial_{u_i u_i}^2 k](x_*, x_*) \\ & + hh' [\partial_{u_i v_i}^2 k](x_*, x_*) \\ & + \frac{h'^2}{2} [\partial_{v_i v_i}^2 k](x_*, x_*) - \frac{h'^2}{2} [\partial_{v_i v_i}^2 k](x_*, x_*) \\ & + O(h^3) + O(h'^3) + O(\|(h, h')\|^3) \end{aligned}$$

All constant, linear h and h' , and h^2 and h'^2 terms cancel, leaving the mixed term and the remainders. $\frac{1}{hh'}$ cancel and the remainders can be combined:

$$\mathbb{E}[B^2] = \partial_{u_i v_i}^2 k(x_*, x_*) + O(h^2) \quad (35)$$

For $\mathbb{E}[A_h B]$, assuming zero-mean:

$$\mathbb{E}[A_h B] = \text{Cov} \left(\frac{f(x_* + h e_i) - f(x_*)}{h}, \frac{\partial f}{\partial x_{*i}}(x_*) \right)$$

Similarly to B , we can remove the scaling factor $\frac{1}{h}$:

$$= \frac{1}{h} \text{Cov} \left(f(x_* + h e_i) - f(x_*), \frac{\partial f}{\partial x_{*i}}(x_*) \right)$$

By additivity of covariance:

$$= \frac{1}{h} \left[\text{Cov}(f(x_* + h e_i), \frac{\partial f}{\partial x_{*i}}(x_*)) - \text{Cov}(f(x_*), \frac{\partial f}{\partial x_{*i}}(x_*)) \right]$$

A fundamental identity for Gaussian fields [21] is:

$$\text{Cov}(f(u), \delta_{u_i} f(v)) = \delta_{u_i} k(u, v)$$

Applying it to both terms:

$$\text{Cov}(f(x_* + h e_i), \frac{\partial f}{\partial x_{*i}}(x_*)) = \delta_{u_i} k(x_* + h e_i, x_*)$$

$$\text{Cov}(f(x_*), \frac{\partial f}{\partial x_{*i}}(x_*)) = \delta_{v_i} k(x_*, x_*)$$

Taylor expanding our first term:

$$\begin{aligned} \delta_{u_i} k(x_* + h e_i, x_*) &= \\ &\delta_{u_i} k(x_*, x_*) \\ &+ h \delta_{u_i v_i}^2 k(x_*, x_*) \\ &+ \frac{h^2}{2} \delta_{u_i u_i u_i}^3 k(x_*, x_*) \\ &+ O(h^3) \end{aligned}$$

We can take advantage of symmetry of k to cancel out the constant term using the second term in $\mathbb{E}[A_h B]$ - when $u = v$, $\delta_{v_i} k(x_*, x_*) = \delta_{u_i} k(x_*, x_*)$. Substituting this into $\mathbb{E}[A_h B]$:

$$\begin{aligned} \mathbb{E}[A_h B] &= \frac{1}{h} \\ &h \delta_{u_i v_i}^2 k(x_*, x_*) + \frac{h^2}{2} \delta_{u_i u_i v_i}^3 k(x_*, x_*) \\ &+ O(h^3) \end{aligned}$$

Dividing by h reduces the order of the remainder. Simplifying for a final expression:

$$\mathbb{E}[A_h B] = \delta_{u_i v_i}^2 k(x_*, x_*) + \frac{h}{2} \delta_{u_i u_i u_i}^3 k(x_*, x_*) + O(h^2) \quad (36)$$

Substituting 34, 35, and 36 into our original limit definition of MS differentiability:

$$\lim_{h \rightarrow 0} \delta_{u_i v_i}^2 k(x_*, x_*) + O(h^3) + \delta_{u_i v_i}^2 k(x_*, x_*) + O(h^2) - 2 \left(\delta_{u_i v_i}^2 k(x_*, x_*) + \frac{h}{2} \delta_{u_i u_i u_i}^3 k(x_*, x_*) + O(h^2) \right) = 0$$

Redefining $A = \delta_{u_i v_i}^2 k(x_*, x_*)$ and $B = \frac{h}{2} \delta_{u_i u_i u_i}^3 k(x_*, x_*)$ for brevity:

$$\lim_{h \rightarrow 0} A + O(h^3) + A + O(h^2) - 2(A + \frac{h}{2} B + O(h^2)) = 0$$

Expanding the last term:

$$\lim_{h \rightarrow 0} A + O(h^3) + A + O(h^2) - 2A - hB - 2O(h^2) = 0$$

All A terms cancel. Combining and reducing the remainders to the lowest order:

$$\lim_{h \rightarrow 0} hB + O(h) = 0$$

We can absorb hB into $O(h)$ since it is a linear term in h :

$$\lim_{h \rightarrow 0} O(h) = 0 \quad (37)$$

$O(h)$ approaches 0 as $h \rightarrow 0$, so our limit definition of MS differentiability is met using a second-order Taylor expansion. We can extend this to any k -th derivative in MS space by using a $2k$ -th order Taylor expansion of the covariance functions, which requires the covariance function to be $2k$ -times continuously differentiable at x_* .

3.2 Stationary covariance functions

3.2.1 Stationarity and isotropicism

A stationary covariance function [21] $k(X - X')$ is some function of $X - X'$, and is invariant to the exact locations of X and X' . An isotropic covariance function $k(|X - X'|)$ is a function of $|X - X'|$, and is invariant to the direction of $X - X'$. For example, SE [eq:se] is both stationary and isotropic because it is a function of $|X - X'|$.

3.2.2 Spectral density

So far, length-scale analysis and MS continuity and differentiability have been used to understand the smoothness of our GP. For stationary covariance functions specifically, we can use the notion of spectral density to understand how evenly spread out variance is within our GP, incorporate length scale into this notion, and use it to derive conditions for MS continuity and differentiability.

Bochner's theorem [9] states that the covariance function of any stationary process, like a stationary GP, can be represented as the Fourier transform of some positive finite measure μ :

$$k(x, x') = \int_{-\infty}^{\infty} e^{2\pi i \omega(x-x')} d\mu(\omega) \quad (38)$$

The Fourier transform decomposes our stationary covariance function into a sum of sines and cosines that occur at some frequency, written in a compact form as a complex exponential:

$$e^{-2\pi i \omega(x-x')} = \cos(2\pi \omega(x-x')) - i \sin(2\pi \omega(x-x')) \quad (39)$$

No such Fourier transformation exists for non-stationary covariance functions, as non-stationary covariance functions depend on variables outside of $X - X'$ and cannot be expressed as a sum of sines and cosines alone.

The "spectral density" $S(\omega)$ is a complex number representing how much total variance is allocated at some frequency ω .

$$S(\omega) = \int_{-\infty}^{\infty} k(x, x') e^{2\pi i \omega(x-x')} d(x-x') \quad (40)$$

To remain integratable, our spectral density function needs to approach 0 as $\omega \rightarrow \infty$. The faster it approaches 0 as $\omega \rightarrow \infty$, the less power it contains at high frequencies, and the smoother our resultant GP will be. High variances allocated at low frequencies correspond to slow-varying, large-scale wiggles, whereas high variances at higher frequencies correspond to rapid oscillations. Since the covariance function itself is included in this definition, any length scale l in the covariance function will also affect the spectral density function.

The Weiner-Khinchin theorem [21] describes how to derive the spectral density. It states that our covariance function and its spectral density are Fourier duals of each other, meaning that we obtain the spectral density by taking the Fourier transform of the covariance function 40. Furthermore, this theorem implies that we can recover the covariance function from the spectral density by taking the Fourier transform of the spectral density:

$$k(x, x') = \int_{-\infty}^{\infty} S(\omega) e^{-2\pi i \omega(x-x')} d\omega \quad (41)$$

Spectral density and MS differentiability We can formalise the relationship between allocations of mass in $S(\omega)$ and smoothness using "moments" of the spectral density, or an average of the total variance of the GP weighted by the frequency ω raised to some power m . The $2m$ -th ω -th moment of the spectral density is:

$$M_{2m} = \int_{-\infty}^{\infty} |\omega|^{2m} S(\omega) d\omega$$

Ritter et. al. [22] found that if this M_{2m} moment exists and is finite, our covariance function is $2m$ differentiable, thus sample functions drawn from our GP are almost-surely m -times MS differentiable.

3.2.3 GPs from stationary covariance functions in MS space

Putting the stationary kernel $k(x, x') = k(x - x')$ into 30:

$$\lim_{x \rightarrow x_*} |k(x - x) - 2k(x - x_*) + k(x_* - x_*)| = 0$$

Simplifying the terms inside the kernels and combining like terms:

$$\lim_{x \rightarrow x_*} 2|k(0) - k(x - x_*)| = 0$$

2 is a constant factor and can be ignored. Because $|k(0) - k(x - x_*)|$ is invariant to direction, we can swap them to align with the definition of continuity at 0:

$$\lim_{x \rightarrow x_*} |k(x_* - x) - k(0)| = 0$$

Thus, a stationary covariance function is MS continuous at x_* if and only if it is continuous at $x_* = 0$. Similarly, it can be shown [21] that stationary covariance functions are MS differentiable at x_* if and only if they are MS differentiable at $x_* = 0$.

3.3 Stationary covariance functions

3.3.1 Squared exponential (SE)

Here is the noiseless form of the already introduced SE: [21]

$$k(X, X*) = \exp\left(-\frac{|X - X'|^2}{2l^2}\right) \quad (42)$$

Using 40, we can derive the spectral density of SE: [21]

$$S(\omega) = (2\pi l^2)^{D/2} \exp(-2\pi^2 l^2 \omega^2) \quad (43)$$

The rate of frequency decline is controlled by the behaviour of the RHS exponential. A larger frequency ω increases the size of the negative term inside the exponential, which exponentially decreases the size of the variance allocated at that frequency. The speed of this exponential approach to zero is controlled by the length scale l - a larger l means that the exponential approaches zero more quickly, and thus less variance is allocated at higher frequencies. In terms of behaviour in MS space, SE's spectral density has infinite moments, that the covariance function is infinitely differentiable, and thus the GP sample functions are infinitely MS differentiable.

These extremely smooth sample functions are not useful for approximating real-world data-generating functions that tend to be much rougher. The three major attempts to generalise the SE to vary the level of smoothness are the rational quadratic, γ -exponential, and Matern classes of covariance functions.

From feature space to SE We can derive SE by expanding X into our feature space ϕ defined by Gaussian-shaped basis functions centred densely in X . Defining this basis function:

$$\phi_c(x) = \exp\left(-\frac{|x - c|^2}{2l^2}\right)$$

c the centre of our basis functions. With our familiar isotropic Gaussian prior on the weights $W \sim \mathcal{N}(0, \sigma_p^2 I)$, we get our familiar covariance function in weight-space:

$$k(x, x') = \sigma_p^2 \sum_{c=1}^N \phi_c(x) \phi_c(x')$$

N represents the number of these basis functions. If $N = \infty$ with centres everywhere between some interval c_{min} and c_{max} , we can simply integrate over the interval:

$$\lim_{N \rightarrow \infty} \sigma_p^2 \sum_{c=1}^N \phi_c(x) \phi_c(x') = \sigma_p^2 \int_{c_{min}}^{c_{max}} \phi_c(x) \phi_c(x') dc$$

Plugging in our basis function:

$$k(x, x') = \sigma_p^2 \int_{c_{min}}^{c_{max}} \exp\left(-\frac{|x - c|^2}{2l^2}\right) \exp\left(-\frac{|x' - c|^2}{2l^2}\right) dc$$

Combining the exponentials:

$$= \sigma_p^2 \int_{c_{min}}^{c_{max}} \exp\left(-\frac{|x - c|^2 + |x' - c|^2}{2l^2}\right) dc$$

Expanding the squared terms, rearranging the fraction and simplifying:

$$= \sigma_p^2 \int_{c_{min}}^{c_{max}} \exp\left(-\frac{1}{l^2} \left[c^2 - c(x + x') + \frac{x^2 + x'^2}{2} \right]\right) dc$$

We can complete the square on the c terms to get a product of two exponentials:

$$= \sigma_p^2 \int_{c_{min}}^{c_{max}} \exp\left(-\frac{1}{l^2} \left[\left(c - \frac{x + x'}{2}\right)^2 - \frac{(x + x')^2}{4} \right]\right) dc$$

Our second term in the exponential does not vary with c and can be safely factored out of the integral:

$$= \sigma_p^2 \exp\left(\frac{(x + x')^2}{4l^2}\right) \int_{c_{min}}^{c_{max}} \exp\left(-\frac{1}{l^2} \left(c - \frac{x + x'}{2}\right)^2\right) dc$$

If we let our c_{min} and c_{max} approach infinity, we can use the standard Gaussian integral:

$$\int_{-\infty}^{\infty} \exp\left(-\frac{1}{l^2} \left(c - \frac{x + x'}{2}\right)^2\right) dc = l\sqrt{\pi}$$

Substituting this in:

$$k(x, x') = \sigma_p^2 l \sqrt{\pi} \exp \left(-\frac{(x - x')^2}{4l^2} \right)$$

We can absorb the $l\sqrt{\pi}$ into the σ_p^2 term to produce our familiar SE with a $\sqrt{2}$ longer length scale:


$$k(x, x') = \sigma_p^2 \exp \left(-\frac{(x - x')^2}{2(\sqrt{2}l)^2} \right) \quad (44)$$

./img/covariance-functions/Gaussian_Smooth_mean.png

Figure 1: Plot of a GP using SE applied to a smooth sin wave $y = \sin(2 * \pi * x) + \epsilon$, $\epsilon \sim N(0, 0.05)$, fitted using the R package GauPro [5].

