Using Gaussian Processes to Approximate Solutions to Differential Equations

Elizabeth Mabbutt

Supervised by Matt Moores and Aidan Sims

University of Wollongong

Abstract

Differential equations are useful for many important applications, but often do not have known solutions. In this report, we introduce Gaussian processes and Gaussian process regression as a way to estimate these unknown solutions. We then introduce a Sobolev space - a special function space that relates a function to its derivatives - and, given some theorems from recent literature, come up with a list of factors that we need to input into our design of Gaussian process regression to ensure convergence and minimise error.

1 Introduction

Differential equations (DEs) are equations involving a function and its derivatives. They are very useful for modelling problems involving a rate of change, especially in physics. One notable example is the Navier Stokes equation to model fluid dynamics, which, as seen in [4], is being used to model the movement of underwater pipes in Western Australia. The problem with this is that the Navier Stokes equation, as well as many other differential equations do not have known solutions. Usually, in this case, we approximate the the solution using numerical methods, however, in the case of the Navier Stokes equation, approximating the solution using numerical methods is extremely time consuming; as stated in [4], it took 6 weeks to approximate the solution at 9 points. It is therefore evident that other methods need to be used to approximate solutions to differential equations.

The method that we will introduce in this report uses a statistical process known as a *Gaussian process* (GP), which, as seen in Definition 2 is a set of random variables equipped with a mean and covariance function. Given some approximations of points on the solution of a differential equation, we can use a method known as *Gaussian process regression* (see section 3.2) to approximate the solution at other points by estimating the mean and covariance functions (see section 3.3). From this, we have a predictive function, shown in Equation (8), which we can use to approximate the solution at new values, as shown in section 3.4. We show an example of approximating a function using Gaussian process regression in section 4.

From here, we want to know how accurate Equation (8) is at approximating the solution to the differential equation, which requires us to introduce a special type of function space, a *Sobolev space*, which, as stated in Definition 4, and more formally in Definition 6 is a space of bounded functions such that all of their derivatives up to order l are also bounded. We then introduce the *Sobolev norm*, which defines the distance between functions as the sum of how far apart all of their derivatives up to order l are.

With this in hand, section 6 presents two important theorems from [4] and [5]. The theorem from [4] allows us to determine what Sobolev space our predictive function lies in, which allows us to determine if the limit of our approximations satisfies the DE. The theorem from [5] then gives conditions for said limit to exist. We then explain how we can adjust the covariance function we use for Gaussian process regression, as well as what domain we are doing it over to get a bound on our estimate and ensure convergence.

Finally, in section 7, we sum all of this up, and discuss possible applications.

1.1 Statement of Ownership

This report was written by Elizabeth Mabbutt with some suggestions from Matt Moores and Aidan Sims. All definitions and theorems are adapted from credited sources and/or discussions with Matt Moores and Aidan Sims. Code is, as credited, from [3], with adaptations made by Elizabeth Mabbutt, and some suggestions by Matt Moores. Figures were generated using the statistical software R. The information about Gaussian processes and Sobolev spaces in sections 3 and 5 are not original ideas, but adapted from sources as credited.

2 Notation

The following table defines the notation used throughout this paper.

Notation	Explanation
d	$d \in \mathbb{N}$ is the dimension
$U \subset \mathbb{R}^d$	An open d-dimensional set
$\mathbb{N}_0$	The natural numbers including zero, i.e $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$.
$\partial_i f$	For $f : U \to \mathbb{R}$ a differentiable function and $i = 1, ..., d$, $\partial_i f$ is the partial derivative of f with respect to the ith input. We write ∂_i^α for the αth partial derivative with respect to the ith coordinate.
$\partial^\alpha f$	For $f : U \to \mathbb{R}$ and $\alpha = (\alpha_1, ..., \alpha_d) \in \mathbb{N}_0^d$, $\partial^\alpha f := \partial_1^{\alpha_1} ... \partial_d^{\alpha_d} f$.
$\partial^{\alpha,\alpha} k$	For $k : U \times U \to \mathbb{R}$, and $\alpha \in \mathbb{N}_0^d$, $\partial^{\alpha,\alpha} k(u, u') := \frac{\partial^{\alpha_1}}{\partial u_1{}^{\alpha_1}} \cdots \frac{\partial^{\alpha_d}}{\partial u_d{}^{\alpha_d}} \frac{\partial^{\alpha_1}}{\partial u_1'{}^{\alpha_1}} \cdots \frac{\partial^{\alpha_d}}{\partial u_d'{}^{\alpha_d}} k(u, u')$
$cl(U)$	The closure of U, as defined in [6], section 1.1
$[a(u_i)]^{1 \leq i \leq n}$	For $a : U \to \mathbb{R}$ some function, and $u_1, ..., u_n \in U$, $[a(u_i)]^{1 \leq i \leq n}$ is a $1 \times n$ matrix (n-dimensional vector) where the ith entry is $a(u_i)$
$[b(u_i, u_j)]^{1 \leq i \leq n}_{1 \leq j \leq n}$	For $b : U \times U \to \mathbb{R}$ some function, and $u_1, ..., u_n \in U$, $[b(u_i, u_j)]^{1 \leq i \leq n}_{1 \leq j \leq n}$ is an $n \times n$ matrix where the ijth entry is $b(u_i, u_j)$.
0_n	For $n \in \mathbb{N}$, 0_n is the vector in $\mathbb{R}^n$ such that $0_n^i = 0$ for $i = 1, ..., n$. That is, the n-dimensional 0 vector.
$\mathcal{N}_n(\mu, \Sigma)$	For $n \in \mathbb{N}$, $\mu \in \mathbb{R}^n$ and $\Sigma \in M_n(\mathbb{R})$, $\mathcal{N}_n(\mu, \Sigma)$ is the multivariate normal distribution (also known as the multivariate Gaussian distribution) for n variables

3 Gaussian Processes

In this section, we will define Gaussian processes (GPs), and then will show the theory behind how they can be used to approximate functions.

3.1 Definition and Properties of Gaussian Processes

First, we will introduce a *stochastic process*, following the definition in [1] section 9.1

Definition 1. *Let $(\Omega, \mathcal{F}, P)$ be a probability space, and let $U \subseteq \mathbb{R}^d$ be some set. A **stochastic process** is a collection, $X = \{X(u) \mid u \in U\}$ of random variables, $X(u) : \Omega \to \mathbb{R}$ which we say are indexed by the set U.*

Section 9.2 of [1] explains the different ways we can think of stochastic processes. The most common way we will view them in this report is as a function from the sample space, Ω, to $\mathbb{R}^U := \{f : U \to \mathbb{R} \mid f \text{ is a function}\}$, that is, the set of all functions from U to $\mathbb{R}$. Then, we can write $X : \Omega \to \mathbb{R}^U$. This way of thinking of stochastic processes allows us use them to estimate functions, but to do this, we need a more well-behaved stochastic process; a special type of stochastic process called a *Gaussian process.* Following the definition in section 9.8 of [1],

Definition 2. *A **Gaussian Process** is a stochastic process, $X = \{X(u) \mid u \in U\}$ equipped with*

1. *a mean function, $\mu : U \to \mathbb{R}$ such that for all $u \in U$, $\mu(u) = E(X(u))$, and*
2. *a covariance function, $\sigma : U \times U \to \mathbb{R}$ such that for all $u, u' \in U$, $\sigma(u, u') = cov(X(u), X(u'))$,*

such that for any finite collection of points, $u_1, ..., u_n \in U$, $(X(u_1), ..., X(u_n)) \sim \mathcal{N}_n\left([\mu(u_i)]^{1\leq i\leq n}, [\sigma(u_i, u_j)]^{1\leq i\leq n}_{1\leq j\leq n}\right)$.

Remark 1. *It is a requirement of the multivariate normal distribution that the covariance matrix, $[\sigma(u_i, u_j)]^{1\leq i\leq n}_{1\leq j\leq n}$ is positive-definite. Therefore, σ must be a* positive definite function, *that is, a function where for any $u_1, ..., u_n \in U$, $[\sigma(u_i, u_j)]^{1\leq i\leq n}_{1\leq j\leq n}$ is a positive definite matrix.*

To make our notation more readable, for any $n \in \mathbb{N}$ and any $D_n := (u_1, ..., u_n) \in U^n$, we define the mean vector function, $\mathcal{M}_n : U^n \to \mathbb{R}^n$ such that

$$\mathcal{M}_n(D_n)_i = \mu(u_i).$$

Similarly, for any $n, m \in \mathbb{N}$, we define the covariance matrix function. For reasons that will be explained in section 3.2, our matrix function has two inputs, D_n as defined in the above paragraph, and $X_m := (x_1, ..., x_m) \in U^m$. Then we define $\Sigma_{n,m} : U^n \times U^m \to M_{n,m}(\mathbb{R})$ such that

$$\Sigma_{n,m}(D_n, X_m)_{ij} = \sigma(u_i, x_j).$$

For simplicity, when $D_n = X_m$, we write $\Sigma_n(D_n, D_n)$ instead if $\Sigma_{n,n}(D_n, D_n)$.

