FYP

Sean White

January 2025

1 GR intro

Gravitational waves (GWs) are small fluctuations of spacetime that propagate at the speed of light. As described in *Gravitational Waves, Vol. 1* by Maggiore [4, p. 29], SA: better to re-write this in your own words

> "In this setting, the definition of GWs is relatively clear: the background spacetime is flat, and the small fluctuations around it have been called 'gravitational waves'. The term 'waves' is justified by the fact that, in a suitable gauge, $h_{\mu\nu}$ indeed satisfies a wave equation."

This is formally appproached by expanding the Einstein equations around the flat Minkowski metric η_{ab}

$$g_{ab} = \eta_{ab} + \epsilon\, h_{ab}, \qquad \epsilon \ll 1, \qquad \eta_{ab} = \mathrm{diag}(-1, 1, 1, 1), \tag{1}$$

where h_{ab} represents the perturbation. Using linear perturbation theory in Lorenz gauge, the Einstein field equations simplify to

$$\Box \bar{h}_{ab} = \frac{-16\pi G}{c^4} T_{\mu\nu}. \tag{2}$$

However, we are interested in this equation outside of the source (i.e, stress-energy tensor $T_{\mu\nu} = 0$) therefore we are left with

$$\Box \bar{h}_{ab} = 0, \tag{3}$$

with $\Box$ denoting the flat spacetime d'Alembertian operator.

Although h_{ab} initially has 10 independent components (as a symmetric 4×4 tensor), 8 of these correspond to gauge freedom and constraints. After imposing the Lorenz and transverse-traceless (TT) gauge conditions, only two physical degrees of freedom remain: the plus (h_+) and cross ($h_\times$) polarizations [4, Sec. 1.2].

This wave equation admits plane-wave solutions. For a wave propagating in the z-direction, the TT-gauge form of the perturbation is:

$$h_{ab}^{(\mathrm{TT})} \propto \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & h_+ & h_\times & 0 \\ 0 & h_\times & -h_+ & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} e^{i(kz - \omega t)}. \tag{4}$$

Leading-Order Power Emission by Gravitational Waves

In general, the power emitted by a radiative field can be expressed schematically as

$$\dot{E} = \sum_{\ell=0}^{\infty} \left\langle \left| \left(\frac{\partial}{\partial t} \right)^{\ell+1} P_\ell(t) \right| \right\rangle. \tag{5}$$

Here, $P_\ell(t)$ represents the multipole moments of the radiating source. The $\ell = 0$ term corresponds to the monopole moment, which in the gravitational case is the total mass of the system. Assuming mass is conserved, this term vanishes.

The $\ell = 1$ term represents the dipole moment, which is also zero in the gravitational case due to conservation of linear momentum. Therefore, the leading-order contribution to gravitational-wave emission arises from the $\ell = 2$ term, known as the quadrupole.

This gives the leading-order expression for the power emitted in gravitational waves:

$$\dot{E} = \frac{G}{5\,c^5} \left\langle \dddot{Q}_{ij}\, \dddot{Q}^{ij} \right\rangle, \tag{6}$$

where Q_{ij} is the mass quadrupole moment of the source. It is related to the mass moment M_{ij} by

$$Q_{ij} := M_{ij} - \frac{1}{3}\delta_{ij} M^k_{\ k}, \tag{7}$$

where $M_{ij} = \int d^3x\, T^{00}\, x_i x_j$.

A Simple Two-Mass Inspiral Model

I took this example and summarised it from Sarp's notes, leads nicely to mismatch

We will now consider a generic problem as illustrated in Ref. [1] where two point masses, $m_1 \geq m_2$, are in a quasi-circular orbit of separation R, each at distances r_1 and r_2 from their common center of mass (CoM), with $R = r_1 + r_2$. We place the orbit in the x–y plane so that mass 1 moves on $\mathbf{x}_1(t) = r_1(\cos\Omega t, \sin\Omega t)$ and mass 2 on $\mathbf{x}_2(t) = r_2(\cos(\Omega t + \pi), \sin(\Omega t + \pi))$. The system's orbital frequency is Ω. A short calculation yields

$$Q^{ij} = 4\Omega^3 \left(m_1 r_1^2 + m_2 r_2^2\right) \begin{pmatrix} \sin(2\Omega t) & -\cos(2\Omega t) \\ -\cos(2\Omega t) & -\sin(2\Omega t) \end{pmatrix}. \tag{8}$$

Introducing the reduced mass $\mu = m_1 m_2/(m_1 + m_2)$, we get that the $\ell = 2$ power emission is

$$\dot{E}_{\ell=2} \;=\; \frac{32}{5}\,\frac{G}{c^5}\,\Omega^6\,\mu^2\,R^4. \tag{9}$$

Both Ω and R are functions of time, but they evolve on a time scale much longer than the orbital time scale and so when averaging over orbits we approximate them as constant. Applying Kepler's law, $\Omega^2 = GM/R^3$, and defining $\omega = 2\Omega$ as the GW frequency, we find

$$\dot{E}_{\ell=2} \;=\; \frac{32}{5}\,\frac{c^5}{G}\left(\frac{G\,\mathcal{M}\,\omega}{2c^3}\right)^{10/3}, \quad \text{where} \quad \mathcal{M} \;=\; \mu^{3/5}\,M^{2/5} \tag{10}$$

is known as the chirp mass where $M = m_1 + m_2$ is the total mass.

Could maybe mention radiation time scale being much smaller than orbit timescale but I don't see what this will add SA: You already kind of mention this below Eq. (9), so maybe provide slightly more detail there.

$\dot{E}$ represents the rate at which the system loses energy due to gravitational-wave emission. This energy loss causes the binary orbit to shrink and the GW frequency to increase, with the chirp mass $\mathcal{M}$ and frequency ω capturing the key features of this inspiral.

SA: From here you should go on to derive the form of a simple waveform in first time domain then frequency domain. Then you can go into the Mismatch

Introducing the Waveform Mismatch

This simplified two-mass, circular-orbit model captures the main physical components of the model such as the "chirp behaviour" (frequency of GW increasing as the orbit shrinks). However we want to quantify how neglecting for example eccentricity of the orbit and the spin vectors of the masses effects the waveform. We compare the simplified model to more complete waveforms using the waveform mismatch discussed in [5] and [3]. The mismatch between two signals is defined by

$$\mathcal{M} \;=\; 1 \;-\; \max_{\lambda_{\mathbf{m}}} \frac{\langle h_{\text{simple}},\, h_{\text{accurate}}\rangle}{\sqrt{\langle h_{\text{simple}}, h_{\text{simple}}\rangle\,\langle h_{\text{accurate}}, h_{\text{accurate}}\rangle}}, \tag{11}$$

where $\langle \cdot, \cdot \rangle$ is the inner product of both GW, and we maximise over a set of (intrinsic or extrinsic) model parameters $\lambda_{\mathbf{m}}$. A mismatch $\mathcal{M} \ll 1$ indicates that our simple waveform faithfully represents the physical signal, whereas larger mismatches highlight missing physics (e.g. not circular orbit).