The noiseless sin-wave is in blue and some sample datapoints taken from this function with noise are in black. The black line represents the expected function from the Gaussian process. The green line represents the 95% confidence interval around the predictive distribution without the σ_n^2 term, representing the uncertainty surrounding predictions of the noise-free mean function $f(X)$. The red line represents the 95% confidence interval with σ_n^2 , representing the uncertainty surrounding predictions of the noisy observations y .

For this extremely smooth function with low noise, SE nearly perfectly captures the original data-generating function. The variances fall as they approach datapoints and rise as they move away from them. Defining our point of interest as x_* , as x_* approaches a training datapoint x , $x - x_*$ approaches zero and the value of our SE kernel approaches one, and the predictive distribution at x_* collapses close to our training data $N(Y \sigma_n^2 I, \sigma_n^2 I)$. This is known as the interpolation property [21] and is valid for all stationary covariance functions we cover.

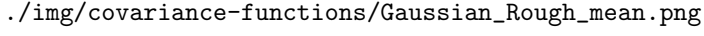


`./img/covariance-functions/Gaussian_Smooth_draws.png`

Figure 2: Plots of some sample functions drawn from a GP trained on the smooth dataset using SE. The blue and black lines are, as before, the noiseless data-generating function and the mean function drawn from the GP respectively. The green lines are 20 functions drawn from the GP - if we superimposed these draws onto the previous GP of means and variances, we would expect approximately 95% of these function draws to be contained within the red variance lines.

Although we can technically draw infinite sample functions from any GP, these function draws all look very similar. This is because very few functions exist that satisfy SE's infinite smoothness and can exist entirely within this variance envelope, and they inevitably share many properties.

These constraints also affect the sample functions' ability to obey the interpolation property and require a trade-off between smoothness and fidelity. For example, there are no sample functions that pass through the fourth point at $(0.21, 0.86)$ because infinitely smooth functions are too rigid to pass through this point and the next two points at $(0.29, 1)$ and $(0.36, 0.81)$. This violation of the interpolation property is a consequence of the noise term $\epsilon \sim N(0, 0.05)$ in the data-generating function. If the noise term was zero and the data-generating function was infinitely differentiable, like this one, then the GP and all sample paths would pass through all datapoints.

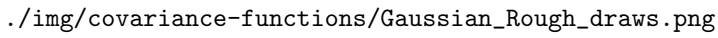


./img/covariance-functions/Gaussian_Rough_mean.png

Figure 3: Plot of the expected function draw and variances as before, using SE fitted to a rougher function:

$$\begin{cases} x < 0.3 & x \\ 0.3 \leq x < 0.7 & 1 - x \\ x \geq 0.7 & x - 0.5 \end{cases}$$

SE performed worse on this rougher data as its extreme smoothness is unable to adapt well. The length scale has decreased (0.55, compared to 0.76 using smooth data), which reduces the rate of variance decay with frequency 43 and increases the oscillatory behaviour of the GP sample functions whilst still producing infinitely smooth functions.



./img/covariance-functions/Gaussian_Rough_draws.png

Figure 4: Plot of some sample functions drawn from a GP as before, trained on the rough dataset using SE.

These functions are much more varied than those drawn from the smooth dataset, as SE is unable to adapt to the rougher data. Some draws have become more oscillatory due to the decrease in length scale, but all draws are still infinitely smooth.

3.3.2 Rational quadratic (RQ)

RQ controls smoothness by using a mixture of different length scales to capture multiple trends in the data at different frequencies.

$$k(X, X') = \left(1 + \frac{|X - X'|^2}{2\alpha l^2}\right)^{-\alpha}$$

We can derive this form by creating an infinite sum of different instances of SE with a vector of different length-scales l and some weights w_i attached to each instance:

$$k(X, X') = \sum_{i=1}^{\infty} w_i \exp\left(-\frac{|X - X'|^2}{2l_i^2}\right)$$

This is a valid covariance function long as the sum remains less than ∞ to maintain PSD. We can turn this into a continuous representation if our l is very dense, where $p(l)$ is the PDF for l :

$$k(X, X') = \int_0^{\infty} p(l) \exp\left(-\frac{|X - X'|^2}{2l^2}\right) dl$$

Defining $\tau = l^{-2}$, $l = \tau^{-1/2}$. Putting this into SE:

$$k_{SE}(X, X') = \exp\left(-\frac{|X - X'|^2}{2[\tau^{-1}]}\right) = \exp\left(-\frac{1}{2}\tau|X - X'|^2\right)$$

Before putting this back into the above integral which is in terms of l , we need to account for the change of variables from l to τ . The Jacobian factor of this transformation is:

$$|\partial_{\tau} l| = \frac{1}{2}\tau^{-3/2}d\tau$$

Defining the transformation $l(x) = \frac{1}{\sqrt{x}}$, putting these into the integral changes the PDF from $p(l)$ to $p(l(\tau))|\partial_{\tau} l|$:

$$k(X, X') = \int_0^{\infty} p(l(\tau)) \frac{1}{2}\tau^{-3/2} \exp\left(-\frac{1}{2}\tau|X - X'|^2\right) d\tau$$

We need an expression for the PDF $p(\tau)$ that integrates to 1 and is PSD. A common choice is the Gamma distribution with shape α and scale β parameters: $\tau \sim \Gamma(\alpha, \frac{\beta}{\alpha})$

$$p(\tau) \propto \tau^{\alpha-1} \exp\left(-\frac{\alpha}{\beta}\tau\right)$$

Putting this into the integral:

$$k(X, X') \propto \int_0^{\infty} \left[\tau^{\alpha-1} \exp\left(-\frac{\alpha}{\beta}\tau\right)\right] \frac{1}{2}\tau^{-3/2} \exp\left(-\frac{1}{2}\tau|X - X'|^2\right) d\tau$$

$\frac{1}{2}\tau^{-3/2}$ from dl and the new $\tau^{\alpha-1}$ combine, and the new exponential joins the existing one:

$$k(X, X') \propto \int_0^{\infty} \frac{1}{2}\tau^{\alpha-\frac{5}{2}} \exp\left[-\tau\left(-\frac{\alpha}{\beta} + \frac{|X - X'|^2}{2}\right)\right] d\tau$$

We can use the Gamma integral here, but need some rearranging. Absorbing $\frac{1}{2}$ into the proportionality, setting $a = \alpha - \frac{3}{2}$ and factoring τ out of the exponent for $c = \frac{\alpha}{\beta}\tau + \frac{|X - X'|^2}{2}$:

$$k(X, X') \propto \int_0^{\infty} \tau^{a-1} \exp(-\tau c) d\tau$$

Applying the Gamma integral and absorbing $\Gamma(a)$ into proportionality:

$$k(X, X') \propto c^{-a} \Gamma(a) \propto \left[\frac{\alpha}{\beta} + \frac{|X - X'|^2}{2}\right]^{-([\alpha - \frac{3}{2}])}$$

Now we need to decide on the parameter values of our Gamma distribution. Since the Gamma distribution strictly takes in one Typically, $\beta = l^{-2}$ such that $\beta^{-1} = l^2$ and $\frac{\alpha}{\beta} = a \cdot \beta^{-1} = \alpha l^2$. Substituting:

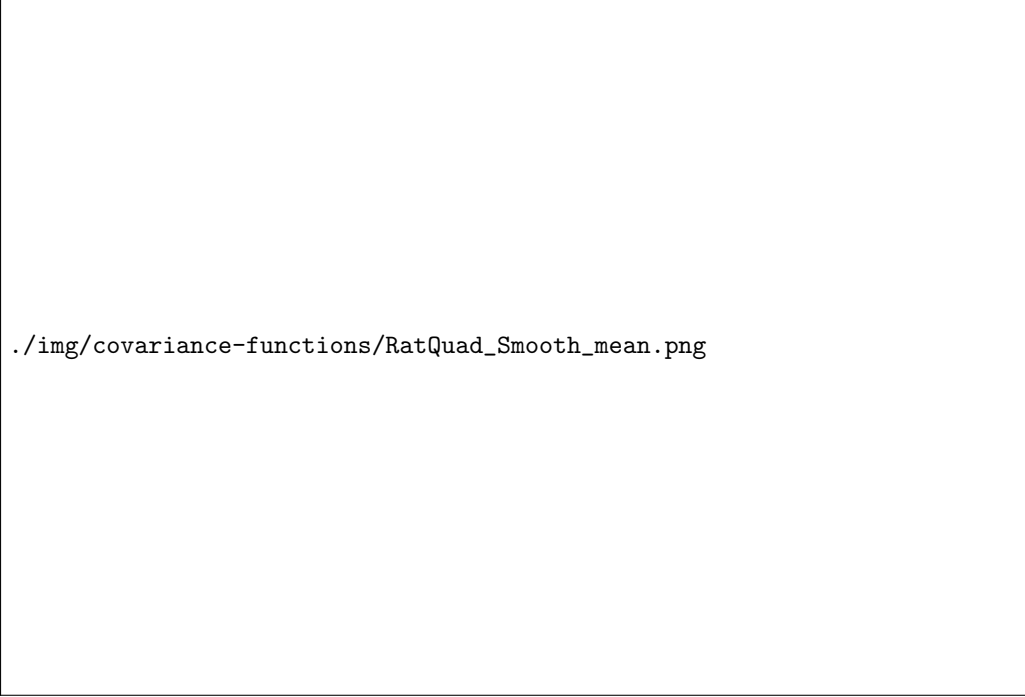
$$k(X, X') = \left(\alpha l^2 + \frac{|X - X'|^2}{2}\right)^{-\alpha - \frac{3}{2}}$$

Factoring out al^2 and absorbing it into the proportionality constant to get our final form:

$$k(X, X') = \left(1 + \frac{|X - X'|^2}{2\alpha l^2}\right)^{-\alpha - \frac{3}{2}}$$

This derivation starts from the discrete length scale l for illustrative purposes, so includes $\frac{3}{2}$ to account for the change from length scale to τ . In practice, derivations start from τ so never have to change units and never include the $\frac{3}{2}$ term in the exponent.

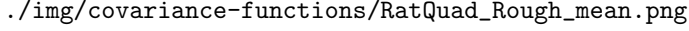
RQ is infinitely MS differentiable, so produces infinitely smooth functions no matter what the shape parameter of our distribution of length scales α is. As $\alpha \rightarrow \infty$, this distribution becomes sharply peaked at a single length scale l and RQ becomes SE. However, as $\alpha \rightarrow 1$, we get a wider spread of length scales and our sample functions capture trends at different frequencies.



`./img/covariance-functions/RatQuad_Smooth_mean.png`

Figure 5: Plot of the expected function draw and variances as before, using RQ fitted to the smooth sin wave dataset.

The GauPro package [5] in R used to fit these GPs has chosen an extremely high $\alpha = 100$ via MLE because of the smoothness of the data-generating function. This high α means we are very likely to choose one length scale l and thus RQ is nearly indistinguishable from SE.

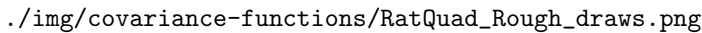


./img/covariance-functions/RatQuad_Rough_mean.png

Figure 6: Plot of the expected function draw and variances as before, using RQ fitted to the rougher dataset.

The rougher dataset has driven α to be much lower than in the smooth dataset, at $\alpha = 1.04$, and RQ becomes a true heavy-tailed mixture distribution of SEs over length scales.

The green variance lines disappear because the expected sample path passes through all datapoints, so any new training point is expected to be exactly equal to the expected function draw at that point, and the variance is zero. If the data-generating function became too rough or we used a more inflexible kernel like SE, we would need these sample variance lines to reflect how far from the expected function draw we expect the sample functions to be.



./img/covariance-functions/RatQuad_Rough_draws.png

Figure 7: Plots of some sample functions drawn from a GP as before, trained on the rough dataset using RQ.

