FYP

Sean White

January 2025

Abstract

- Motivated by the need for fast and accurate models of gravitational waveforms, this project explores Gaussian Process Regression (GPR) as a way to model the waveform mismatch in black hole binary mergers.
- Begin by understanding GPR fundamentals on a low-dimensional toy problem following [6],.
- We extend this approach to a 4-dimensional and 7-dimensional parameter space, derived from the 8D intrinsic black hole parameter set.
- Multiple kernels (RBF, Matern, RationalQuadratic, Laplacian) and noise modelling techniques (homoscedastic, heteroscedastic) are compared.
- Using cross-validation and metrics [7], we find that several models are very accurate with the best models achieving .....
- We also apply MCMC to sample hyperparameter posteriors and understand how uncertain our predictions are. Emphasis maybe how MCMC is ideal but too computationally expensive
- Visualise the GPR across its parameter space taking cross-cuts
- Finally our GPR provides a fast and accurate mismatch model that will help the work being carried out in [3].

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],

> "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 formully 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. In the linearized regime of general relativity, 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 $T_{\mu\nu} = 0$) therefore we are left with

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

with $\Box$ the d'Alembertian operator in flat spacetime.
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 Lorentz 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 illistrated in [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. SA: You already kind of mention this below Eq. (9), so maybe provide slightly more detail there.
Could maybe mention radiation time scale being much smaller than orbit timescale but I don't see what this will add
$\dot{E}$ 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 Regression Background:** 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 the basic assumptions about smoothness and structural properties of the underlying function. I look at how kernel hyper-parameters effect the shape of samples from our prior distribution.
3. **Noise Modeling:** Section 3.4 covers several approaches for incorporating observational noise into the GP framework. I look at how noise effects the samples from our prior distribution.

4. **Hyperparameter Optimization:** Finally, in Section 3.5, I discuss how kernel and noise hyper-parameters can be optimised resulting in a posterior distribution that better explains our data.

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

$$f(X) \sim \mathcal{GP}(\mu(X), k(X, X')). \tag{12}$$

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

$$\mu(X) = E[f(X)], \tag{13}$$

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

$$k(X, X') = \text{Cov}(f(X), f(X')). \tag{14}$$

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

$$X = \{X_1, X_2, \ldots, X_n\}, \tag{15}$$

the corresponding function values

$$f = \{f(X_1), f(X_2), ..., f(X_n), \} \tag{16}$$

follow a multivariate normal distribution

$$f \sim \mathcal{N}\left(\mu(X), K(X, X)\right). \tag{17}$$

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

$$\begin{bmatrix} f(X) \\ f(X_*) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(X) \\ \mu(X_*) \end{bmatrix}, \underbrace{\begin{bmatrix} K(X, X) & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}}_{\mathcal{C}=\text{Covariance Matrix}} \right). \tag{18}$$

The corresponding joint probability density function (pdf) is given by:

$$p(f, f_*) = \frac{1}{(2\pi)^{n/2}\sqrt{|\mathbf{C}|}} \exp\left(-\frac{1}{2} \left(\begin{bmatrix} f \\ f_* \end{bmatrix} - \begin{bmatrix} \mu(X) \\ \mu(X_*) \end{bmatrix} \right)^T \mathbf{C}^{-1} \left(\begin{bmatrix} f \\ f_* \end{bmatrix} - \begin{bmatrix} \mu(X) \\ \mu(X_*) \end{bmatrix} \right) \right) \tag{19}$$

After accounting for the mean, the resulting distribution is entirely determined by its kernel function. The kernel governs how the model generalizes to unseen data. There are many kernel choices, each encoding different structural assumptions about the function, such as smoothness, periodicity, or linearity. 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.

3.3 Kernel Functions

The kernel function encodes our assumptions about the relationship between input points in a Gaussian Process (GP). It defines the covariance between any two function values and thereby determines the smoothness, periodicity, or other properties of the functions drawn from the GP prior. Fundamentally, kernels reflect the idea of similarity: input points x and x' that are close together are assumed to have highly correlated outputs $f(x)$ and $f(x')$, while distant inputs are assumed to produce less correlated values. This notion of similarity, as emphasized in [6, p. 79], is central to how Gaussian processes learn from and generalize beyond training data.

In Figure 1, we illustrate the effect of the kernel on the GP prior. We draw three functions from the multivariate Gaussian prior defined in Equation 18, using a zero mean and an RBF kernel. The first subplot shows these samples, while the second subplot visualizes the corresponding covariance matrix as a heatmap. The matrix reveals that correlations are strongest when input points are close together (near the diagonal) and decay as the distance between inputs increases. This is evident also from the samples as we can see nearby points often move in similar directions, while distant points diverge more significantly.

The final three subplots highlight how this distance-based correlation manifests in the joint distribution of pairs of function values. For closely spaced inputs, such as $(x, x') = (0, 0.1)$, the joint distribution of $(f(0), f(0.1))$ forms a narrow elliptical contour, indicating strong correlation (approximately 0.9). As the distance increases, such as in the pairs $(0, 0.5)$ and $(0, 1)$, the ellipses widen, reflecting weaker correlation. This visualization reinforces the intuition that kernel functions govern how input proximity translates to output similarity.

Figure 1: Sampling from the GP prior with zero mean and an RBF kernel ($\ell = 0.5$, $\sigma_f^2 = 1$). The first plot shows three sample functions drawn from the prior distribution. The second plot visualizes the covariance matrix as a heatmap, revealing the strength of correlations between inputs. The final three subplots display joint distributions between selected input pairs, illustrating how output correlation diminishes with increasing input distance.

We have discussed how the kernel function encodes the covariance structure of the GP prior. This structure depends on the choice of kernel. According to [[6]] kernels can be divided 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. While each kernel imposes a distinct structural pattern on the functions—such as smoothness, periodicity, or linearity—they are all similarly influenced by shared hyperparameters like the lengthscale. In addition, many kernels

include unique internal parameters that further shape the behaviour of the modeled functions. In the following subsections, we explore each of these kernels in detail and discuss the role of their associated hyperparameters.

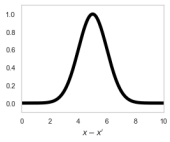
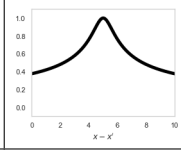
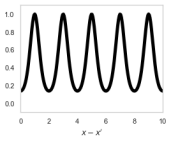
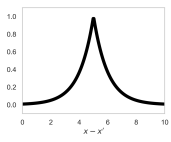
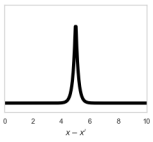
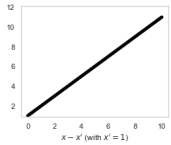
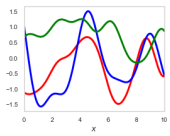
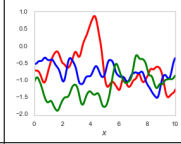
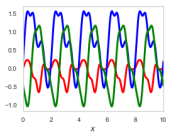
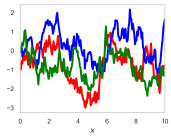
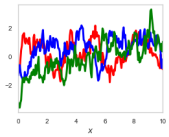
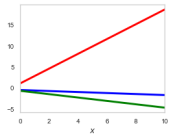
Kernel name:	**RBF (SE)**	**Rational Quadratic**	**Periodic**	**Matern**	**Laplace**	**Linear (Dot Product)**
Plot of $k(x, x')$:						
GP Prior Samples:						
Key Hyperparameters	ℓ (Lengthscale)	ℓ α (Scale-mix)	ℓ p (Period)	ℓ ν (Smoothness)	γ (Decay rate)	None or variance
Structure type:	Local variation	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 $p = 2$. Detailed formula and graphs for each kernel are provided in the appendix B.

Note: In practice, we scale each kernel by a signal variance hyperparameter σ_f^2, which governs the overall vertical variation in the function. This scaling is applied consistently across all kernel types and is discussed further in the 3.4.