2 Background

This section was written ages ago and needs to be updated but I think could still be worth time just going over how bayesian methods are currently used in GW and how making a mismatch predictor model may help moving forward

The gold standard gravitational waveforms are obtained by numerical relativity simulations which involve directly solving Einstein's equations of general relativity which is massively computationally expensive. For this reason, only several thousand simulations are currently available. As a result, GW models rely on analytical or semi-analytical prescriptions that are calibrated to the numerical relativity simulations.

Each of these modelling approaches will incur a certain amount of error. We quantify this error as the mismatch between the "true" relativity simulation of a gravitational waveform and the analytic model for the waveform.The mismatch [5] varies between 0, signifying that the model and the true waveform are identical (up to an overall amplitude rescaling), and 1, meaning that the two are completely orthogonal. To begin we will delve into the methods used before to "combine" these models and then discuss the new proposed method. This will lead us onto my project.

Methods Used Before

Standard Method

We build a distribution based off of each of the 3 models. We then combine these distributions with equal weight to form a new total distribution from which we make our predictions from.

Evidence-Informed Method

This method again builds a distribution based off of each of the 3 models. The distributions are then combined with weights which are determined by how well the model explains the observed data. From this new distribution predictions are made.

New Proposed Method: Numerical Relativity Informed Method

Numerical Relativity Informed Method

This method builds a distribution based off of each of the 3 models. Then the mismatch (difference between NR simulations and model) is calculated. We then find the ratios of the mismatch between each model, for example:

$$\frac{\text{mismatch(model 1)}}{\text{mismatch(model 2)}}.$$

A distribution using the ratio of mismatch of each model is built. This distribution is used to determine the probability of selecting each model in each region of the parameter space. Predictions are then made using the model selected in that region of the parameter space using the ratio distribution. This selection of the model is probabilistic and so the best model in a certain region may be chosen 85% of the time and the second best 10% etc etc.

Comparison

The Standard Method and the Evidence-Informed Method build a distribution from which we make our predictions. The Numerical Relativity Informed method builds a distribution which helps us choose the best model in that

parameter space "most of the time"".

Now the issue is that to calculate the mismatch for the models needed in the NR informed method, we must have gravitational waveforms generated from NR simulations. This as we discussed is a computationally expensive process and is often not feasible. The goal of my project is to build a Gaussian Process Regression model to model the mismatch of the model's to the NR simulations.

3 Gaussian Process Regression GPR Background

3.1 Introduction and Roadmap

In the following subsections, I discuss the foundational concepts of Gaussian Process Regression **(GPR)** in four main steps:

1. **Gaussian Processes:** Section 3.2 introduces Gaussian Processes and explains how their priors and posteriors are constructed from finite sets of points.
2. **Kernel Functions:** In Section 3.3, I explore how kernels encode assumptions about smoothness, periodicity, and structural properties of the underlying function.
3. **Noise Modeling:** Section 3.4 covers several approaches for incorporating observational noise into the GP framework.
4. **Hyperparameter Optimization:** Finally, in Section 3.5, I discuss how kernel and noise hyperparameters influence the posterior and how they can be optimized to improve model performance.

3.2 Gaussian Proces Regression Background

Definition of a Gaussian Process

A Gaussian Process **(GP)** defines a probabilistic model over all possible functions rather than assuming a single function to be true

$$f(X) \sim \mathcal{GP}(\mu(X), k(X, X')).$$

where $\mu(X)$ is the mean function, specifying the expected function value at each X:

$$\mu(X) = E[f(X)],$$

$k(X, X')$ is the covariance function (kernel), encoding the relationships between function values at different points:

$$k(X, X') = \mathrm{Cov}(f(X), f(X')).$$

Since the input space is continuous, the GP represents an infinite-dimensional distribution. In practice, we approximate the process by evaluating the GP at a finite set of inputs. These function values are then assumed to follow a multivariate normal Gaussian distribution.

Mathematically, for a finite set of input points

$$X = \{X_1, X_2, \ldots, X_n\},$$

the corresponding function values

$$f = \{f(X_1), f(X_2), ..., f(X_n), \}$$

follow a multivariate normal distribution

$$f \sim \mathcal{N}(\mu(X), K(X, X)).$$

Each sample from this multivariate distribution represents a function evaluated at n different points.

The Prior Distribution

Before observing any data, we assume a joint Gaussian distribution over both training and test points. Let X denote training inputs and X_* test inputs. The joint prior over their function values is

$$\begin{bmatrix} f(X) \\ f(X_*) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(X) \\ \mu(X_*) \end{bmatrix}, \underbrace{\begin{bmatrix} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}}_{\mathcal{C}=\text{Covariance Matrix}} \right).$$

After accounting for the mean, the kind of structure that can be captured by a GP model is entirely determined by its kernel. The kernel determines how the model generalizes, or extrapolates to new data.There are many possible choices of covariance function, all encoding different shapes.

Adding Data: Prior to Posterior

One of the key strengths of Gaussian Processes is that, given observations at training inputs X, we can make predictive inferences about the function value at any new test location x_*. By applying the standard conditional Gaussian formulas (see appendix A for the full derivation), the posterior distribution of $f(x_*)$ given $\{X, f(X)\}$ is Gaussian and given by:

$$p\big(f(x_*) \mid f(X), X, \mu(\cdot), k(\cdot,\cdot)\big) \;=\; \mathcal{N}\Big(m(x_*),\; \sigma^2(x_*)\Big),$$

where

$$m(x_*) \;=\; \mu(x_*) \;+\; k(x_*, X)\, k(X,X)^{-1}\big[f(X) - \mu(X)\big],$$
$$\sigma^2(x_*) \;=\; k(x_*, x_*) - k(x_*, X)\, k(X,X)^{-1}\, k(X, x_*).$$

In these expressions:

- $\mu(\cdot)$ is the mean function (I take this to be zero since we centre the data prior to prediction),
- $k(X, X)$ is the covariance matrix among the observed training points,
- $k(x_*, X)$ is the vector of cross-covariances between the test point x_* and the training inputs,
- $k(x_*, x_*)$ is the prior variance at the test point itself.

We now have an analytic function that can be evaluated to find the mean function value and variance at any input point. This is a very useful property that Gaussian Processes possess. Figure 2 shows how we update our distributions based on the training points. We initially plot samples taken from the prior distribution (Equation 3.2). After conditioning this distribution on one training point and obtaining the new predictive posterior distribution (Equation 3.2), we see that the predictive mean passes exactly through the training point, has small variance around it, and then fans out farther away. Conditioning on two training points at opposite ends of our input domain creates an ellipse-shaped credible interval whose largest radius appears midway between the two training points. With more training points, the predictions align progressively closer to the true function and the variance becomes smaller.

