Notes on Bayesian Theorem, Multivariate Gaussian Density, Gaussian Processes, and Reflection on Paper “Resource-efficient machining through physics-informed machine learning”[1]

1. Notes on the Bayesian Theorem:

Bayes’ Theorem:

Let $\boldsymbol{x} = (x_1, \dots, x_n)$ is a vector of “*n*” observations/features -here it is assumed that each observation,$x \in \mathbb{R}$, so $\boldsymbol{x} \in \mathbb{R}^n$.The probability distribution of $\boldsymbol{x}$ is given by $P(\boldsymbol{x}|\boldsymbol{w})$, which depends on the values of “*k*” parameters/categories $\boldsymbol{w} = (w_1, \dots, w_k)$ Let the “$\boldsymbol{w}$” follows a probability distribution $P(\boldsymbol{w})$. Then,

$$P(\boldsymbol{x}|\boldsymbol{w})P(\boldsymbol{w}) = P(\boldsymbol{w}|\boldsymbol{x})P(\boldsymbol{x}) \qquad 1\text{-}1$$

Given the observed data $\boldsymbol{x}$**,** the conditional distribution of $\boldsymbol{w}$ is:

$$P(\boldsymbol{w}|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|\boldsymbol{w})}{P(\boldsymbol{x})}P(\boldsymbol{w}) \qquad 1\text{-}2$$

$P(\boldsymbol{w})$, tells about the distribution information about categories or values of $(w_1, \dots, w_k)$ without knowledge of the data, and it is called the **prior** distribution of, $\boldsymbol{w}$. Correspondingly, $P(\boldsymbol{w}|\boldsymbol{x})$ is informing about the distribution of $\boldsymbol{w}$ gives the knowledge of data $\boldsymbol{x}$**,** and it is called the posterior. Where the $P(\boldsymbol{X}|\boldsymbol{W})$ is called the Likelihood function/probability, which is the hypothetical class-conditional probability distribution of data given special categories or values, or in other words the likelihood of $\boldsymbol{w}$ given $\boldsymbol{x}$, sometimes it is written as $l(\boldsymbol{w}|\boldsymbol{x})$ [1], so equation(1-2) will be written in form,

$$P(\boldsymbol{w}|\boldsymbol{x}) = l(\boldsymbol{w}|\boldsymbol{x})P(\boldsymbol{w}) \qquad 1\text{-}3$$

So Bayes’ theorem tells us that the probability distribution for $\boldsymbol{w}$ posterior to the data $\boldsymbol{x}$ is proportional to the product of the distribution for $\boldsymbol{w}$ prior to the data and the likelihood for $\boldsymbol{w}$ given $\boldsymbol{x}$**.** So it is written in the general form:

$$posterior = \frac{likelihood}{evidence} \times prior \qquad 1\text{-}4$$

Notice that it is the product of the likelihood and the prior probability that is most important in determining the posterior probability; the evidence factor, $P(\boldsymbol{x})$, can be viewed as merely a scale factor that guarantees that the posterior probabilities sum to one, as all good probabilities must.

A Clarifying Example for Bayes’ Theorem [2]:

Assume two categories of wood samples for the two main categories (w_1, w_2), where the differentiation is based on a brightness factor, where, $P(w_1) = 2/3$ and $P(w_2) = 1/3$. This likelihood is given in Figure 1.

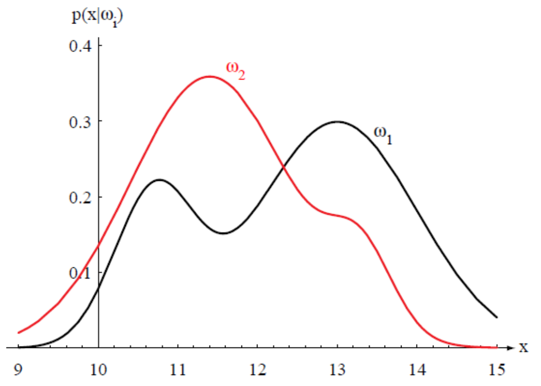
And the evidence $P(x) = \sum_{i=1}^{i=2} P(x|w_i)P(w_i)$. It is noticed that at any point in Figure 1 (b)$P(w_1|x) + P(w_2|x) = 1$, which also can be shown by using (1-2), $P(w_1|x) + P(w_2|x) = \frac{P(x|w_1)}{P(x)}P(w_1) + \frac{P(x|w_2)}{P(x)}P(w_2) = 1$. So equation 1-2 can be written in the form,

$$P(\boldsymbol{w}|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|\boldsymbol{w})}{\sum_{\boldsymbol{W}} P(\boldsymbol{x}|\boldsymbol{w})P(\boldsymbol{w})}P(\boldsymbol{w}) \qquad 1\text{-}5$$

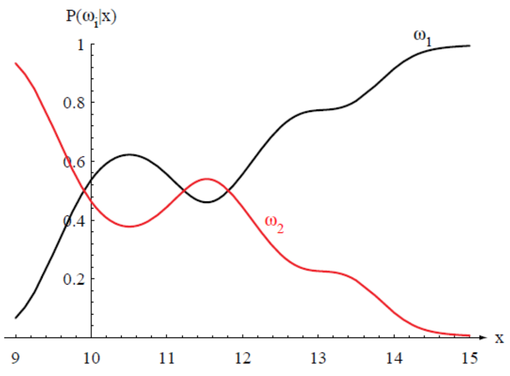
A decision could be made in favour of w_i, if $P(w_i|x) > P(w_j|x)$, given $i \neq j$. The error in this decision-making arises from the overlap of $P(w_i|x)$ in the region where $x \cong 10, 11, 12$ as shown in Figure 1 (b). For example given the brightness

[1] For the notation: $\boldsymbol{x} \in \mathbb{R}^{\boldsymbol{n}}$, $\boldsymbol{X} \in \mathbb{R}^{\boldsymbol{n \times m}}$, and $x \in \mathbb{R}$

factor $x = 13.5$ for a piece of wood whose category is unknown to the observer, as shown in Figure 1 (b) $P(w_1|x = 13.5) \cong 0.8$, while $P(w_2|x = 13.5) \cong 0.2$. So the decision goes in favour for w_1.