From Table 1, we observe that the RBF, Rational Quadratic, Matern, Laplace, and Periodic kernels are all examples of stationary kernels (i.e depend on $|x - x'|$). Many of these—such as the RBF, Matern,Rational Quadratic and Laplace—exhibit "bell-shaped" structures: inputs x and x' that are close together yield high covariance, which then decays as the distance $\|x - x'\|$ increases.The Periodic kernel, while also stationary, has a unique structure. Instead of decaying monotonically with distance, it assigns high covariance to inputs that are separated by integer multiples of a fixed period p. This leads to a repeating pattern of similarity, making the kernel ideal for modeling functions which are periodic.

We have examined multiple kernel types and their properties.We will now briefly visualise and examine the effect of the lengthscale hyperparameter and the signal variance hyperparameter on the GP prior. We will use the RBF kernel as an example, but the same principles apply to other kernels.

Figure 2: Sampling from the GP prior with mean 0 and covariance given by the RBF kernel. The first plot shows the effect of the lengthscale hyperparameter ℓ on the GP prior.We fix the signal variance to 1. The second plot shows the effect of the signal variance hyperparameter σ_f^2 on the GP prior.We fix the lengthscale to 0.5.

Sean: Comment on how each effects prior. Below is waffle from a while ago could potentially add

Sean: Could mention something about credible interval here, Maybe next section

Adding Data: Prior to Posterior

We have discussed how our prior distribution is dependent on the choice of kernel and the hyper-parameters of said kernel. In this section we will discuss how we can update our prior distribution 18 to achieve our new prediction distribution distribution from which we can make inferences. One of the key strengths of Gaussian Processes is that, given observations at training inputs X and setting kernel hyper-parameters θ we can make predictive inferences about the function value at any new test location x_*. By applying the standard conditional Gaussian formulas (see appendix Must clean up this derivation in appendix A for the full derivation), the posterior distribution of $f(x_*)$ given $\{X, f(X)\}$ is Gaussian and given by:

$$p\big(f(x_*) \mid f(X), X, X_*, \theta\big) \sim \mathcal{N}\big(m(x_*),\ \sigma^2(x_*)\big)\,, \tag{20}$$

where

$$m(x_*) = \mu(x_*) + k(x_*, X)\, k(X, X)^{-1}\, [f(X) - \mu(X)]\,, \tag{21}$$

$$\sigma^2(x_*) = k(x_*, x_*) - k(x_*, X)\, k(X, X)^{-1}\, k(X, x_*). \tag{22}$$

In these expressions:

- $\mu(\cdot)$ is the mean function (We 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 3 shows how we update our distributions based on the training points. We initially plot samples taken from the prior distribution (Equation 18). After conditioning this distribution on one training point and obtaining the new predictive posterior distribution (Equation 20), 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 3: 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.

Sean: can add a potential link to animation here illustrating the nice distribution formed at each point

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 applicable to this real-world data, we now explore how to incorporate noise into the GP framework.

We assume that each observation includes the true function value plus Gaussian noise:

$$y_i = f(x_i) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma_{n,i}^2), \tag{23}$$

where $\sigma_{n,i}^2$ is the noise variance associated with input x_i. This allows us to model both homoscedastic and heteroscedastic noise under a unified notation.

Under this model, we have the following Gaussian assumptions:

$$f \sim \mathcal{N}(0, K), \qquad \text{(prior over the true function)} \tag{24}$$

$$y \sim \mathcal{N}(0, K + \Sigma), \qquad \text{(distribution over noisy observations)}, \tag{25}$$

where $\Sigma = \text{diag}(\sigma_{n,1}^2, \sigma_{n,2}^2, \ldots, \sigma_{n,n}^2)$ is the noise covariance matrix.

This now updates our previous posterior mean and variance (eq: 21 and 22) to a revised posterior:

$$P(f_*|X, X_*, \theta, y) \sim \mathcal{N}(m(f_*), \text{Var}(f_*)), \tag{26a}$$

$$m(f_*) = K_*^T (K_y)^{-1} y, \tag{26b}$$

$$\text{Var}(f_*) = K_{**} - K_*^T (K_y)^{-1} K_*, \tag{26c}$$

where K_y depends on how we handle our noise. There are three main cases of how we handle our noise

Homoscedastic Noise

In the homoscedastic case, we assume that all observations have the same noise level, meaning the noise variance is constant across the dataset:

$$\sigma_{n,i}^2 = \sigma_n^2 \quad \forall i.$$

This simplifies the noise covariance matrix Σ to a scalar multiple of the identity matrix:

$$\Sigma = \sigma_n^2 I.$$

The total covariance matrix of the observed data becomes:

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

Our prior distribution now becomes:

$$y \sim \mathcal{N}(0, K+\sigma_n^2 I) \tag{27}$$

where σ_n^2 is a new parameter which effects the shape of the distribution. In figure 2 we explored how the internal kernel hyper-parameters effect the shape of the sampes from our prior distribution. We now examin how the noise (i.e σ_n^2) effects samples from our prior distribution from 26a. Our predictive distribution remains as in 26a

Figure 4: Sampling from the GP prior with mean 0 and covariance given by the RBF kernel. The plot shows the effect of the noise hyperparameter σ_n^2 on the GP prior.We fix the signal variance to 1 and length scale to 0.5.

Our posterior distribution is as in eqn 26a with $K_y = K(X,X)+\sigma_n^2 I$ for some constant σ_n

Heteroscedastic Noise

Known 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 $\Sigma = \text{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2)$. In this case the noise is not found as a hyper-parameter but instead just added to the diagonal of the covariance matrix. Our posterior distribution is as in eqn 26a with $K_y = K(X,X)+\Sigma$ where Σ is the known noise of our data.

Learning Noise over the Input Space If the noise variance is unknown but varies across the input space, we can model it as a function. This is done by building a kernel that captures both smooth, global trends and rough, local fluctuations. In practice, this means building an additive kernel made up of sub-kernels.For example, as seen in Table 1, some kernels like the Matern, Laplacian, or Rational Quadratic capture local variations well (interpreted as noise), while others like the RBF capture broader, smoother structure. By combining these, we can allow one kernel component to model the general structure of the function, and the other to model the heteroscedastic noise behavior. Our posterior distribution is as in eqn 26a with :

$$K_y = \theta_1 K_1(X,X) + \theta_2 K_2(X,X), \tag{28}$$

where K_1 and K_2 are distinct kernels chosen to capture different aspects of the data. The coefficients θ_1 and θ_2 are parameters that control the relative contribution of each kernel component.

Monte Carlo Sampling of Noise

This technique can be applied to both homoscedastic and heteroscedastic noise settings. Instead of modifying the kernel matrix (adding noise to the diagonal), we mimic the effect of noise by adding

random noise to our observed outputs. We assume the observation noise is Gaussian:

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

we have the observed data y which is a noisy version of our true function values

$$y = f + \epsilon, \quad \text{with } y \sim \mathcal{N}(f, \Sigma),$$

where $\Sigma = \text{diag}(\sigma_1^2, \ldots, \sigma_n^2)$. Each true function value corresponds to our observed value $\pm\epsilon$ To account for this, we generate M noisy samples of the observations

$$y^{(s)} = y + \epsilon^{(s)}, \quad \epsilon^{(s)} \sim \mathcal{N}(0, \Sigma). \tag{29}$$

For each sampled dataset $y^{(s)}$, we compute a GP posterior

$$p(f_* \mid X, X_*, \theta, y^{(s)}). \tag{30}$$

To obtain the final predictive distribution, we marginalize over these sampled posteriors:

$$p(f_* \mid X, X_*, \theta, y) = \int p(f_* \mid X, X_*, \theta, y^{(s)})\, p(y^{(s)} \mid y)\, dy^{(s)}. \tag{31}$$

This integral is intractable so we approximate it using Monte Carlo integration where we get the average of each of our predictions on sampled datasets $y^{(s)}$:

$$p(f_* \mid X, X_*, \theta, y) \approx \frac{1}{M} \sum_{s=1}^{M} p(f_* \mid X, X_*, \theta, y^{(s)}). \tag{32}$$

Figure 5: Samples drawn from a zero-mean Gaussian Process prior with varying noise assumptions. **Left:** Homoscedastic noise, where a constant noise variance $\sigma_n^2 = 0.5$ is added uniformly across all inputs. **Middle:** Heteroscedastic noise, where individual noise variances σ_i^2 are known and drawn from $\mathcal{N}(0, 0.5)$, resulting in a diagonal noise covariance. **Right:** Monte Carlo sampling of noisy observations, where multiple noisy realizations are generated from $\mathcal{N}(f(x), \epsilon^2)$

From Figure 5 we see how different noise assumptions influence samples drawn from the Gaussian Process prior. In the homoscedastic case (left), our samples exhibit consistent fluctuations across the entire domain due to a constant noise variance applied uniformly to all inputs. The heteroscedastic case (middle) introduces input-dependent noise, this results in regions of smoothness followed by abrupt variations—reflecting the fact that each output has its own associated noise level. Finally, in the Monte Carlo noise sampling approach (right), we generate multiple samples by adding different levels of noise to our prior distribution on our true function values. This highlight all the plausible functions consistent with the observed data and captures the full range of uncertainty introduced by noisy observations.