3.2 Using Gaussian Processes to Estimate Functions

When we view a Gaussian process as a random function $X : \Omega \to \mathbb{R}^U$, we can use it to approximate functions.

There are a lot of scenarios in applied mathematics where we have a finite set of approximations $(u_1, y_1), ..., (u_1, y_n)$ from the function $f : U \to \mathbb{R}$, that is, $f(u_i) = y_i + \epsilon_i$ for each $i = 1, ..., n$. We often write $D_n := (u_1, ..., u_n)^T \in U^n$ as the vector of points in U that we know the function values of, and refer to these as the design points, we write $Y_n := (y_1, ..., y_n)^T$ as the vector of approximations of function values. Together, we call D_n and Y_n the training data. A common scenario where this may come about is if f is the unknown solution to a differential equation (DE) and our training data are numerical approximations of the solution.

We now want to use our training data to approximate the values of the function at some other points. To do this, we will assume that the image of f is the realisation of a Gaussian process. That is, for some Gaussian process $Z = \{Z(u) \mid u \in U\}$, each $f(u)$ is a realisation of the random variable $Z(u)$. However, the problem is, we do not know which Gaussian process $f(u)$ are realisations of (and it is not unique). Therefore, we will use a technique known as *Gaussian process regression.* As is explained in [3] chapter 5, this is where we find a Gaussian process with a mean and covariance function that fits our data. An implication of Definition 2 is that a Gaussian process is completely defined by its mean and covariance functions, so we perform Gaussian process regression by approximating the mean and covariance function. This is commonly done by starting with a base function with a set of parameters that we fit to the data. To make it clear that they are estimated, and also to show what parameters we are using, for θ a vector of parameters we denote the estimated mean function $m(\theta; \cdot) : U \to \mathbb{R}$ and the estimated covariance function, also called the kernel function $k(\theta; \cdot, \cdot) : U \times U \to \mathbb{R}$. Likewise to section 3.1, $M_n(\theta; \cdot) : U^n \to \mathbb{R}^n$ is our estimated mean vector function, such that $M(\theta; D_n)_i = m(\theta; u_i)$, and $K_{n,m}(\theta; \cdot, \cdot) : U^n \times U^m \to \mathbb{R}$ is our estimated kernel matrix function such that $K_{n,m}(\theta; D_n, X_m)_{ij} = k(\theta; u_i, x_j)$ for X_m as defined in section 3.1.

3.3 Estimating the Parameters

In this section we will go through how we define the base functions $m(\theta; \cdot)$ and $k(\theta; \cdot, \cdot)$, and how we estimate the parameters θ, which are sometimes referred to as the hyperparameters.

As presented in [3], section 5.1, for the mean function it is a common practice to assume it is zero. That is, for any vector of parameters, θ, and any $u \in U$, $m(\theta; u) = 0$. A Gaussian process with a mean function as such is referred to as a centered Gaussian process.

If we assume that the mean function is zero everywhere, it means all of the estimation is focused on the kernel function. We generally use a different base kernel function to fit the known properties of our function f. Generally, for f the solution of a DE, the only properties we know about f is a lower bound for $\tilde{\nu}$ - the number of times f is differentiable. However, given a well-behaved DE, we often assume infinite differentiability. We like to choose a kernel function that fits the differentiability of f. This will be further explored in section 6. In either of these cases, the kernel function has parameters that we then fit to the data. The most common parameters, as presented in [3] sections 5.2.1, 5.2.4 and 5.2.2 respectively, are $\theta = (\tau^2, \lambda, g)$, where;

- $\tau^2 \in (0, \infty)$ is called the amplitude, and measures how spread out the values of f are,
- $\lambda \in (0, \infty)$ is called the lengthscale, and measures how quickly $Z(u)$ and $Z(u')$ become uncorrelated as u and u' become further apart in U, and
- $g \in (0, \infty)$ is called the nugget. It represents the error, ϵ_i in our observed data.. For example, if Y_n is found using numerical approximations, g would be the variance of the numerical error.

Note that here, our parameter space, Θ is $(0, \infty)^3$.

The first kernel function we will introduce is the Gaussian kernel, also called the squared exponential kernel. Summarising the information presented in [3] sections 5.2.1-5.2.4, the Gaussian kernel is used for infinitely differentiable functions, and is itself infinitely differentiable. We denote it $k_G(\theta; \cdot, \cdot)$, and for $\theta = (\tau^2, \lambda, g)$ it is given by

$$k_G(\theta; u, u') := \tau^2 \left(\exp\left\{ -\frac{\|u - u'\|_2}{\lambda} \right\} + g\delta_{u,u'} \right), \tag{1}$$

where $\delta_{u,u'}$ is the Kronecker delta, such that $\delta_{u,u'} = 1$ when $u = u'$, and $\delta_{u,u'} = 0$ otherwise.

The other kernel that we will introduce is the Matérn kernel. Summarising [3], section 5.3.3, the Matérn kernel uses the parameters λ, τ^2 and g, but also uses a parameter $\nu \in (0, \infty)$, which is the number of times f is differentiable. We denote the Matérn kernel $k_M(\theta; \cdot, \cdot)$, and for $\theta = (\lambda, \tau^2, g, \nu)$, it is given by

$$k_M(\theta; u, u') = \tau^2 \left[\frac{2^{1-\nu}}{\Gamma(\nu)} \left(\|u - u'\|_2 \sqrt{\frac{2\nu}{\lambda}} \right)^{\nu} K_\nu \left(\sqrt{\frac{2\nu}{\lambda}} \right) + g\delta_{u,u'} \right], \tag{2}$$

where K_ν is the modified Bessel function of the second kind, and Γ is the gamma function. This formula does not have an analytical solution, and is therefore hard to work with. However, when $\nu = p + \frac{1}{2}$ for $p \in \mathbb{N}_0$, there is an exact formula. As a result, in practice, if we think f is p times differentiable, we often use $\nu = p + \frac{1}{2}$, and then

$$k_M(\theta; u, u') = \tau^2 \left[\exp\left\{ \|u - u'\|_2 \sqrt{\frac{2\nu}{\lambda}} \right\} \frac{(\nu - \frac{1}{2})!}{(2\nu - 1)!} \left(\sum_{i=1}^{\nu - \frac{1}{2}} \frac{(\nu - \frac{1}{2} + i)!}{i!(\nu - \frac{1}{2} - i)!} \right) \left(\|u - u'\|_2 \sqrt{\frac{8\nu}{\lambda}} \right)^{\nu - \frac{3}{2}} + g\delta_{u,u'} \right].$$

Admittedly, for large p, this requires a lot of computational power to compute, however, ν is usually only introduced as a parameter for $\nu \leq \frac{5}{2}$, otherwise, the function is assumed to be ‘smooth enough’.

Something that is also interesting to note is that as $\nu \to \infty$, $k_M((\tau^2, \lambda, g, \nu)) \to k_G((\tau^2, 2\lambda, g))$, meaning that any theorems applying to Matérn kernels can be extended to Gaussian kernels.

Similarly to how we use a subscript G or M to specify if our kernel function uses a base Gaussian or Matérn kernel respectively, for the estimated kernel matrix function, we write $K^G_{n,m}(\theta; \cdot, \cdot)$ if each entry is defined using the Gaussian kernel, and $K^M_{n,m}(\theta; \cdot, \cdot)$ if each entry is defined using the Matérn kernel. When referring to a kernel function or matrix that uses an arbitrary kernel, we will denote it $k(\theta; \cdot, \cdot)$ or $K_{n,m}(\theta; \cdot, \cdot)$ as before.

