Applications for Machine Learning coursework

Xinyu Zhong
Queens' College

Word count:

1 Introduction

Gaussian process provides alternative ways to simulate the time dependence of partial differential equations (PDEs).

This project aims to reproduce the result from the paper by [4] and [3] on solving time-dependent partial differential equations using Gaussian processes. The project will focus on solving the Burger's equation and the wave equation using Gaussian processes. The project will also compare the results from Gaussian processes with the results from finite difference methods. The choice of the kernel in the Gaussian process will also be discussed.

The problem to solve is to find the solution to the time-dependent partial equation with the given initial condition and boundary condition. Currently, the most popular solution to the time-dependent partial equation is solved using finite difference methods such as the forward Euler method, backward Euler method, and Crank-Nicolson method. These methods are widely used in solving time-dependent partial equations due to their simplicity and efficiency[1]. In this project, these finite difference methods will be incorporated into the Gaussian process to solve the time-dependent partial equation, generating result that not only including the solution but also the uncertainty of the solution.

The report is structured as follows. Section 2 provides the background theory on Gaussian processes. Section 3 discusses the implementation of the backward Euler method using Gaussian processes. Section 4 compares the backward Euler method using Gaussian processes with the backward Euler method using finite differences.

2 Background theory on Gaussian processes

2.1 Gaussian process

Gaussian Process is a non-parametric method that can be used for regression, classification, and optimization problems. It is a collection of random variables, any finite number of which have a joint Gaussian distribution. It is completely specified by its mean function and covariance function[5].

The advantage of Gaussian Process includes a few key points [5]:

- It is a non-parametric method, which means that it does not assume a specific functional form for the data.
- It provides a measure of uncertainty for the predictions.
- It has a flexible kernel function that can be chosen based on the physical properties of the data.

Although it comes with a few drawbacks[5]:

- It is computationally expensive for large datasets.
- It is not suitable for high-dimensional data.

This project will explore the desirable property of Gaussian Process that predictions comes with an uncertainty estimation, while keep the dimension and the size of the data small so that the computational cost is manageable on a personal device.

Gaussian process can be used solve regression problems, classification problems, and optimization problems[5]. In this project, we will focus on using Gaussian processes regression problems, in particular, solving partial differential equations.

2.2 Gaussian process regression

Gaussian process can be used to solve regression problems. A Gaussian process is completely specified by its mean function and covariance function. i.e.

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')) \tag{1}$$

where $m(x)$ is the mean function and $k(x, x')$ is the covariance function. The mean function is usually set to zero, and the covariance function is usually chosen to be a squared exponential kernel, or other kernels that are suitable for the data.

Suppose we have measured value at n points $\mathbf{X} = \{x_1, x_2, \ldots, x_n\}$, and the corresponding values $\mathbf{f} = \{f(x_1), f(x_2), \ldots, f(x_n)\}$, and we are interested in predicting the value at a new point x_*, the joint distribution of the observed values and the predicted values is given by:

The joint distribution of the observed values and the predicted values is given by:

$$\begin{bmatrix} \mathbf{f}_* \\ \mathbf{f} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_* \\ \mu \end{bmatrix}, \begin{bmatrix} \mathbf{k}_{**} & \mathbf{k}_*^{\mathbf{T}} \\ \mathbf{k}_* & \mathbf{K} \end{bmatrix} \right) \tag{2}$$

where $\mathbf{f}_*$ is the predicted value at x_*, μ_* is the mean of the predicted value. And the block matrices are defined as:

$$\begin{aligned} \mathbf{k}_{**} &= k(x_*, x_*) \\ \mathbf{k}_* &= k(x_*, \mathbf{X}) \\ \mathbf{K} &= k(\mathbf{X}, \mathbf{X}) \end{aligned} \tag{3}$$

The mean and variance of the predicted value is given by:

$$\mu_* = \mathbf{k}_*^{\mathbf{T}} \mathbf{K}^{-1} \mathbf{f} \tag{4}$$

$$\sigma_*{}^2 = \mathbf{k}_{**} - \mathbf{k}_*^{\mathbf{T}} \mathbf{K}^{-1} \mathbf{k}_* \tag{5}$$

The equation (**??**) is the mean of the predicted value. The equation (**??**) is the variance of the predicted value. It provides a measure of uncertainty for the prediction, which is a key advantage of Gaussian processes.