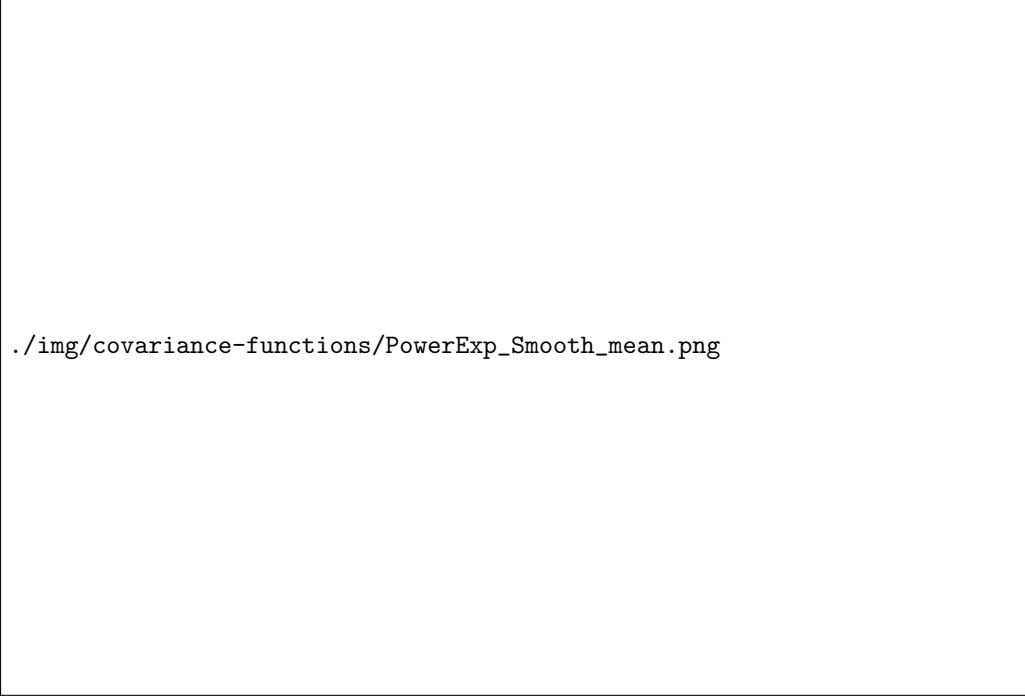
Each function draw expresses a different combination of high and low length scale components drawn from this fixed inverse-gamma distribution, and exhibit different behaviours at different frequencies. This is especially visible as the difference between function draws widens between data points - some function draws oscillate less than others as they are capturing a longer-term, low-frequency trend in the data. These more flexible functions are enough for the expected function draw to pass through all datapoints and satisfy the interpolation property.

3.3.3 γ -exponential and exponential

RQ controlled smoothness by using a mixture of different length scales to capture multiple trends in the data at different frequencies. Another approach to controlling smoothness involves introducing a differentiability parameter γ to control how differentiable the covariance function is and thus how smooth the sample functions are:

$$k(X, X') = \exp\left(-\frac{|X - X'|^\gamma}{l^\gamma}\right)$$

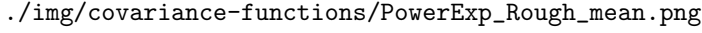
where $0 < \gamma \leq 2$ controls the smoothness of the covariance function. $\gamma = 2$ produces SE for maximal smoothness. For all other values of γ other than 2, the covariance function is continuous but not differentiable at all at $|x - x'| = 0$ because this point coincides with the undifferentiable turning point of the modulus function at $x - x' = 0$.



`./img/covariance-functions/PowerExp_Smooth_mean.png`

Figure 8: Plot of the expected function draw and variances as before, using γ -exponential fitted to the smooth sin wave dataset.

The GauPro package [5] in R has chosen $\gamma = 2$ via MLE, which produces SE. This is because the smoothness of the data-generating function is so high that no other covariance function can capture it.

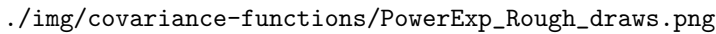


./img/covariance-functions/PowerExp_Rough_mean.png

Figure 9: Plot of the expected function draw and variances as before, using γ -exponential fitted to the rougher dataset.

$\gamma = 1.51$ has been selected by GauPro [5] via MLE, which produces a covariance function that is not differentiable at $|x - x'| = 0$. This is because the roughness of the data-generating function is too high for SE to capture, and the γ -exponential covariance function is able to adapt to this roughness.

These variances appear not to collapse to zero at the training points. In fact, this is a graphical issue caused by the very rough functions that the γ -exponential covariance function produces. Unlike the functions drawn from an RQ-powered GP that remain close to the training points for some time before and after the training points, functions drawn from this distribution only remain close to the training points for a very short distance before diverging - a distance that is too small to be visible in this plot.



./img/covariance-functions/PowerExp_Rough_draws.png

Figure 10: Plots of some sample functions drawn from a GP as before, trained on the rough dataset using γ -exponential.

These function draws are much rougher than those drawn from RQ or SE thanks to their non-differentiability.

Exponential covariance function $\gamma = 1$ produces the exponential covariance function:

$$k(X, X') = \exp\left(-\frac{|X - X'|}{l}\right) \quad (45)$$

Restating:

$$k(X, X') = a \exp(-c|X - X'|)$$

Setting $a = 1$ and $c = \frac{1}{l}$ recovers our familiar exponential kernel 45. This form relates to the Ornstein-Uhlenbeck [26] stochastic differential equation (SDE):

$$dX_t = -\lambda X_t dt + \sqrt{2\lambda} dW_t$$

This SDE has a unique stationary Gaussian solution [26] with a covariance equal to that of the exponential covariance function where $a = \sigma^2$ and $c = \lambda$:

$$\mathbb{E}[X_t] = 0$$

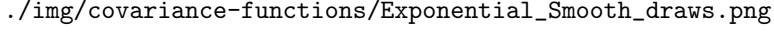
$$\text{Cov}(X_t, X_s) = \sigma^2 \exp(-\lambda|t - s|)$$

Because this SDE is of order one, the OU process is Gauss-Markovian, meaning that the future state of the process only depends on the current state X_t and not on any previous states X_{t-1} . Higher order SDEs with integer order m produce Matern covariance functions with $\nu = m - \frac{1}{2}$.

`./img/covariance-functions/Exponential_Smooth_mean.png`

Figure 11: Plot of the expected function draw and variances as before, using exponential fitted to the smooth sin wave dataset.

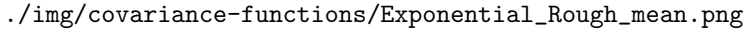
The expected function draw is practically a straight line connecting each data point, as the exponential covariance function is not smooth enough to capture the sin wave. The graphical issue observed in γ -exponential is more pronounced here, with the variances barely approaching zero as x_* approach a training point.



./img/covariance-functions/Exponential_Smooth_draws.png

Figure 12: Plots of some sample functions drawn from a GP as before, trained on the smooth dataset using exponential.

These function draws are effectively white noise that collapses at training points.



./img/covariance-functions/Exponential_Rough_mean.png

Figure 13: Plot of the expected function draw and variances as before, using exponential fitted to the rougher dataset.

The behaviour of GPs using the exponential covariance function is invariant to the data it is trained on - as with the smooth dataset, the expected function draw is a straight line connecting each data point, and the variances barely approach zero as x_* approaches a training point.

3.3.4 Matern-class

The γ -exponential generalisation is not useful due to its brittleness - it can only produce two covariance functions representing the extremes of the smoothness-roughness scale. The Matern class of covariance functions addresses this by introducing a parameter $\nu > 0$ that controls the smoothness of the function. The Matern class is defined as:

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

l is our familiar length scale hyperparameter, and K_ν is a modified Bessel function. This class has a spectral density:

$$S(\omega) = \frac{2^D \pi^{D/2} \Gamma(\nu + D/2) (2\nu)^\nu}{\Gamma(\nu) l^{2\nu}} \left(\frac{2\nu}{l^2} + 4\pi^2 s^2 \right)^{-(\nu + D/2)} \quad (46)$$

A Gaussian process using a Matern class kernel is k -times MS differentiable if and only if $\nu > k$.

Using half-integers, i.e. $\nu = p + 1/2$ where p is a non-negative integer, the covariance function becomes a product of a polynomial and an exponential:

$$k_{\nu=p+1/2}(X, X') = \exp\left(-\frac{\sqrt{2\nu}|X - X'|}{l}\right) \frac{\Gamma(p+1)}{\Gamma(2p+1)} \sum_{i=0}^p \frac{(p+i)!}{i!(p-i)!} \left(\frac{\sqrt{8\nu}r}{l}\right)^{p-i}$$

As $\nu \rightarrow \infty$, the Matern covariance function approaches the SE covariance function. $\nu = 1/2$ yields the exponential covariance function, and $\nu \geq 7/2$ produces functions that are as smooth as SE, leaving us with two cases of interest: $\nu = 3/2$ and $\nu = 5/2$.

Matern 3/2 $\nu = 3/2$ produces the following covariance function:


$$k_{3/2}(X, X') = \left(1 + \frac{\sqrt{3}|X - X'|}{l}\right) \exp\left(-\frac{\sqrt{3}|X - X'|}{l}\right) \quad (47)$$

$3/2$ is one-time MS differentiable, leading to much rougher functions than SE but smoother than exponential.

`./img/covariance-functions/Matern32_Smooth_mean.png`

Figure 14: Plot of the expected function draw and variances as before, using Matern 3/2 fitted to the smooth sin wave dataset.


This GP's expected function draw goes directly to the training points, similar to the behaviour of the exponential covariance function. Unlike the exponential covariance function, the one-time MS differentiability constraint on the sample functions restricts the sample functions from resembling white noise, which produces narrower variances closer to the expected draw.



`./img/covariance-functions/Matern32_Smooth_draws.png`

Figure 15: Plots of some sample functions drawn from a GP as before, trained on the smooth dataset using Matern 3/2.

These function draws are much smoother than those drawn from the exponential covariance function.



`./img/covariance-functions/Matern32_Rough_draws.png`

Figure 16: Plots of some sample functions drawn from a GP as before, trained on the rough dataset using Matern 3/2.

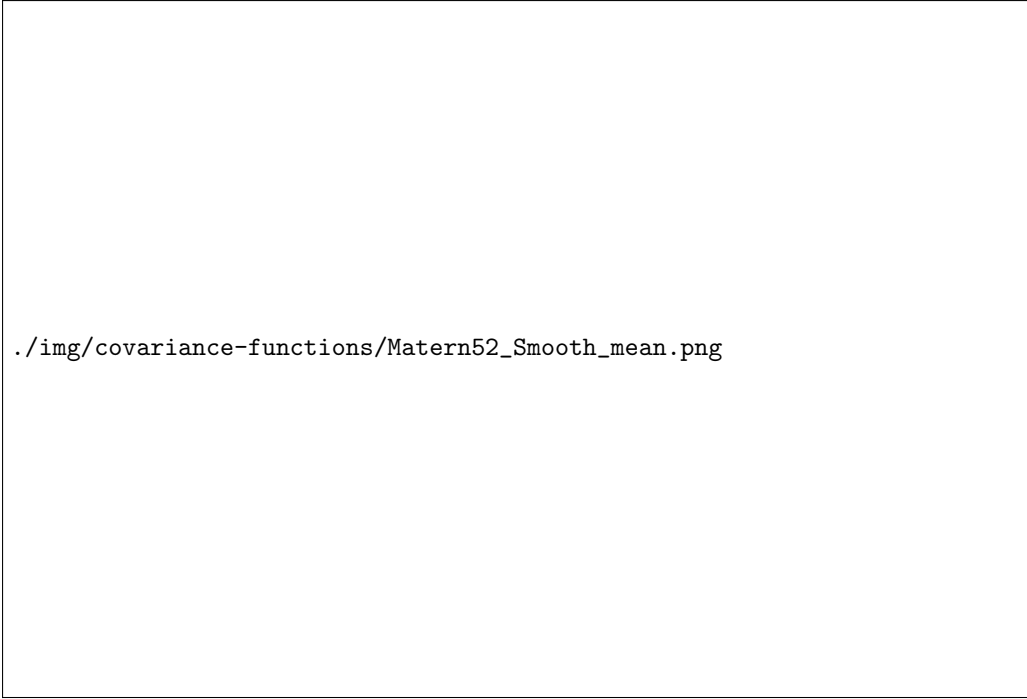
Although these function draws are much smoother than those produced by the exponential covariance function, they are still rough between points.

Matern 5/2 $\nu = 5/2$ produces the following covariance function:

$$k_{5/2}(X, X') = \left(1 + \frac{\sqrt{5}|X - X'|}{l} + \frac{5|X - X'|^2}{3l^2}\right) \exp\left(-\frac{\sqrt{5}|X - X'|}{l}\right)$$

5/2 is twice MS differentiable, producing functions that are slightly smoother than 3/2.


This is the most commonly used stationary covariance function [5] because it represents a sweet-spot between the smoothness of SE and the roughness of exponential, and is able to adapt to a wide range of data-generating functions.



`./img/covariance-functions/Matern52_Smooth_mean.png`

Figure 17: Plot of the expected function draw and variances as before, using Matern 5/2 fitted to the smooth sin wave dataset.


Here, Matern 5/2's expected function is smooth enough to resembles SE - a smooth function that did not pass through all training points. The variance of the noise-free function evaluations and the noisy observations are narrower than SE's, as the additional roughness allows the expected function draw to pass closer to the training points. For example, the previously examined point at (0.21, 0.86) is now much closer to the expected function draw than under SE.



`./img/covariance-functions/Matern52_Rough_mean.png`

Figure 18: Plot of the expected function draw and variances as before, using Matern 5/2 fitted to the rougher dataset.