Now that we have explored the different kernel functions we use, we want to estimate θ. To see how, consider that by the definition of a Gaussian process, for any finite set of points, the Gaussian process at those points is multivariate normally distributed. Therefore, if we consider the finite set (vector) of points, D_n, we have that

$$(Z(u_1), ..., Z(u_n)) \sim \mathcal{N}_n \left(M_n(\theta; D_n), K_n(\theta; D_n, D_n) \right).$$

If we consider the multivariate normal pdf with $M_n(\theta; D_n) = 0_n$, we have that

$$P\left((Z(u_1), ..., Z(u_n)) = Y_n\right) = \frac{1}{\sqrt{(2\pi)^n \det\left(K_n(\theta; D_n, D_n)\right)}} \exp\left\{-\frac{1}{2} Y_n^T K_n(\theta; D_n, D_n)^{-1} Y_n\right\}, \tag{3}$$

so we can use maximum likelihood estimation (MLE) to estimate θ. That is, estimate θ by the value $\hat{\theta}$ that maximises Equation (3). Since ln is an increasing function, maximising Equation (3) is equivalent to maximising

$$\begin{aligned} \ell(\theta; Y_n) &:= \ln\left(P\left((Z(u_1), ..., Z(u_n)) = Y_n\right)\right) \\ &= -\frac{1}{2}\left(n \ln(2\pi) + \ln\left(\det\left(K_n(\theta; D_n, D_n)\right)\right) + Y_n^T K_n(\theta; D_n, D_n)^{-1} Y_n\right). \end{aligned} \tag{4}$$

In the case of either the Gaussian or Matérn kernel, τ^2 has an exact estimate that can be found using calculus, while and λ and g need to be estimated using numerical methods. Usually ν is selected based on the properties of f, for example, if it is the solution to a DE we know how many derivatives it must have, and can infer how smooth it is based on how nice the DE looks. More formally, we want the solution f and the Gaussian process Z to belong to the same function space. This will be further explored in sections 5 and 6.

To see how we estimate τ^2, first observe that for either the Matérn or Gaussian kernel, we can write

$$K_n(\theta; D_n, D_n) = \tau^2 \left(K_n((1, \lambda, 0, (\nu)); D_n, D_n) + Ig\right).$$

For ease of notation, let $C := K_n((1, \lambda, 0, (\nu))$. Then it is shown in appendix D that the maximum likelihood estimator for τ^2 is given by

$$\hat{\tau^2} = \frac{1}{n} Y_n^T \left(C + Ig\right)^{-1} Y_n.$$

Then to find λ and g, maximise Equation (4) with $\hat{\tau^2}$ plugged in, that is

$$\ell_1(\theta | Y_n) := -\frac{1}{2}\left(n \ln(2\pi) + \ln\left(\det\left(Y_n^T \left(C + Ig\right)^{-1} Y_n \left(C + Ig\right)\right)\right) + n\right).$$

Since $n\ln(2\pi)+n$ is a constant, is equivalent to maximising

$$\ell_2(\theta|Y_n) := -\frac{1}{2}\ln\left(\det\left(Y_n^T\left(C+Ig\right)^{-1}Y_n\left(C+Ig\right)\right)\right).$$

We did this in R using the code in appendix A. With $\hat{\theta}$ in hand, we have defined a Gaussian process $Z = \{Z(u) \mid u \in U\}$ with the mean function $m(\hat{\theta};\cdot) := 0$, and kernel function $k(\hat{\theta};\cdot,\cdot)$ such that, by our assumption, the image of f is a realisation of Z.

3.4 Approximating New Function Values

Now that we have estimated a Gaussian process, Z such that f is a realisation of Z, we now want to use Z to estimate f at some new values. Firstly, given our estimator $\hat{\theta}$, we can say explicitly that

$$(Z(u_1),...,Z(u_n)) \sim \mathcal{N}_n\left(0_n, K_n(\hat{\theta};D_n,D_n)\right). \tag{5}$$

We have assumed that the image of f is a realisation of Z, which means we can approximate $f(u)$ by $E(Z(u))$ for all $u \in U$. However, given that we know the values of f at D_n, we can get an even better estimate by approximating $f(u)$ by $E(Z(u) \mid Z(u_1),...,Z(u_n))$ for all $u \in U$. There are two different ways we can do this depending on how many points we want to approximate.

Method 1: If we only want to approximate f at a finite vector of m new points, which we will denote $X_m := (x_1,...,x_m)$ such that for all i,j, $x_i \neq u_j$, then, as is explained in [3] section 5.1.1, we can find the distribution of $(Z(x_1),...,Z(x_m)) \mid (Z(u_1),...,Z(u_n)) = Y_n$, and estimate $f(x_i)$ by the ith entry of the mean vector of that distribution. To determine the distribution, first consider that under the assumptions of a Gaussian process,

$$(Z(x_1),...,Z(x_m)) \sim \mathcal{N}_m\left(0_m, K_m(\hat{\theta};X_m,X_m)\right) \tag{6}$$

Then, considering Equations (5) and (6), by properties of the Gaussian distribution, it can be shown that the conditional distribution, $(Z(x_1),...,Z(x_m)) \mid (Z(u_1),...,Z(u_n)) = Y_n$ is given by

$$(Z(x_1),...,Z(x_m)) \mid (Z(u_1),...,Z(u_n)) = Y_n \sim \mathcal{N}_m(\mu_p,\Sigma_p), \text{ where} \tag{7}$$
$$\mu_p = K_{m,n}(\hat{\theta};X_m,D_n)K_n(\hat{\theta};D_n,D_n)^{-1}Y_n, \text{ and}$$
$$\Sigma_p = K_m(\hat{\theta};X_m,X_m) - K_{m,n}(\hat{\theta};X_m,D_n)K_n(\hat{\theta};D_n,D_n)^{-1}K_{n,m}(\hat{\theta};D_n,X_m).$$

So we can estimate $f(x_i)$ by μ_p^i, and we can also estimate the covariance between $Z(x_i)$ and $Z(x_j)$ by Σ_p^{ij}. This method is slightly more accurate as the estimates take into account the covariance between all of the points we are estimating, however, for large samples, this method can be extremely computationally expensive.

Method 2: As is presented in equations 2.2 and 2.3 from [5], if we want to estimate f at a large number of points in U, we can estimate them one at a time. We can use the same procedure as method 1, but with

$X_m = u$. That is, X_m is a single point in U. Plugging this into Equation (7),

$$E\left(Z(u) \mid (Z(u_1), ..., Z(u_n)) = Y_n\right) = K_{1,n}(\hat{\theta}; u, D_n) K_n(\hat{\theta}; D_n, D_n)^{-1} Y_n.$$

So we can estimate f by the function

$$f_n^p(u) := K_{1,n}(\hat{\theta}; u, D_n) K_n(\hat{\theta}; D_n, D_n)^{-1} Y_n. \tag{8}$$

Furthermore, if we want to estimate the covariance between $f(u), f(u')$, by letting $X_m = (u, u')$ and considering the 1,2-entry of Σ_p (or equivalently the 2,1-entry), by Equation (7), we can estimate it by

$$k_n^p(\hat{\theta}; u, u') = k(\hat{\theta}; u, u') - K_{1,n}(\hat{\theta}; u, D_n) K_n(\hat{\theta}; D_n, D_n)^{-1} K_{n,1}(\hat{\theta}; D_n, u'). \tag{9}$$

This has much fewer operations required. If X_m is large, if we used the technique in method 1, then we could need to perform matrix multiplication between an $m \times n$ matrix and an $n \times n$ matrix, which would be very computationally expensive. Therefore, method 2 is sometimes the only feasible method. Furthermore, Equation (8) allows us to approximate the function f for different values of n, meaning we can characterise the convergence of the approximation to the true function f as $n \to \infty$, as was done in [5], and will be explained further in Section 6.

4 Applying Gaussian Processes to Differential Equations

We saw in section 3 how to use Gaussian process regression to estimate functions. Now we want to use Gaussian process regression to estimate the solution to a differential equation given some numerical approximations. For $d \in \mathbb{N}, f : U \to \mathbb{R}$ and $u \in U$, consider an arbitrary lth order differential equation,

$$0 = g(u, \partial^\alpha f(u) \text{ for all } \alpha \in \mathbb{N}_0^d, \|\alpha\|_1 \leq l)$$

with an initial value / boundary value, and suppose that we do not know the analytical solution. If we are able to determine points $(u_1, y_1), ..., (u_n, y_n)$ such that $y_i = f(u_i) + \epsilon_i$, which could be done using numerical methods, then we have n approximations of values from the solution function f. This means that we can approximate the values of our solution function f at other $u \in U$ using Gaussian process regression.

In this section, we will go through an example of using a Gaussian process to approximate the solution to a differential equation. The equation in question is an ordinary differential equations (ODE) of order 1. That is, the case where $d = 1$ and $l = 1$. In this case, it is common to arbitrarily represent the DE as

$$f'(u) = g(u, f(u)), f(0) = f_0$$

for $f_0 \in \mathbb{R}$. Through this example, we will explore some of the problems that arise when fitting Gaussian processes to differential equations, and how we often rectify them.