Note that this example uses a one-dimensional, noise-free function and a basic RBF kernel with fixed hyperparameters ($\ell = 1$, $\sigma^2 = 0.5$). In practice the choice of Kernel function plays a pivotal role in the assumptions we make about the shape and general behaviour of our model. In the next section we examine the different kernel choices available and the assumptions that each kernel encodes about our function structure, such as smoothness and periodicity.

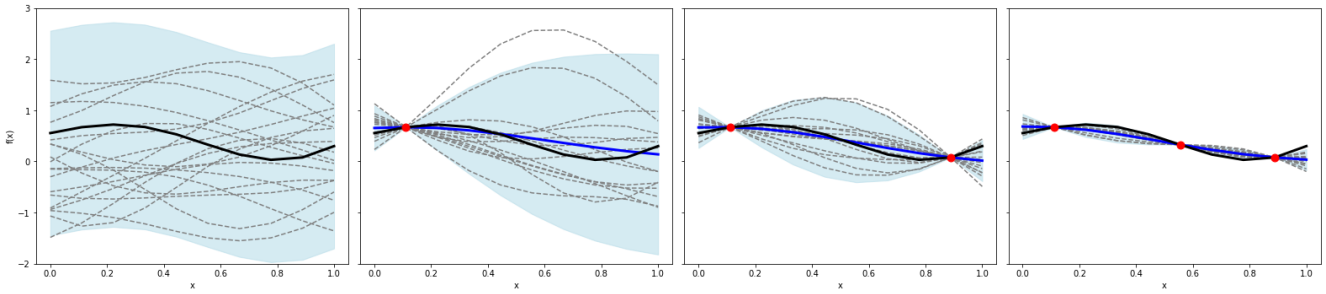


Figure 1: Prior Distribution

Figure 2: 1D Gaussian Process Regression: Prior to Posterior. This sequence shows how the GP prior transforms into a posterior as more data points are added. The RBF kernel was used with hyper-parameters: $l = 1$, $\sigma^2 = 0.5$.The black line represents the true function. The blue is the mean of each posterio distribution. The light blue shaded region is the credible interval and the grey lines are the samples drawn from each posterior/prior.

3.3 Kernel Functions

In Section 3.2, we derived the posterior mean and variance (Equations 3.2 and 3.2), both of which depend massively with choice of kernel function. The kernel represents our assumptions about how function values depend on different inputs across different inputs.

"From a slightly different viewpoint it is clear that in supervised learning the notion of similarity between data points is crucial; it is a basic similarity assumption that points with inputs x which are close are likely to have similar target values y, and thus training points that are near to a test point should be informative about the prediction at that point. Under the Gaussian process view it is the covariance function that defines nearness or similarity." [6, p. 79]

In other words, the kernel function enforces how strongly different input values effect each of their outputs . [6] divide kernels into two major sub-groups, stationary kernels and non-stationary kernels. Stationary kernels depend only on the relative (often radial) distance between inputs $\|x - x'\|$ and are invariant to translations in the input domain. By contrast, non-stationary kernels depend explicitly on the absolute values of x and x', allowing the function's properties—such as smoothness or amplitude—to vary across the domain. For more detailed discussion on building, combining, and customizing these kernels, see [2] and [6, Ch. 4].

In Table 1, we provide an overview of several common kernel types, showing both their functional form and samples drawn from the corresponding GP priors. Although each kernel imparts a distinct structure on the functions (e.g., local variation, multi-scale behavior, periodicity, or linearity), all have hyperparameters that significantly shape the resulting GP. In the following subsections, we examine these kernels and their hyperparameters in more detail.

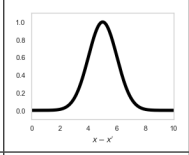
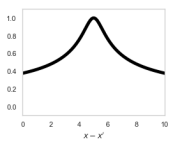
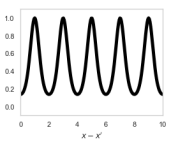
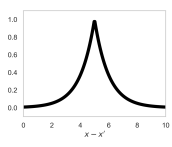
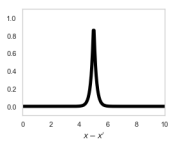
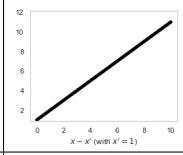
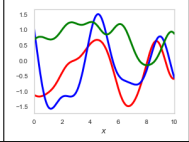
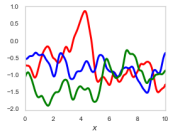
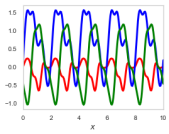
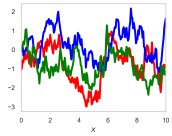
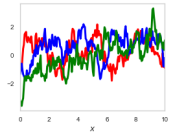
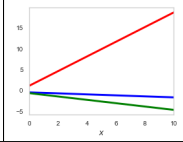
Kernel name:	**RBF (SE)**	**Rational Quadratic**	**Periodic**	**Matern**	**Laplace**	**Linear (Dot Product)**
Plot of $k(x, x')$:						
GP Prior Samples:						
Structure type:	Local variation	Multi-scale local variation	Repeating structure	Rough to smooth	Rougher variation	Linear functions

Table 1: Visual comparison of common kernel functions and their effect on Gaussian process priors. Each column shows the kernel shape $k(x, x')$, samples from the corresponding GP prior, and a summary of the structure it imposes. All kernels were evaluated using a lengthscale parameter $\ell = 1$ (except where noted). For the Matern kernel, $\nu = 0.5$; Laplace kernel, $\gamma = 6$; Rational Quadratic kernel, $\alpha = 0.25$; and Periodic kernel, period $p = 2$.

In this section going to do a deep dive on each kernel and how hyper-parameters effect each one. Just pick main ones. depending on graphs might group some together. Exhausitive work here is needed I think otherwise too many questions later about parameters etc. I do this later at the minute in Hyper-param optimisatinbut this flow is more logical I think

Note: When implementing we multiply every kernel by a constant hyper-parameter that represents the variance. In this section we keep this variance constant at 1 since without noise in the data the variance only effects the predicted variance which increases directly proportionally to this hyper-parameter.

Radial Basis Function (RBF) Kernel

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

This kernel assumes smooth and infinitely differentiable functions, modeling local variations.