3.5 Hyper-parameters

Until now, all predictive distributions such as Equations 26a and 20 have been conditioned on fixed kernel hyperparameters. As demonstrated in Figures 2 and 4, these hyperparameters have a significant influence on the structure and behaviour of the Gaussian Process, shaping both the prior and posterior distributions. In practice, these hyperparameters are not known and must be inferred from the data. To do so, we aim to find the set of hyperparameters that best explain the observed data by maximising the log marginal likelihood, a method detailed in [Ch5 [6].]

As outlined in Section 3.4 we have different methods of handling noise resulting in different hyperparameters to be optimised. We will focus on the general case here which can be easily manipulated for each spefici method. From Equation 27, we have:

$$y \sim \mathcal{N}(0, K + \Sigma I),$$

where K is the kernel matrix computed from the training inputs X, and Σ is the noise variance. This implies that the marginal likelihood which is 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 $K_y = K + \Sigma I$, and Σ may be constant or input-dependent depending on the noise model used. The hyperparameters are given by:

$$\theta = \left\{\sigma_f^2, \ell, (\text{other internal kernel params}), \sigma_n^2 \text{ (if noise is modelled as a hyperparameter)}\right\}.$$

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

$$\log p(y \mid X, \theta) = -\frac{1}{2} y^\top K_y^{-1} y - \frac{1}{2} \log |K_y| - \frac{n}{2} \log 2\pi \tag{33}$$

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. Sean: Explain the below Figure 6

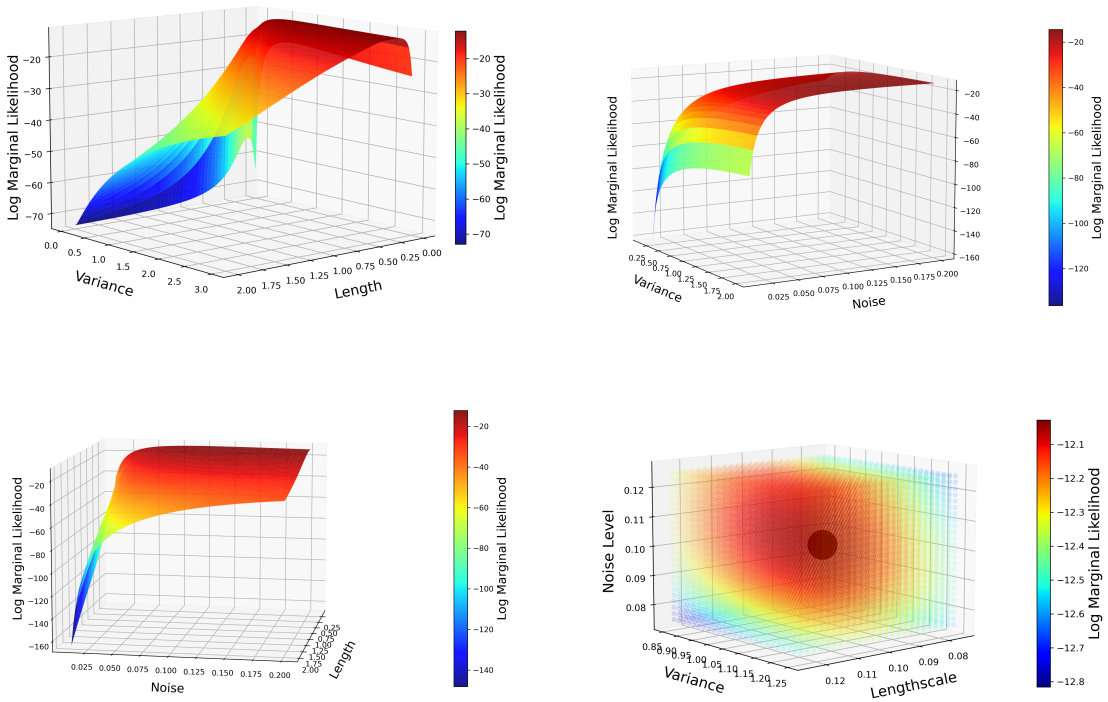


Figure 6: For a GPR with noise where we have the noise as a hyper-parameter we are forced to optimise a length hyper-parameter, a variance hyper-parameter and a noise hyper-parameter. Here we have kept one parameter constant on each graph and compared the log likelihood space varying the other two parameters. In the upper left panel the noise is set at 0.1, in the upper right panel the length is set at 0.5, in the lower left panel the variance is set at 1.5. In the final panel 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$= 0.105$).

Sean: Moved my cross evaluation section to methods

Sean: Cut to 1.5 pgs and clean. add into GPR background

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 2 and 6, 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 33. 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 7: 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).

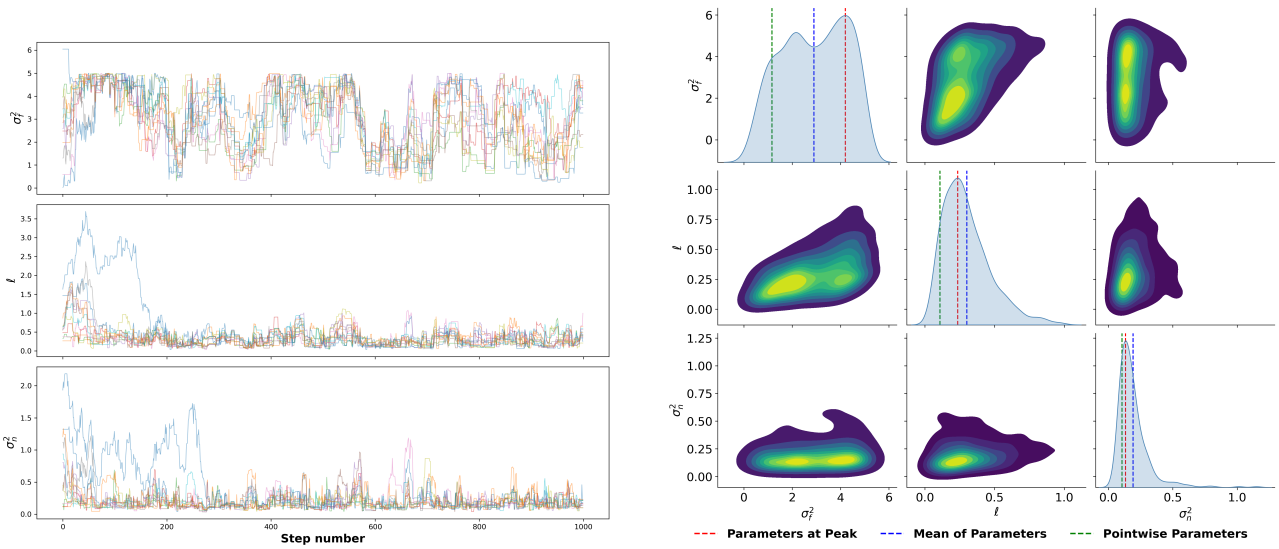


Figure 8: This MCMC is run using a gpr with a RBF Kernel with White Kernel for noise.Left: A plot of the space sampled by each walker in my MCMC run (burnin=100 and thin=15). Right: A plot of the distribution of the hyper-parameters built using a Gaussian KDE from the MCMC samples.