Matern 5/2 has adapted well to the rough dataset - the expected function draw is smoother than the expected function draw from Matern 3/2, but remains rough enough to pass through all training points. The variances are the narrowest of all the covariance functions because the smoothness constraint is severe enough to prevent exponential-style white noise, but the roughness is high enough to approach the training points closely.



`./img/covariance-functions/Matern52_Rough_draws.png`

Figure 19: Plots of some sample functions drawn from a GP as before, trained on the rough dataset using Matern $5/2$.

These function draws are much smoother than those drawn from Matern $3/2$, particularly between data points, whilst still passing through all datapoints.

Computational Issues

4.1 Cholesky decomposition

Cholesky decomposition [21] is a matrix factorisation technique specialised for Gram matrices. Analogous to taking the square root of a Gram matrix, Cholesky finds a unique lower-triangular matrix L with positive diagonal entries for some Gram matrix A such that:

$$A = LL^T \quad (48)$$

For example, let:

$$A = \begin{pmatrix} 4 & 12 & -16 \\ 12 & 37 & -43 \\ -16 & -43 & 98 \end{pmatrix}$$

L would be:

$$L = \begin{pmatrix} 2 & 0 & 0 \\ 6 & 1 & 0 \\ -8 & 5 & 3 \end{pmatrix}$$

such that:

$$LL^T = \begin{pmatrix} 2 & 0 & 0 \\ 6 & 1 & 0 \\ -8 & 5 & 3 \end{pmatrix} \begin{pmatrix} 2 & 6 & -8 \\ 0 & 1 & 5 \\ 0 & 0 & 3 \end{pmatrix} = \begin{pmatrix} 4 & 12 & -16 \\ 12 & 37 & -43 \\ -16 & -43 & 98 \end{pmatrix}$$

We can assemble L using the Cholesky algorithm [19]. For diagonal entries in A , this costs $O(n)$:

$$L_{i,i} = \sqrt{A_{i,i} - \sum_{k=1}^{i-1} L_{i,k}^2}$$

For off-diagonal entries:

$$L_{i,j} = \begin{cases} \frac{1}{L_{j,j}} \left(A_{i,j} - \sum_{k=1}^{j-1} L_{i,k} L_{j,k} \right) & \text{if } i > j \\ 0 & \text{if } i < j \end{cases}$$

The computational cost of obtaining the Cholesky factorisation is $O(\frac{1}{3}n^3)$ thanks to the \sum terms in the off-diagonal entries where $i > j$.

Once we have L , we can represent A^{-1} in terms of Cholesky factors:

$$A^{-1} = L^{-T} L^{-1}$$

To obtain A^{-1} , we need to solve the forward substitution problem $LY = I$ for Y , where I is an $N \times N$ identity matrix, and plug this solution into the backward substitution problem $L^T X = Y$ to obtain $X = A^{-1}$. Solving for one column of A^{-1} requires solving the forward substitution problem for one column of I , which costs $O(n^2)$, and solving the backward substitution problem for one column of Y , which also costs $O(n^2)$. Thus, solving for all n columns of A^{-1} multiplies this cost to $O(n^2m)$, where m is the number of columns in the product. Since inverting the matrix alone requires a self-product, $m = n$ and the total cost of inverting A is $O(n^3)$.

The inversions of $K(X, X)$ that we need to form our predictive distribution 22 and likelihood 26 exist as products of other terms ($K(X, X)^{-1}y$ or $K(X, X)^{-1}K(X, X_*)$). We can obtain these products directly by solving these triangular systems:

$$K(X, X)^{-1}Y = L^{-T}(L^{-1}Y)$$

Y is the product of the $K(X, X)^{-1}$ that we need, e.g. the vector of noisy observations y or $K(X, X_*)$. y is a single column vector and $K(X, X_*)$ may not contain n columns, so we can solve these triangular problems in $O(n^2)$ and $O(n^2m)$ time respectively, where m is the number of testing points in X_* .

Cholesky is the preferred method of handling the inversion of covariance matrices because of its extreme numerical stability (thanks to the positive diagonal entries), and its guarantee to produce a valid Gram matrix. However, its cubic complexity is still too high for large datasets.

4.2 Subset-of-data (SoD)

One naive method to reduce this cubic complexity is to simply fit the GP on a subset M of our training data X to reduce the cost of inversion to $O(m^3)$, where m is the number of training points in M .

The central issue of this method is how to choose the subset. Choosing these points randomly maintains the highly desirable $O(m^3)$ complexity at the cost of accuracy. Hayashi et. al. [11] showed that this computational complexity is practically required to work with extremely large datasets. They also showed that at this scale, SoD is more accurate than other $O(m^3)$ methods [11] (i.e. SVGP).

To remain accurate, random subsampling requires a large enough dataset such that some of the data is redundant and M becomes a representative subset of X . This phenomenon is not possible for more reasonably sized datasets. The

size of M required to achieve the same level of accuracy can be reduced with a greedy approach. This greedy algorithm determines the gain in likelihood from including each data point x_i in X , then adds the "centroid", or the point that produces the maximal gain in likelihood, to M . This process is repeated until the size of M reaches m . However, computational savings from reducing m are smaller than the cost of searching X for these centroids $O(n^2m)$. Instead, Keerthi et. al. [15] developed a "matching pursuit" approach - maintain a cache of the already precomputed kernel values, and use these to compute the gain in likelihood for each point in X in $O(nm^2)$ time.

4.3 Eigenfunction and eigenvalue approximation of covariance matrices

Even with matching pursuit, SoD is still too inaccurate for reasonably sized datasets. Mercer's representation of the kernel 27 provides a way of approximating any arbitrary Gram matrix. Instead of summing all the eigenfunctions from 1 to ∞ , we can approximate our kernel to any degree of accuracy $M < N$ by restricting our sum to the largest M eigenvalues:

$$K(X, X') \approx \sum_{i=1}^M \lambda_i \phi_i(X) \phi_i(X') \quad (49)$$

and invert this smaller matrix at a lower computational cost $O(M^3)$. However, obtaining the eigenfunctions and eigenvalues of an N -size covariance matrix matrix is $O(N^3)$ for a total cost of $O(N^3) + O(M^3)$. We can reduce the cost of obtaining these eigenfunctions and eigenvalues by solving our eigenproblem on a smaller random sample of X .

Instead of our integral weighing our kernel and eigenfunction with the measure μ as before, they become weighed in terms of the PDF of the training data $p(x)$:

$$\lambda_i \phi_i(x') = \int K(x, x') p(x) \phi_i(x) dx$$

We can approximate this continuous representation and make it discrete by representing our selected samples as a vector X_l of size l :

$$\lambda_i \phi_i(X') \approx \frac{1}{l} \sum_{j=1}^l K(x_j, X') \phi_i(x_j)$$

Defining a vector Φ_i of all eigenfunction evaluations $\phi_i(x_j)$ on the X_l sample:

$$\lambda_i \phi_i(X') \frac{1}{l} K(X_j, X') \Phi_i$$

Multiplying both sides by l :

$$l \lambda_i \phi_i(X') = K(X_j, X') \Phi_i$$

Setting $\lambda_i^{mat} = l \lambda_i$:

$$K(X_l, X') \phi_i(X') = \lambda_i^{mat} \Phi_i$$

This arrangement requires our covariance matrix to be evaluated with some X' . Without creating another sample or defeating the point of this approximation by using the whole of X , the only candidate for X' is the X_l sample. Setting $X' = X_l$, our $\phi_i(X')$ becomes the definition of Φ_i :

$$K(X_l, X_l) \Phi_i = \lambda_i^{mat} \Phi_i \quad (50)$$

Here, Φ_i also plays the role of the eigenvector u_i of $K(X_l, X_l)$, and λ_i^{mat} plays the eigenvalue of $K(X_l, X_l)$.

Although we introduced $\lambda_i^{mat} = l \lambda_i$, λ_i^{mat} are the eigenvalues of the eigenproblem produced by sampling, whereas λ_i are the eigenvalues from the original eigenproblem. Thus, it acts as an approximation:

$$\lim_{l \rightarrow \infty} \lambda_i = \frac{1}{l} \lambda_i^{mat}$$

4.3.1 Approximating covariance matrices with Nystrom

Solving this eigenproblem is $O(l^3)$ - an improvement on the $O(n^3)$ cost of solving the previous eigenproblem, and the resulting $O(l^3) + O(M^3)$ could be lower than the naive $O(n^3)$ inversion cost (if we use sufficiently low degrees of accuracy l and M). We can improve this further by approximating Φ_i .

u_i is an obvious estimator for Φ_i , but they are not exactly equivalent thanks to differing normalisations. Φ_i 's normalisation comes from the original Monte Carlo approximation $\frac{1}{l} \sum_{j=1}^l \phi_i(x_j)^2 \approx 1$, thus the sum and our $\|\Phi_i\|_2^2 \approx l$ and $\|\Phi_i\| \approx \sqrt{l}$. u_i 's normalisation comes from solving the eigenproblem, which returns $\|u_i\|_2^2 = 1$ and $\|u_i\| = 1$. We can convert between the two by scaling Φ_i by $\frac{1}{\sqrt{l}}$ to produce u_i .

The Nystrom method [2] extends this approximation from the sample points X_l to the full set of points X :

$$\phi_i(X) \approx \frac{\sqrt{l}}{\lambda_i^{mat}} K(X, X_l) u_i$$

Assembling the full covariance matrix We can now assemble a full approximation for our covariance matrix by substituting this approximation of ϕ_i and $\lambda_i = \frac{\lambda_i^{mat}}{l}$ into the original Mercer approximation 49:

$$K(X, X') \approx \sum_{i=1}^M \left(\left[\frac{\lambda_i^{mat}}{l} \right] \left[\frac{\sqrt{l}}{\lambda_i^{mat}} K(X, X_l) u_i \right] \left[\frac{\sqrt{l}}{\lambda_i^{mat}} K(X', X_l) u_i \right]^T \right)$$

Expanding and simplifying:

$$K(X, X') \approx \sum_{i=1}^M \frac{1}{\lambda_i^{mat}} K(X, X_l) u_i u_i^T K(X', X_l)$$

The spectral theorem representation of the solution for $K(X_l, X_l)$ to the sampling eigenproblem 50 is:

$$K(X_l, X_l) = \sum_{i=1}^l \lambda_i^{mat} u_i u_i^T$$

The pseudo-inverse of this expression corresponds exactly to the middle term of our approximation, so we can substitute $K(X_l, X_l)^\dagger$ into our approximation:

$$K(X, X') \approx K(X, X_l) K(X_l, X_l)^\dagger K(X', X_l)$$

If we set $M = l$, the pseudo-inverse becomes a full inverse for the final Nystrom approximation $Q(X, X')$

$$K(X, X') \approx Q(X, X') = K(X, X_l) K(X_l, X_l)^{-1} K(X', X_l) \quad (51)$$

The final computational complexity comes from assembling these matrices. Assembling $K(X, X_l)$ and $K(X', X_l)$ costs $O(nl)$, and assembling $K(X_l, X_l)$ costs $O(l^2)$. Inverting $K(X_l, X_l)^{-1}$ costs $O(l^3)$, but we can compute the product $K(X_l, X_l)^{-1} K(X', X_l)$ in $O(l^2 m)$ time using Cholesky decomposition, where m is the number of columns in $K(X', X_l)$. The process of assembling the Cholesky factors themselves cost $O(l^3)$. Once we have these matrices, the cost of assembling their products are $O(nlm)$, leading to a final computational complexity of $O(nlm + l^3)$, or $O(nm^2 + m^3)$ if we set $l = m$. However, inverting this matrix is $O(m^3)$ which is equal to the cost of Cholesky decomposition, so we can use the Nystrom approximation to reduce the cost of assembling the covariance matrix and its inverse to $O(nm^2 + m^3)$.

PSD issues using naive Nystrom Williams and Seeger [28] developed the naive Nystrom GP by applying the Nystrom approximation 51 to the training data's covariance matrix $K(X, X)$ that required inverting for the predictive distribution and likelihood. This effectively specified a new joint prior on $f(X_*)$ that only replaced $K(X, X)$ with $Q(X, X_*)$ in 23:

$$p(f(X), f(X_*)) = \mathcal{N} \left(0, \begin{pmatrix} Q(X, X) + \sigma_n^2 & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{pmatrix} \right)$$

The resulting predictive distribution has the following variance:

$$\text{Var}(f(X_*)|y) = K(X_*, X_*) - K(X_*, X)(Q(X, X) + \sigma_n^2)^{-1} K(X, X_*)$$

To be a valid predictive distribution, this overall variance must satisfy PSD because the predictive covariance between two points must be positive or zero.

$Q(X, X)$ and the other components of the prior are guaranteed to be PSD, but this variance is guaranteed to be non-negative only if the entire joint prior is PSD. To illustrate, here is an example 2×2 matrix A :

$$A = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$$

A satisfies PSD if $a \geq 0$, $c \geq 0$ and $ac - b^2 \geq 0$. Even if a and c are individually PSD, if b is sufficiently large and ac is sufficiently small, the entire matrix can become non-PSD. This is the case for the naive Nystrom approximation, where a sufficiently small approximation $Q(X, X)$ (e.g. if the inducing points X_l are far from the training points X) or a sufficiently large $K(X, X_*)$ (e.g. if the testing points X_* are close to the training points), can produce a non-PSD variance.