(a)



(b)

Figure 1: (a) Hypothetical class-conditional probability density functions show the probability density of measuring a particular feature value x given the pattern is in category w_i. If x represents the brightness of wood, the two curves might describe the difference in brightness of variations of two types of wood. Density functions are normalized; thus, the area under each curve is 1.0. (b) Posterior probabilities for the particular priors $P(w_1)$= 2/3 and $P(w_2)$=1/3 for the class-conditional probability densities shown in (a), just multiplying figure (a) with these values and normalize with $P(x) = \sum_{i=1}^{i=2} P(x|w_i)P(w_i)$ results in figure (b).

2. Multivariate Gaussian density:

The structure of a Bayes classifier is determined by the conditional densities $P(x|w_i)$ as well as by the prior probabilities, in this part multivariate normal or Gaussian density will discussed. First, recall the definition of the expected value of a scalar function $f(x)$, defined expectation for some density $P(x)$:

$$\mathbb{E}[f(x)] = \int_{-\infty}^{\infty} f(x)P(x)\,dx \qquad 2\text{-}1$$

And for discrete data in a data set $\mathcal{D}$,

$$\mathbb{E}[f(x)] = \sum_{x \in \mathcal{D}} f(x)P(x) \qquad 2\text{-}2$$

Just to simplify Equation 2-2, let denote $\boldsymbol{x}$ is the random variable taking the value of the outcome of thrown fair dice, then $\mathbb{E}(\boldsymbol{x})$, will denote mean of values in $\boldsymbol{x}$ [3].Applying equation (2-1) : $\mu \equiv \mathbb{E}(\boldsymbol{x}) = \sum_{x \in \mathcal{D}} xP(x) = 1 \cdot \frac{1}{6} + \cdots + 6 \cdot \frac{1}{6}$. In the cases of normal distribution, the Equation 2-1 will be used to evaluate the mean and variance. We begin with the continuous univariate normal/Gaussian density.

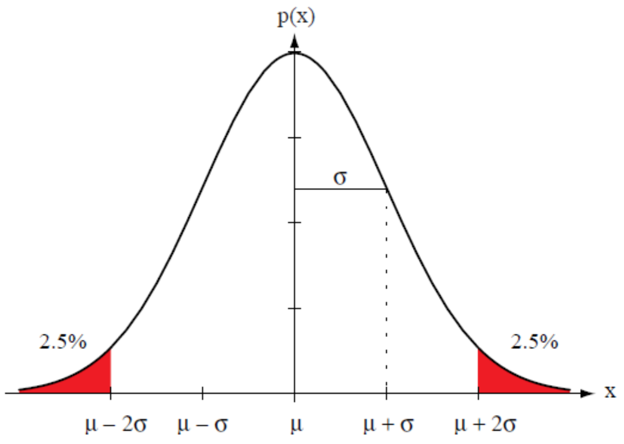


Figure 2: A univariate normal distribution has roughly 95% of its area in the range $|x - \mu| \leq 2\sigma$, as shown. The peak of the distribution has value $P(\mu) = \frac{1}{\sqrt{2\pi}\sigma}$.

Univariate Gaussian density:

$$P(x) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right] \qquad 2\text{-}3$$

For which the expected value of x (an average, here taken over the feature space) is,

$$\mu \equiv \mathbb{E}[x] = \int_{-\infty}^{\infty} xP(x)\,dx \qquad 2\text{-}4$$

and where the expected squared deviation or variance[2] is,

$$\sigma^2 \equiv \mathbb{E}[(x-\mu)^2] = \int_{-\infty}^{\infty} (x-\mu)^2 P(x)\,dx \qquad 2\text{-}5$$

For simplicity, the Equation 2-3 is abbreviated by writing $P(x) \sim \mathbb{N}(\mu, \sigma^2)$, an example of this distribution is shown in Figure 2[2]. Using this abbreviated way of writing it, equation 2-3, sometimes written in the form [4],

$$\mathbb{N}(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right] \qquad 2\text{-}6$$

Reflecting on the wood problem shown in Figure 1, where there are $P(x|w_i)$ represents the brightness distribution over category,w_i. These could be done if the density function assumed for example Univariate Gaussian, then the mean and variance could be calculated using two approaches maximum likelihood and Bayesian estimation, further discussion about them is given in chapter three [2], the both will be discussed later but the important equations resulted from this analysis are presented here[3]. Also assuming that the density function is unknown, two methods are widely used to estimate the density function, they are the Parzen-window approach, which is based on considering the window function, used usually in terms of Gaussian kernel, the other method is k_n–Nearest-Neighbour estimation, is based on averaging the number of elements surrounding a special point to the total number of samples, the probability is evaluated at this special point, further information is given in chapter 4 [2] this will not be discussed here.

Multivariate Gaussian Density

Let's assume having "*n*" features so $\boldsymbol{x} = (x_1, \dots, x_n)$, the same as defined previously, thus the Gaussian distribution in this multivariable sense is given by,

$$P(\boldsymbol{x}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}}\exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right] \qquad 2\text{-}7$$

So $\boldsymbol{x} \in \mathbb{R}^n$, Covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{n\times n}$, it is symmetric, positive semidefinite, but the notes here will be restricted to the case of positive definite case, as we assume that no identical data sets will be used. Note that the evaluation of $\boldsymbol{\Sigma^{-1}}$ is usually expensive, thus let assume that exist $\boldsymbol{y} \neq \boldsymbol{0}$, such that $\boldsymbol{y} \in \mathbb{R}^n$, and $(\boldsymbol{x}-\boldsymbol{\mu}) = \boldsymbol{\Sigma}\,\boldsymbol{y}$, this could reduce the computation significantly as the computation of the inverse of matrix is *O(n^3),* which is *O(n^2)* for the case of solving system of linear equations. For simplicity equation 2-7 could be written in the form, $P(\boldsymbol{x}) \sim \mathbb{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Using this abbreviated way of writing it, equation 2-6, sometimes written in the form [4],

[2] The square root of the variance, given by σ, is called the standard deviation, and the reciprocal of the variance, written as $\beta = 1/\sigma^2$, is called the precision [4].