We can see from the distributions in 8 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 9: SA: We agreed that this figure be removed. 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

5 Multi-Dimensional Gaussian Process Regression

Sean: Add in

- 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

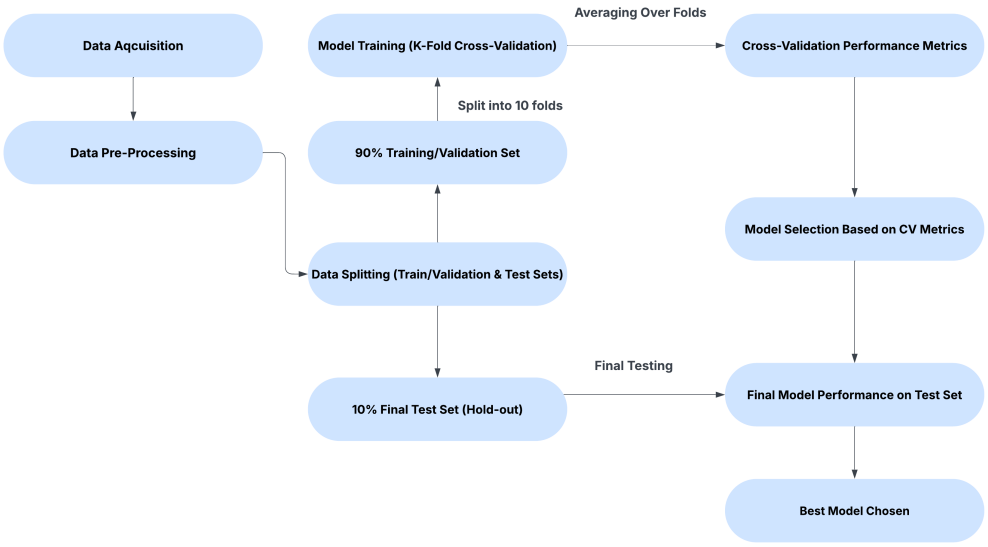


Figure 10: SA: Acquisition is misspelled My Process Flowchart

6.1 Data Description

SA: This section should go probably after the intro. Sean: How data was asquired. What is the data what simulation number SA: The data is mismatches between waveform model SEOBNRv5PHM and the NR surrogate NRSur7dq4 for a set of 250 intrinsic parameters such that it covers 5 different mass ratios and a grid of concentric ellipses in the spin projection space. This 250-element set is then repeated for 4 different masses. Each mismatch is further an average of 294 mismatches computed over a grid of 3 different extrinsic parameters. **4D**
The inrinsic parameter space of a binary black hole is 8 dimension. This accounts for two masses and two spin vectors both in 3 dimensions. For our 4d model we reduce this 8 dimensional parameter space to 4. These 4 parameters are:

- **Total Mass:** $M_1 + M_2$

- **Symmetric Mass Ratio:** $\frac{q}{(1+q)^2}$ where $q = \frac{M_2}{M_1}$
- **Parallel Spin Component:** $\chi_{\parallel} = \frac{|\mathbf{S}_{1,\parallel} + \mathbf{S}_{2,\parallel}|}{M^2}$, the magnitude of the component of the spin vectors parallel to the orbital angular momentum
- **Perpendicular Spin Component:** $\chi_{\perp} = \frac{|\mathbf{S}_{1,\perp} + \mathbf{S}_{2,\perp}|}{M^2}$, the magnitude of the component perpendicular to the orbital angular momentum

Here, the spin vectors are defined as $\mathbf{S} = (S_x, S_y, S_z)$

Sean: Maybe Redundant Graph. Almost definitely leave out

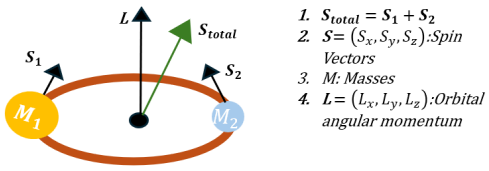


Figure 11: GR diagram

We scale our Total Mass data to lie between 0 and 1 (4 different total masses), and denote the scaled total mass by w. For each total mass, we define 5 different symmetric mass ratios, which are scaled to lie between -1 and 1; we denote these scaled values by z. Next, we transform our $X_{\parallel}$ and $X_{\perp}$ data onto a square grid of size 10×25, with $X_{\parallel} \in [0.1, 1]$ and $X_{\perp} \in [-\pi/2, \pi/2]$. We now have:

$$(x, y, z, w) = \text{transformed}(X_{\parallel}, X_{\perp}, \text{Symmetric Mass Ratio}, \text{Total Mass}) \tag{34}$$

We can visualise our data by holding parameters fixed and extracting cross-sections. In figure 12 we first show a 3d scatter plot of all our data for $w = 0.25$. We then cut our 3d surface at $z = 0$ and interpolate over our values to visualise a plane. Finally to extract a 1d-cut from the plane we cut the plane at $x = 0.8$ and interpolate, helping visualise how our data changes over y. When working directly with the true data, interpolation is necessary to estimate values between sample points. However, a major advantage of using Gaussian Process Regression (GPR) is that it gives us an analytic model that can be evaluated at any point within the domain. This means that when visualising GPR predictions, we do not need to interpolate, instead we can directly evaluate the GPR while holding selected dimensions constant.

Sean: Not nice visual, looks very bad

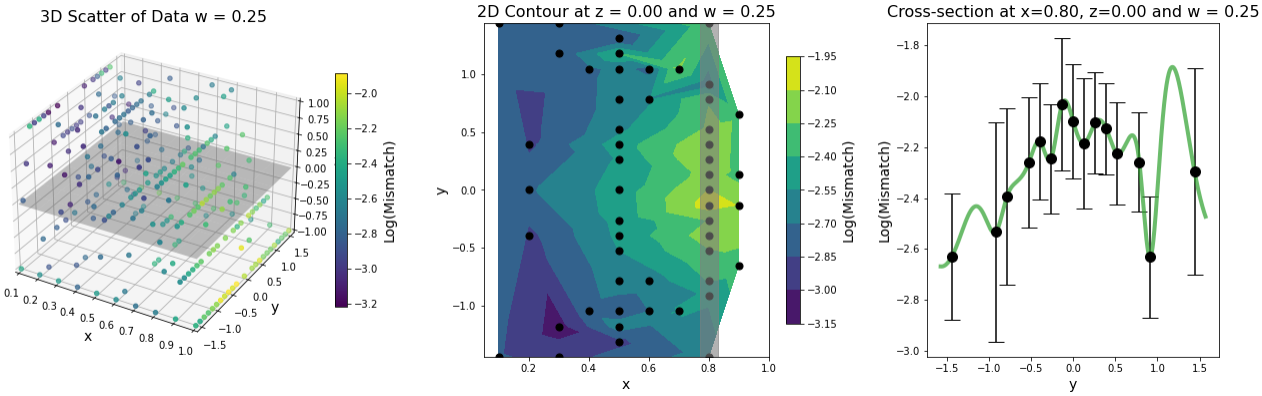


Figure 12: Visualising our data by moving from 4d to 1d

7D

6.2 The Models

To build a robust Gaussian Process Regression model, I built an extensive selection of models for comparison. These models included all combinations of kernels introduced in Section 3.3 and noise-handling strategies discussed in Section 3.4. A summary of all these models is provided in Table 2. I began with the `fixedalpha` model, which assumes known noise added directly to the diagonal of the kernel matrix discussed in Section. Following this I added noise to my model as a hyper-parameter in the selection of White Kernel models I used. Sean: Define White Kernel its purpose . Upon initial visualisations it was clear that the resulting optimised noise depended massively on the optimisation bounds. I therefore evaluated different models with different noise bounds. These included `whitenoerror` model which used loose bounds $(10^{-6}, 10^{6})$, along with two additional variants: `whiteminmaxerror`, which uses bounds set by the 5% and 95% quantiles of the observed uncertainty, and `whitemeanerror`, which constrains the noise using scaled bounds around the mean of the observed uncertainties. A hybrid model (`hybrid`) combining a fixed alpha with a learnable white noise term was also tested. To better capture uncertainty in input observations, I included a Monte Carlo model (`montecarlo`) that samples from the input distribution instead of optimising a noise parameter. Finally, I experimented with additive kernel structures (`combinekernel`) by summing the RBF kernel with Matern, RationalQuadratic, and Laplace kernels to model more complex function behaviours. For each configuration, hyperparameters were optimised by maximising the log marginal likelihood as detailed in Section 3.5.