2.3 Kernel function

The choice of the kernel function is important in Gaussian processes. A well-chosen kernel function can capture the physical properties of the data, and provide a good prediction. It should also be easy to implement, i.e. the derivatives of the kernel function can be computed analytically.

The squared exponential kernel is a popular choice for the kernel function, and it is defined as:

$$k(x, x') = \sigma^2 \exp\left(-\frac{(x - x')^2}{2l^2}\right) \tag{6}$$

The squared exponential kernel is smooth and infinitely differentiable, which makes it a good choice for the kernel function for describing continuous data. The squared exponential kernel is also known as the radial basis function (RBF) kernel, and it is a stationary kernel, which means that the covariance between two points is only dependent on the distance between the two points. This property may not be suitable for some data, such as the Burger's equation, where the kernel function should be non-stationary. The kernel function chosen for the Burger's equation will be discussed in the next section.

2.4 Gaussian process for solving partial differential equations

3 Implementation

3.1 Implementation

The implementation of Gaussian Process on differential equation is as follows:

1. The initial data $\{\mathbf{x}^0, \mathbf{u}^0\}$ and boundary data$\{\mathbf{x}_b^1, \mathbf{u}_b^1\}$ are collected or generated.[1]
2. A model is trained using the initial data and boundary data, to find the hyperparameters of the kernel functions.
3. The model is used to predict the value of the function at the next time step $\mathbf{u}^1$.
4. The predicted value $\{\mathbf{x}^1, \mathbf{u}^1\}$ and the boundary data $\{\mathbf{x}_b^2, \mathbf{u}_b^2\}$ are used as the initial data for the next time step. The process is repeated until the desired time step is reached.

The parameters of the kernel function are updated in every time step to improve the prediction accuracy. To speed up the training process, each training uses the hyperparameters of the previous time step as the initial guess for the hyperparameters of the current time step. The hyperparameters are updated using Adam optimizer with a learning rate of 0.01.

3.2 Kernel Selection

The kernel function is an important component of the Gaussian process, as it captures the physical properties of the data. The choice of the kernel function in the projects involves the following criteria:

1. **Ease of implementation:** The kernel function should be easy to implement and understand. The implementation of the kernel function should be straightforward and should not require complex mathematical operations. The analytic form of the derivative of the kernel function should also be easy to compute.
2. **Capture of physical properties:** The kernel function should capture the physical properties of the data. For example, the kernel function for the Burger's equation should be non-stationary, as the covariance between two points is dependent on the location of the points. Whereas, the kernel function for the wave equation should be stationary, as the covariance between two points is only dependent on the distance between the two points.

The choice of the kernel function for the Burger's equation is:

$$k(x, x') = \frac{2}{\pi} \sin^{-1}\left(\frac{2(\sigma_0^2 + \sigma^2 x^T x')}{\sqrt{(1 + (2\sigma_0^2 + \sigma^2 x^T x))(1 + 2((2\sigma_0^2 + \sigma^2 x'^T x')))}}\right) \tag{7}$$

The derivative of the kernel function is obtained using Mathematica and shown in the appendix.

[1] The exponent term is used to denote the time step, and the subscript b is used to denote the boundary data.

The choice of the kernel function for the wave equation is the RBF kernel, which is defined as:

$$k(x, x') = \sigma^2 \exp\left(-\frac{(x - x')^2}{2l^2}\right) \tag{8}$$

The derivative of the kernel function is less tedious to compute and is shown in the appendix.

3.3 Library

This project is built on the GPJax Library, which is a Gaussian process library built on JAX. The library provides a simple interface to train and predict using Gaussian processes. Apart from the standard kernel functions, the library also provides a ways to implement custom kernel functions[2].

3.4 Implementation of Burger's Equation

The Burger's equation is a non-linear partial differential equation, which is given by:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2} \tag{9}$$

The target is to simulate the propagation of the Burger's equation in the domain of $x \in [-1, 1]$ and $t \in [0, 0.5]$.

We solve this equation together wit the Dirichlet boundary condition that $u(t, -1) = u(t, 1) = 0$ and the initial condition that $u(0, x) = -\sin(\pi x)$.