4.4 Sparse approximations

Sparse approximation methods attempt to build global GPs that represent the entire training data using a set of potentially imaginary inducing points X_m selected via MLE to best approximate the full data. These methods attempt to resolve the PSD issues with naive Nystrom whilst maintaining its computational efficiency. For example, the Nystrom approximation 51 using inducing points becomes:

$$K(X, X) \approx Q(X, X) = K(X, X_m) K(X_m, X_m)^{-1} K(X_m, X)$$

The joint prior 21 to relate the real and inducing point datasets by introducing the Nystrom approximation:

$$\begin{pmatrix} f(X) \\ f(X_m) \end{pmatrix} \sim \mathcal{N} \left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} K(X, X), Q(X, X_*) \\ Q(X_*, X), K(X_*, X_*) \end{pmatrix} \right)$$

Using the Gaussian conditioning rule 22, we can obtain the the prior conditional distribution $p(f(X_*)|f(X_m))$:

$$q(f(X)|f(X_m)) = \mathcal{N} \left(\begin{aligned} &K(X, X_m)K(X_m, X_m)^{-1}f(X_m), \\ &K(X, X) - Q(X, X) \end{aligned} \right) \quad (52)$$

Since X_m is imaginary, we need to integrate out $f(X_m)$ and condition on y to obtain the "posterior" distribution $p(f(X), f(X_m)|y)$, or the new predictive distribution after observing the noisy observations y . If we do this naively, we simply recover our original predictive distribution 22 and its cubic inference cost:

$$p(f(X_*), f(X_m)|y) = \mathcal{N}(K(X_*, X)(K(X, X) + \sigma_n^2 I)^{-1}y, K(X_*, X_*) - K(X_*, X)(K(X, X) + \sigma_n^2 I)^{-1}K(X, X_*))$$

Additionally, this posterior produces the same marginal likelihood as before.

There are two classes of strategies to avoid this: prior approximations $q(f(X)|f(X_m))$ which change the conditional distribution prior to observing y $p(f(X)|f(X_m))$ 52 and perform the conditioning process exactly, and posterior approximations that use the exact conditional distribution $p(f(X)|f(X_m))$ 52 but approximate the final posterior distribution $q(f(X), f(X_m)|y)$.

4.4.1 Prior approximations

Prior approximations modify approximate the variance of the conditional distribution 53 with $\bar{Q}(X, X)$:

$$q(f(X)|f(X_m)) = \mathcal{N}(K(X, X_m)K(X_m, X_m)^{-1}f(X_m), \bar{Q}(X, X))$$

This yields the following predictive distribution:

$$\begin{aligned} q(f(X_*), f(X_m)|y) &= \mathcal{N} \left(\begin{aligned} &Q(X_*, X) (\bar{Q}(X, X) + Q(X, X) + \sigma_n^2 I)^{-1} y, \\ &(Q(X_*, X_*) + \bar{Q}(X_*, X_*)) - Q(X_*, X) (\bar{Q}(X, X) + Q(X, X) + \sigma_n^2 I)^{-1} Q(X, X_*) \end{aligned} \right) \end{aligned} \quad (53)$$

This also produces a new likelihood:

$$\begin{aligned} q(y) &= -\frac{n}{2} \log 2\pi - \frac{1}{2} \log |\bar{Q}(X, X) + Q(X, X) + \sigma_n^2 I| \\ &\quad - \frac{1}{2} Y^T [\bar{Q}(X, X) + Q(X, X) + \sigma_n^2 I]^{-1} y \end{aligned} \quad (54)$$

The only term being inverted in the likelihood and the predictive distribution is $\bar{Q}(X, X) + Q(X, X) + \sigma_n^2 I$, and it is always a product of the vector y . If $\bar{Q}(X, X)$ is chosen to be rank m or lower and is easy to invert, then obtaining this product of an inversion costs $O(nm^2)$ plus the cost of Cholesky factorisation $O(m^3)$, leading to an overall training and inference complexity of $O(nm^2 + m^3)$.

As our inducing points become more representative of the data and $X_m \rightarrow X$, our Nystrom approximation improves and $Q(X, X) \rightarrow K(X, X)$. If we use a $\bar{Q}(X, X)$ that does not interfere with this process, then we can theoretically recover the full GP exactly as $X_m \rightarrow X$. However, Titsias [25] found that maximising the approximate likelihood $q(y)$ when $X_m = X$ does not produce the same hyperparameters as maximising the full GP likelihood $p(y)$, meaning the approximate likelihood is not a good approximation of the full GP likelihood and can produce biased hyperparameter estimates.

Subset-of-Regression (SoR) Joukov and Kulic's Subset-of-Regression (SoR) [13] sets $\bar{Q}(X, X) = 0$, producing this noisy predictive distribution:

$$q_{\text{SoR}}(f(X_*), f(X_m)|y) = \mathcal{N}(Q(X_*, X) (Q(X, X) + \sigma_n^2 I)^{-1} y, Q(X_*, X_*) - Q(X_*, X) (Q(X, X) + \sigma_n^2 I)^{-1} Q(X, X_*))$$

This approximation replaces $K(\cdot)$ in the full predictive distribution 22 with the Nystrom approximation $Q(\cdot)$, which is the simplest way of addressing the PSD issues with naive Nystrom and maintains the favourable $O(nm^2 + m^3)$ training and inference complexity. However, the variance of the predictive distribution does not reflect the approximate nature of $Q(\cdot)$ and produces variances that are generally too low, especially in regions far from the inducing points X_m [19] that force $Q(X, X) \rightarrow 0$.

Fully independent training conditional (FITC) Snelson and Ghahramani’s Fully Independent Training Conditional (FITC) [24] improves on SoR by introducing information on the accuracy of the Nystrom approximation into the \bar{Q} assumption:

$$\begin{aligned} V(X, X') &= K(X, X) - Q(X, X) \\ \bar{Q}(X, X') &= \text{diag}[V(X, X')] \end{aligned}$$

Substituting this into the posterior predictive distribution 53 produces:

$$q_{\text{FITC}}(f(X_*), f(X_m)|y) = \mathcal{N}(Q(X_*, X) (\text{diag}[V(X, X)] + Q(X, X) + \sigma_n^2 I)^{-1} y, (Q(X_*, X_*) + \text{diag}[V(X_*, X_*)]) - Q(X_*, X) (\text{diag}[V(X, X)] + Q(X, X) + \sigma_n^2 I)^{-1} y)$$

Since this diagonal is a vector and is simple to invert, we maintain our $O(nm^2 + m^3)$ training and inference complexity.

$V(X, X)$ directly measures the differences between the full covariance matrix $K(X, X)$ and the Nystrom approximation $Q(X, X)$, and its diagonal is a vector of residuals from the Nystrom approximation for each point x_i in X . By using the diagonal, FITC enforces independence between the Nystrom residuals since the diagonal does not incorporate information on the residuals of other points. Bauer et. al. [3] found that if the residuals are heteroskedastic, then FITC interprets this as input-dependent noise, which can lead to a severe underestimation of σ_n^2 and a biased predictive distribution mean.

4.4.2 Posterior approximations

Variational free energy (VFE) Titsias [25] introduced a variational distribution to approximate the posterior that uses the full joint prior $p(f(X)|f(X_m))$:

$$q(f(X), f(X_m)|y) = p(f(X)|f(X_m)) \cdot q(f(X_m)|y) \quad (55)$$

Our optimal variational distribution $q^*(f(X_m)|y)$ is a Gaussian with mean μ^* and covariance Σ^* . Using these parameters and our full joint prior $p(f(X)|f(X_m))$ 52, we can assemble our predictive distribution $q(f(X_*), f(X_m)|y)$:

$$\begin{aligned} q(f(X_*), f(X_m)|y) &= \mathcal{N}(\mu^*, \Sigma^*) \\ &= \mathcal{N}(\mu^*, \Sigma^* + K(X_*, X_m)K(X_m, X_m)^{-1}\Sigma^* \\ &\quad + V(X_*, X_*) + K(X_*, X_m)K(X_m, X_m)^{-1}\Sigma^* (K(X_m, X_m)^{-1}K(X_m, X_*))^T) \end{aligned}$$

$V(X_*, X_*)$ is the FITC-style Nystrom residuals $K(X_*, X_*) - Q(X_*, X_*)$.

Obtaining estimates for Σ^* and μ^* requires assembling a marginal likelihood. The Kullback-Leibler (KL) divergence measures how different two probability distributions are. The KL divergence between the true posterior $p(f(X), f(X_m)|y)$ and our variational distribution $q(f(X), f(X_m)|y)$ is:

$$D_{KL}(q(f(X), f(X_m)|y)||p(f(X), f(X_m)|y)) = \int q(f(X), f(X_m)|y) \log \frac{q(f(X), f(X_m)|y)}{p(f(X), f(X_m)|y)} df(X) df(X_m)$$

We need this expression in terms of $\log p(y)$ to form an approximation for the likelihood. We can introduce this term into D_{KL} by Bayes’ theorem:

$$D_{KL}(q(\cdot)||p(\cdot)) = \int q(f(X), f(X_m)|y) \log \frac{q(f(X), f(X_m)|y)}{p(y, f(X), f(X_m)|y)/p(y)} df(X) df(X_m)$$

Simplifying:

$$D_{KL}(q(\cdot)||p(\cdot)) = \int q(f(X), f(X_m)|y) \log \frac{q(f(X), f(X_m)|y)p(y)}{p(y, f(X), f(X_m)|y)} df(X) df(X_m)$$

Splitting the log term:

$$D_{KL}(q(\cdot)||p(\cdot)) = \int q(f(X), f(X_m)|y) \log \frac{q(f(X), f(X_m)|y)}{p(y, f(X), f(X_m)|y)} df(X) df(X_m) + \int q(f(X), f(X_m)|y) \log p(y) df(X) df(X_m)$$

$\log p(y)$ is a constant with respect to $f(X)$ and $f(X_m)$ and $\int q(f(X), f(X_m)|y) df(X) df(X_m) = 1$, so our second integral is just $\log p(y)$. Thus:

$$D_{KL}(q(\cdot)||p(\cdot)) = \int q(f(X), f(X_m)|y) \log \frac{q(f(X), f(X_m)|y)}{p(y, f(X), f(X_m)|y)} df(X) df(X_m) + \log p(y)$$

Rewriting to isolate $\log p(y)$:

$$\log p(y) = \int q(f(X), f(X_m)|y) \log \frac{p(y, f(X), f(X_m))}{q(f(X), f(X_m)|y)} df(X) df(X_m) + D_{KL}(q(\cdot)||p(\cdot)) \quad (56)$$

Our first term $F = \int q(f(X), f(X_m)|y) \log \frac{p(y, f(X), f(X_m))}{q(f(X), f(X_m)|y)} df(X) df(X_m)$ is called the Evidence Lower Bound (ELBO), or the VFE term. Titsias [25] found that minimising $D_{KL}(q(\cdot)||p(\cdot)) \geq 0$ is equivalent to maximising $F \leq \log p(y)$

Maximising F using the model hyperparameters directly improves the model, whilst reducing $D_{KL}(q(\cdot)||p(\cdot))$ using the parameters of the variational distribution drives the variational distribution closer to the true posterior. Unlike prior approximations, using the whole training data X in F recovers $\log p(y)$ exactly which removes the bias in hyperparameter estimates as $X_m \rightarrow X$.

Titsias [25] computed the Gaussian integrals in 56, and applied calculus of variations to eliminate $q(f(X_m)|y)$ and represent the final "collapsed" ELBO F_{VFE} in terms of SoR's likelihood (54, $\hat{Q}(X, X) = 0$) and the Nystrom residuals from FITC $V(X, X)$:

$$F_{VFE} = \log q_{SoR}(y) - \frac{1}{2\sigma_n^2} \text{tr}[V(X, X)] \geq F$$

Similar to its usage in FITC, the trace term $\text{tr}[V(X, X)]$ represents the residual variance the introduced Nystrom approximations left unexplained. The cost of assembling F_{VFE} for training is dominated by the likelihood term $\log q_{SoR}(y)$, which is $O(nm^2 + m^3)$. VFE achieves a more accurate approximation of the marginal likelihood than FITC without additional computational complexity.

Deriving FITC through the VFE framework Bui et. al. [4] explored the links to FITC that are introduced in VFE by using the exact sparse prior 52. They found that applying expectation propagation, or minimising the "inclusive" KL where p and q are swapped: $D_{KL}(p(\cdot)||q(\cdot))$ recovers the FITC approximate prior exactly. They defined the power expectation propagation (PEP) objective:

$$\log q_{PEP}(y) = \log q(y) - \frac{1-\alpha}{2\alpha} \text{tr} \left[\log(I_n + \frac{\alpha}{\sigma_n^2} V(X, X)) \right]$$

$q(y)$ represents our FITC prior and the trace term represents VFE . When $\alpha = 1$, the trace term vanishes and we recover FITC exactly. As $\alpha \rightarrow 0$, the expansion of the $\frac{1-\alpha}{\alpha} \log(I + \alpha A) \rightarrow \text{tr}[A]$ and we recover VFE in its collapsed form.