Model Label	**Noise Optimisation Bounds**	**Kernels**
`fixedalpha`	–	All kernels
`whitenoerror`	$(10^{-6}, 10^{6})$	All kernels
`whiteminmaxerror`	5% lower and 95% upper error	All kernels
`whitemeanerror`	$(0.7\mu_{\text{error}}, 1.3\mu_{\text{error}})$	All kernels
`hybrid`	$(10^{-6}, 10^{6})$	All kernels
`montecarlo`	Sampling	All kernels
`combinekernel`	–	RBF + Matern
`combinekernel`	–	RBF + RationalQuadratic
`combinekernel`	–	RBF + Laplace

Table 2: Summary of the Gaussian Process Regression models evaluated. "All kernels" refers to the RBF, Matern, Rational Quadratic, ExpSine Squared and the Laplace kernels discussed in Section 3.3.

Sean: Optimisation details, algorithm used

6.3 Model Evaluation Metrics

To assess the performance of each Gaussian Process Regression model, I used six evaluation metrics. These metrics were discussed and chosen in [7] for the specific reason that they offer a robust metrics that capture different information about the model. In this paper they divide the metrics into two types Average Expected Error **AEE** metrics and correlation metrics.

AEE Metrics

- **Root Mean Squared Error (RMSE)** $\left(\sqrt{\frac{1}{N}\sum(y_i - \hat{y}_i)^2}\right)$: Measures the average distance between true values and the prediction. Since at each point the error is squared the RMSE has a heavier penalty for larger errors.

- **Mean Absolute Error (MAE)** $\left(\frac{1}{N}\sum|y_i - \hat{y}_i|\right)$: The MAE measures the average absolute difference between predicted and true values. It is less sensitive to larger errors than the RMSE since it is not quadratically scaled.
- **Figure of Merit (FOM)** $\left(\frac{\text{RMSE}}{\sigma}\right)$: This is the ratio of our RMSE to the standard deviation. It can be interpreted as the average expected error scaled by the spread of the data. Lower FOM values indicate higher model accuracy, as they imply that the model's predictive error is small compared to the variation in the data. A value near zero reflects excellent predictive performance, while larger values suggest less accurate predictions.

Correlation Metrics

- **Coefficient of Determination** $\mathbf{(R^2)}\left(1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}\right)$: Finds 1 - the ratio of how well the mean predicts the true values compared to the model predictions. Measures how much better our model is at predicting than a baseline mean prediction.
- **Adjusted** $\mathbf{R^2}$ $\left(\bar{\mathbf{R}}^{\mathbf{2}}\right)\left(1 - (1 - R^2) \cdot \frac{n-1}{n-p-1}\right)$: Updates our R^2 value to take account the number of predictors compared to the number of observed points. This helps to prevent overfitting because trivially if we used $p = n$ predictors we should get perfect results but our model would be drastically over-fitted.
- **Pearson Correlation Coefficient** $\left(\frac{\text{cov}(y,\hat{y})}{\sigma_y \sigma_{\hat{y}}}\right)$: Quantifies the linear relationship between true and predicted values.

In [7] the author concluded that lower AEE metrics (close to 0) correspond to higher regressor accuracy, and higher correlation metrics (closer to 1) correspond to better predictions. We set out to evaluate our models with these metrics.

6.4 Training, Comparing, and Testing My Models

To ensure that each of my models was not overly dependent on a particular dataset and to mitigate overfitting, I implemented K-fold cross-validation, evaluating the metrics outlined in Section 6.3 on each validation fold. This approach is supported by Rasmussen and Williams in Chapter 5 of [6], where cross-validation is discussed as a method for model selection. I divided the full dataset into a 90–10 split, where 90% of the data was used for 10-fold cross-validation, and the remaining 10% was held out as an untouched test set for final evaluation. During cross-validation, each model was trained on 9 out of the 10 folds and evaluated on the remaining fold, with the process repeated such that every fold served as the validation set once. For each fold, model predictions were compared against the true values using the six metrics described in Section 6.3. To analyse model stability and consistency, I visualised the distribution of each metric across folds using box plots for each model type. Additionally, I plotted the mean performance for every metric across all folds and model types. Each model was then ranked for each metric individually (e.g., lower is better for AEE, higher is better for correlation), and these rankings were averaged across all metrics to produce an overall ranking table. To understand relationships between metrics, I constructed a dendrogram based on the correlation of their model rankings, enabling the identification of clusters of similar metrics and those with differing behaviours. Finally, to ensure robust generalisation, I used the most distinct metrics identified from the dendrogram to create a scatter plot and selected a subset of models that consistently performed best across these dimensions.

Final Testing

In the final stage, each of the shortlisted models was retrained on the full 90% cross-validation training set and evaluated on the held-out 10% test set. Model predictions were compared to the true values using the same six metrics, and performance was visualised to identify the most

accurate and robust candidates. Two final models were selected: one with the overall best test performance, and a second with simpler hyperparameters to serve as a baseline for comparison and to help guard against overfitting. For both models, I performed Markov Chain Monte Carlo (MCMC) sampling as demonstrated in Section 4.1 to construct posterior distributions over their hyperparameters, allowing for a visual and probabilistic assessment of hyperparameter uncertainty.

Implementation Details

All models were implemented in Python. Gaussian Process Regression was carried out using the `GaussianProcessRegressor` class from the `scikit-learn` library. For MCMC sampling, I used the `emcee` package. Interpolation was performed using `scipy.interpolate`. To ensure reproducibility, I consistently set the random seed to 42 throughout all experiments.

7 Results

7.1 4D Results

Sean: Unsure about visuals here need to build them in better, don't know which models to plot, I am plotting too much atm maybe combined noerror and minmax

Cross-Validation Performance

After running cross-validation for all model types discussed, we summarize the results in Figure 13. A few notable observations emerge. Firstly, the `monte_carlo` noise modeling method performs poorly. This may be partly due to computational limitations (we only ran 10 samples per fold) but also due to instability in optimization. For noisy systems, sampling extreme values for noise led to difficulty during optimization, and averaging across such varied samples likely caused over-generalized predictions. Sean: Add graph to demonstrate a GP prediction after Monte Carlo sampling of the noise. Incorporating the true noise (standard deviation of the data) values resulted in worse performance than learning the noise as a hyperparameter. This is evidenced by the relatively poor results from `fixed_alpha` and the slightly improved but still limited performance of `hybrid`. We conclude that learning noise through a hyperparameter is the most effective approach in these models.

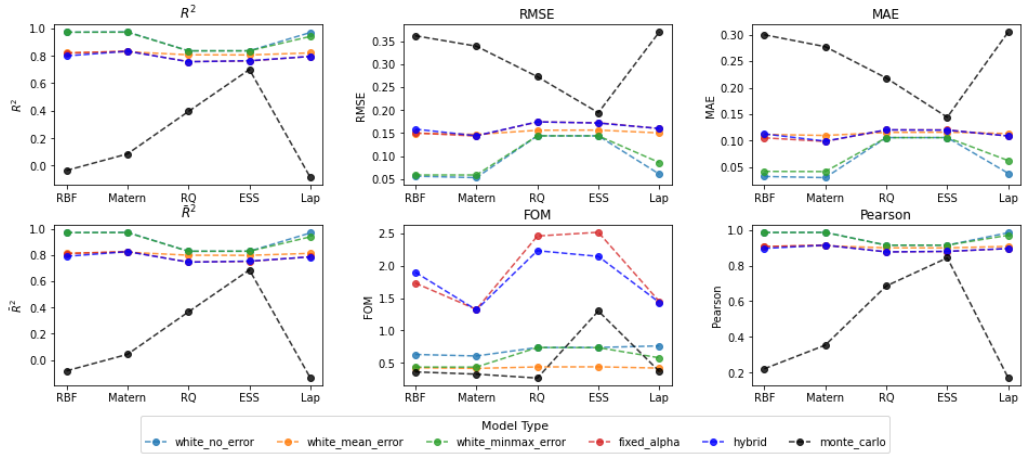


Figure 13

Note:In figure 13 we focused on known noise and noise being modeled as a hyper-parameter we did not visualise the combined kernel approach. They are included in all subsequent evaluations.