[3] The idea behind maximum likelihood estimation/Bayesian estimation to calculate the mean and variance will be covered in the section Maximum likelihood and least squares.

$$\mathbb{N}(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right] \quad 2\text{-}8$$

As shown in Equation 2-1 the mean and covariance given set of samples will be calculated,

$$\boldsymbol{\mu} \equiv \mathbb{E}[\boldsymbol{x}] = \int \boldsymbol{x}P(\boldsymbol{x})\,d\boldsymbol{x} \quad 2\text{-}9$$

and,

$$\boldsymbol{\Sigma} \equiv \mathbb{E}[(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^{\boldsymbol{T}}] = \int (\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^T P(\boldsymbol{x})\,d\boldsymbol{x} \quad 2\text{-}10$$

In other words, if (x_i) is the ith component of $\boldsymbol{x}$, μ_i is the ith component of $\boldsymbol{\mu}$, and σ_{ij} is the ijth component of $\boldsymbol{\Sigma}$, then

$$\mu_i \equiv \mathbb{E}[x_i] \quad 2\text{-}11$$

and,

$$\sigma_{ij} \equiv \mathbb{E}\big[(x_i-\mu_i)(x_j-\mu_j)\big] \quad 2\text{-}12$$

For both cases, the univariate and multivariate Gaussian distributions, assume there are “m” samples withdrawn, and that they follow a Gaussian distribution, then the distribution parameters in terms of mean and variance in the univariate case, and the covariance in the multidimensional case, could be estimated using the maximum likelihood principle (chapter three [2]). For the univariate case, these estimates are given by,

$$\hat{\mu} = \frac{1}{m}\sum_{s=1}^{m} x_s \quad 2\text{-}13$$

and,

$$\hat{\sigma}^2 = \frac{1}{m}\sum_{s=1}^{m}(x_s-\hat{\mu})^2 \quad 2\text{-}14$$

For the multivariate case, these estimates are given by,

$$\hat{\boldsymbol{\mu}} = \frac{1}{m}\sum_{s=1}^{m} \boldsymbol{x}_s \quad 2\text{-}15$$

and,

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{m}\sum_{s=1}^{m}(\boldsymbol{x}_s-\hat{\boldsymbol{\mu}})(\boldsymbol{x}_s-\hat{\boldsymbol{\mu}})^T \quad 2\text{-}16$$

Before proceeding in combining the previous discussion with the Gaussian process (GP), the plotting for multivariate Gaussian in two dimensions will be shown, and simple calculations as described in the Equations 2-15 and 2-16 will be presented. Also, the calculation of $\boldsymbol{\Sigma}$ based on some kernel function [5]-this is the heart of GP [6]- will be clarified here.

<u>A Clarifying Example for Multivariate Density</u>

Let's pick a randomly data set of 40 samples from the Gaussian distribution $P(\boldsymbol{x}) \sim \mathbb{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$, where $\boldsymbol{\mu} = \boldsymbol{0}$, $\boldsymbol{\Sigma} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$, these picked data are picked from the surface of the multivariate Gaussian distribution, in this example a set of 40 samples have been picked and plotted in a scatter plot on the surface of the distribution, to clarify this point as shown in Figure 3.

The code for this figure and sample data are given in the appendix. The average sum using the equation (2-15) for x_1, x_2 is given by $\begin{bmatrix}\mu_1\\ \mu_2\end{bmatrix} = \frac{1}{40}\sum_{s=1}^{s=40}\begin{bmatrix}x_1\\ x_2\end{bmatrix}_s$, for the data set given in the Appendix, for Figure 3 this values (μ_1, μ_2) are -0.291, -0.151. These values are not so close to 0,0 as the samples did not cover the whole surface. For the calculation of the covariance matrix, using the Equation 2-16, so Equation 2-17 is equal to $\begin{bmatrix}2.03 & -0.55\\ -0.55 & 3.07\end{bmatrix}$, the code for the calculation is in the Appendix.

$$\begin{bmatrix}\sigma_{11} & \sigma_{12}\\ \sigma_{21} & \sigma_{22}\end{bmatrix} = \frac{1}{40}\sum_{s=1}^{s=40}\begin{bmatrix}x_1-\mu_1\\ x_2-\mu_2\end{bmatrix}_s [x_1-\mu_1 \quad x_2-\mu_2]_s$$
$$= \frac{1}{40}\sum_{s=1}^{m}\begin{bmatrix}(x_1-\mu_1)^2 & (x_2-\mu_2)\times(x_1-\mu_1)\\ (x_1-\mu_1)\times(x_2-\mu_2) & (x_2-\mu_2)^2\end{bmatrix}_s \qquad 2\text{-}17$$