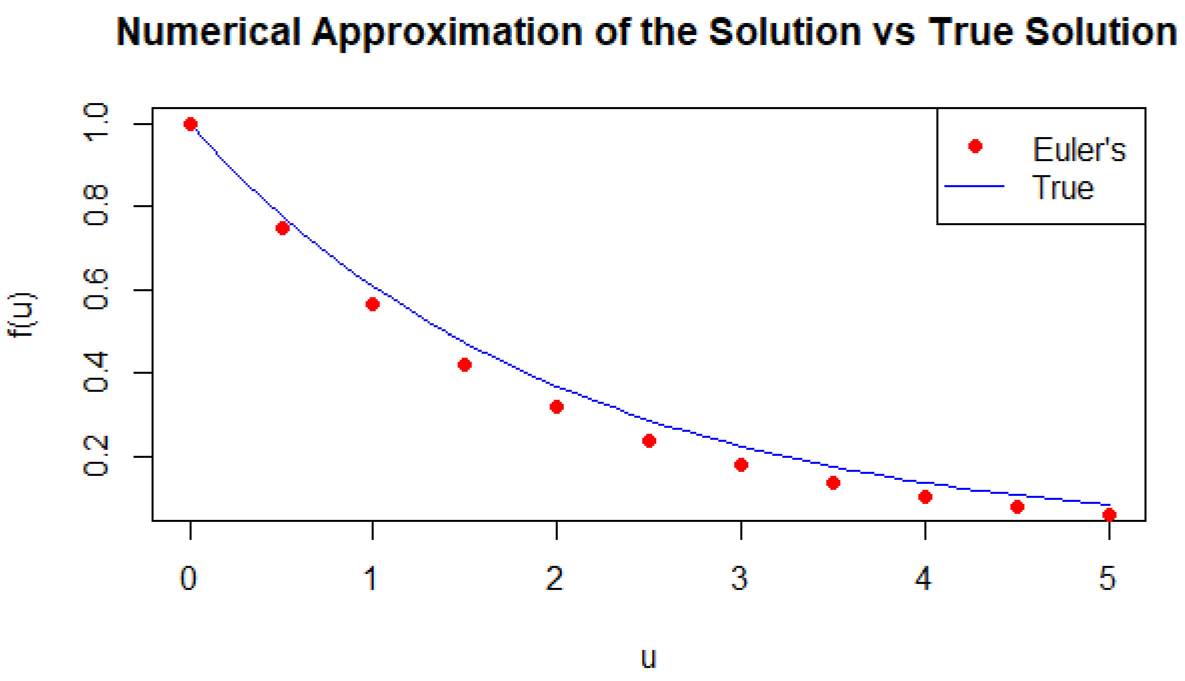

Figure 1: True Solution of Equation 10 vs Euler's Method Approximation

Consider

$$0 = f'(u) + \frac{1}{2}f(u), f(0) = 1. \tag{10}$$

We can show that the function $f(u) = e^{-\frac{1}{2}u}$ is a solution to this DE, but for the sake of demonstrating Gaussian process regression, we will pretend we do not know this.

In R, we approximated the solution to Equation (10) for $0 \leq u \leq 5$ using Euler's method with a step size of $h = 0.5$. A plot of the approximated solution vs the true solution can be seen in Figure 1, where it is evident that the numerical approximations are not completely accurate. This is because Euler's method has local truncation error of order $\mathcal{O}(h)$, meaning it is not the most accurate numerical method, and also because we are using a relativity large step size.

We now want to fit a GP to these numerical approximations. Here, the $n = 11$ design points are given by $D_n = (0, 0.5, ..., 5)^T$, and Y_n is the numerical approximations shown in Figure 1. We named these variables `Dn` and `Yn` in R, and then fitted a Gaussian process through these points using the code in Appendix A, giving us the MLE estimates $\hat{\lambda} = 1, \hat{\tau^2} = 1.129483$ and $\hat{g} = 1.490116 \times 10^{-8}$. We then attempted to approximate the solution to Equation (10) at $m = 111$ new points, $X_m = (0.05, 0.1, ..., 5.55)$ by using the predictive mean in Equation (7) as an estimator. The approximations can be seen in Figure 2. Note that there is no 95% confidence

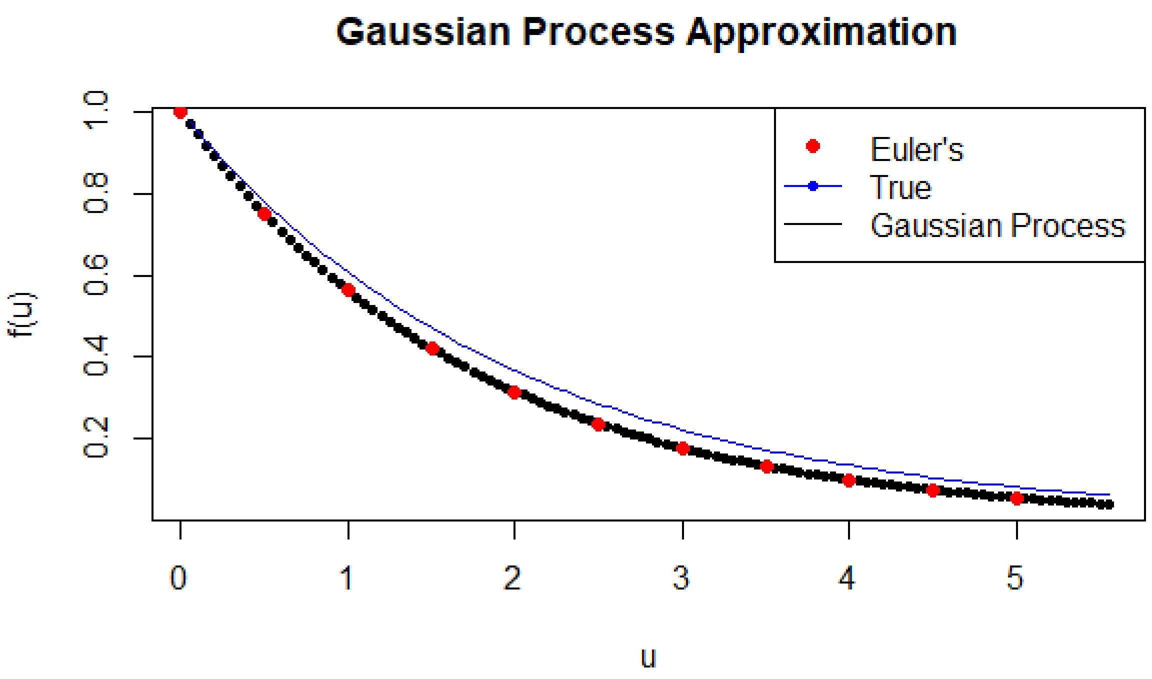

Figure 2: Gaussian process approximation of Equation (10)

interval in Figure 2. This is because the diagonal entries of Σ_p are so small that the 95% interval cannot be clearly seen on the plot. The true solution is not inside the interval.

Figure 2 demonstrates one of the main problems that arise with Gaussian process regression - given numerically inaccurate points, the Gaussian process often approximates what the numerical method would be doing at those points, rather than the solution function. We explained in section 3.2 that the hyperparameter g is introduced to rectify the numerical error in our observations, however, determining the best way to quantify this error remains an open problem, as stated in [2].

We previously stated that the maximum likelihood estimate for g is 1.490116×10^{-8}, however, we notice in Figure 1 that the error is a lot larger. In fact, if for $i = 1, ..., 11$ we write $f(u_i) = y_i + \epsilon_i$, then the vector of errors, $\epsilon := (\epsilon_1, ..., \epsilon_{11})$, correct to 3 decimal places is given by

$$\epsilon = (0, 0.029, 0.044, 0.05, 0.051, 0.049, 0.045, 0.04, 0.035, 0.03, 0.026), \tag{11}$$

with $\text{var}(\epsilon) = 0.0002273649$. This is 4 magnitudes larger than our predicted value of g. This is because, when using MLE, we are assuming that our numerical errors (ϵ) are random, but really, they are determined by our choice of numerical method in combination with the properties of our differential equation. It is hard to model errors as such, so statisticians often assume they are random, which leads to the Gaussian process approximating the wrong function.

Of course, in this example, we know the true error. In practice, we do not know the true numerical error: if we knew the true function values, numerical approximations would be unnecessary, so we want a way to model the error when it is unknown. Often, numerical errors do have bounds, however, these are often magnitudes larger than the true errors, so are often not helpful. [2] provides an analysis on ways of modeling errors, but in practice, often the only feasible solution to this is to use numerical methods with very low error that it is essentially zero. For example, instead of using Euler's method, we can use a method with lower truncation error, such as the classical 4th order Runge Kutta method (rk4), which only has $\mathcal{O}(h^4)$ truncation error, and is therefore more accurate. Appendix C repeats this example, but instead calculating Y_n using rk4, where it is evident μ_p approximates the solution more accurately.