To rank models across multiple metrics, we compute the average rank of each model across all metrics. These rankings are shown in Table 4. To investigate which metrics produce similar or different model rankings, we plot a dendrogram of the ranking distances between metrics in (Figure 14,left). From this, we observe that the Figure of Merit (FoM), MAE, and R^2 yield notably different rankings. To visualise model performance and how the model generalises over all metrics we compare all models in a scatter plot of R^2 versus FoM (Figure 14, right) with the color of each scatter point representing the MAE. We label each point with its ranking from Table 4, which reveals that the top 8 models form a clearly distinguishable set of optimal performers. These are examined more closely in the next section.

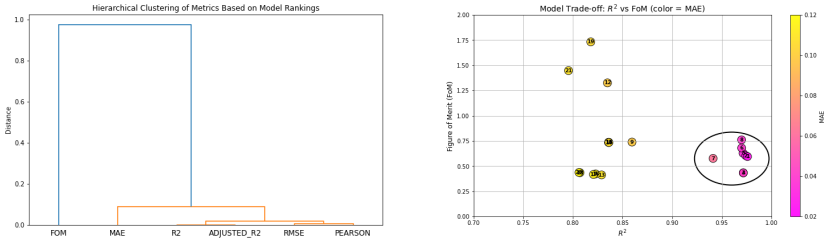


Figure 14: Left: Dendrogram of difference in ranking between metrics, Right: Comparing clustered model performance by R^2 and FoM.

Training on 90% of Data

Now selecting our 8 models that formed that cluster of high performing models in (Figure 14, right) we train all these models on the full 90% of the data that cross validation was corried out with. The resulting model hyper-parameterers are available in 3. With these trained models we predict on our test set (10% of data completely unseen) and like in (Figure 14, right) we plot our results as a scatter showing how models generalise over the R^2,FOM and MAE metrics.

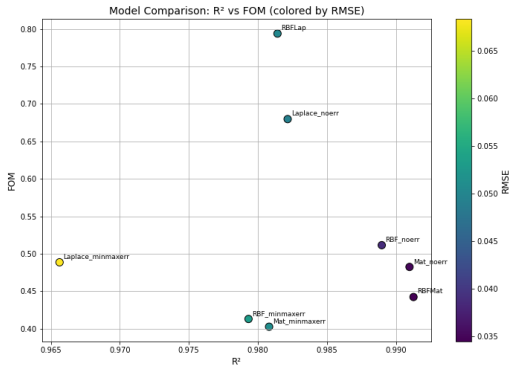


Figure 15: Comparing metrics

To understand the general shape of all our models we pick two cross-sections of our data and plot all our models for each.

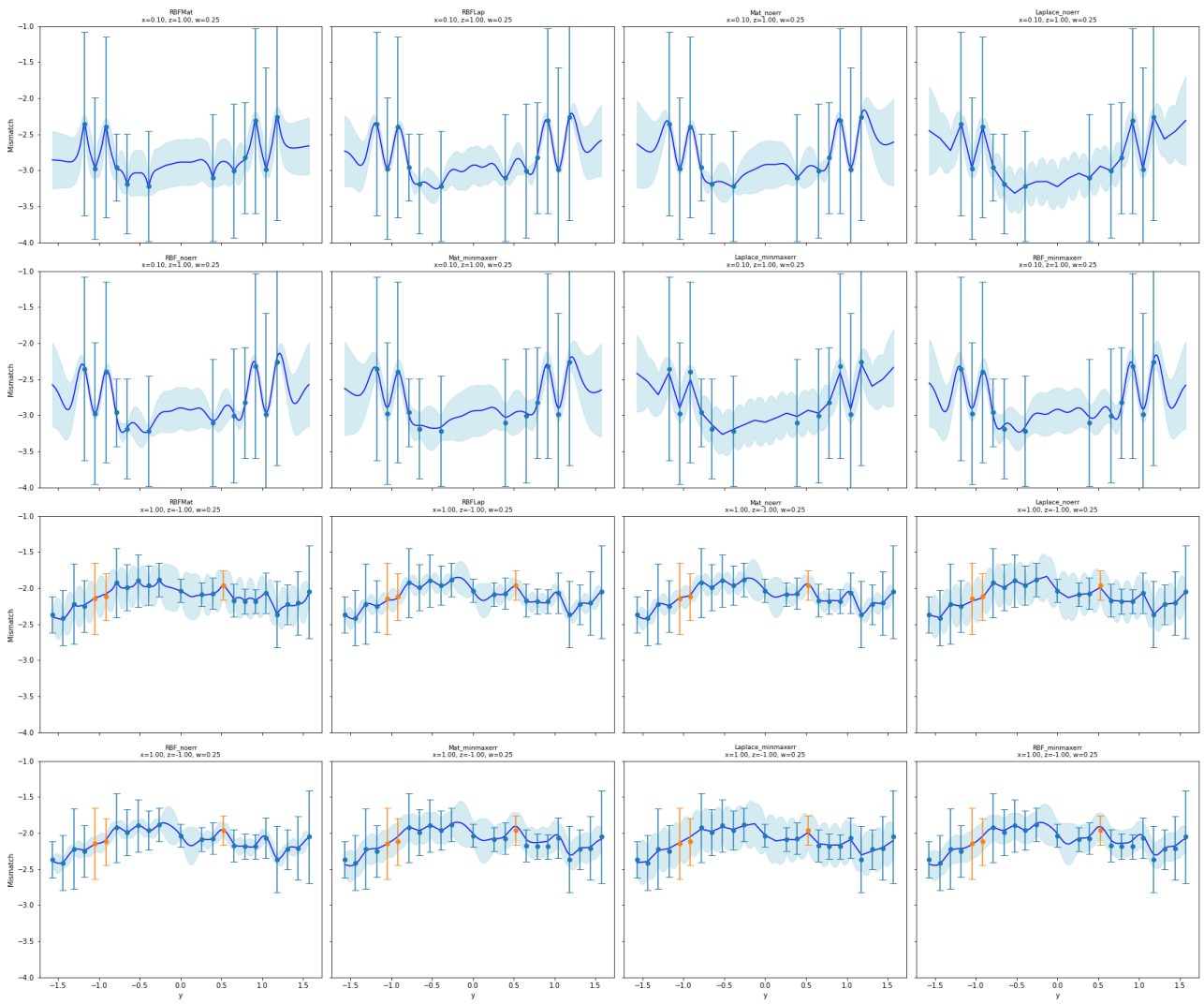


Figure 16: TODO: Make labels bigger Comparing Cross-cuts of best models

From these visualizations, we observe that while all models perform well, they differ in how they handle local noise. Let's examine the `RBFMat` model. It remains smooth where the data is clean but responds sharply in regions containing outliers or local irregularities. This behaviour is reflected in its optimized hyperparameters (Table 3). The RBF component has long characteristic length scales (around 1.0–1.5) and a large scale factor (2.34), which captures the global smooth structure of the data. Meanwhile, the Matern component has much shorter length scales (mostly ¡ 0.5) and a smaller scale (0.207), allowing it to respond to local variability—effectively modeling input-dependent noise.

In contrast, models using a single kernel with a `WhiteKernel` for noise (e.g., `Mat_noerr`, `RBF_noerr`) appear smoother overall. These models result in smaller length scales and relatively small noise variances ($\sigma_n^2 < 0.01$). When using tighter optimisation bounds for noise in the `min_maxerr` models, we observe from Table 3 that the noise hyperparameter consistently sits at or near the lower bound. This suggests that the GP is compensating for noise directly through the kernel, not through the explicit noise model. This provides further credibility to the `RBFMat` combined model, where the noise appears to be effectively captured within the `Matern` kernel. Finally, examining Laplacian-based kernel models (`Laplace_noerr` and `Laplace_minmaxerr`), which only have a single hyperparameter (γ), they still perform reasonably well but tend to produce more linear predictions. This is likely due to their limited flexibility compared to other kernels.

Based on our metrics calculated and visualised in Figure 15 and our qualitative examination of the crosscuts in Figure 16, we conclude that the `RBFMat` model offers the best trade-off between

flexibility, interpretability, and robustness. However, since the `RBFMat` combined kernel model optimises 10 hyperparameters, there is a risk we are overfitting to the data. Therefore, we retain the `Mat_noerr` model for comparative purposes, as it offers a smooth fit with fewer hyperparameters.