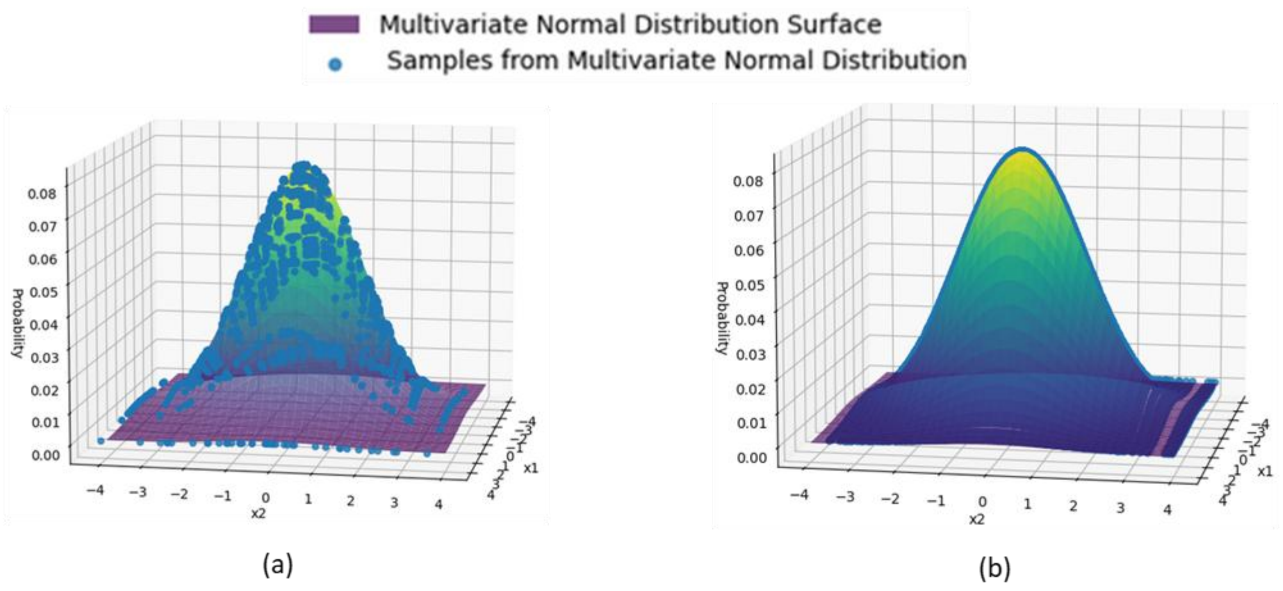


Figure 3: Bivariate Gaussian distribution for the case $\boldsymbol{\mu} = \mathbf{0}, \boldsymbol{\Sigma} = \begin{bmatrix}2 & 0\\ 0 & 2\end{bmatrix}$, (a) is the case while picking a data set for 40 samples from $P(\boldsymbol{x}) \sim \mathbb{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and plotting these data on the surface of the source distribution, (b) doing the same using a data set of 500 samples.

As the calculation shows that the covariance and mean are far away from the right ones for the data set of 40 samples, for data of 500 samples, these values (μ_1, μ_2) are 0.0159, 0.046 and $\boldsymbol{\Sigma}$ is $\begin{bmatrix}1.85 & -0.01\\ -0.01 & 2.00\end{bmatrix}$, for the final case. Sometimes the values of the feature plots of "*m*" samples $(x_{11}, x_{12} \dots, x_{1n}) \dots (x_{m1}, x_{m2} \dots, x_{mn})$ vs dimension "*n*" to study the behaviour of the kernel used for calculating the covariance matrix [5] for the Gaussian distribution as shown in Figure 3. This analysis will be presented in the following section regarding the different kernels used in GP method.

Examples of some kernel functions [5][4]:

1. Squared Exponential Kernel (Gaussian/Radial basis function kernel RBF):

[4] It is preferred here to show how these plots are created to study behaviour of multivariate Gaussian distribution based on the covariance kernel/matrix, as further things will get more complex later while discussing the GP method . Also based on my understanding -as I did not reference it, so it need to be verified by expert-these functions will serve as the basis functions for GP regression. You can notice that the plotting of the dimensions related features $((x_{m1}, x_{m2} \dots, x_{mn})$ of each picked sample "*m*" as shown in Figure 4, will create a set of functions.

$$k(x_i, x_j) = \exp\left(\frac{-(x_i - x_j)^2}{2\gamma^2}\right) \tag{2-18}$$

where γ is the length scale of the kernel.

2. Laplace Kernel :

$$k(x_i, x_j) = \exp\left(\frac{-|x_i - x_j|}{\gamma}\right) \tag{2-19}$$

3. Indicator Kernel:

$$k(x_i, x_j) = I(x_i = x_j) \tag{2-20}$$

Where I, is the indicator function.

4. Linear Kernel :

$$k(x_i, x_j) = x_i^T x_j \tag{2-21}$$

To visualize the effect of kernel function on GP behaviour, a sample input x_i is used then the feature samples picked from the multivariate Gaussian distribution $\mathbb{N}(\mathbf{0}, \boldsymbol{\Sigma})$, is plotted Figure 4. These plots where the values of features of "*m*" samples $(x_{11}, x_{12} \ldots, x_{1n}) \ldots (x_{m1}, x_{m2} \ldots, x_{mn})$ plotted on Y-axis vs dimension (1…*n*) on X-axis as it is multivariate density of "*n*" dimension. This showed that the picked samples could generate random set of functions, which have special behaviour following the kernel used. Note that more complicated kernels can be constructed by adding known kernel functions together, as the sum of two kernel functions is also a kernel function.