5 Sobolev Spaces

Now we want to find a way to determine how well a Gaussian process approximates the solution of a DE. To do this we will first introduce a special type of function space.

A *Sobolev space*, denoted, $W^{l,p}(U)$ is a function space with a norm that incorporates distances between derivatives of elements. For $l \geq 1$ and $p \in [1, \infty)$, $W^{l,p}(U)$ is the norm completion of the space of functions whose derivatives up to order l are p-integerable. The rigorous definition appears in appendix B, but for the purposes of this report, it suffices to think of elements of $W^{l,p}(U)$ as functions from U to $\mathbb{R}$, and work with the subspace $W_c^{l,p}(U)$ consisting of continuous functions. We will first define a norm on continuous functions called the $L_c^p - norm$, which we use to measure the distance between functions.

Definition 3. *For $U \subseteq \mathbb{R}^d$, and $p \in [1, \infty)$ the **$\mathbf{L_c^p}$-norm** of a continuous function $f : U \to \mathbb{R}$ is*

$$\|f\|_{L_c^p(U)} := \left(\int_U |f(u)|^p du \right)^{\frac{1}{p}}.$$

We write $L_c^p(U)$ for the collection $\{f : U \to \mathbb{R} \mid f$ is continuous and $\|f\|_{L_c^p(U)} < \infty\}$, and sometimes refer to it as an L_c^p-space.

It is easy to show that $(L_c^p(U), \|\cdot\|_{L_c^p(U)})$ is a normed vector space, but for the sake of brevity, we will omit the details. Note that it is common to prefer $p = 2$, because using the proper definition of an L^p space from Definition 6, $L^2(U)$ is what is known as a Hilbert space, meaning we can define an inner product on it.

The Sobolev norm builds on the L_c^p norm by incorporating the L_c^p norms of derivatives of the function of interest.

Definition 4. *For $U \subseteq \mathbb{R}^d$, $p \in [1,\infty)$, and $l \geq 1$, we define*

$$W_c^{l,p}(U) := \left\{ f \in L_c^p(U) \mid \partial^\alpha f \in L_c^p(U) \text{ for all } \alpha \in \mathbb{N}_0^d \text{ with } \|\alpha\|_1 \leq l \right\}.$$

*The **Sobolev norm** of $f \in W_c^{l,p}(U)$ is*

$$\|f\|_{W_c^{l,p}(U)} := \left(\sum_{\|\alpha\|_1 \leq l} \|\partial^\alpha f\|_{L_c^p(U)}^p \right)^{\frac{1}{p}}$$

So given a sequence $(\phi_n)_{n=1}^\infty$ in $W_c^{l,p}(U)$, $\phi_n \to f \in W_c^{l,p}(U)$ with respect to the $W_c^{l,p}(U)$ norm if and only if for every $\alpha \in \mathbb{N}_0^d$ such that $\|\alpha\|_1 \leq l$, $\partial^\alpha \phi_n \to \partial^\alpha f$ with respect to the L_c^p norm. However, note that $W_c^{l,p}(U)$ and $L_c^p(U)$ as we have defined them are not complete, so $\|\cdot\|_{W_c^{l,p}(U)}$ and $\|\cdot\|_{L_c^p(U)}$-Cauchy sequences need not converge in $W_c^{l,p}(U)$ and $L_c^p(U)$ respectively.

In the context of Gaussian process and differential equations, for $(f_n^p)_{n=1}^\infty$ a sequence of function approximations to the solution to an lth order DE using Equation (8), if $(f_n^p)_{n=1}^\infty$ converges to $f \in W_c^{l,p}(U)$ as $n \to \infty$ with respect to the $\|\cdot\|_{W_c^{l,p}(U)}$ norm, then $\partial^\alpha f_n^p \to \partial^\alpha f$ with respect to the L_c^p norm whenever $\|\alpha\|_1 \leq l$, meaning the limit f will satisfy the DE.

6 Convergence of Gaussian Processes

In this section, we want to link together Gaussian processes and Sobolev spaces by presenting the ways in which Sobolev spaces can be used to characterise the convergence of the function approximations resulting from Gaussian process regression. The results in this section are ideas presented in [5] and [4], reworded into the notation used in this report.

Firstly, proposition 3.1 from [4] highlights the link between Sobolev spaces, our choice of kernel function and the function we are trying to approximate.

Theorem 1. *Let $Z = \{Z(u) \mid u \in U\}$ be a Gaussian process with mean function zero and covariance function $k(\theta;\cdot,\cdot)$. Then, for $p \in (1,\infty)$, the sample paths of Z lie in $W_c^{l,p}(U)$ almost surely if and only if for all $\|\alpha\|_1 < l$, $\partial^{\alpha,\alpha} k(\theta;\cdot,\cdot) \in L_c^p(U \times U)$, and $s_\alpha \in L_c^p(U)$ for $s_\alpha(\theta;u) := \sqrt{\partial^{\alpha,\alpha} k(\theta;u,u)}$. (proposition 3.1 in [4])*

Proof. [4], section 3. □

Note for a Gaussian process Z, a sample path is a function $h : U \to \mathbb{R}$ where $h(u)$ is a realisation of $Z(u)$ for all $u \in U$. The most relevant example of a sample path is f_n^p, so Theorem 1 this is essentially telling us that our kernel function determines the Sobolev space that our predictive function lives in.

This idea of sample paths being in a specific Sobolev space can be extended to convergence using Theorem 3.5 from [5], and applies in the case of a Matérn covariance kernel or Gaussian kernel as defined in Equations (2) and (1) respectively.

Theorem 2. *Suppose that we want to approximate the function $f : U \to \mathbb{R}$ using Gaussian process regression with some training data, D_n and Y_n as defined in section 3.2. Suppose we have a sequence of estimates for θ, $(\hat{\theta}_n)_{n=1}^{\infty}$, that is, our maximum likelihood estimate, $\hat{\theta}$ for θ given training data of size n (which design points we are using are irrelevant). Let $(f_n^p)_{n=1}^{\infty}$ be a sequence of function approximations as defined in Equation (8). $\tilde{\nu}$ is the true smoothness of f, as introduced in section 3.3. Suppose that the following hold*

1. *$(\hat{\theta}_n)_{n=1}^{\infty}$ is uniformly bounded, that is $\{\hat{\theta}_n \mid n \in \mathbb{N}\} \subset S \subseteq \Theta$ for some compact set S.*

2. *$cl(U)$ is compact.*

3. *For any $\theta \in S$, $k(\theta; \cdot, \cdot) \in V \cong W_c^{\hat{\nu},2}(U)$, that is, the space that our kernel function lives in is isomorphic to $W_c^{\hat{\nu},2}(U)$.*

4. *$f \in W_c^{\tilde{\nu},2}(U)$ for $\tilde{\nu} > \max\left\{1, \frac{d}{2}\right\}$*

5. *For all possible estimates $\theta \in S$, and all $n \in \mathbb{N}$, $f_n^p \in W_c^{\tilde{\nu},2}(U)$ for f_n^p as defined in Equation (8).*

6. *There exists an $N \in \mathbb{N}$ such that $\hat{\nu}_{min} := \inf_{n \geq N} \hat{\nu}_n$ satisfies $\hat{\nu}_{min} = c$ for some fixed $c \geq \max\left\{1, \frac{d}{2}\right\}$.*

Then $\|f - f_n^p\|_{W_c^{\beta,2}(U)}$ is bounded by a decreasing bound for all $\beta \leq \tilde{\nu}$.

Proof. [5], section 3.1.1. $\square$

Of the list of requirements in Theorem 2 for f_n^p to converge, we can check 3 and 5, as these are essentially determined by what choice of kernel function we are using. 1 and 4 have to be assumed, as we cannot know what all our possible estimated parameters could be, however, when applying to differential equations, we will know the minimal number of derivatives that f must have, and 2 and 5 can be ensured by design - we can make sure that the closure of the set U we choose is compact by restricting our DE to only that domain, and we can fix our estimate for $\hat{\nu}$ (choose the same value for each $\hat{\nu}_n$), therefore ensuring that it is bounded below.