Stochastic variational GP (SVGP) Instead of collapsing the approximate variational distribution $q(f(X_m)|y)$ in the final ELBO by calculus of variations, Hensman et. al. [12] explicitly retained it as something to maximise:

$$q(f(X_m)|y) = \mathcal{N}(\mu, \Sigma) \quad (57)$$

μ is an m -sized vector representing the best estimates of the function at each inducing point, and Σ is an $m \times m$ covariance matrix representing the covariance between the inducing points. These are free variational parameters optimised to maximise this ELBO:

$$F_{SVGP} = \mathbf{E}_{q(f(X_m)|y) \cdot p(f(X)|f(X_m))} [\log p(y|f(X))] - D_{KL}(q(f(X_m)|y)||p(f(X_m)))$$

The KL distance between our approximation and the prior is:

$$D_{KL}(q(f(X_m)|y)||p(f(X_m))) = \frac{1}{2} \left(\log \frac{|K(X_m, X_m)|}{|S|} - m + \text{tr}(K(X_m, X_m)^{-1} \Sigma) + \mu^T K(X_m, X_m)^{-1} \mu \right)$$

For the i -th point, we can find the squared difference between the true y_i and the predicted $f(x_i)$ by producing predictions using the posterior mean μ_i and variance Σ_{ii} at i :

$$\mathbf{E}_{q(f(X_i))} [\log p(y_i|f(X_i))] = \mathbf{E} [(y_i - f(x_i))^2] = (y_i - \mu_i)^2 + \Sigma_{ii}$$

Assuming these remainders are Gaussian and independent of each other, we can sum all i remainders together and plug them into the Gaussian likelihood to get a global expression:

$$\log p(y|f(X)) = -\frac{n}{2} \log(2\pi\sigma_n^2) - \frac{1}{2\sigma_n^2} \sum_{i=1}^n [(y_i - \mu_i)^2 + \Sigma_{ii}]$$

The only reason both training and inference using this approximation is $O(mn^2 + m^3)$ is because the posterior means and variances currently cost $O(nm^2)$ to compute. Spending this during inference is unavoidable, but Kingma and Ba's Stochastic Gradient Descent (SGD) [17] suggests that we can reduce training costs by only summing a batch of b points at a time:

$$\log p(y|f(X)) = -\frac{n}{2} \log(2\pi\sigma_n^2) - \frac{1}{2\sigma_n^2} \frac{n}{b} \sum_{i=1}^b [(y_i - \mu_i)^2 + \Sigma_{ii}]$$

This results in a training complexity of $O(bm^2 + m^3)$, which is significantly lower than the $O(nm^2 + m^3)$ complexity of VFE and FITC. However, since the optimal $q(f(X_m)|y)$ is not analytically determined and collapsed, F_{SVGP} will be lower than F_{VFE} and produce less accurate hyperparameter estimates. Practically, SGD itself requires hyperparameter estimates such as learning-rate and schedule choices, and optimising m and S in addition to inducing locations and model hyperparameters increases epoch time.

4.5 Celerite kernels

The approximation methods we have seen so far are designed to approximate any valid covariance matrix. The celerite kernel is an example of a structured approximation approach, whereby the kernel itself is specifically designed to be computationally efficient to invert.

A naive example of a structured approximation is a sparse kernel. Sparse kernels [19] are particularly designed kernels that impose $k(X, X') = 0$ if $|X - X'|$ is larger than some threshold d to create a sparse covariance matrix. This reduces the number of calculations that need to be performed and computational complexity to $O(an^3)$, where a is the proportion of non-zero entries remaining. However, the kernel needs to be carefully designed to work with zeroes and ensure all entries are PSD.

Celerite models achieve a much smaller computational complexity $O(NJ^2)$ for one-dimensional stationary kernels expressible as sums of dampened sinusoids and exponentials, such as the Matern 3/2 kernel.

4.5.1 The celerite model

Rybicki and Press [23] used the Markov property of the exponential covariance function to produce a fast algorithm for computing its inverse. Since the exponential kernel is Markov, the only non-zero inverse covariance at i is $i - 1$ and $i + 1$, and the inverse covariance matrix is tridiagonal whose inverse can be computed in $O(n)$ time.

Kelly et. al. [16] generalised this to J arbitrary mixtures of exponentials:

$$K(X, X') = \sum_{j=1}^J a_j \exp(-c_j |X - X'|)$$

Although the inverse of this is dense, Kelly et. al. [16] showed that it can be computed in $O(NJ^2)$ time.

Foreman-Mackay et. al. [7] used the Bochner representation 38 of this generalised covariance function:

$$\begin{aligned} K(X, X') = & \sum_{j=1}^J \\ & \text{frac12}(a_j + ib_j) \exp(-(c_j + id_j)|X - X'|) \\ & + \frac{1}{2}(a_j - ib_j) \exp(-(c_j - id_j)|X - X'|) \end{aligned}$$

Using the Fourier transform 39:

$$\begin{aligned} K(X, X') = & \sum_{j=1}^J \\ & a_j \exp(-c_j |X - X'|) \cos(d_j |X - X'|) \\ & + b_j \exp(-c_j |X - X'|) \sin(d_j |X - X'|) \end{aligned} \tag{58}$$

The argument within this sum is called a celerite term. Each celerite term contributes two ranks in the semiseparable factorisation; with J celerite terms, the off-diagonal rank is $2J$. Setting $b_j = 0$ and $d_j = 0$ recovers the original exponential kernel, but Foreman-Mackay et. al.'s [7] generalisation can be extended to approximate more complex kernels than sums of exponentials whilst maintaining Kelly et. al. [16]'s $O(NJ^2)$ complexity.

4.5.2 Building Matern 3/2 from celerite

Applying the Fourier representation to the kernel to obtain its spectral density 40:

$$S(\omega) = \sum_{j=1}^J \sqrt{\frac{2}{\pi}} \frac{(a_j c_j + b_j d_j)(c_j^2 + d_j^2) + (a_j c_j - b_j d_j)\omega^2}{\omega^4 + 2(c_j^2 - d_j^2)\omega^2 + (c_j^2 + d_j^2)^2}$$

A stochastically-driven damped simple harmonic oscillator (SHO) is a system that oscillates with a frequency ω and is dampened by a factor c . The differential equation for this system is:

$$\left[\frac{d^2}{dt^2} + \frac{\omega_0}{Q} \frac{d}{dt} + \omega_0^2 \right] y(t) = \eta(t)$$

Q is the "quality factor" of the oscillator, which determines how quickly it loses energy, and $\eta(t)$ is the stochastic force driving the oscillation. Anderson et. al. [1] found that if $\eta(t)$ is white noise, the spectral density of the SHO is:

$$S(\omega) = \sqrt{\frac{2}{\pi}} \frac{S_0 \omega_0^4}{(\omega^2 - \omega_0^2)^2 + \frac{\omega_0^2 \omega^2}{Q^2}}$$

where S_0 is proportional to the power at $\omega = \omega_0$, $S(\omega_0) = \sqrt{\frac{2}{\pi}} S_0 Q^2$.

Foreman-Mackay et. al. [7] showed that the celerite spectral density matched the SHO spectral density if $Q \geq \frac{1}{2}$:

$$\begin{aligned} a_j &= S_0 \omega_0 Q \\ b_j &= \frac{S_0 \omega_0 Q}{\sqrt{4Q^2 - 1}} \\ c_j &= \frac{\omega_0}{2Q} \\ d_j &= \frac{\omega_0}{2Q} \sqrt{4Q^2 - 1} \end{aligned}$$

For $0 < Q \leq \frac{1}{2}$, Foreman-Mackay et. al. [7] used a pair of celerite terms with parameters:

$$\begin{aligned} a_{j\pm} &= \frac{1}{2} S_0 \omega_0 Q \left[1 \pm \frac{1}{\sqrt{1 - 4Q^2}} \right] \\ b_{j\pm} &= 0 \\ c_{j\pm} &= \frac{\omega_0}{2Q} \left[1 \pm \sqrt{1 - 4Q^2} \right] \\ d_{j\pm} &= 0 \end{aligned}$$

As $Q \rightarrow 0$, our spectral density indicates that the process has no oscillatory behaviour and over-damped, e.g. $Q = 1/\sqrt{2}$ recovers an astrophysics model [14] for background granulation noise in stars. Conversely, the process exhibits strong oscillatory behaviour from $Q = \frac{1}{2}$ onwards.

We can use the Wiener-Khinchin theorem 41 to obtain the SHO kernel from these spectral densities:

$$K(X, X'; S_0, Q, \omega_0) = S_0 \omega_0 Q \exp\left(-\frac{\omega_0 |X - X'|}{2Q}\right) \begin{cases} \cosh(\eta \omega_0 |X - X'|) + \frac{1}{2\eta Q} \sinh(\eta \omega_0 |X - X'|) & 0 < Q < \frac{1}{2} \\ 2(1 + \omega_0 |X - X'|) & Q = \frac{1}{2} \\ \cos(\eta \omega_0 |X - X'|) + \frac{1}{2\eta Q} \sin(\eta \omega_0 |X - X'|) & Q > \frac{1}{2} \end{cases}$$

$$\eta = |1 - (4Q^2)^{-1}|^{1/2}.$$

At $Q = \frac{1}{2}$, the kernel becomes:

$$K(X, X')_{Q=\frac{1}{2}} = S_0 \omega_0 \exp^{-\omega_0 |X - X'|} (1 + \omega_0 |X - X'|)$$

Setting $\omega_0 = \frac{\sqrt{3}}{l}$ and $\sigma^2 = S_0 \omega_0$ recovers the Matern 3/2 kernel :

$$K(X, X')_{Q=\frac{1}{2}} = \sigma^2 \exp\left(-\frac{\sqrt{3}}{l} |X - X'| \right) \left(1 + \frac{\sqrt{3}}{l} |X - X'| \right)$$

4.5.3 Cholesky factorisation and inversion of celerite kernels

Foreman-Mackay et. al. [7] develop a Cholesky factorisation method for these celerite kernels in $O(NR^2)$ complexity by exploiting their semiseparable structure. This methods reduces the $O(n^3)$ complexity of standard Cholesky factorisation 48 whilst retaining its numerical stability.

Semiseparable matrices A rank- R semiseparable matrix K is a matrix that can be expressed as a sum of a diagonal matrix A and two low-rank matrices U and V :

$$K_{nm} = \begin{cases} \sum_{r=1}^R U_{nr} V_{mr} & m < n \\ A_{nn} & m = n \\ \sum_{r=1}^R U_{mr} V_{nr} & m > n \end{cases} \quad (59)$$

In matrix notation:

$$K = A + \text{tri}_{\text{upper}}(UV^T) + \text{tri}_{\text{lower}}(VU^T) \quad (60)$$

A contains all the diagonal elements of the matrix $i = j$, and the second and third terms cover all the elements below and above the diagonal respectively.

Celerite kernels as semiseparable matrices For celerite kernels 58, the diagonal component A is simply:

$$A_{nn} = \sum_{j=1}^J a_j$$

[7] For the off-diagonal components, defining $R = 2J$, we can assemble U and V [7]:

$$\begin{aligned} U_{n,2j-1} &= a_j \exp(-c_j t_n) \cos(d_j t_n) + b_j \exp^{-c_j t_n} \sin(d_j t_n) \\ U_{n,2j} &= a_j \exp(-c_j t_n) \sin(d_j t_n) - b_j \exp^{-c_j t_n} \cos(d_j t_n) \\ V_{m,2j-1} &= \exp(+c_j t_m) \cos(d_j t_m) \\ V_{m,2j} &= \exp(+c_j t_m) \sin(d_j t_m) \end{aligned}$$

The resulting covariance matrix K_{nm} can be assembled using the rules for semiseparable matrices 59.

Cholesky factorisation of semiseparable matrices The goal of Cholesky factorisation 48 remains to find the square root of the matrix. For standard Cholesky factorisation, the diagonal is already baked into L . However, for semiseparable matrices the diagonal is not part of L and needs to be taken into account in the Cholesky representation of K :

$$K = LDL^T \quad (61)$$

Foreman-Mackay et. al. [7] use the ansatz that our Cholesky factor has the form:

$$L = I + \text{tri}_{\text{lower}}(UW^T)$$

I is the identity matrix, U is from the definition of the semiseparable matrix 60. After we determine the unknown $N \times N$ matrix W , we can determine our Cholesky factor. Substituting this ansatz into 61:

$$K = [I + \text{tri}_{\text{lower}}(UW^T)]D[I + \text{tri}_{\text{upper}}(WU^T)]$$

Simplifying:

$$K = D + \text{tri}_{\text{lower}}(UW^T)D + D\text{tri}_{\text{upper}}(WU^T) + \text{tri}_{\text{lower}}(UW^T)D\text{tri}_{\text{upper}}(WU^T)$$

Setting each element on the right-hand side equal to the corresponding element on the left-hand side of 59 (e.g. $A = D$), we derive the following recursive definition W [7]:

$$\begin{aligned} S_{n,j,k} &= S_{n-1,j,k} + D_{n-1,n-1}W_{n-1,j}W_{n-1,k} \\ D_{n,n} &= A_{n,n} - \sum_{j=1}^R \sum_{k=1}^R U_{n,j}S_{n,j,k}U_{n,k} \\ W_{n,j} &= \frac{1}{D_{n,n}} \left[V_{n,j} - \sum_{k=1}^R U_{n,k}S_{n,j,k} \right] \end{aligned}$$

Each iteration of this recursion costs $O(R^2)$, and this needs to be run for all N rows in W for an overall complexity of $O(NR^2)$. This cost dominates the $O(N)$ cost of obtaining the triangle matrix U and the diagonal matrix D .