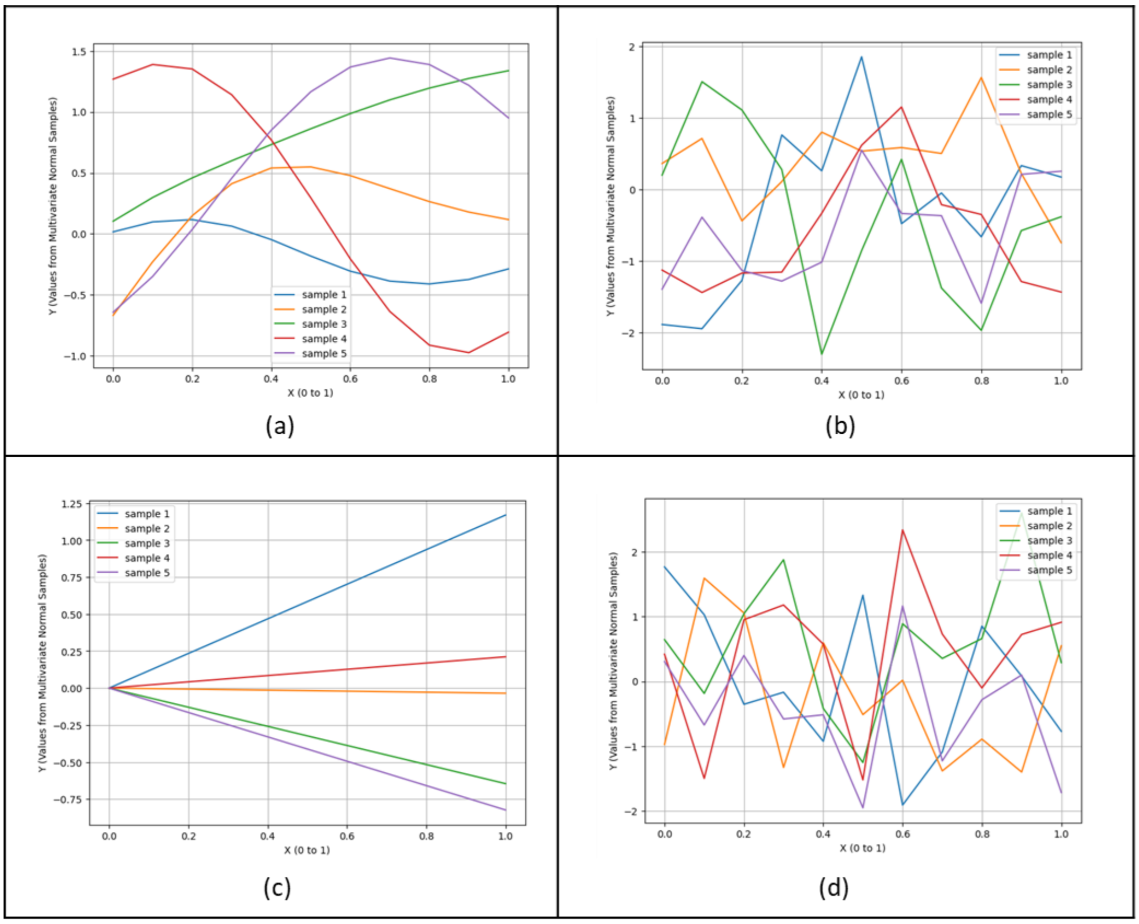


Figure 4: Examples of some kernel functions [5], (a) Gaussian kernel, $\gamma = 0.5$, (b) Laplace kernel, $\gamma = 0.5$, (c) Linear kernel, and (d) Indicator kernel.

3. Maximum likelihood, least squares and Bayesian linear regression[5]

In the last two sections, the Bayes' theorem, Gaussian distribution, and GP kernels representation have been clarified. in this section, the maximum likelihood and the Bayesian linear regression will be clarified as an important introduction to GP. Let's start with the linear regression model (will approached from likelihood and Bayesian viewpoint) for a set of data as shown in Table 1[7]. Where $\boldsymbol{x} \in \mathbb{R}^{2\times1}$, $\boldsymbol{X} \in \mathbb{R}^{6\times2}$, $\boldsymbol{w} \in \mathbb{R}^{2\times1}$ and $\boldsymbol{\epsilon} \in \mathbb{R}^{6\times1}$.

Let assume a linear regression model with noise (ϵ) where this noise follows normal distribution with mean 0, so $\epsilon \sim \mathbb{N}(0, \sigma^2)$, let the weights of this model be ($\boldsymbol{w}$), so the output y could be written in the form ,

$$y_i = \boldsymbol{x}_i^T \boldsymbol{w} + \epsilon_i \qquad 3\text{-}1$$

and in the matrix form ;

$$\begin{bmatrix} 0.1 \\ 1.2 \\ \vdots \\ 1.2 \end{bmatrix} = \begin{bmatrix} 1 & 0.9 \\ 1 & 3.8 \\ \vdots & \vdots \\ 1 & 9.6 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_6 \end{bmatrix} \qquad 3\text{-}2$$

Where $\boldsymbol{x} \in \mathbb{R}^{2\times1}$ $\boldsymbol{X} \in \mathbb{R}^{6\times2}$, $\boldsymbol{w} \in \mathbb{R}^{2\times1}$ and $\boldsymbol{\epsilon} \in \mathbb{R}^{6\times1}$.

Table 1:Observations for the regression example. Inputs x_t and corresponding outputs y_t observed at 6 different times i = 1, . . . , 6.

i	x_i	y_i
1	0.9	0.1
2	3.8	1.2
3	5.2	2.1
4	6.1	1.1
5	7.5	1.5
6	9.6	1.2

Maximum Likelihood and least squares

So first the solution (the same as least square method as the constant arise from noise will die during the differentiation with respect the fitting parameters) will be showed using maximum likelihood , to show its connection to least square method, then the Bayesian approach to relate it to the previous discussion and link it to the GP.

Equation 3-1 could be written in the form ,

$$y_i - \boldsymbol{x}_i^T \boldsymbol{w} = \epsilon_i \sim \mathbb{N}(0, \sigma^2) \qquad 3\text{-}3$$

As clarified in equation 2-6, Equation 3-3 could be viewed in the form [4][6],

[5] For both Maximum likelihood, and Bayesian regression replacing $\boldsymbol{X}$ with $\boldsymbol{\phi}(X) = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^p \\ 1 & x_2 & x_2^2 & \dots & x_2^p \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^p \end{bmatrix}$ in Equations 3-10, 3-17, and 3-18 , will make the regression using polynomial of degree (p), also will solve for $\boldsymbol{w} \in \mathbb{R}^{p+1}$.

[6] If you struggle to understand this equation, assume that the output from $\boldsymbol{x}_i^T \boldsymbol{w} + \epsilon_i$ follows a normal distribution, this assumption holds because we assumed that $\epsilon_i \sim \mathbb{N}(0, \sigma^2)$, so probability to get y_i based on special input $\boldsymbol{x}_i^T \boldsymbol{w}$ have to follow a normal distribution its mean value is the $\boldsymbol{x}^T \boldsymbol{w}$ and its uncertainty or deviation is driven by σ, based on this perspective $y_i \sim \mathbb{N}(\boldsymbol{x}_i^T \boldsymbol{w}, \sigma^2)$. Or you can

$$p(y|\boldsymbol{x},\boldsymbol{w},\sigma^2) = \mathbb{N}(y|\boldsymbol{x}^T\boldsymbol{w},\sigma^2) \tag{3-4}$$