The behaviour of Burger's equation, simulated using Rutta-Kunge method, is shown in the Figure **??**. The simulation is done with $\nu = 0.01$ and $\Delta x = 0.01$. The simulation shows the formation of sharp and continuous gradient at the centre of the domain, which is the characteristic of the Burger's equation.

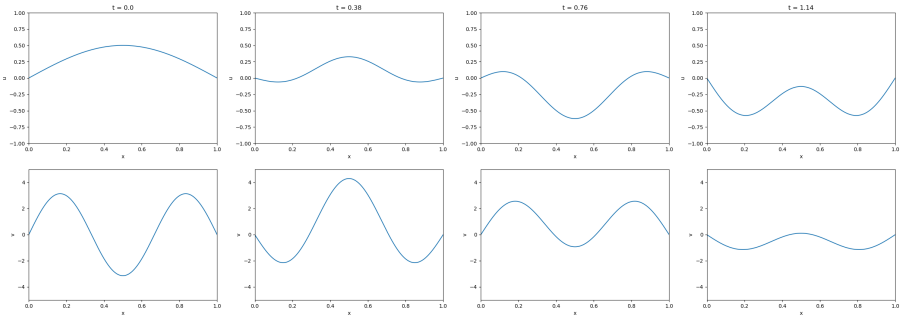

Fig. 1: Simulation of Burger's equation using Rutta-Kunge method, with $\nu = 0.01$ and $\Delta x = 0.01$. The simulation shows the formation of sharp and continuous gradient at the centre of the domain.

The implementation of the Burger's equation using Gaussian processes is shown in the appendix.

We construct the following Gaussian Process:

$$\begin{bmatrix} \mathbf{f}_* \\ \mathbf{f} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} k_{u,u}^{n,n} & k_{u,u}^{n,n-1} \\ k_{u,u}^{n-1,n} & k_{u,u}^{n-1,n-1} \end{bmatrix}\right) \tag{10}$$

with the covariance matrices defined as:

$$\begin{aligned}
k_{u,u}^{n,n} &= k := k(x, x') \\
k_{u,u}^{n,n-1} &= k + \Delta t \mu^{n-1}(x')\frac{d}{dx'}k - \nu\Delta t\frac{d^2}{dx'^2}k \\
k_{u,u}^{n-1,n-1} &= k + \Delta t \mu^{n-1}(x')\frac{d}{dx'}k - \nu\Delta t\frac{d^2}{dx'^2}k, \\
&\quad + \Delta t \mu^{n-1}(x)\frac{d}{dx}k + \Delta t^2 \mu^{n-1}(x)\mu^{n-1}(x')\frac{d}{dx}\frac{d}{dx'}k \\
&\quad - \nu\Delta t^2 \mu^{n-1}(x)\frac{d}{dx}\frac{d^2}{dx'^2}k - \nu\Delta t\frac{d^2}{dx^2}k \\
&\quad - \nu\Delta t^2 \mu^{n-1}(x')\frac{d^2}{dx^2}\frac{d}{dx'}k + \nu^2\Delta t^2\frac{d^2}{dx^2}\frac{d^2}{dx'^2}k
\end{aligned} \tag{11}$$

where the kernel function is defined as:

$$k(x, x') = \frac{2}{\pi}\sin^{-1}\left(\frac{2(\sigma_0^2 + \sigma^2 x^T x')}{\sqrt{(1 + (2\sigma_0^2 + \sigma^2 x^T x))(1 + 2((2\sigma_0^2 + \sigma^2 x'^T x')))}}\right) \tag{12}$$

where σ_0 and σ are the hyperparameters of the kernel function. The derivative of the kernel function are shown in the appendix.

3.4.1 Implementation of Wave Equation

The wave equation is a second-order partial differential equation, which is given by:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2} \tag{13}$$

The target is to simulate the propagation of the wave equation in the domain of $x \in [0, 1]$ and $t \in [0, 1]$. Given that it is not a linear equation, we need to split the equation into two first-order equations:

$$\begin{aligned} \frac{\partial u}{\partial t} &= v \\ \frac{\partial v}{\partial t} &= c^2 \frac{\partial^2 u}{\partial x^2} \end{aligned} \tag{14}$$

The true value of the wave equation is shown in the Figure **??**. The simulation is done with $c = 1$ and $\Delta x = 0.01$. The simulation shows the formation of the wave propagating from the left to the right of the domain.