Inversion of semiseparable matrices with a Cholesky factorisation As with standard regular Cholesky inversion, we can avoid the full cost of inversion since GPs only require the inverse multiplied by a product. This process is also cheaper than under regular Cholesky factorisation. As before, our goal is to solve $z = K^{-1}y$:

$$z = K^{-1}y = L^{T^{-1}}D^{-1}L^{-1}y$$

Recursively solving for $L^{-1}y$ using forward substitution, and defining the solution as z' :

$$\begin{aligned} f_{n,j} &= f_{n-1,j} + W_{n-1,j}z'_{n-1} \\ z'_n &= y_n - \sum_{j=1}^R U_{n,j}f_{n,j} \end{aligned}$$

where $f_{0,j} = 0$ for all j . Using z' and backward substitution, we can compute z recursively:

$$\begin{aligned} g_{n,j} &= g_{n+1,j} + U_{n+1,j}z_{n+1} \\ z_n &= \frac{z'_n}{D_{n,n}} - \sum_{j=1}^R W_{n,j}g_{n,j} \end{aligned}$$

The total cost of both the forwards and backwards substitution is $O(NR)$, equivalently $O(NJ)$.

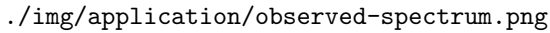
Celerite models achieve extreme numerical stability and accuracy, but have several disadvantages. The $O(NJ^2)$ complexity, whilst an improvement over the naive Cholesky inversion $O(n^3)$ or the Nystrom-based approximations $O(nm^2 + m^3)$, is inferior to the $O(bm^2 + m^3)$ complexity of SVGP. The Markov property that this alternative Cholesky factorisation exploits requires that the data is sequentially ordered. Celerite approximations rely on the off-diagonal blocks being uniformly low rank $2J$ which is not the case if the training data is more than one dimension. These two issues restrict celerite models to one-dimensional time-series data only.

Applying Gaussian Processes to Astrophysics

5.1 Introduction

5.1.1 Astronomical background

The optical light from a distant galaxy, integrated over its entire size in the sky, comes from stars and gas in the galaxy that shine. We can find the spectral energy distribution (SED) of the galaxy by measuring the brightness at different wavelength bands. Brightness at specific wavelengths can be used to infer the existence of certain behaviours in the galaxy's gases and stars.



./img/application/observed-spectrum.png

Figure 20: SED of a sample galaxy (id 51613-0305-552) [18].

The 4000-4500 wavelength region of this SED contains two examples of how a galaxy's age can be inferred after observing their SED. The upward narrow emission band highlighted in green is from ionised gas, mostly Hydrogen and Oxygen in this wavelength range, produced by younger stars. The downward dipping absorption band in red is from electronic transitions in the stars atmospheres, such as Hydrogen and Calcium, that occurs in older stars [27]. Astronomers use very complex models with many free parameters and hidden variables to understand the effect of characteristics of the stars and gases of the galaxy on the SED [27].

./img/application/model.png

Figure 21: SED estimated using Leung’s model of the the sample galaxy alongside its observed SED [18].

Leung’s model [18] agreed with the observed SED in that the regions we identified as emissions and absorption bands but failed to predict brightness in other regions of the SED. In particular, large amounts of true emmissions were missed in a low wavelength region (<4500 Angstroms) and a high wavelength region (6500-7500 Angstroms), while the model predicted a large absorption band in the 4100 Angstrom region that did not exist.

5.1.2 Low-frequency wobbles

./img/application/raw_residuals.png

Figure 22: Residuals of the model fit to the SED of the same galaxy [18].

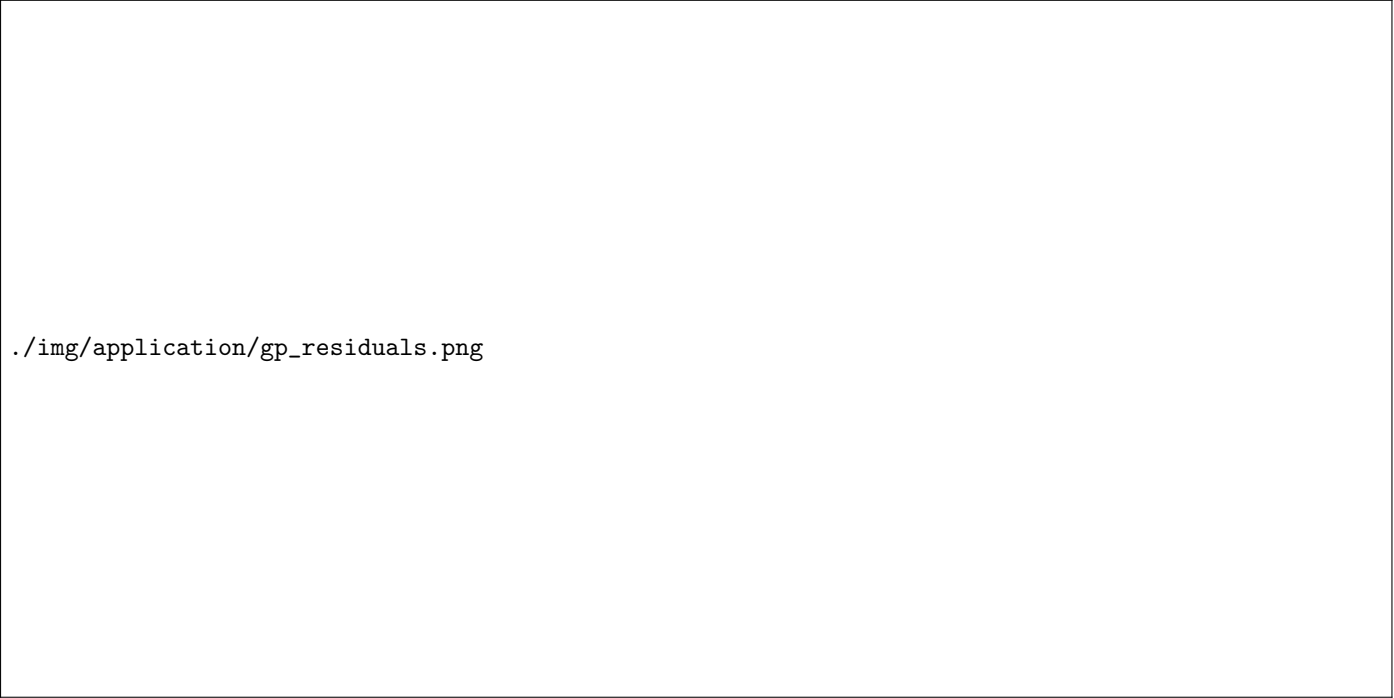
Residuals that behave as zero-mean noise are dependent on how long we observe the galaxy for and are not of interest [27]. However, the two regions identified previously are reflected in the model’s residuals - the residuals oscillate above zero in the low and high wavelength regions, with the exception of the 4100 Angstrom negative residual spike. We refer to these regions as low frequency wobbles because these oscillations occur at a lower frequency than the oscillations around

zero in the remaining SED. SED models exhibiting low-frequency wobbles are problematic to use for inference as they violate the zero-mean noise assumption 2, leading to biased estimates of the model parameters and their uncertainties.

As the source of low-frequency wobbles is not well understood and cannot be incorporated into an SED model [27], we need a method to remove these low-frequency wobbles without introducing identifiability issues by altering or biasing the high-frequency noise that is required for inference (e.g. standard errors in frequentist regression). Signals processing methods would require a fixed frequency cutoff, which does not exist as the amplitude and scale of the low-frequency wobbles vary between galaxies and needs to be learned from the data. Although large non-parametric models (e.g. neural networks) could learn from data, both methods produce point estimates of residuals, the uncertainty of which cannot be propagated through to the SED model parameters and used for inference.

5.2 Methodology

5.2.1 Using GPs for SED residual modelling



./img/application/gp_residuals.png

Figure 23: Remaining residuals after fitting a full GP using SE [18] to the raw residuals of the previous SED model.

GPs are a natural choice for this problem as they can identify the low-frequency wobbles on a per-galaxy basis, whilst doing so in a probabilistic framework that allows the uncertainty in this estimation to be propagated. SE 42 is likely the best kernel for SED modelling because its spectral density converges to zero faster than any other kernel, meaning it ignores high-frequency noise more than other kernels.

Unfortunately, GPs have a naive training and inference cost of $O(n^3)$ and each SED data contains thousands of data points (e.g. the example galaxy has 3000 data points), which makes training and inference on a single galaxy infeasible.

5.2.2 Approximations and kernels considered

We compare two methods of addressing this cubic complexity to enable the use of GPs for SED residual modelling on large numbers of galaxies. SVGP is an approximation-based method that sacrifices accuracy, computational complexity and wall-clock speed in exchange for generality. SVGPs can be used with any kernel and can be applied to any dataset [12]. Conversely, a celerite speedup achieves near-perfect accuracy and significant improvements in computational complexity and wall-clock speed, but can only be used with Matern 3/2 kernels and on particular datasets [7]. Although this dataset meets all the conditions for celerite models (one dimensional and data is sorted by wavelength), using a rougher kernel like Matern 3/2 will capture too much noise compared to SE. Our goal is to determine if the loss in identifiability from using Matern 3/2 outweighs the computational gain from using celerite.

5.3 Results

./img/application/celerite_svgp.png

Figure 24: Comparison of the residuals produced by GPs using different kernels and approximation methods to Leung’s SED model [18]. SVGP was trained using the SE kernel, while celerite was trained using the Matern 3/2 kernel. We used GPy’s SVGP implementation [10] and celerite’s implementation [7] to fit these models.

Matern 3/2 ultimately performed worse on this task because it produced a GP whose expected function draw oscillated too much around zero in high frequency regions, unlike SE which remained at zero in high frequency regions but rose in the low frequency regions. Thus, Matern 3/2 overfitted to the data while SE captured the low frequency wobbles without overfitting.

The variance in SE is much larger than the variance in Matern 3/2 and rise with wavelength.

Kernel	Approximation	Theoretical complexity	Training time (s)	Inference time (s)	Epochs
SE	SVGP	$O(bm^2 + m^3)$	662.48	0.02	2500
Matern 3/2	Celerite	$O(n)$	0.01	0.84	10

Table 1: Comparison of the training and inference times of SVGP and celerite approximations to Leung’s SED model [18]. Theoretical complexities are given in terms of dataset size n , batch size b and number of inducing points m .

SVGP took 65,000 times longer than celerite and required 250 times as many epochs to converge.

5.4 Discussion

5.4.1 Computational cost

In total, SVGP required 15 billion operations to complete its training cycle. This incredible cost was largely caused by SVGP’s sensitivity to the noise in the dataset, which in turn raised the per-epoch cost and number of epochs required to converge.

SVGP’s $O(bm^2 + m^3)$ training complexity appears theoretically attractive due to its lack of dependence on dataset size n . However, the noise of this dataset raised the number of inducing points (min. 256) and batch sizes (1024) required to produce a subset of the data that was representative enough for the model to learn from. This directly raised the number of operations required per epoch to a minimum of 80 million operations, in contrast to celerite’s $O(n)$ training complexity only necessitating 3000 operations per epoch.

Furthermore, the raised m and batch size meant the optimiser required far more epochs to converge on the optimal parameters compared to celerite. Celerite is only optimising two kernel parameters, signal noise σ_s^2 and length scale l , while SVGP has to additionally optimise a full set of variational parameters for the SVGP posterior for each inducing point. Thus, the increase in m necessitated by noise also increased the difficulty of the optimisation problem, which meant that SVGP took 2500 epochs to converge compared to celerite’s ten epochs.

5.4.2 Identifiability

Similar to its behaviour on the toy datasets, Matern 3/2 is a rough process and its expected function draw passes through all the datapoints. This is an undesirable property for SED residual modelling, as it means that the GP will capture both high and low frequency noise. In contrast, SE’s highly limited flexibility disincentivises it from modelling the noise in favour of approaching the large number of points in the low-frequency wobble regions. This is similar to its behaviour on the smooth toy dataset, where SE avoided one stray point so its expected function draw could get closer to two other points. SE’s large variance are another symptom of this behaviour - similar to the rough toy dataset, it has a significantly higher variance than Matern 3/2 because the extreme smoothness requirement forces SE to produce more diverse functions.