This for one input and one output, for series N of independent inputs and outputs , the joint probability [7] [3] or likelihood function is given by,

$$p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{w},\sigma^2) = \prod_{i=1}^{N} \mathbb{N}(y_i|\boldsymbol{x}_i^T\boldsymbol{w},\sigma^2) \tag{3-5}$$

To find the $\boldsymbol{w}$ for the linear regression model, the previous equation (likelihood function 3-5) is to be maximized, actually this will show that this maximization derivation will lead to least square error function, also as this probability function has a mean of $\boldsymbol{x}^T\boldsymbol{w}$, and variance σ^2 , thus the maximum probability (optimum solution) is when the output y approaches the mean, this will give maximum probability (substitute $y_i = \boldsymbol{x}_i^T\boldsymbol{w}$ in equation 3-6, that will give the supremum probability) . Then the logarithm of this likelihood will be taken, in practice, it is more convenient to maximize the log of the likelihood function. Because the logarithm is a monotonically increasing function of its argument, maximization of the log of a function is equivalent to maximization of the function itself [4]. Just to simplify, the equation 3-5, is

$$\prod_{i=1}^{N} \mathbb{N}(y_i|\boldsymbol{x}_i^T\boldsymbol{w},\sigma^2) = \left(\frac{1}{\sqrt{2\pi\sigma}}\right)^N \exp\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - \boldsymbol{x}_i^T\boldsymbol{w})^2\right] \tag{3-6}$$

and logarithm of 3-6

$$\text{In} \prod_{i=1}^{N} \mathbb{N}(y_i|\boldsymbol{x}_i^T\boldsymbol{w},\sigma^2) = \text{In}\left(\frac{1}{\sqrt{2\pi\sigma}}\right)^N - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - \boldsymbol{x}_i^T\boldsymbol{w})^2 \tag{3-7}$$

and it is equal to ,

$$\text{In} \prod_{i=1}^{N} \mathbb{N}(y_i|\boldsymbol{x}_i^T\boldsymbol{w},\sigma^2) = -\frac{N}{2}\text{In}\,(2\pi) - N\,\text{In}(\sigma) - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - \boldsymbol{x}_i^T\boldsymbol{w})^2 \tag{3-8}$$

It is clear that last term of equation 3-8 is the sum of squares error. Taking the gradient of this function with respect to $\boldsymbol{w}$,

$$\nabla\,\text{In} \prod_{i=1}^{N} \mathbb{N}(y_i|\boldsymbol{x}_i^T\boldsymbol{w},\sigma^2) = \frac{1}{\sigma^2}\sum_{i=1}^{N}(y_i - \boldsymbol{x}_i^T\boldsymbol{w})\boldsymbol{x}_i^T \tag{3-9}$$

Setting the gradient to zero, and solving for $\boldsymbol{w}_{opt}$ (opt for optimum),

$$\boldsymbol{w}_{opt} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{3-10}$$

And differentiating 3-8 with respect σ ,

alternatively view it from the perspective of Equation 3-3, where $y_i - \boldsymbol{x}_i^T\boldsymbol{w} = \epsilon_i \sim \mathbb{N}(0,\sigma^2)$, so $y_i - \boldsymbol{x}_i^T\boldsymbol{w}$ has mean of zero and deviation of σ.

[7] Remember : the joint probability for independent events is by multiply the probability of each event.Just to simplify equation 3-5, it is the equation multiplied itself N times, but every time you have different y_i , $\boldsymbol{x}_i^T\boldsymbol{w}$

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \boldsymbol{x}_i^T \boldsymbol{w}_{opt}\right)^2 \qquad 3\text{-}11$$

This shows the derivation for equation 2-14, for equation 2-13, replace in equation 3-8 $\boldsymbol{x}_i^T\boldsymbol{w}$ with μ , and differentiate with respect to μ.The calculation for $\boldsymbol{w}_{opt}$, σ^2 following these derivation for the data given in

Where $\boldsymbol{x} \in \mathbb{R}^{2\times1}$ $\boldsymbol{X} \in \mathbb{R}^{6\times2}$, $\boldsymbol{w} \in \mathbb{R}^{2\times1}$ and $\boldsymbol{\epsilon} \in \mathbb{R}^{6\times1}$.

Table 1, $w_1 = 0.551$, $w_2 = 0.117$, σ^2= 0.248, the code for the calculation , and the plotting in Figure 5 are given in the appendix.

It was interesting to show here the derivation of least square from the likelihood probability, and as it shown in Figure 5, the confidence region has a constant width reflecting the $\mathbb{N}(0,\sigma^2)$ noise in the output. Also because the selected noise has mean zero, the samples are very close to the best fit line, if a probability is drawn between the two bounds in this graph, it will have the structure of $\mathbb{N}(\boldsymbol{x}_i^T\boldsymbol{w}_{opt},\sigma^2)$, so when you input new $\boldsymbol{x}_*^T$ in $\mathbb{N}(\boldsymbol{x}_*^T\boldsymbol{w}_{opt},\sigma^2)$, you are picking a sample from this distribution. In other words, given new $\boldsymbol{x}_*^T$, the new output y_* is following the Gaussian posterior $p(y_*|\boldsymbol{x}_*,\boldsymbol{w}_{opt},\sigma^2) = \mathbb{N}(\boldsymbol{x}_*^T\boldsymbol{w}_{opt},\sigma^2)$, that holds following Equation 1-4, as $p(\boldsymbol{w})$=1, as the only $\boldsymbol{w}$ is the $\boldsymbol{w}_{opt}$**.** That will change slightly in the next part of this section while discussing Bayesian linear regression . To further simplify the results, $y_* = \boldsymbol{x}_*^T\boldsymbol{w}_{opt} + \mathbb{N}(0,\sigma^2)$

So in the next part of this section, the Bayesian linear regression model will discussed where the confidence region will reflect the uncertainty estimate for parameter $\boldsymbol{w}_{opt}$. This will show the contrast between classical linear regression model and Bayesian linear regression model [8].