Once we have made sure of all of these, we can be sure that f_n^p does converge, so we can be sure that our prediction is accurate, and also find a bound. The bound can be seen in [5] Theorem 3.5. We will not define that bound here, as it requires the introduction of lots of new notation. The main takeaway is that we can use Gaussian processes to approximate the solutions to differential equations, and by ensuring that our chosen kernel function lives in an appropriate Sobolev space aligning with the smoothness of our differential equation, we can be sure that our Gaussian process approximation is accurate.

7 Conclusion

Given the motivation of wanting to approximate unknown solutions to differential equations given numerical solutions, this report explores a statistical process known as a Gaussian process. We define a Gaussian process in section 3.1, and in sections 3.2-3.4, we explain how, using an approach known as Gaussian process regression, we can estimate the solutions to differential equations. We want to know how accurate these approximations were, so section 5 explores special types of function spaces – Sobolev spaces – that can be used to relate a function to its first l derivatives. Finally, in section 6, we explore key theorems from [4] and [5] that can be used to characterise if Gaussian process approximations of our solution converge to the true solution as the number of numerical approximations goes to infinity, and determine how large the error bounds are given n numerical approximations. From this, we know what factors of our design we need to fix to ensure convergence of our approximations to the true solution of the differential equation, and can ensure that we apply Gaussian process regression in such a way that our approximation will converge.

Given characteristics that determine convergence, and the link between convergence of Gaussian processes, Sobolev spaces and differential equations, we can extend these results to many differential equations with unknown solutions.

Differential equations are very useful in physics and other fields for modeling. Given a differential equation with an unknown solution, this research can be used to approximate the solutions to that differential equations for the purpose of using them for modeling in physics or otherwise. It is common for mathematicians to be able to characterise what Sobolev space differential equations, especially partial differential equations (PDEs) lie in. [4], section 1 provides a list of PDEs where we know the Sobolev space the solution lies in, including the wave equation, heat equation, Laplace's equation and Schrodinger's equation, with lie in the Hilbert-Sobolev spaces over $\mathbb{R}^2$ or $\mathbb{R}^2$ corresponding to $l = 1$. Given this, we can use Gaussian process regression to approximate the solutions to differential equations in such a way that we can ensure convergence.

[10] also explores the use of the Navier Stokes equation in modeling the movement of underwater pipes in Western Australia, and provides numerical approximations of the solution at 9 points. Given an analysis on what Sobolev space the solution of the Navier Stokes equation could live in, we could model the solution using Gaussian process regression and save much computational power.

References

[1] Jørgensen, J 1994, *Probability With a View Towards Statistics, Volume II*, Chapman & Hall/CRC

[2] Chkrebtii, O.A, Campbell, D.A, Calderhead, B & Girolami, M.A, 2016, 'Bayesian Solution Uncertainty Quantification for Differential Equations', *Bayesian Analysis*, vol. 11, no. 4, pp. 1239–1267.

[3] Gramacy, R.B 2020, Surrogates: *Gaussian Process Modeling, Design and Optimization for the Applied Sciences*, Chapman & Hall/CRC, Boca Raton, Florida.

[4] Henderson, I 2024, 'Sobolev regularity of Gaussian random fields', *Journal of Functional Analysis*, vol. 286, no. 3, pp.110241

[5] Teckentrup, A.L 2020, 'Convergence of Gaussian Process Regression with Estimated Hyper-Parameters and Applications in Bayesian Inverse Problems', *SIAM/ASA Journal on Uncertainty Quantification*, vol. 8, no. 4, pp. 1310–1337

[6] Stein, E.M & Shakarchi, R 2005, *Real Analysis: Measure Theory, Integration, and Hilbert Spaces*, Princeton University Press, Princeton, New Jersey.

[7] Wikipedia 2001, *Equivalence relation*, viewed 15 December, 2023, <https://en.wikipedia.org/wiki/Equivalence_relation#Definition>

[8] Wikipedia 2002, *L^p space*, viewed 10 December, 2023, <https://en.wikipedia.org/wiki/Lp_space#Lp_spaces_and_Lebesgue_integrals>

[9] Wikipedia, 2004, *Sobolev space*, viewed 10 December, 2023, <https://en.wikipedia.org/wiki/Sobolev_space>

[10] Jiang, H 2022 "Flow past a circular cylinder: Fundamentals and engineering applications" *Oceans Graduate School, University of Western Australia*, viewed 18th September 2023, <https://www.bilibili.com/video/BV1a44y1D7yk/>

Appendix A Code used to find μ_p and Σ_p

Code adapted from [3] sections 5.1-5.3.

```
true_soln <- function(u){
  return(exp(-0.5*u))
}
f_dash <- function(u,fu){ #of the form f'(u) = g(u,f(u))
  return(-0.5*fu)
}
#re-writing the function in the way that "euler" likes it
f_dash2 <- function(t,y,parms){
  return(list(f_dash(t,y)))
}
#numerically calculate the the training data
library(deSolve)
Dn <- seq(0,5,by=0.5)
Yn <- euler(y=1,times = Dn,func = f_dash2,parms = c(0))[,2]
#get the true values and determine the error
fus <- true_soln(Dn)
error <- fus-Yn
#get the true values of f in a dense vector for the sake of plotting
```

```
us <- seq(Dn[1],Dn[length(Dn)],by=0.1)
true_fus <- true_soln(us)

plot(us,true_fus, type = "l", col = "blue",xlab = "u",ylab="f(u)",
main="Numerical Approximation of the Solution vs True Solution")
points(Dn,Yn,col="red",pch= 16)
legend(x = "topright",legend = c("Euler's","True"),col = c("red","blue"),lty = c(0,1),
pch = c(16,NA))

#now we want to fit a GP through these
norm2Vec <- function(x1,x2){ #function to determine the 2-norm of a vector
  n <- length(x1)#note this still works for matrices
  if (n == 1){
    return(abs(x1-x2))
  }
  else{
    sum = 0
    for (i in 1:n){
      sum = sum + (x1[i]-x2[i])^2
    }
    return (sum)^(1/2)
  }
}

cGausEnt <- function(x1,x2,lambda){ #exponential part of the Gaussian kernel
  r<- norm2Vec(x1,x2)
  return( exp(-r^2/lambda) )
}
cGausMatrix<- function(D1,D2,lambda){ # exponential part of the Gaussian kernel matrix
  D1 = as.vector(D1)
  D2 = as.vector(D2)
  n = length(D1)
  m = length(D2)
  K = matrix(rep(NA,n*m),nrow = n,ncol=m)
  if(identical(D1,D2) == TRUE){ #since it's symmetric
    for (i in 1:n){
      for (j in i:m){
        ijent <- cGausEnt(D1[i],D2[j],lambda)
        K[i,j] <- ijent -> K[j,i]
      }
    }
  }else{
  for (i in 1:n){
    for (j in 1:m){
      K[i,j] <- cGausEnt(D1[i],D2[j],lambda)
    }
  }}
  return(K)
}

loglikelihood <- function(par, Dn, Y) #concentrated log likelihood
{
  lambda <- par[1]
  g <- par[2]
  n <- length(Dn)
  K <- cGausMatrix(Dn,Dn,lambda) + diag(g, n)
  Ki <- solve(K)
  ldetK <- determinant(K, logarithm=TRUE)$modulus
  ll <- - (n/2)*log(t(Y) %*% Ki %*% Y) - (1/2)*ldetK
  return(ll)
}

negloglikelihood <- function(par,Dn,Y){ # negative log likelihood so we can use optim
  return(-1*loglikelihood(par,Dn,Y))
}

#find the maximum of the log likelihood - use the optim function
eps <- sqrt(.Machine$double.eps)
parm <- optim(c(0.1, 0.1*var(Yn)), fn = negloglikelihood , lower = eps,upper = c(10,var(Yn)),
method="L-BFGS-B",Dn = Dn, Y=Yn)

lambda = parm$par[1]
g = parm$par[2]
c(lambda,g)
Cg <- cGausMatrix(Dn,Dn,lambda) +  diag(g, length(Yn)) #covariance matrix
cholCg <- t(chol(Cg)) #the cholesky decomposition of C (lower triangular)
Cginv <- chol2inv(t(cholC))
tau2hat <- drop((t(Yn) %*% Cginv %*% Yn) / length(Yn))
tau2hat
K = tau2hat*Cg
```

```
Kinv = (1/tau2hat)*Cginv

#now, say our new data is;
Xm <- seq(0.05,5.55,by = 0.05)
length(Xm)
#the correlation matrix between the new data and old data is
KXm_Dn <- tau2hat*cGausMatrix(Xm,Dn,lambda)
KXm_Xm <- tau2hat*(cGausMatrix(Xm,Xm,lambda) + g*diag(1,length(Xm)))

#predictive mean and covariance
mup = KXm_Dn%*% Kinv %*%Yn
sigmap = KXm_Xm - KXm_Dn%*%Kinv%*%t(KXm_Dn) + g*diag(1,length(Xm))
dense_True_fus <- rep(NA,length(Xm))
for (i in 1:length(Xm)){ #get the true values of f at each of our new points.
  dense_True_fus[i] <- true_soln(Xm[i])
}

plot(Xm,mup,col="black",xlab = "u",ylab = "f(u)",main = "Gaussian Process Approximation",
type="p",pch = 20)
points(Dn,Yn,col="red",pch=16)
lines(Xm,dense_True_fus, col = "blue")
legend(x = "topright",legend=c("Euler's","True","Gaussian Process"),
col = c("red","blue","black"),lty=c(0,1,1),pch=c(16,20,NA))
```

Note for Appendix C, all code is the same except for labels on graphs, and `Yn` is determined by

```
Yn <- rk4(y=1,times = Dn,func = f_dash2,parms = c(0))[,2]
```

Appendix B Proper Definition of L^p Spaces and Sobolev Spaces

In section 5, we defined a Sobolev space and L^p space only over continuous functions. The proper definition of an L^p space and Sobolev space are over all measurable functions.