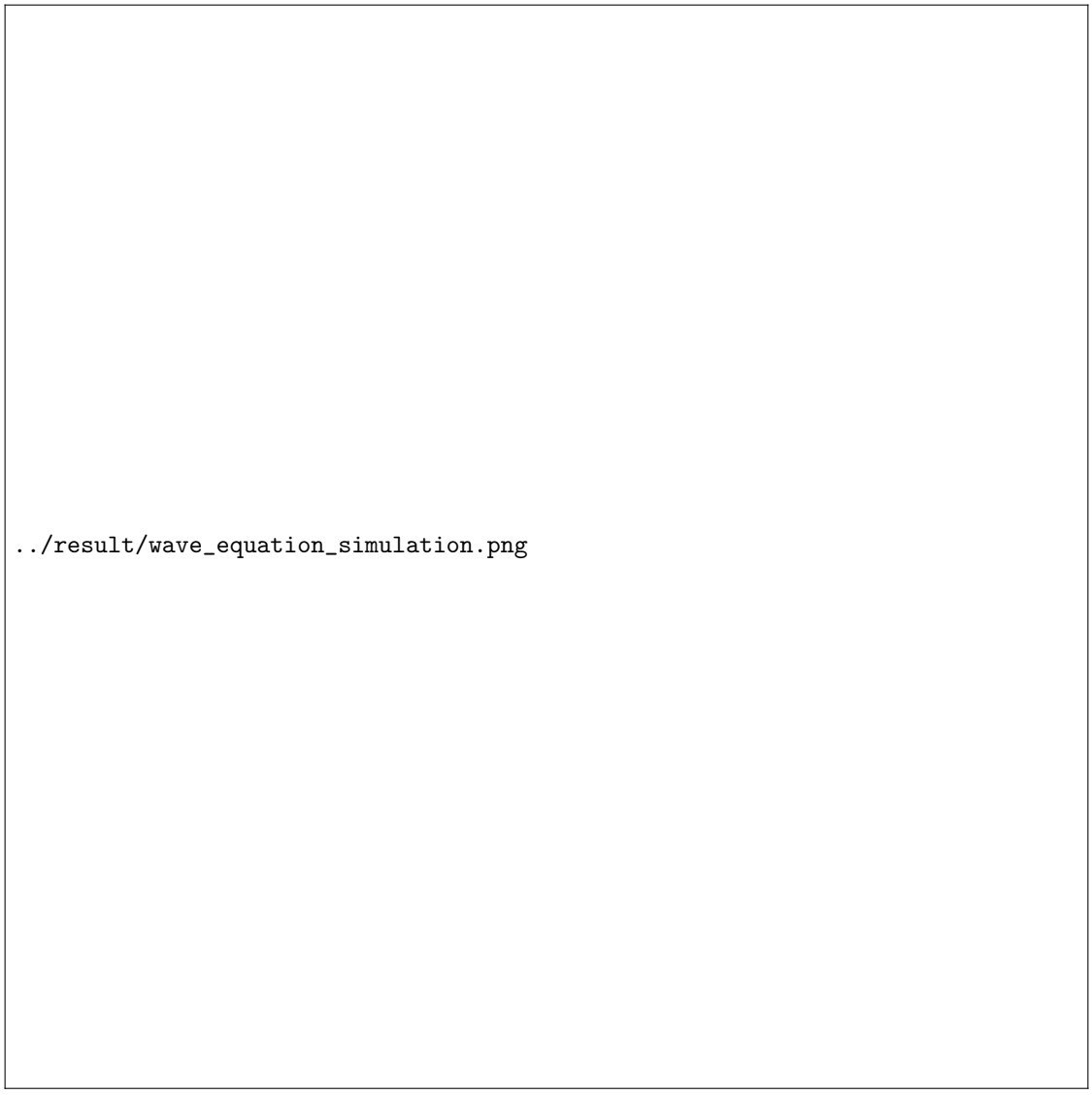

Fig. 2: Simulation of the wave equation using Rutta-Kunge method, with $c = 1$ and $\Delta x = 0.01$. The simulation shows the formation of the wave propagating from the left to the right of the domain.