5.5 Conclusion

SVGP is not a suitable method for SED residual modelling because of its extreme computational cost. However, SVGP’s batching and use of an iterative optimiser means it lends itself well to GPU processing. SVGP on GPUs have recently become supported by more recent SVGP implementations such as GPyTorch [8] and GPFlow [20], and training SVGP on a GPU using these frameworks could significantly reduce its training time.

Another issue with SVGPs that was not explored in the case study is their inherent inaccuracy compared to celerite or more analytical ELBO methods such as VFE. If GPUs make SVGP computationally feasible, we would need to determine if the accuracy degradation is acceptable for SED residual modelling since a high accuracy degradation could introduce additional errors into the high frequency noise, which would bias the SED model parameters and their uncertainties.

Although celerite was highly computationally performant on this dataset, its inability to use SE means it is not suitable for SED residual modelling. It can only use Matern 3/2’s kernel, and this kernel’s roughness means it overfits to the data and removes both high and low frequency noise. A naive remedy would be to increase the length scale of Matern 3/2 beyond its optimal, increasing its rate of spectral decay 46 to mimic SE’s spectral decay. While this would achieve the desired behaviour of removing low frequency wobbles without removing high frequency noise, this would likely produce worse estimates of SED residuals than SE. Leung [18] empirically determined a non-linear relation between a celerite’s Matern 3/2 length scale ρ and the SE length scale l . He produced a celerite Matern 3/2 kernel which approximated SE’s behaviour well enough to be used for SED residual modelling. Instead of using celerite kernels, Wilson and Nickisch [29] introduced another structured approximation to GPs that take advantage of specific relationships between inducing points and the kernel to achieve linear complexity in low-dimensional datasets.

Conclusion

Analysis of covariance functions Conceptually, reframing smoothness as a single lens that unites length-scale, mean-square (MS) differentiability, and spectral decay proved useful. The Taylor-upcrossing analysis shows how length-scale governs the density of wiggles in typical function draws; shorter scales imply more upcrossings, while longer scales enforce broader, slower variations. In MS space, continuity and differentiability conditions translate directly into constraints on the kernel. For stationary kernels the spectral viewpoint makes the trade visible: fast high-frequency spectral decay yields very smooth sample paths. Together, these perspectives clarify why ostensibly similar kernels behave so differently downstream and proved fundamental in understanding the different performances of Matern 3/2 versus SE in the case study.

Computational efficiency versus generality The work established and disentangled several approaches to making GPs computationally tractable. Stochastic variational GPs (SVGP) buy generality at the price of accuracy and wall-clock time, while structured methods, exemplified by celerite, achieve near-exact linear-time algebra on restricted kernel classes and data layouts. Even though SVGP had a theoretically attractive computational complexity, its training took 65,000 times longer than celerite training and ultimately was too computationally infeasible to recommend for this application. In this case study, practical cost depended more on optimiser dynamics and noise levels than on theoretical computational complexity.

The case study’s objective was to remove correlated, low-frequency ”wobbles” while leaving high-frequency noise intact so downstream inference remains valid. SE’s rapid spectral decay matches that need by resisting high-frequency noise but rising and falling to capture correlated residuals. By contrast, Matérn 3/2’s rougher prior tends to pass through more datapoints and, in this setting, overfits both low-frequency and high-frequency structure.

These tradeoffs exposed a central tension between efficiency and generality that exists in GP modelling. SE avoided enough high-frequency noise to be identifiable, but required a generalised approximation like SVGP that was not computationally feasible. Celerite was computationally feasible for the case study, but it could only work with Matern 3/2 which overfit to the high-frequency noise and produced unidentifiable results.

There are three paths to addressing this tradeoff. Firstly, recent software developments have enabled generalised frameworks such as SVGP to run on GPUs, which could make generalised models practically feasible without needing to address their complexity issues. Although similar successes have been achieved with neural networks, this approach relies on GPUs which are themselves expensive and not always available and do not address the fundamental complexity issues that generalised frameworks have. Secondly, the existing structured approximations can be manipulated to approximate other covariance functions, e.g. Leung’s approximation of SE with celerite [18]. However, these extension methods produce approximations that sometimes differ from the target of the approximation. For example, Leung found that celerite’s approximation of SE produces functions that are more flexible at the tails than SE, which is an issue for applications that require perfect smoothness like SED residual modelling.

New structured approximations The most promising path forward is via new structured approximation methods that retain dimensionality restrictions but can produce more classes of covariance functions. One example of a newer structured approximation framework is ”structured kernel interpolation” (SKI) [29], which has been shown to approximate a wide range of covariance functions with linear complexity in similar one-dimensional time series applications. The major drawbacks are its tendency to produce discontinuous predictions and its reliance on inducing points - our case study demonstrated that inducing point methods are less reliable in high noise scenarios such as SED residual modelling. Evans and Nair [6] exploited a different structure in the covariance matrix rather than using inducing points to overcome these discontinuities, and this technique could readily be adapted to the case study’s setting.

References

- [1] Edwin R. Anderson, Thomas L. Duvall Jr., and Stuart M. Jefferies. “Modeling of Solar Oscillation Power Spectra”. In: *apj* 364 (Dec. 1990), p. 699. DOI: 10.1086/169452.
- [2] Christopher T H Baker. *The numerical treatment of integral equations*. en. Monographs on Numerical Analysis. London, England: Oxford University Press, Jan. 1978, pp. 67, 67.
- [3] Matthias Bauer, Mark van der Wilk, and Carl Edward Rasmussen. *Understanding Probabilistic Sparse Gaussian Process Approximations*. 2017. arXiv: 1606.04820 [stat.ML]. URL: <https://arxiv.org/abs/1606.04820>.
- [4] Thang D. Bui, Josiah Yan, and Richard E. Turner. “A Unifying Framework for Gaussian Process Pseudo-Point Approximations using Power Expectation Propagation”. In: *Journal of Machine Learning Research* 18.104 (2017), pp. 1–72. URL: <http://jmlr.org/papers/v18/16-603.html>.
- [5] Collin Erickson. *GauPro: Gaussian Process Fitting*. R package version 0.2.15.9000, commit 6c7a7a8055506229b6d02650aca960. 2025. URL: <https://github.com/CollinErickson/GauPro>.
- [6] Trefor Evans and Prasanth Nair. “Scalable Gaussian Processes with Grid-Structured Eigenfunctions (GP-GRIEF)”. In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 1417–1426. URL: <https://proceedings.mlr.press/v80/evans18a.html>.
- [7] D. Foreman-Mackey et al. “Fast and Scalable Gaussian Process Modeling with Applications to Astronomical Time Series”. In: *aj* 154 (Dec. 2017), p. 220. DOI: 10.3847/1538-3881/aa9332.
- [8] Jacob R. Gardner et al. *GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration*. 2021. arXiv: 1809.11165 [cs.LG]. URL: <https://arxiv.org/abs/1809.11165>.
- [9] Iosif I Gikhman and Anatoli V Skorokhod. *The theory of stochastic processes I*. en. Classics in mathematics. Berlin, Germany: Springer, Mar. 2004.
- [10] GPy. *GPy: A Gaussian process framework in python*. <http://github.com/SheffieldML/GPy>. since 2012.
- [11] Kohei Hayashi, Masaaki Imaizumi, and Yuichi Yoshida. “On Random Subsampling of Gaussian Process Regression: A Graphon-Based Analysis”. In: *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. Ed. by Silvia Chiappa and Roberto Calandra. Vol. 108. Proceedings of Machine Learning Research. PMLR, 26–28 Aug 2020, pp. 2055–2065. URL: <https://proceedings.mlr.press/v108/hayashi20a.html>.
- [12] James Hensman, Nicolo Fusi, and Neil D. Lawrence. *Gaussian Processes for Big Data*. 2013. arXiv: 1309.6835 [cs.LG]. URL: <https://arxiv.org/abs/1309.6835>.
- [13] Vladimir Joukov and Dana Kulić. “Fast Approximate Multioutput Gaussian Processes”. In: *IEEE Intelligent Systems* 37.4 (July 2022), pp. 56–69. ISSN: 1541-1672. DOI: 10.1109/MIS.2022.3169036. URL: <https://doi.org/10.1109/MIS.2022.3169036>.
- [14] Kallinger, T. et al. “The connection between stellar granulation and oscillation as seen by the Kepler mission”. In: *A&A* 570 (2014), A41. DOI: 10.1051/0004-6361/201424313. URL: <https://doi.org/10.1051/0004-6361/201424313>.
- [15] Sathya Keerthi and Wei Chu. “A matching pursuit approach to sparse Gaussian process regression”. In: *Advances in Neural Information Processing Systems*. Ed. by Y. Weiss, B. Schölkopf, and J. Platt. Vol. 18. MIT Press, 2005. URL: https://proceedings.neurips.cc/paper_files/paper/2005/file/1ec3e7af38e33222bde173fecaf6bfa-Paper.pdf.
- [16] Brandon C. Kelly, Małgorzata Sobolewska, and Aneta Siemiginowska. “A STOCHASTIC MODEL FOR THE LUMINOSITY FLUCTUATIONS OF ACCRETING BLACK HOLES”. In: *The Astrophysical Journal* 730.1 (Mar. 2011), p. 52. ISSN: 1538-4357. DOI: 10.1088/0004-637x/730/1/52. URL: <http://dx.doi.org/10.1088/0004-637X/730/1/52>.
- [17] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG]. URL: <https://arxiv.org/abs/1412.6980>.
- [18] Ho-Hin Leung. “Unveiling the evolutionary history of local post-starburst galaxies through careful Bayesian modelling of their spectra”. PhD thesis. PhD thesis. University of St Andrews, 2025. DOI: 10.17630/sta/1254. URL: <https://research-repository.st-andrews.ac.uk/bitstream/handle/10023/31630/Thesis-Ho-Hin-Leung-complete-version.pdf?sequence=5&isAllowed=y>.
- [19] Haitao Liu et al. “When Gaussian Process Meets Big Data: A Review of Scalable GPs”. In: *IEEE Transactions on Neural Networks and Learning Systems* 31.11 (2020), pp. 4405–4423. DOI: 10.1109/TNNLS.2019.2957109.
- [20] Alexander G. de G. Matthews et al. “GPflow: A Gaussian process library using TensorFlow”. In: *Journal of Machine Learning Research* 18.40 (Apr. 2017), pp. 1–6. URL: <http://jmlr.org/papers/v18/16-537.html>.
- [21] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Nov. 2005. ISBN: 9780262256834. DOI: 10.7551/mitpress/3206.001.0001. eprint: <https://direct.mit.edu/book-pdf/2514321/book\9780262256834.pdf>. URL: <https://doi.org/10.7551/mitpress/3206.001.0001>.

- [22] Klaus Ritter, Grzegorz W. Wasilkowski, and Henryk Wozniakowski. “Multivariate integration and approximation for random fields satisfying Sacks-Ylvisaker conditions”. In: *The Annals of Applied Probability* 5.2 (May 1995). DOI: 10.1214/aoap/1177004776.
- [23] George B. Rybicki and William H. Press. “Class of Fast Methods for Processing Irregularly Sampled or Otherwise Inhomogeneous One-Dimensional Data”. In: *Phys. Rev. Lett.* 74 (7 Feb. 1995), pp. 1060–1063. DOI: 10.1103/PhysRevLett.74.1060. URL: <https://link.aps.org/doi/10.1103/PhysRevLett.74.1060>.
- [24] Edward Snelson and Zoubin Ghahramani. “Sparse Gaussian Processes using Pseudo-inputs”. In: *Advances in Neural Information Processing Systems*. Ed. by Y. Weiss, B. Schölkopf, and J. Platt. Vol. 18. MIT Press, 2005. URL: https://proceedings.neurips.cc/paper_files/paper/2005/file/4491777b1aa8b5b32c2e8666dbe1a495-Paper.pdf.
- [25] Michalis Titsias. “Variational Learning of Inducing Variables in Sparse Gaussian Processes”. In: *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*. Ed. by David van Dyk and Max Welling. Vol. 5. Proceedings of Machine Learning Research. Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA: PMLR, 16–18 Apr 2009, pp. 567–574. URL: <https://proceedings.mlr.press/v5/titsias09a.html>.
- [26] G. E. Uhlenbeck and L. S. Ornstein. “On the Theory of the Brownian Motion”. In: *Phys. Rev.* 36 (5 Sept. 1930), pp. 823–841. DOI: 10.1103/PhysRev.36.823. URL: <https://link.aps.org/doi/10.1103/PhysRev.36.823>.
- [27] Vivienne Wild. “Fitting Galaxy Spectra 101”. Unpublished manuscript, created 4 July 2025. 2025.
- [28] Christopher Williams and Matthias Seeger. “Using the Nyström Method to Speed Up Kernel Machines”. In: *Advances in Neural Information Processing Systems*. Ed. by T. Leen, T. Dietterich, and V. Tresp. Vol. 13. MIT Press, 2000. URL: https://proceedings.neurips.cc/paper_files/paper/2000/file/19de10adbaa1b2ee13f77f679fa1483a-Paper.pdf.
- [29] Andrew Wilson and Hannes Nickisch. “Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP)”. In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 1775–1784. URL: <https://proceedings.mlr.press/v37/wilson15.html>.