Note that this section will assume some measure theory background, including the definition of a measure, a measure space, and a measurable function. All of these definitions, as well as other relevant background information can be found in [6] chapter 1 and 2. This section also requires knowledge of what an equivalence relation is, and what an equivalence class is, which can be found in [7], and knowledge of what a weak derivative is, which is defined in the introduction of [4]. We will first properly define an L^p space following from [8]. This firstly requires us to first define a precursor to an L^p space; an L_0^p space.

Definition 5. *For U a measurable set, and $(U, \mathcal{B}(U), \gamma)$ a measure space, where $\mathcal{B}(U)$ is a Borel-algebra on U,* $\mathbf{L_0^p(U)}$ *is given by*

$$L_0^p(U) := \left\{ \textit{measurable functions } f : U \to \mathbb{R} \mid \left(\int_U |f|^p d\gamma \right)^{\frac{1}{p}} < \infty \right\}.$$

It can be shown that $L^p(U)$ is a real vector space with zero element $0_{L_0^p(U)} : U \to \mathbb{R}$ such that $0_{L_0^p(U)}(u) = 0$ for all $u \in U$. We want L_0^p to be a normed vector space, so we will attempt to define a norm on L_0^p, given by $N : L_0^p(U) \to [0, \infty)$ such that

$$N(f) := \left(\int_U |f|^p d\gamma \right)^{\frac{1}{p}}. \tag{12}$$

Lemma 1. *N is not a norm*

Proof. Consider the function $g \in L_0^p(U)$, where for some $u^* \in U$,

$$g(u) := \begin{cases} 1 & u = u^* \\ 0 & \text{otherwise.} \end{cases}$$

It can be shown that $N(g) = 0$. The definition of a norm requires that for N to be a norm, $N(f) = 0$ if and only if $f = 0_{L^p(U)}$. We saw that $N(g) = 0$, but $g \neq 0_{L^p(U)}$. Therefore, N is not a norm. $\square$

In order for $L_0^p(U)$ to be a normed vector space with norm N we need to prevent functions other than $0_{L_0^p(U)}$ from having a norm of zero. Therefore, $L^p(U)$ is defined on equivalence classes.

Definition 6. *Consider the equivalence relation $\sim$ such that $f \sim g$ if $N(g - f) = 0$. That is, $g \sim f$ if the subset $A \subset U$ given by $A := \{a \mid f(a) \neq g(a)\}$ has measure zero ($\gamma(A) = 0$).*

Let $(U, \mathcal{B}(U), \gamma)$ be some measure space for $U \subseteq \mathbb{R}^d$. Then for $p \in [1, \infty)$, $\mathbf{L^p(U)}$ *is given by*

$$L^p(U) := L_0^p(U)/\sim .$$

Here, $L_0^p(U)/\sim$ represents the quotient space of $L_0^p(U)$ with respect to $\sim$.

It can be shown that $\|\cdot\|_{L^p(U)} : L^p(U) \to \mathbb{R}$ given by

$$\|[f]\|_{L^p(U)} = N(f) \text{ for some } f \in [f]$$

is a norm on $L^p(U)$. In fact, it can also be shown using dual spaces that $L^p(U)$ is complete with respect to $\|\cdot\|_{L^p(U)}$.

We can now use this to define $W^{l,p}(U)$ following from [9]. For $f \in [f] \in L^p(U)$, let $\partial^\alpha f$ denote the αth weak derivative of f, then,

Definition 7. *For $p \in [1, \infty)$, $l \geq 1$, the **Sobolev space** $W^{l,p}(U)$ is defined as*

$$W^{l,p}(U) = \left\{ [u] \in L^p(U) \mid \textit{ for all } \alpha \in \mathbb{N}_0^d \textit{ such that } \|\alpha\|_1 < l, \partial^\alpha u \in L^p(U) \textit{ for all } u \in [u] \right\}$$

Definition 8. *The* ***Sobolev norm*** *for* $[f] \in W^{l,p}(U)$ *is given by*

$$\|[f]\|_{W^{l,p}(U)} := \left(\sum_{\|\alpha\|_1 \leq l} \|\partial^\alpha f\|^p_{L^p(U)} \right)^{\frac{1}{p}} \quad \textit{for any } f \in [f].$$

Note that each equivalence class $[f] \in L^p(U)$ and $[f] \in W^{l,p}(U)$ only has one $f \in [f]$ that is continuous, so in section 5, given the context of differential equations, where we only care about continuous functions, defining $L^p_c(U)$ and $W^{l,p}_c(U)$ as the continuous elements of each equivalence class is valid, however, note there are $[f] \in L^p(U)$ and $[f] \in W^{l,p}(U)$ where no $f \in [f]$ is continuous, hence why $L^p_c(U)$ and $W^{l,p}_c(U)$ are not complete.

Appendix C Gaussian Process Regression on Training Data Calculated Using rk4

Following on from section 4, where we stated it is difficult to quantify the error resulting in numerical approximations, and often the only solution is to use numerical methods with negligible error, we performed Gaussian process regression on numerical approximations of Equation (10) that were determined using rk4 with a step size of $h = 0.5$. The approximations are shown in Figure 3, where we can see visually that the errors are close to zero. The errors are given by

$$\epsilon = (0, -0.781, -1.22, -1.42, -1.48, -1.44, 1.34, -1.22, -1.09, -0.951, -0.823) \times 10^{-5},$$

rounded to 3 significant figures, with $\operatorname{var}(\epsilon) = 1.844461 \times 10^{-11}$, which we notice is significantly less than those resulting from Euler's method, as seen in Equation (11). Our maximum likelihood estimates for θ are $\hat{\lambda} = 1, \hat{\tau^2} = 0.7931441$ and $\hat{g} = 1.490116 \times 10^{-8}$. Here, we notice that g is absolutely closer to the true value of $\operatorname{var}(\epsilon)$, and in fact over predicts our error. Then, using μ_p as defined in Equation (7) as an estimator for the values of f at the same X_m used in section 4, we can see in Figure 4 that the GP approximation of the solution to the DE is much more accurate.

Appendix D Maximum Likelihood Estimation for $\hat{\tau^2}$

Given the log likelihood function in Equation (4), for $\theta = (\tau^2, \lambda, g, (\nu))$, we want to determine the maximum likelihood estimator for τ^2.

Lemma 2. *The maximum likelihood estimator for* τ^2 *is*

$$\hat{\tau^2} := \frac{1}{n} Y_n^T \left(C + Ig\right)^{-1} Y_n.$$

Y_n, g and C are defined in sections 3.2, 3.3 and 3.4 respectively.