We apply the trapezoidal rule to solve the wave equation, which is given by:

$$\begin{aligned} u^{n+1} &= u^n + \frac{\Delta t}{2}(v^n + v^{n+1}) \\ v^{n+1} &= v^n + \frac{\Delta t}{2}(c^2 \frac{\partial^2 u^n}{\partial x^2} + c^2 \frac{\partial^2 u^{n+1}}{\partial x^2}) \end{aligned} \tag{15}$$

We solve this equation together with the Dirichlet boundary condition that $u(t,-1) = u(t,1) = 0$ and the initial condition that $u(0,x) = -\sin(\pi x)$ and $\frac{\partial u}{\partial t}(0,x) = 0$.

4 Analysis

4.1 Accuracy of the Gaussian process

4.2 Uncertainty Estimation

Uncertainty estimation is one of the unique features of Gaussian Process. The uncertainty of the prediction is accumulated at every time step, and is dependent on the error in the initial data. The magnitude of the uncertainty, with respect to the error in the initial data, is shown in Figure This feature is useful in many applications,

Note that it is not a feature of finite difference method. In the case of the finite difference method, the solution is deterministic, and the uncertainty of the solution is not available. Being able to evaluate the uncertainty of the solution is one of the main motivations to use Gaussian Process in this project.

4.3 Computational Cost

The downside of Gaussian Process is that it is computationally expensive for large datasets. The computational cost of Gaussian Process is $O(n^3)$, where n is the number of data points. This is because the inversion of the covariance matrix is required in the prediction step, and the inversion of a matrix is $O(n^3)$.

In practise, a few tricks are used to reduce the computational cost of Gaussian Process. For example, the Cholesky decomposition can be used to avoid the inversion of the covariance matrix, however, the computational cost is still $O(n^3)$.

Also, the derivatives of simpler kernels, e.g.the covariance can be computed analytically, which can reduce the computational cost of the kernel function evaluation.

The number of data points n, can be further decomposed into the number of points sampled n_{points} and the dimension of the data D. If we are dealing with high-dimensional data, the test vector x_* will be a described by a matrix of shape (n_{points}, D) that is reshaped into a vector of shape $(n_{points} \times D, 1)$. The computational cost of the Gaussian Process will be $O(n_{points}^3) \times O(D^3)$. Therefore, in practise, the Gaussian Process is not suitable for high-dimensional data. Also, the number of points sampled n_{points} should be chosen carefully to balance the computational cost and the coverage of the data.

A Use of generative AI

Chat-GPT 3.5, accessible for free at https://chat.openai.com/, was utilized for certain small sections of the source code. These instances, where using Chat-GPT provided a more efficient solution than other sources could offer, are clearly marked in the code with comments. Additionally, Chat-GPT helped in formatting some of the tables in this report using LATEX. The concepts and execution of this project are entirely my own, built upon the knowledge gained from this course's materials, my own experiences, and the cited references.

B Derivatives of the kernel function for Burger's equation

The derivatives of the kernel function for Burger's equation is a straightforward but tedious process. They can be computed using Mathematica. The kernel function for Burger's equation is defined as:

$$k(x,x') = \frac{2}{\pi}\sin^{-1}\left(\frac{2(\sigma_0^2+\sigma^2 xx')}{\sqrt{(1+(2\sigma_0^2+\sigma^2x^2))(1+2((2\sigma_0^2+\sigma^2x'^2)))}}\right) \tag{16}$$

The first order derivative of the kernel function with respect to x is:

$$\frac{\partial k(x,x')}{\partial x} = \frac{4\sigma^2(x'-2(x-x')\sigma_0^2)}{\pi(1+2x^2\sigma^2+2\sigma_0^2)^{3/2}\sqrt{1+2x'^2\sigma^2+2\sigma_0^2}\sqrt{1-\frac{4(xx'\sigma^2+\sigma_0^2)^2}{(1+2x^2\sigma^2+2\sigma_0^2)(1+2x'^2\sigma^2+2\sigma_0^2)}}} \tag{17}$$

The first order derivative of the kernel function with respect to x' is:

$$\frac{\partial k(x,x')}{\partial x'} = \frac{4\sigma^2(x+2(x-x')\sigma_0^2)}{\pi\sqrt{1+2x^2\sigma^2+2\sigma_0^2}(1+2x'^2\sigma^2+2\sigma_0^2)^{3/2}\sqrt{1-\frac{4(xx'\sigma^2+\sigma_0^2)^2}{(1+2x^2\sigma^2+2\sigma_0^2)(1+2x'^2\sigma^2+2\sigma_0^2)}}} \tag{18}$$