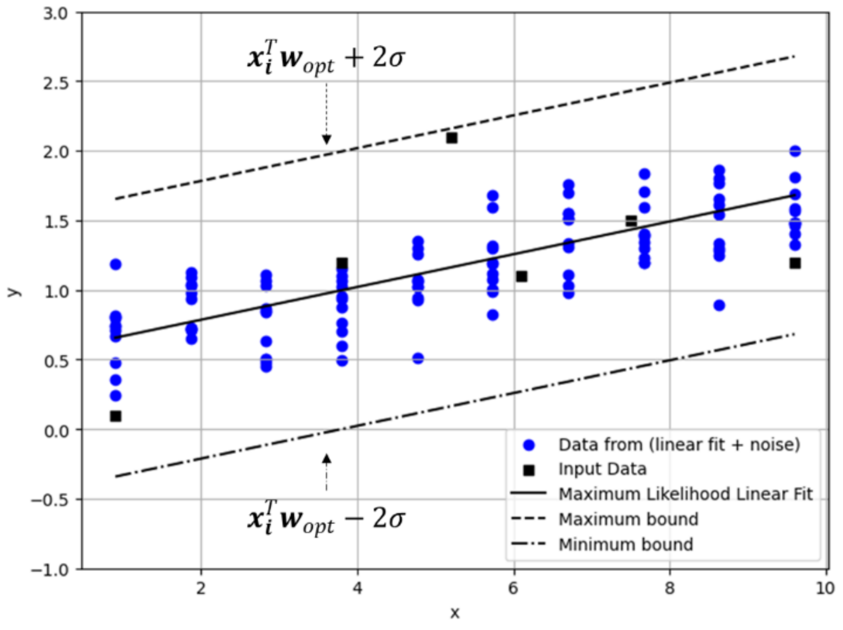


Figure 5: In this plot the input data have been plotted together with the data as output from the fitted linear model with additive noise following the normal distribution ($\boldsymbol{x}_i^T\boldsymbol{w}_{opt}$ +samples from ($\mathbb{N}(0,\sigma^2)$)). As clarified in Figure 2, the region $\mu \pm 2\sigma$ denotes the 95% confidence in the model.

Bayesian linear regression

Starting from the linear regression model with Gaussian noise and the likelihood the probability density as in the previous part of this section[9],

$$y_i - \boldsymbol{x}_i^T \boldsymbol{w} = \epsilon_i \sim \mathbb{N}(0, \sigma^2) \qquad \text{3-12}$$

and the likelihood the probability density,

$$p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w}) = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^N \exp\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{N}\left(y_i - \boldsymbol{x}_i^T \boldsymbol{w}\right)^2\right] = \mathbb{N}(\boldsymbol{X}^T \boldsymbol{w}, \sigma^2 \boldsymbol{I}) \qquad \text{3-13}$$

The priori $p(\boldsymbol{w})$, is assumed to follow the Gaussian with covariance matrix $\boldsymbol{\Sigma_p}$,

$$\boldsymbol{w} \sim \mathbb{N}(\boldsymbol{0}, \boldsymbol{\Sigma_p}) \qquad \text{3-14}$$

The results Equation 3-17 for $\boldsymbol{w}$ will give the same results as in Equation 3-10 if the $p(\boldsymbol{w})$ assumed to have no density distribution, but given that this is not the case, the results will differ by factor $\boldsymbol{\Sigma_p}$.

Following the Bayes' theorem,

$$\text{posteriori} = \frac{\text{likelihood} \times \text{priori}}{\text{evidence}}, \quad p(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{X}) = \frac{p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w})p(\boldsymbol{w})}{p(\boldsymbol{y}|\boldsymbol{X})} \qquad \text{3-15}$$

and,

$$p(\boldsymbol{y}|\boldsymbol{X}) = \int p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w})p(\boldsymbol{w})d\boldsymbol{w} \qquad \text{3-16}$$

Thus evaluating the posteriori 3-15 is combining the priori 3-14, and the likelihood 3-13, and it is given by,

$$p(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{X}) = \mathbb{N}\left(\frac{1}{\sigma^2}A^{-1}\boldsymbol{X}^T\boldsymbol{y}, A^{-1}\right), \qquad A = \frac{1}{\sigma^2}\,\boldsymbol{X}^T\boldsymbol{X} + \boldsymbol{\Sigma}_{\boldsymbol{p}}^{-1} \qquad \text{3-17}$$

So given new $\boldsymbol{x}_*^T$, the prediction y_* is given by averaging the output of all possible linear models w.r.t. the Gaussian posterior[9],

$$p(y_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) = \int p(y_*|\boldsymbol{x}_*, \boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y})d\boldsymbol{w} = \mathbb{N}\left(\frac{1}{\sigma^2}\boldsymbol{x}_*^T\boldsymbol{A}^{-1}\boldsymbol{X}^T\boldsymbol{y}, \boldsymbol{x}_*^T\boldsymbol{A}^{-1}\boldsymbol{x}_*\right) \qquad \text{3-18}$$

To simplify this results, given $\boldsymbol{x}_*^T$, new $y_* = \frac{1}{\sigma^2}\boldsymbol{x}_*^T\boldsymbol{A}^{-1}\boldsymbol{X}^T\boldsymbol{y} + \mathbb{N}(0, \boldsymbol{x}_*^T\boldsymbol{A}^{-1}\boldsymbol{x}_*)$, it is noticeable, that in contrast to likelihood method, the input $\boldsymbol{x}_*^T$, which the results should be evaluated at has influence on the uncertainty. For simplicity[8] σ^2, $\boldsymbol{\Sigma_p}$ is 1, $\boldsymbol{I}$; to simplify the calculation[9].This assumption is not naïve in this example working on the data Table 1, as using $A = \frac{1}{\sigma^2}\,\boldsymbol{X}^T\boldsymbol{X} + \boldsymbol{\Sigma}_{\boldsymbol{p}}^{-1}$, using $\boldsymbol{\Sigma}_{\boldsymbol{p}}^{-1} = \boldsymbol{I}$, while $\boldsymbol{X}^T\boldsymbol{X} \gg \boldsymbol{I}$ so $A \cong \boldsymbol{X}^T\boldsymbol{X}$, using this in Equation 3-18 without noise part, $y_* = \boldsymbol{x}_*^T(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$, this is the same result obtained in likelihood results Equation 3-10. These results are shown in Figure

[8] The details concerning optimization regarding σ^2, $\boldsymbol{\Sigma_p}$, is a bit tricky, and is not of significant benefits here, as this is only used as introduction to Gaussian process.