Credible Interval Behavior: In Gaussian Process regression, the predictive variance at a new input x^* is given by

$$k(x^*, x^*) - k(x^*, X)(K_{XX}^{-1})k(X, x^*).$$

With an RBF kernel, $k(x^*, X)$ will be near-zero if x^* is farther than a few ℓ's away from all training inputs X. For small ℓ, this situation happens frequently – any point outside a tiny neighborhood of training data effectively has $k(x^*, X) \approx 0$, so its predictive variance is roughly

$$k(x^*, x^*) = \sigma_f^2$$

(the prior variance). Thus, the 95% credible interval reverts to roughly $\pm 2\sigma_f$ in large gaps or outside the data range. For large ℓ, on the other hand, $k(x^*, X)$ remains sizable over a much broader range, reducing the variance term. The uncertainty only approaches the prior level far outside the data (several ℓ's away). In effect, a large ℓ flattens the covariance function so that the GP "remembers" the training set far out, maintaining narrower error bars over a wider domain.

Rational Quadratic Kernel

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

This kernel can be seen as a scale mixture of RBF kernels, allowing for multi-scale behavior.

Periodic Kernel

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

This kernel models repeating structures with period p.

Matern Kernel

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

The Matern kernel allows for controlling the smoothness of functions via the parameter ν.

Laplace (Exponential) Kernel

$$k(x, x') = \sigma_f^2 \exp\left(-\gamma|x - x'|\right)$$

Equivalent to the Matern kernel with $\nu = \frac{1}{2}$, this kernel models rougher functions.

Linear (Dot-Product) Kernel

$$k(x, x') = \sigma_b^2 + x^\top x'$$

This kernel grows with the similarity (inner product) between inputs, and it allows the function to vary globally. Since it depends directly on the values of x and x', not just their difference, it is non-stationary. It is particularly useful for modeling linear trends.

3.4 Handling Noise in our Data

So far, our discussion has assumed noise-free observations. However, real-world data is rarely clean—measurements often include some form of uncertainty. To make our Gaussian Process models more realistic and applicable, we now explore how to incorporate noise into the GP framework.

We model noise by assuming that the observations y are related to the latent function values f through additive Gaussian noise:

$$y = f + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_n^2 I),$$

where σ_n^2 is the noise variance. We now have:

$$f \sim \mathcal{N}(0, K)$$

and

$$y \sim \mathcal{N}(0, K + \sigma_n^2 I) \tag{12}$$

which leads to a revised posterior:

$$m(f_*) = K_*^T (K + \sigma_n^2 I)^{-1} y,$$
$$\mathrm{Var}(f_*) = K_{**} - K_*^T (K + \sigma_n^2 I)^{-1} K_*.$$

There are three common approaches to handling noise in GPR:

- **Noise as a Hyper-parameter (Homoscedastic Noise)**: Here, we assume that all observations have the same level of noise, i.e., the noise is constant throughout the dataset:

$$\sigma_i^2 = \sigma_n^2 \quad \forall i.$$

This leads to a covariance matrix modified by adding a constant term to the diagonal:

$$K(X,X) + \sigma_n^2 I = \begin{bmatrix} k(x_1,x_1) + \sigma_n^2 & \cdots & k(x_1,x_n) \\ \vdots & \ddots & \vdots \\ k(x_n,x_1) & \cdots & k(x_n,x_n) + \sigma_n^2 \end{bmatrix}.$$

The noise variance σ_n^2 is treated as a learnable hyper-parameter during model training.

- **Known Noise (Heteroscedastic Noise)**: In this case, the noise variance changes across the input space—some observations are noisier than others. If we know the individual noise variances σ_i^2 for each training input x_i, we incorporate them by adding a diagonal noise matrix to the kernel:

$$K(X,X) + \Sigma = \begin{bmatrix} k(x_1,x_1) + \sigma_1^2 & \cdots & k(x_1,x_n) \\ \vdots & \ddots & \vdots \\ k(x_n,x_1) & \cdots & k(x_n,x_n) + \sigma_n^2 \end{bmatrix},$$

where $\Sigma = \text{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2)$.

- **Monte Carlo Sampling of Noise**: This technique can be used with either homoscedastic or heteroscedastic noise. Instead of modifying the kernel matrix directly, we explicitly simulate noise. We assume:

$$\epsilon_i \sim \mathcal{N}(0, \sigma_i^2),$$

and create new noisy training sets by adding a sampled noise value to each function value:

$$y_i^{(s)} = f_i + \epsilon_i^{(s)}.$$

Repeating this sampling multiple times allows us to generate multiple noisy versions of the dataset. We then average predictions over these Monte Carlo samples to better capture the uncertainty introduced by noise.

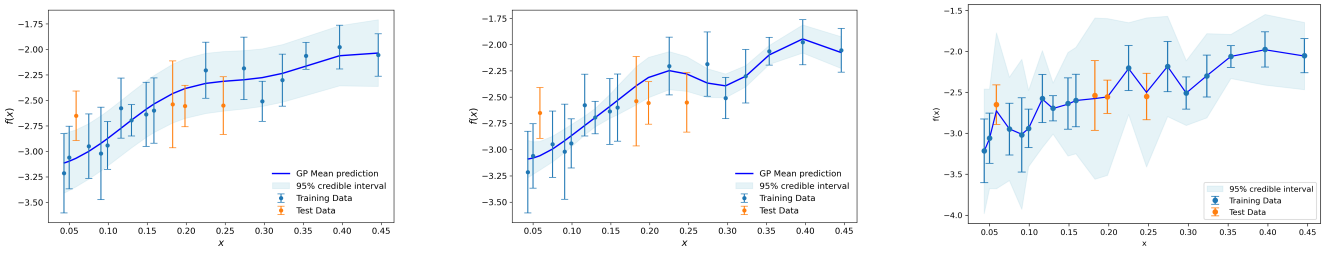


Figure 3: Left: A white kernel is used to add the noise as a hyper-parameter which is optimised (homoscedastic noise). Middle: We update our covariance function with the true variance at the training points (heteroscedastic noise). Right: Result of Monte Carlo sampling of normally distributed noise at each training point (heteroscedastic noise).

Add a justification as to the shapes. Why the larger and different confidence intervals

3.5 Hyper-parameters

Having now incorporated both prior structure (via kernels) and observational noise, the next step is to fine-tune the model by optimizing its hyperparameters—such as the kernel lengthscale, amplitude, and noise variance—based on

the observed data. Lets examine how the choice of these hyper-parameters effect our mean and our 95 % credible interval