The other derivatives of the kernel function with respect to x and x' are:

$$\frac{\partial^2 k(x,x')}{\partial x\partial x'} = \frac{4\sigma^2(1+4\sigma_0^2)}{\pi\sqrt{1+2x^2\sigma^2+2\sigma_0^2}\sqrt{1+2x'^2\sigma^2+2\sigma_0^2}(1+2x^2\sigma^2+2x'^2\sigma^2+4(1+x^2\sigma^2-2xx'\sigma^2+x'^2\sigma^2)\sigma_0^2)\sqrt{1-\frac{4(xx'\sigma^2+\sigma_0^2)^2}{(1+2x^2\sigma^2+2\sigma_0^2)(1+2x'^2\sigma^2+2\sigma_0^2)}}} \tag{19}$$

$$\frac{\partial^2 k(x,x')}{\partial x^2} = \frac{8\sigma^2(-xx'\sigma^2(3+6x^2\sigma^2+4x'^2\sigma^2)+(-1+8x^4\sigma^4-24x^3x'\sigma^4-2xx'\sigma^2(9+8x'^2\sigma^2)+2x^2(\sigma^2+12x'^2\sigma^4))\sigma_0^2+2(-3+8x^4\sigma^4-24x^3x'\sigma^4-4xx'\sigma^2(3+2x'^2\sigma^2)+4x^2$$
$$\pi(1+2x^2\sigma^2+2\sigma_0^2)^{5/2}\sqrt{1+2x'^2\sigma^2+2\sigma_0^2}(1+2x^2\sigma^2+2x'^2\sigma^2+4(1+x^2\sigma^2-2xx'\sigma^2+x'^2\sigma^2)\sigma_0^2)\sqrt{1-\frac{4(xx'\sigma^2+\sigma_0^2)^2}{(1+2x^2\sigma^2+2\sigma_0^2)(1+2x'^2\sigma^2+2\sigma_0^2)}}} \tag{20}$$

$$\frac{\partial^2 k(x,x')}{\partial x'^2} = -\frac{8\sigma^2(xx'\sigma^2(3+4x^2\sigma^2+6x'^2\sigma^2)+(1-2x'^2\sigma^2+16x^3x'\sigma^4-24x^2x'^2\sigma^4-8x'^4\sigma^4+6xx'\sigma^2(3+4x'^2\sigma^2))\sigma_0^2+2(3-4x'^2\sigma^2+8x^3x'\sigma^4-24x^2x'^2\sigma^4-8x'^4\sigma^4+12x($$
$$\pi\sqrt{1+2x^2\sigma^2+2\sigma_0^2}(1+2x'^2\sigma^2+2\sigma_0^2)^{5/2}(1+2x^2\sigma^2+2x'^2\sigma^2+4(1+x^2\sigma^2-2xx'\sigma^2+x'^2\sigma^2)\sigma_0^2)\sqrt{1-\frac{4(xx'\sigma^2+\sigma_0^2)^2}{(1+2x^2\sigma^2+2\sigma_0^2)(1+2x'^2\sigma^2+2\sigma_0^2)}}} \tag{21}$$

$$\frac{\partial^3 k(x,x')}{\partial x\partial x'^2} = -\frac{24\sigma^4(1+4\sigma_0^2)(x'-2(x-x')\sigma_0^2)}{\pi\sqrt{1+2x^2\sigma^2+2\sigma_0^2}\sqrt{1+2x'^2\sigma^2+2\sigma_0^2}(1+2x^2\sigma^2+2x'^2\sigma^2+4(1+x^2\sigma^2-2xx'\sigma^2+x'^2\sigma^2)\sigma_0^2)^2\sqrt{1-\frac{4(xx'\sigma^2+\sigma_0^2)^2}{(1+2x^2\sigma^2+2\sigma_0^2)(1+2x'^2\sigma^2+2\sigma_0^2)}}} \tag{22}$$