Table 3: Optimized Hyperparameters for Final GPR Models

Model	Kernel 1	Scale 1	Length Scales 1	Kernel 2	Scale 2	Length Scales 2 / Noise
`RBFMat`	RBF	2.34	$[1.00, 1.51, 1.38, 1.36]$	Matern ($\nu = 0.75$)	0.207	$[0.0996, 0.0582, 0.414, 2.31]$
`RBFLap`	RBF	0.354	$[0.10, 0.10, 1.18, 2.91]$	Laplacian ($\gamma = 0.964$)	0.292	—
`Mat_noerr`	Matern ($\nu = 1.75$)	0.926	$[0.227, 0.20, 1.15, 2.85]$	White	—	$\sigma_n^2 = 0.00637$
`Laplace_noerr`	Laplacian ($\gamma = 0.358$)	7.24	—	White	—	$\sigma_n^2 = 10^{-6}$
`RBF_noerr`	RBF	0.728	$[0.112, 0.112, 0.958, 1.6]$	White	—	$\sigma_n^2 = 0.00728$
`Mat_minmaxerr`	Matern ($\nu = 1.75$)	1.14	$[0.27, 0.22, 1.34, 4.73]$	White	—	$\sigma_n^2 = 0.0439$
`Laplace_minmaxerr`	Laplacian ($\gamma = 0.284$)	6.60	—	White	—	$\sigma_n^2 = 0.0439$
`RBF_minmaxerr`	RBF	0.821	$[0.12, 0.115, 1.19, 2.52]$	White	—	$\sigma_n^2 = 0.0439$

MCMC

7.2 7D Results

SA: Don't do anything for this now, simply discuss in the Discussion section as the next thing to do. Sean: Ask Sarp how to compare 4D with 7D obviously need to use transformation formulas to get equivalent Sean: Ask Sarp if it is at all possible to plots of other other functional fits of Mismatch like he had in his plot Sean: Ask Sar if possible to get true mismatch data from other data-set to test, if model generalises

8 Conclusion

- successfully developed accurate models
- Evaluated a wide variety of kernels
- Used MCMC to visuale hyper-parameters posteriors
- Compared 4d to 7d model how did reducing the parameters space effect the model, loose much information??
- Potential Scaling issues

9 Discussion

- In future would like to build a model with kernel uncertainty built into model, full gpr `TODO: Add the 7D discussion here`
- Combined GPR , including multiple posteriors weighted
- Use other GPR methods, sparse GPR Sean: (read and bried comment)
- Maybe using physical constraints somewhat in the model

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:**

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

B Appendix B: Kernel Formulas

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.

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.