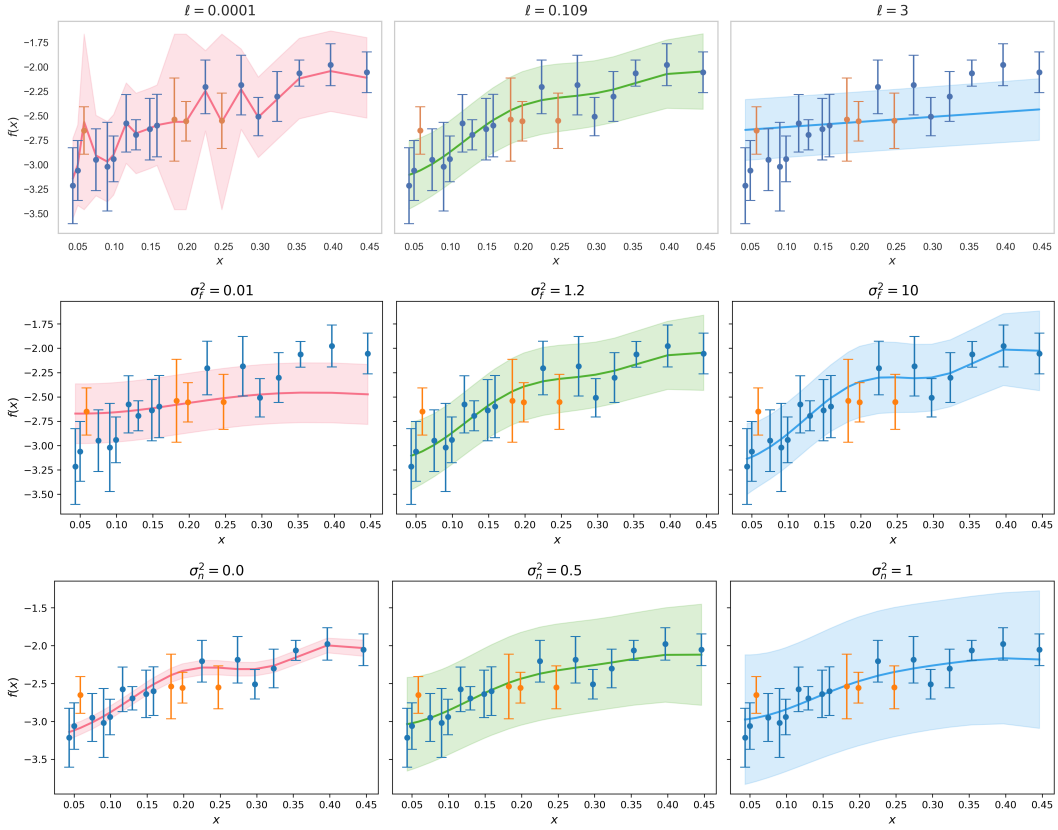


Figure 4: This figure outlines the effect of different RBF kernel hyperparameters on Gaussian Process Regression predictions.For each graph the mean is plotted along with the 95% credible region. Each hyper-parameter is set at $\ell = 0.109$, $\sigma_f^2 = 1.2$ and $\sigma_n^2 = 0.15$ unless explicitly changed

Above graph is misaligned

It is evident that the choice of kernel hyperparameters has a significant impact on the shape of the posterior mean and its uncertainty. The RBF kernel with noise is given by:

$$k(x, x') = \sigma_f^2 \exp\left(-\frac{(x - x')^2}{2\ell^2}\right) + \sigma_n^2 I,$$

where ℓ controls the length scale, σ_f^2 represents the signal variance, and σ_n^2 denotes the noise level.

From Figure 4, we observe that a very small length scale $\ell \approx 0$ leads to overfitting: the GP closely tracks the training data, with sharp fluctuations between points. This occurs because the kernel's covariance rapidly decays with distance, making function values at different inputs nearly uncorrelated unless they are extremely close. As a result, the model fits the noise and exhibits a jagged appearance. In contrast, a large length scale such as $\ell \approx 3$ causes over-smoothing. Distant points remain highly correlated, producing a nearly linear mean function that fails to capture local variations. A moderate length scale ($\ell = 0.109$ in this case) provides a balance—producing a smooth yet flexible mean function that captures trends in the data without overfitting.

Regarding the signal variance σ_f^2, we find that a very small value (e.g., $\sigma_f^2 \approx 0.01$) causes the GP mean to flatten, as the prior assumes the function to vary very little. As the variance increases (e.g., $\sigma_f^2 = 1.2$ and $\sigma_f^2 = 10$), the credible intervals widen slightly, but the overall mean function shape remains relatively stable. This suggests that posterior inference over σ_f^2 may exhibit considerable uncertainty, a point we explore further using MCMC sampling in Section 4.1.

Lastly, varying the noise level σ_n^2 primarily affects the width of the credible interval, while the mean function

remains largely unchanged. As σ_n^2 increases, the model accounts for higher observation noise, and uncertainty in the predictions increases accordingly. This relationship appears approximately linear in the visualizations.

Learning Hyperparameters by Maximising the Log Marginal Likelihood

Now that we have seen how the choice of hyperparameters affects the posterior distribution, we aim to find the optimal hyperparameters that allow the model to best explain the observed data. This is done by maximising the *log marginal likelihood*, as described in [6].

From Equation 12, we have:

$$y \sim \mathcal{N}(0, K + \sigma_n^2 I)$$

where K is the kernel matrix computed from the training inputs X, and σ_n^2 is the noise variance.

This implies that the marginal likelihood (i.e., the probability of the observed outputs y given the inputs X and hyperparameters θ) is given by the multivariate Gaussian density:

$$p(y \mid X, \theta) = \frac{1}{(2\pi)^{n/2}|K_y|^{1/2}} \exp\left(-\frac{1}{2} y^\top K_y^{-1} y\right)$$

where $K_y = K + \sigma_n^2 I$.

Taking the logarithm of this expression yields the *log marginal likelihood*:

$$\log p(y \mid X, \theta) = -\frac{1}{2} y^\top K_y^{-1} y - \frac{1}{2} \log |K_y| - \frac{n}{2} \log 2\pi \tag{13}$$

Our goal is to maximise this log marginal likelihood with respect to the hyperparameters θ, which typically includes the kernel lengthscale, signal variance, and noise variance. Once optimal values are found, we can use them to make accurate posterior predictions.