$$\frac{\partial^3 k(x,x')}{\partial x^2\partial x'} = -\frac{24\sigma^4(1+4\sigma_0^2)(x+2(x-x')\sigma_0^2)}{\pi\sqrt{1+2x^2\sigma^2+2\sigma_0^2}\sqrt{1+2x'^2\sigma^2+2\sigma_0^2}(1+2x^2\sigma^2+2x'^2\sigma^2+4(1+x^2\sigma^2-2xx'\sigma^2+x'^2\sigma^2)\sigma_0^2)^2\sqrt{1-\frac{4(xx'\sigma^2+\sigma_0^2)^2}{(1+2x^2\sigma^2+2\sigma_0^2)(1+2x'^2\sigma^2+2\sigma_0^2)}}} \tag{23}$$

$$\frac{\partial^4 k(x,x')}{\partial x^2\partial x'^2} = -\frac{48\sigma^4(1+4\sigma_0^2)(-5xx'\sigma^2+(-1+8x^2\sigma^2-20xx'\sigma^2+8x'^2\sigma^2)\sigma_0^2+4(-1+4x^2\sigma^2-8xx'\sigma^2+4x'^2\sigma^2)\sigma_0^4)}{\pi\sqrt{1+2x^2\sigma^2+2\sigma_0^2}\sqrt{1+2x'^2\sigma^2+2\sigma_0^2}(1+2x^2\sigma^2+2x'^2\sigma^2+4(1+x^2\sigma^2-2xx'\sigma^2+x'^2\sigma^2)\sigma_0^2)^3\sqrt{1-\frac{4(xx'\sigma^2+\sigma_0^2)^2}{(1+2x^2\sigma^2+2\sigma_0^2)(1+2x'^2\sigma^2+2\sigma_0^2)}}} \tag{24}$$

C Derivatives of the squared exponential kernel

The squared exponential kernel is defined as:

$$k(x,x') = \sigma^2\exp\left(-\frac{(x-x')^2}{2l^2}\right) \tag{25}$$

The first derivative of the squared exponential kernel with respect to x is:

$$\frac{\partial k(x,x')}{\partial x} = \frac{\sigma^2}{l^2}(x-x')\exp\left(-\frac{(x-x')^2}{2l^2}\right) \tag{26}$$

The second derivative of the squared exponential kernel with respect to x is:

$$\frac{\partial^2 k(x,x')}{\partial x^2} = \frac{\sigma^2}{l^2}\left(\frac{(x-x')^2}{l^2} - 1\right)\exp\left(-\frac{(x-x')^2}{2l^2}\right) \tag{27}$$

The third derivative of the squared exponential kernel with respect to x is:

$$\frac{\partial^3 k(x,x')}{\partial x^3} = \frac{\sigma^2}{l^2}\left(1 - \frac{3(x-x')^2}{l^2}\right)(x-x')\exp\left(-\frac{(x-x')^2}{2l^2}\right) \tag{28}$$

The fourth derivative of the squared exponential kernel with respect to x is:

$$\frac{\partial^4 k(x,x')}{\partial x^4} = \frac{\sigma^2}{l^2}\left(\frac{3(x-x')^4}{l^4} - \frac{6(x-x')^2}{l^2} + 1\right)\exp\left(-\frac{(x-x')^2}{2l^2}\right) \tag{29}$$

It should be noted that the derivatives of the squared exponential kernel with respect to x' are the negative of the derivatives with respect to x, as the kernel is a function for the difference between x and x'.

References

[1] Titus A. Beu. *Introduction to numerical programming : a practical guide for scientists and engineers using Python and C/C++*. CRC Press, 2014. DOI: 10.1201/B17384.

[2] Thomas Pinder and Daniel Dodd. "GPJax: A Gaussian Process Framework in JAX". In: *Journal of Open Source Software* 7.75 (2022), p. 4455. DOI: 10.21105/joss.04455. URL: https://doi.org/10.21105/joss.04455.

[3] Maziar Raissi and George Em Karniadakis. "Hidden physics models: Machine learning of nonlinear partial differential equations". In: *Journal of Computational Physics* 357 (2018). ISSN: 0021-9991. DOI: 10.1016/j.jcp.2017.11.039. URL: http://dx.doi.org/10.1016/j.jcp.2017.11.039.

[4] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. *Numerical Gaussian Processes for Time-dependent and Non-linear Partial Differential Equations*. 2017. arXiv: 1703.10230 [stat.ML].

[5] Carl Edward Rasmussen and Christopher Williams. *Gaussian Processes for Machine Learning*. 2005.