C Appendix C: Graphs of 4d finalist GPs

Figure 17: All 8 gps with cutting their y-axis

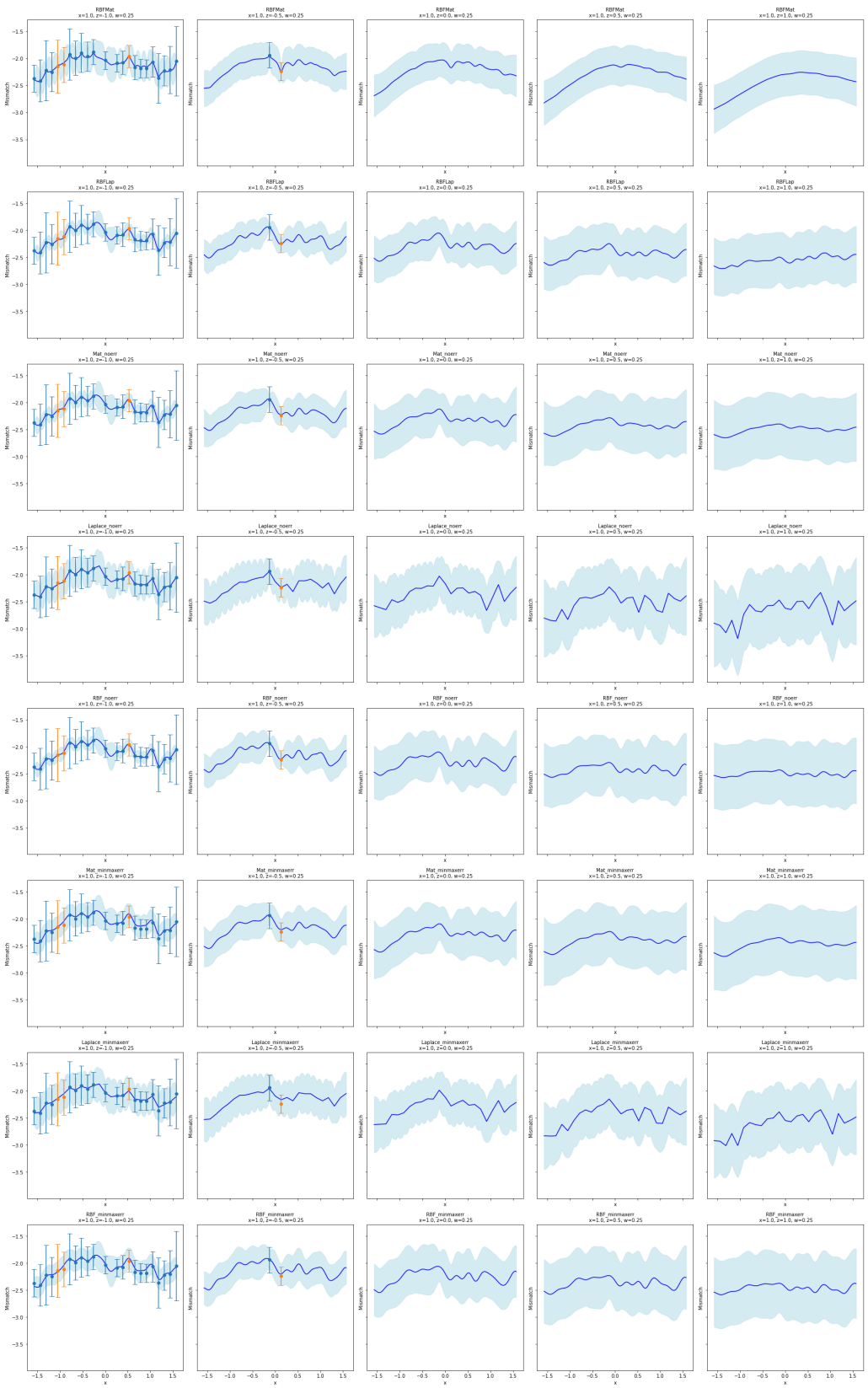


Figure 18: All 8 gps with cutting their x-axis

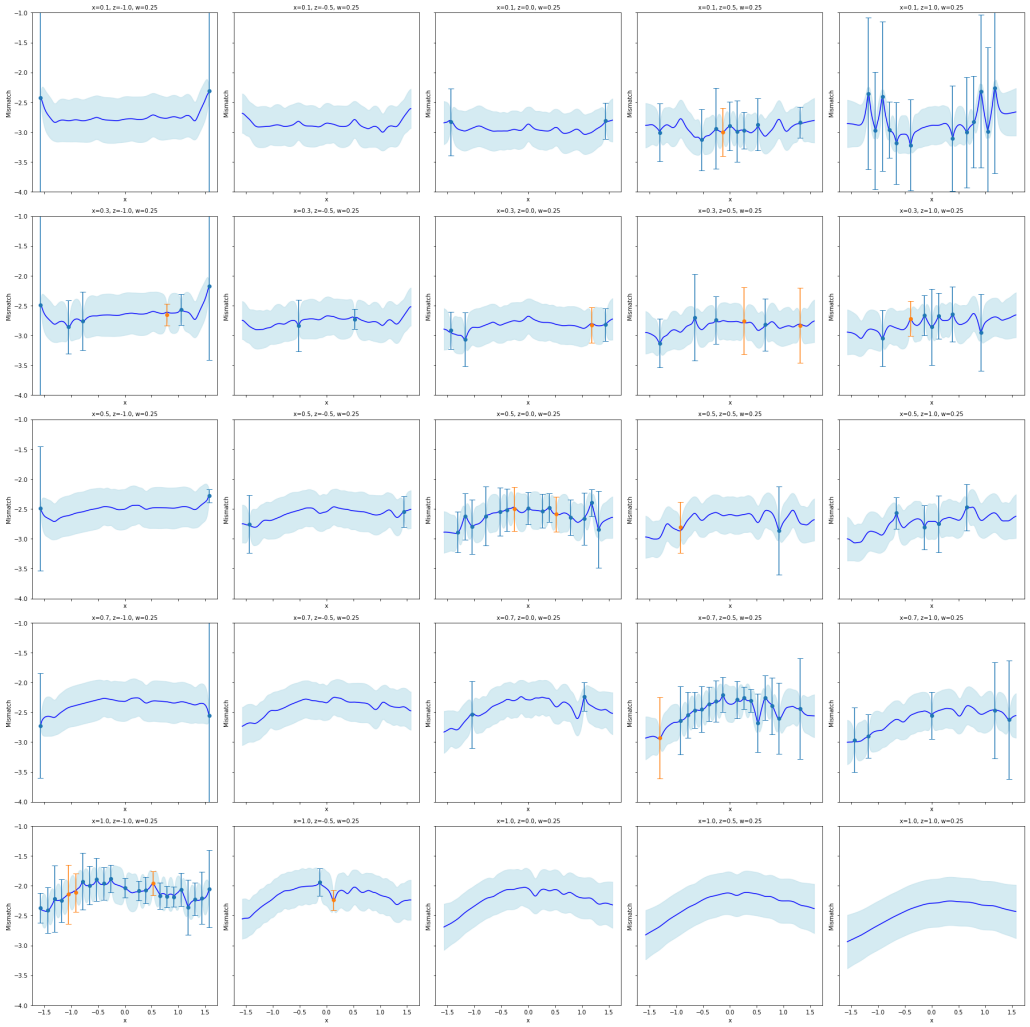


Figure 19: Examining my chosen model RBF Matern kernel

D Appendix D: Model Evaluation Table and graphs

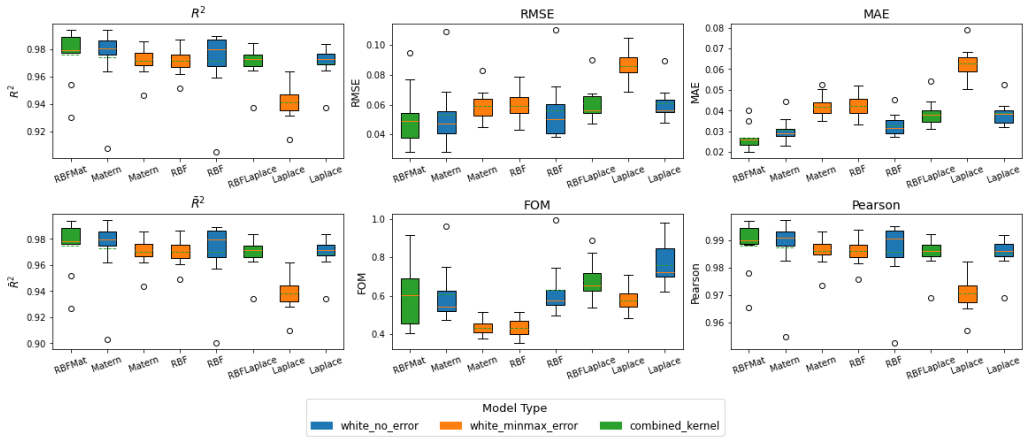


Figure 20: Seeing how the best models performed over different clusters

Table 4: Final Model Rankings from CSV

Kernel	Model	R2	R	RMSE	R	MAE	R	adj R2	R	FOM	R	Pearson	R	Final R
RBFMat	combinedkernel	0.98	1	0.05	1	0.03	1	0.97	1	0.6	9	0.99	1	1
Matern	whitenoerror	0.97	2	0.05	2	0.03	2	0.97	2	0.61	10	0.99	2	2
Matern	whiteminmaxerror	0.97	3	0.06	4	0.04	6	0.97	3	0.43	4	0.99	4	3
RBF	whiteminmaxerror	0.97	4	0.06	5	0.04	7	0.97	4	0.43	5	0.99	3	4
RBF	whitenoerror	0.97	5	0.06	3	0.03	3	0.97	5	0.63	11	0.99	5	5
RBFLaplace	combinedkernel	0.97	6	0.06	6	0.04	5	0.97	6	0.68	12	0.99	6	6
Laplace	whiteminmaxerror	0.94	8	0.09	8	0.06	8	0.94	8	0.58	8	0.97	8	7
Laplace	whitenoerror	0.97	7	0.06	7	0.04	4	0.97	7	0.76	18	0.99	7	8
RBFRad	combinedkernel	0.86	9	0.13	9	0.1	9	0.85	9	0.74	17	0.93	9	9
ExpSineSquared	whiteminmaxerror	0.84	10	0.14	11	0.11	12	0.83	10	0.73	13	0.92	11	10
ExpSineSquared	whitenoerror	0.84	11	0.14	12	0.11	15	0.83	11	0.74	16	0.92	12	11
Matern	fixedalpha	0.83	14	0.14	10	0.1	10	0.83	14	1.33	19	0.92	10	12
Matern	whitemeanerror	0.83	15	0.15	15	0.11	17	0.82	15	0.41	1	0.91	15	13
RationalQuadratic	whitenoerror	0.84	12	0.14	13	0.11	14	0.83	12	0.74	15	0.91	13	14
RationalQuadratic	whiteminmaxerror	0.84	13	0.14	14	0.11	13	0.83	13	0.74	14	0.91	14	15
RBF	whitemeanerror	0.82	16	0.15	16	0.11	18	0.81	16	0.42	3	0.91	17	16
Laplace	whitemeanerror	0.82	17	0.15	18	0.11	19	0.81	17	0.42	2	0.91	16	17
RationalQuadratic	whitemeanerror	0.81	19	0.16	19	0.12	20	0.8	19	0.44	6	0.9	19	18
RBF	fixedalpha	0.82	18	0.15	17	0.11	11	0.81	18	1.73	21	0.91	18	19
ExpSineSquared	whitemeanerror	0.81	20	0.16	20	0.12	21	0.8	20	0.44	7	0.9	20	20
Laplace	fixedalpha	0.8	21	0.16	21	0.11	16	0.79	21	1.45	20	0.9	21	21
ExpSineSquared	fixedalpha	0.76	22	0.17	22	0.12	22	0.75	22	2.52	23	0.88	22	22
RationalQuadratic	fixedalpha	0.76	23	0.17	23	0.12	23	0.75	23	2.46	22	0.88	23	23

E Appendix E: Bin

Sean: Moving Evaluation Metrics to Methods

In figure 5, 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.

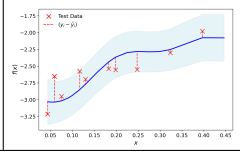
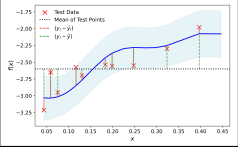
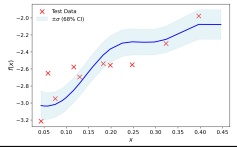
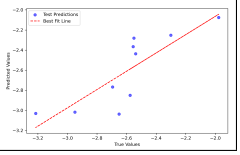
Metric Name	**RMSE**	R^2	**FOM**	**Pearson Coefficient**
Formula	$\sqrt{\frac{1}{N}\sum(y_i - \hat{y}_i)^2}$	$1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$	$\frac{RMSE}{\sigma}$	$\frac{\text{cov}(y, \hat{y})}{\sigma_y \sigma_{\hat{y}}}$
Visual Illustration				

Table 5: 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.

References

[1] Akcay, S. Forecasting Gamma-Ray Bursts Using Gravitational Waves. Annalen Phys. 531, 1 (2019), 1800365.

[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., Akcay, S., Mac Uilliam, J., and Thompson, J. E. Incorporating model accuracy into gravitational-wave Bayesian inference.

[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.

[7] Zaman, Q., Alraho, S., and König, A. Gaussian process regression based robust optimization with observer uncertainty for reconfigurable self-x sensory electronics for industry 4.0. tm - Technisches Messen 88, S1 (2021), S83–S88.