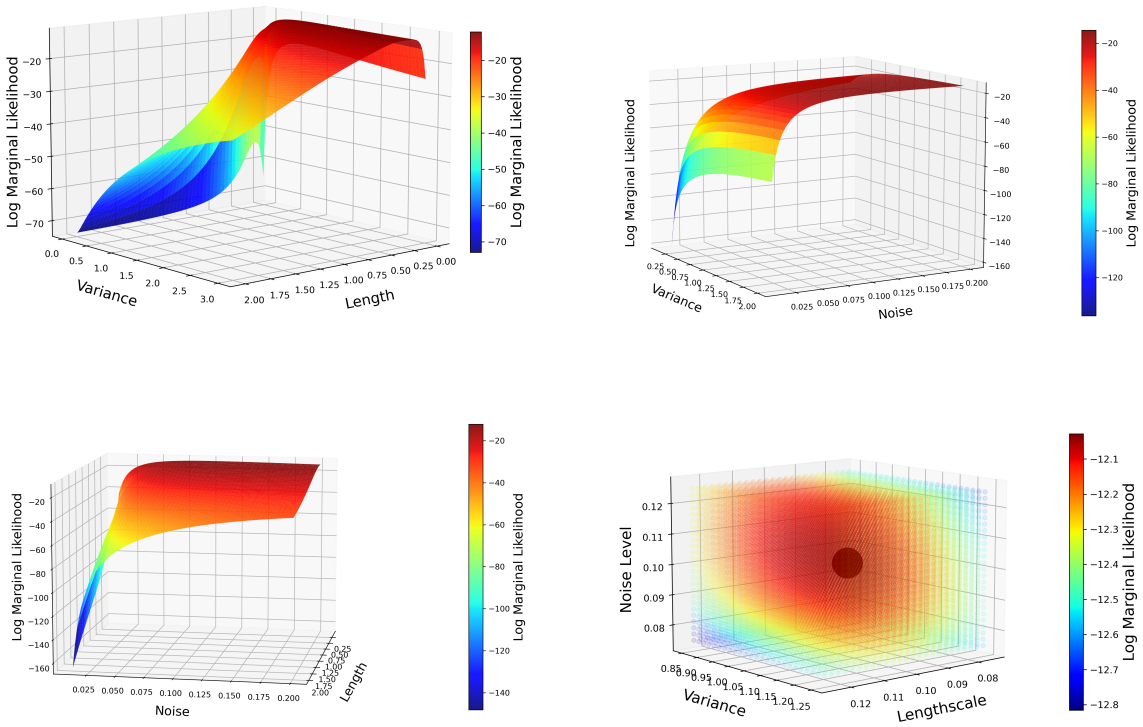


Figure 5: For a GPR with noise we are forced to optimise a length hyper-parameter, a variance hyper-parameter and a noise hyper-parameter. Here we have kept one parameter constant on each graph and compared the log likelihood space varying the other two parameters. In graph one the noise is set at 0.1, in graph 2 the length is set at 0.5, in graph 3 the variance is set at 1.5. In the final graph we plot a 3-dimensional scatter plot and illustrate the point estimate given by the optimisation algorithm as the black dot.This point is located at ($\sigma^2 = 1.16, l = 0.109$ noise= 0.105)

3.5.1 Cross Validation

4 Quantifying Uncertainty and Evaluating Model Accuracy

4.1 Monte Carlo Markov Chain (MCMC): A Posterior over Hyperparameters

Why MCMC and the Mathematics Behind It

We have discussed the role of kernels and how their hyperparameters affect the posterior distribution. From Figures 4 and 5, we observed that when optimising hyperparameters, there is not necessarily a single "best" solution, but rather a region of valid values that explain the data well.

Up to this point, the models we considered have relied on point estimates—selecting the hyperparameters that maximise the log marginal likelihood and using them for predictions. However, this approach ignores the uncertainty between hyperparameters that yield similarly high log-likelihood scores. To address this uncertainty, we adopt a Bayesian perspective by using Markov Chain Monte Carlo (MCMC) methods to sample from the posterior distribution over hyperparameters. This approach allows us to move beyond point estimates and instead capture a full distribution that reflects uncertainty and variability in the hyperparameters. By leveraging this posterior distribution, we gain a more comprehensive understanding of the model's behaviour. It allows for more robust predictions, better uncertainty quantification, and improved decision-making—especially in cases where multiple hyperparameter settings are plausible.

When getting a point-estimate we maximised the log likelihood from equation 13. Now we want to build a distribution over the hyper-parameters. We have that:

$$p(\theta \mid \mathbf{y}, X) = \frac{p(\mathbf{y} \mid X, \theta)\, p(\theta)}{p(\mathbf{y} \mid X)}$$

where:

- $p(\mathbf{y} \mid X, \theta)$ is the likelihood,
- $p(\theta)$ is the prior over hyperparameters,
- $p(\mathbf{y} \mid X)$ is the marginal likelihood, acting as a normalising constant.

Since $p(\mathbf{y} \mid X)$ is often intractable, we sample from the unnormalised posterior using MCMC:

$$p(\theta \mid \mathbf{y}, X) \propto p(\mathbf{y} \mid X, \theta)\, p(\theta)$$

Using MCMC, we generate samples $\{\theta^{(s)}\}_{s=1}^{S} \sim p(\theta \mid \mathbf{y}, X)$ from this posterior.

Implementing MCMC

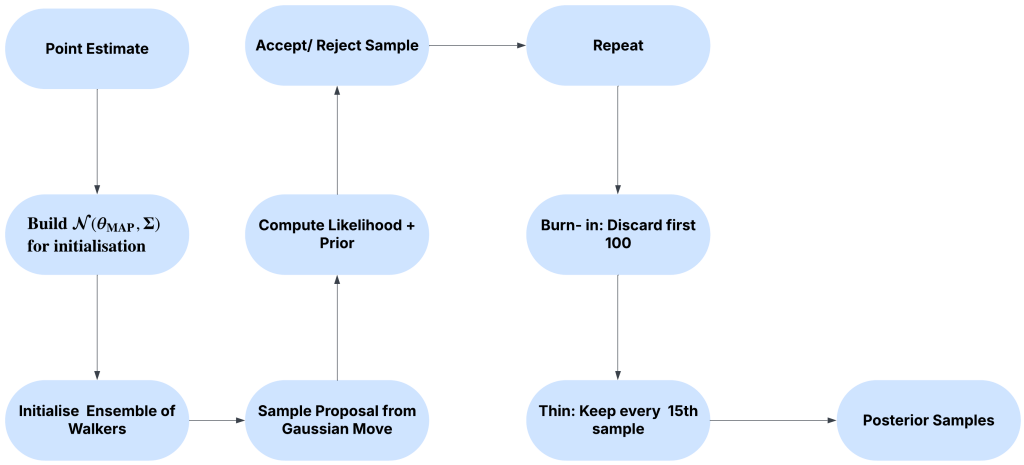


Figure 6: Overview of the MCMC sampling procedure for Gaussian Process hyperparameter inference. This pipeline samples from the posterior $p(\theta \mid \mathbf{y}, X)$ using a Metropolis-Hastings Gaussian proposal and an ensemble of walkers.

Before starting MCMC, we must decide which model we want to build the hyperparameter posterior for. This involves selecting one of the six kernels outlined in Section 3.3, along with one of the three noise-modelling approaches described in Section 3.4. Once this model structure is fixed, we obtain initial point estimates for the hyperparameters by maximising the log marginal likelihood, as discussed in Section 3.5.

We then construct a multivariate normal distribution centred at this point estimate and sample from it to initialise each walker. From there, the walkers explore the hyperparameter space using a Gaussian proposal distribution with a specified covariance. At each step, we compute the sum of the log likelihood and log prior. The proposed sample is then accepted or rejected using the Metropolis-Hastings criterion Could give more detail here. This process is repeated for a fixed number of steps to generate a large set of samples.