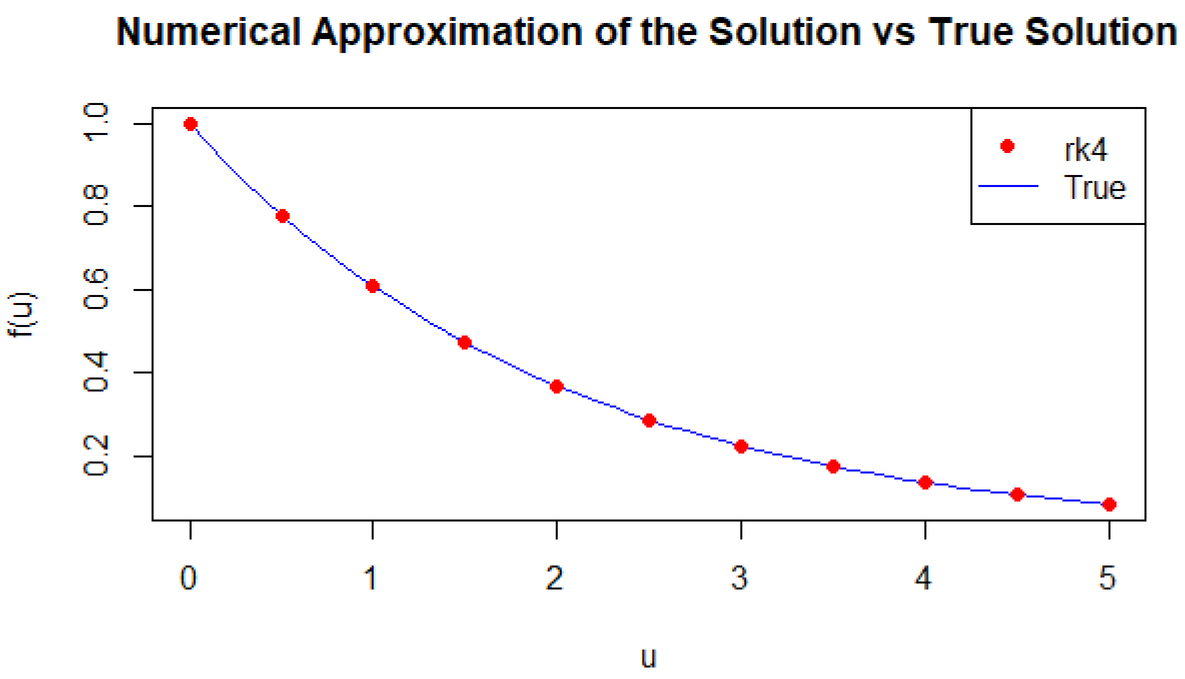

Figure 3: Numerical approximation of the solution of Equation (10) using the Classical 4th Order Runge Kutta Method

Proof. We find the maximum likelihood estimator for $\hat{\tau^2}$ by maximising the log likelihood function shown in Equation (4), that is

$$\ell(\theta; Y_n) = -\frac{1}{2}\left(n\ln(2\pi) + \ln\left(\det\left(K_n(\theta; D_n, D_n)\right)\right) + Y_n^T K_n(\theta; D_n, D_n)^{-1} Y_n\right).$$

Recalling that by definition of C, $K_n(\theta; D_n, D_n) = \tau^2\left(C + Ig\right)$, so we can re-write the log likelihood function as

$$\ell(\theta; Y_n) = -\frac{1}{2}\left(n\ln(2\pi) + \ln\left(\det\left(\tau^2(C+Ig)\right)\right) + Y_n^T(\tau^2(C+Ig))^{-1}Y_n\right).$$

Using the fact that $(\tau^2(C+Ig))^{-1} = \frac{1}{\tau^2}(C+Ig)^{-1}$ and $\det\left(\tau^2(C+Ig)\right) = (\tau^2)^n \det(C+Ig)$,

$$\begin{aligned} &= -\frac{1}{2}\left(n\ln(2\pi) + \ln\left((\tau^2)^n \det\left(C+Ig\right)\right) + \frac{1}{\tau^2}Y_n^T(C+Ig)^{-1}Y_n\right) \\ &= -\frac{1}{2}\left(n\ln(2\pi) + n\ln\left(\tau^2\right) + \ln\left(\det\left(C+Ig\right)\right) + \frac{1}{\tau^2}Y_n^T(C+Ig)^{-1}Y_n\right). \end{aligned}$$

Differentiating,

$$\frac{\partial}{\partial\tau^2}\ell(\theta; Y_n) = -\frac{1}{2}\left(\frac{n}{\tau^2} - \frac{1}{(\tau^2)^2}Y_n^T(C+Ig)^{-1}Y_n\right).$$

To find the stationary points, we want to find $\hat{\tau^2}$ such that

$$0 = -\frac{1}{2\hat{\tau^2}}\left(n - \frac{1}{\hat{\tau^2}}Y_n^T(C+Ig)^{-1}Y_n\right).$$

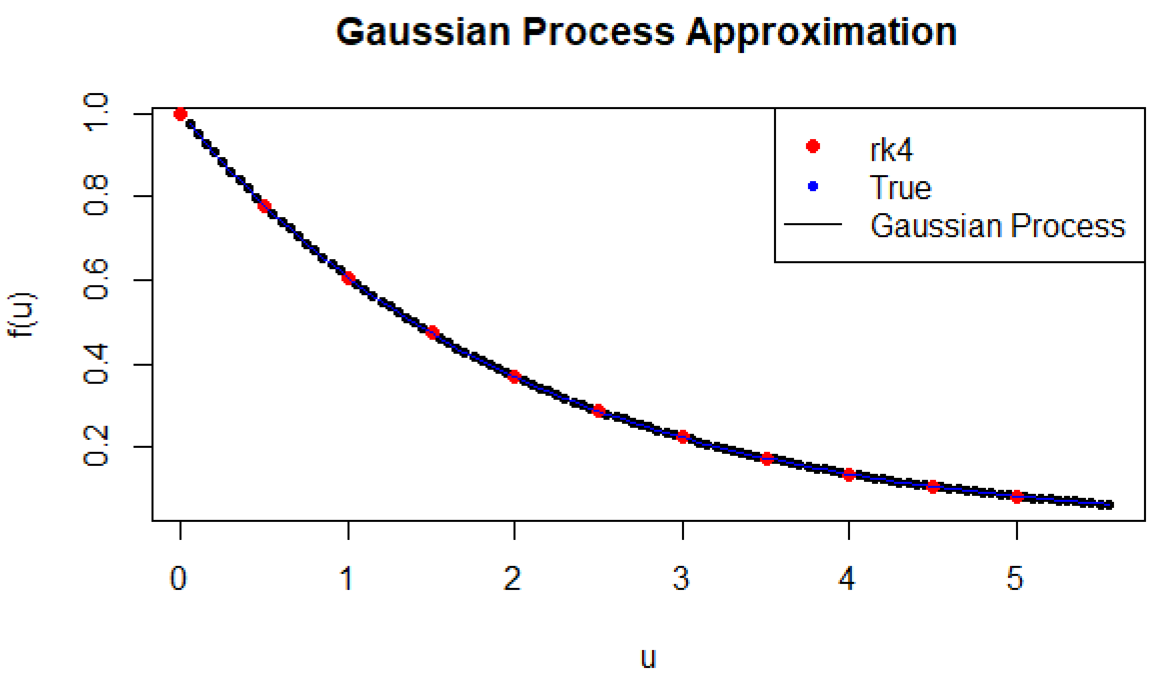

Figure 4: Gaussian process approximation of Equation (10), with training data calculated using the Classical 4th Order Runge Kutta Method

$\frac{1}{2\hat{\tau^2}}$ cannot be equal to zero, therefore, the only way this can be equal to zero is if

$$\begin{aligned} 0 &= n - \frac{1}{\hat{\tau^2}} Y_n^T (C + Ig)^{-1} Y_n, \text{ i.e,} \\ \hat{\tau^2} &= \frac{1}{n} Y_n^T (C + Ig)^{-1} Y_n. \end{aligned}$$

To show that $\hat{\tau^2}$ is in fact a maximum, we will take the second derivative.

$$\begin{aligned} \frac{\partial^2}{\partial(\tau^2)^2} \ell(\theta; Y_n) &= -\frac{1}{2}\left(-\frac{n}{(\tau^2)^2} + \frac{2}{(\tau^2)^3} Y_n^T (C + Ig)^{-1} Y_n\right) \\ &= -\frac{1}{2}\left(-\frac{n}{(\tau^2)^2} + \frac{2n}{(\tau^2)^3}\left(\frac{1}{n} Y_n^T (C + Ig)^{-1} Y_n\right)\right). \end{aligned}$$

Subbing in $\hat{\tau^2}$,

$$\begin{aligned} \frac{\partial^2}{\partial(\tau^2)^2} \ell(\hat{\tau^2}, \lambda, g, (v); Y_n) &= -\frac{1}{2}\left(-\frac{n}{(\hat{\tau^2})^2} + \frac{2n}{(\hat{\tau^2})^2}\right) \\ &= -\frac{1}{2} \frac{n}{(\hat{\tau^2})^2} < 0, \end{aligned}$$

therefore showing that $\hat{\tau^2}$ is in fact a maximum. Then, since $\hat{\tau^2}$ is a stationary point and a maximum, it is a maximum likelihood estimator for $\hat{\tau^2}$. □