To ensure convergence and sample independence, we discard the first 100 samples from each walker as burn-in and apply a thinning factor of 15—retaining every 15^{th} sample. The resulting collection of samples forms our posterior distribution over hyperparameters, which we visualise using a kernel density estimate (KDE).

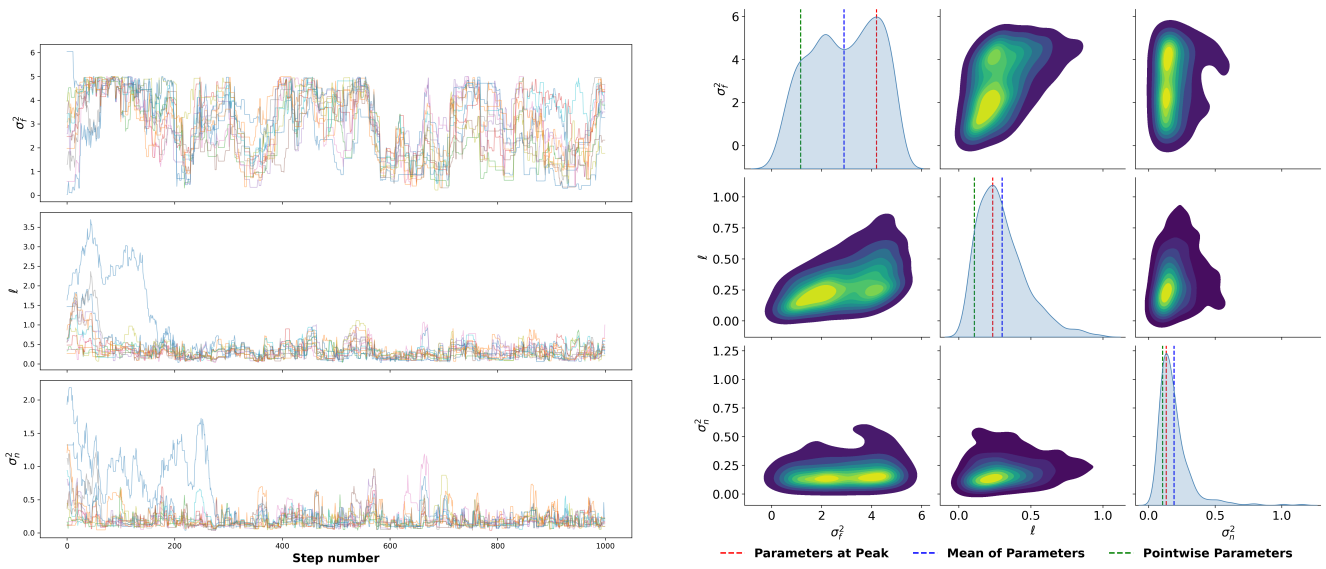


Figure 7: This MCMC is run using a gpr with a RBF Kernel with White Kernel for noise.Left: A plot of the space sampled by each walker in my MCMC run (burnin=100 and thin=15). Right: A plot of the distribution of the hyper-parameters built using a Gaussian KDE from the MCMC samples.

We can see from the distributions in 7 that the variance hyper-paramterer is almost bi-modal (i.e has two significant peaks). For our bi-model hyper-parameter we can see that the mean parameter, peak parameter and point estimate parameter differ significantly. In this scenario any singular point estimate will lose a lot of information about the distribution particularly for the variance hyper-parameter. To resolve this we can instead of picking a singular point build the hyper-parameter uncertainty into our final predictive distribution.

We previously had our predictive distribution as :

$$p(f_* \mid \mathbf{y}, X, X_*, \theta),$$

where predictions were made conditional on a fixed set of hyperparameters θ. Now, we marginalise over the posterior distribution of θ to account for hyperparameter uncertainty:

$$p(f_* \mid \mathbf{y}, X, X_*) = \int p(f_* \mid \mathbf{y}, X, X_*, \theta)\, p(\theta \mid \mathbf{y}, X)\, d\theta.$$

Since this integral is analytically intractable, we approximate it using the MCMC samples $\{\theta^{(s)}\}_{s=1}^{S}$:

$$p(f_* \mid \mathbf{y}, X, X_*) \approx \frac{1}{S} \sum_{s=1}^{S} p(f_* \mid \mathbf{y}, X, X_*, \theta^{(s)}).$$

This entails constructing a full posterior predictive distribution for each set of sampled hyperparameters and then averaging across all predictions. In practice, we compute the final predictive mean and variance using the law of total variance:

$$\mathbf{E}[f_*] \approx \frac{1}{S} \sum_{s=1}^{S} \mu^{(s)}(f_*),$$

$$\mathrm{Var}[f_*] \approx \frac{1}{S} \sum_{s=1}^{S} \left[\sigma^{2(s)}(f_*) + \left(\mu^{(s)}(f_*) \right)^2 \right] - \left(\mathbf{E}[f_*] \right)^2,$$

where $\mu^{(s)}(f_*)$ and $\sigma^{2(s)}(f_*)$ are the predictive mean and variance obtained from the s-th sampled hyperparameter configuration $\theta^{(s)}$.

This procedure yields a predictive distribution that fully reflects both the uncertainty in the data and the uncertainty in the model's hyperparameters

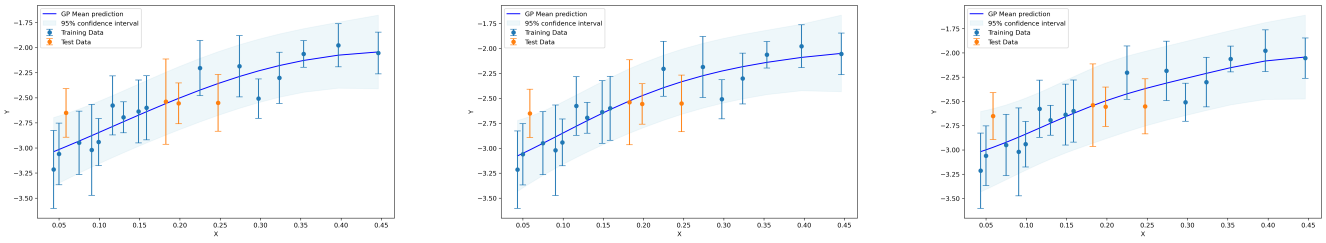


Figure 8: Left: This is the GPR plotted with the mean hyper-parameters Middle: This is the GPR plotted with the peak hyper-parameters Right: This is the full predictive distribution over all the hyper-parameter samples

4.2 Model Evaluation Metrics

In figure 2, we made a graphical representation of four out of six metrics used to evaluate our model's accuracy. [6] discusses how these metrics provide a balanced assessment of model performance. The Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) measure the average deviation of predictions from the true values, with RMSE penalizing larger errors more heavily. The coefficient of determination R^2 quantifies how well the model predicts relative to the mean of the test set. It is computed as 1 minus the ratio of the squared residuals to the total variance. A value closer to 1 indicates better predictive performance. The adjusted R^2 ($\bar{R}^2$) accounts for model complexity by penalizing excessive predictor variables, preventing overfitting. The Figure of Merit (FOM) evaluates the ratio of a point's prediction error to its associated standard deviation. A FOM near 1 is ideal, indicating that the model's uncertainty estimates are well-calibrated. A FOM $\ll 1$ suggests an overly conservative model with large uncertainty, while a FOM $\gg 1$ may indicate overconfidence, failing to capture true variability. The Pearson correlation coefficient measures the linear relationship between predictions and true values. A correlation of 1 (-1) signifies a perfect positive (negative) linear relationship, whereas a correlation of 0 indicates no linear association.

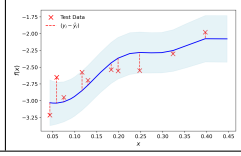
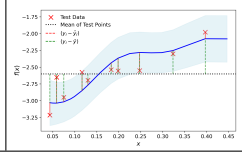
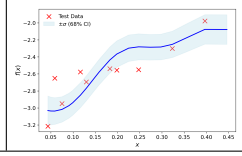
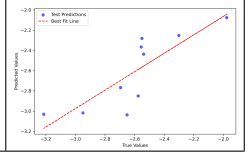
Metric Name	**RMSE**	R^2	**FOM**	**Pearson Coefficient**
Formula	$\sqrt{\frac{1}{N}\sum(y_i - \hat{y}_i)^2}$	$1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$	$\frac{RMSE}{\sigma}$	$\frac{\text{cov}(y, \hat{y})}{\sigma_y \sigma_{\hat{y}}}$
Visual Illustration				

Table 2: Comparison of different performance metrics used in evaluating models. RMSE, R^2, FOM, and the Pearson Coefficient are included. MAE is similar to RMSE but without squaring errors. Adjusted R^2 accounts for the number of predictors and is slightly modified from R^2. The actual metrics for each graph are: RMSE = 0.2, R^2 = 0.6, FOM = 1.09, Pearson correlation = 0.8.

5 Multi-Dimensional Gaussian Process Regression

- Here we want to bring up how we can have a different length parameter for each dimension of the data.
- Get a few plots to demonstrate this maybe.
- Important to note noise modelling for all dimensions is the same since just adding to the diagonal of the covariance matrix.
- Mention how hyper-parameter optimisatino changes in the larger parameter space

6 Method

Figure 9: My Process Flowchart

6.1 Method Over-view

Here I will refer back to my 1d examples. The three ways I have to model noise discussed in section 3.4 are

- Using the true error associated with the data.
- Using a White Kernel to create a noise hyper parameter which can be optimised
- Using Monte Carlo Sampling to sample from the error

My different kernels are

- RBF
- Matern
- Laplace
- Periodic
- Rational Quadratic

6.2 My data 4D - 7D

Could be discussed in the intro also

6.3 Training and Testing Protocol

As 9 details I divided my data in a 90% train and validation data set and then a final 10% data test left untouched for testing. I split my training and validation data into 10 folds and trained my model on each of these 10 folds.

- Different Models (handling error and optimisation)

- split data into 90-10. 90 train and validation and 10 test
- Evaluated using cross validation and multiple metrics
- compared and found best model
- built these final models on the 90% of the data
- tested on the 10% of the data
- Also built MCMC using the gpr methods that made to final
- Used this to build a posterior on the hyper-parameters
- Compared MCMC solutions to the optimised solutions

7 Different Models

- White Kernel without Error Knowledge
- White Kenrel With Error Knowledge
- Using $\alpha = err^2$
- Using Monte Carlo Sampling, Sampling from the Error
- Using MCMCing to build a posterior on hyper-parameter distribution
- Picking values from this posterior distribution to use

A Appendix A: Derivation of Predictive Distribution

Using Bayes' Theorem applied to continuous probabilities, we have:

$$p(f_*|f) = \frac{p(f_*, f)}{p(f)}.$$

we have:

$$p(f_*, f) = \frac{1}{2\pi\sqrt{|\mathbf{C}|}} \exp\left(-\frac{1}{2}\begin{bmatrix} f \\ f_* \end{bmatrix}^T \mathbf{C^{-1}} \begin{bmatrix} f \\ f_* \end{bmatrix}\right)$$

and

$$p(f) = \frac{1}{\sqrt{2\pi}|K|^{1/2}} \exp\left(-\frac{1}{2} f^T K^{-1} f\right).$$

Legend

- **Covariance matrices:**
  - $K = K(X, X)$: Covariance matrix of the training inputs.
  - $K_{**} = K(X_*, X_*)$: Covariance matrix of the test inputs.
  - $K_* = K(X, X_*) = K(X_*, X)^\top$: Cross-covariance between training and test inputs.
- **Joint covariance matrix:**

$$C = \begin{bmatrix} K & K_* \\ K_*^\top & K_{**} \end{bmatrix}$$

- **Determinant of C:**

$$|\mathbf{C}| = KK_{**} - K_* K_*^\top$$

- **Inverse of C:**

$$\mathbf{C}^{-1} = \frac{1}{|\mathbf{C}|} \begin{bmatrix} K_{**} & -K_* \\ -K_*^\top & K \end{bmatrix}$$

- **Mean functions:**

$$m(X) = m(X_*) = 0$$

We obtain the posterior predictive distribution:

$$p(f_*|f) = \frac{1}{(2\pi)^{n/2}|K_{**} - K_*^T K^{-1} K_*|^{1/2}} \exp\left(-\frac{1}{2}(f_* - K_*^T K^{-1} f)^T (K_{**} - K_*^T K^{-1} K_*)^{-1} (f_* - K_*^T K^{-1} f)\right). \tag{14}$$

References

[1] Akcay, S. Forecasting gamma-ray bursts using gravitational waves. arXiv preprint arXiv:1808.10057 (2018).

[2] Duvenaud, D. The kernel cookbook: Advice on covariance functions. `https://www.cs.toronto.edu/~duvenaud/cookbook/`, 2014. Accessed: 2024-03-14.

[3] Hoy, C., Akçay, S., Uilliam, J. M., and Thompson, J. E. Incorporating multi-model uncertainty into gravitational-wave bayesian inference. In preparation (2024). To be published.

[4] Maggiore, M. Gravitational Waves. Volume 1: Theory and Experiments. Oxford University Press, Oxford, 2008.

[5] Owen, B. J. Search templates for gravitational waves from inspiraling binaries: Choice of template spacing. arXiv preprint gr-qc/9511032 (1995). Submitted to Physical Review D: November 7, 1995.

[6] Rasmussen, C. E., and Williams, C. K. I. Gaussian Processes for Machine Learning. MIT Press, Cambridge, MA, USA, 2006.