[9] In the GP $\boldsymbol{\Sigma_p}$ will be calculated using the GP kernel, and σ^2 will represent noise in data.

6, which is similar to the figures shown in (Figure 1 [8], Figure 2.1[9]). Both Bayesian estimation and the likelihood fit are close, as described previously, the region of confidence [10]is between the two bounds, further the region of confidence get narrower as the data tends to locate, as the data get far away near the boundary, the confidence region get wider. In contrast to likelihood method as shown in Figure 5, this width is constant. The code used for the calculation and the plotting of Figure 6 is in the appendix.

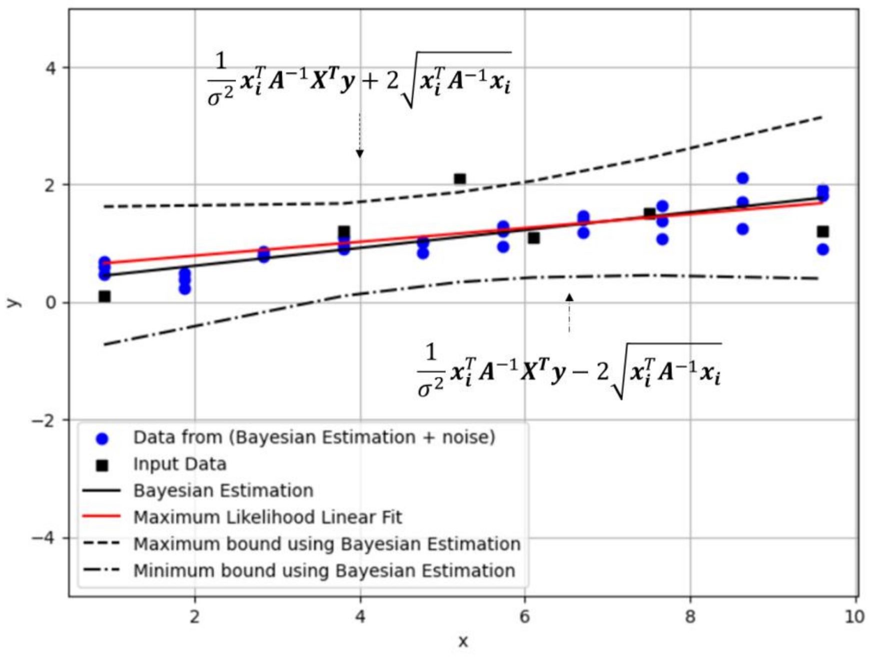


Figure 6: The mean as calculated using Bayesian estimation and maximum likelihood linear fit. The description and comments is in the text.

4. Gaussian Processes

Gaussian Processes with Zero mean

A Gaussian process defines a distribution over functions such that, if we pick any two or more points in a function (i.e., different input–output pairs), observations of the outputs at these points follow a joint (multivariate) Gaussian distribution. More formally, a Gaussian process is defined as a collection of random variables, any finite number of which have a joint (multivariate) Gaussian distribution[7]. Let given a training data as given in Table 1, the goal of Gaussian process regression is to predict the new y_* given new x_* , the same as done previously in Bayesian linear regression and maximum likelihood linear fitting. Let consider the noise-free case , so $y = f(x)$, GP is a multivariate Gaussian[6], a zero-mean Gaussian process prior is assumed

$$f(\boldsymbol{x}) \sim \mathcal{GP}(\mu(\boldsymbol{x}) = 0, k(\boldsymbol{x}, \boldsymbol{x}')) \qquad 4\text{-}1$$

Where $\mu(\boldsymbol{x}) = 0$ is the zero mean function, and $k(\boldsymbol{x}, \boldsymbol{x}')$ is the covariance matrix, this has been clarified using the kernel functions in Page 5.

As clarified in previous section, assuming that the data following Gaussian distribution,

$$p(\mathbf{y}|\mathbf{X}) = \mathbb{N}(\mu(\boldsymbol{X}), k(\boldsymbol{X}, \boldsymbol{X})) \qquad 4\text{-}2$$

[10] But it is not strict confidence as the σ^2, Σ_p have not been optimized.

And the predictive estimate also follows Gaussian distribution,

$$p(y_*|\boldsymbol{x}_*) = \mathbb{N}(\mu(\boldsymbol{x}_*), k(\boldsymbol{x}_*, \boldsymbol{x}_*)) \qquad 4\text{-}3$$

The joint distribution of y_* and $\boldsymbol{y}$ is given by ,

$$\begin{bmatrix} y_* \\ y_1 \\ \vdots \\ y_n \end{bmatrix} \sim N\left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} k(x_*, x_*) & k(x_*, \boldsymbol{x})^T \\ k(x_*, \boldsymbol{x}) & k(\boldsymbol{x}, \boldsymbol{x}') \end{bmatrix}\right) \qquad 4\text{-}4$$

This is referred to the Gaussian process with zero mean $\mathcal{GP}(0, k)$. In next sections the case for Gaussian process with $\mu(\boldsymbol{x}) \neq 0$ will be discussed.

$k(\boldsymbol{x}, \boldsymbol{x}')$ is the covariance matrix ,

$$k(\boldsymbol{x}, \boldsymbol{x}') = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \cdots & k(x_1, x_n) \\ k(x_2, x_1) & k(x_2, x_2) & \cdots & k(x_2, x_n) \\ \vdots & \vdots & \vdots & \vdots \\ k(x_n, x_1) & k(x_n, x_2) & \cdots & k(x_n, x_n) \end{bmatrix} \qquad 4\text{-}5$$

The matrix $k(x_*, \boldsymbol{x})$ is defined by,

$$k(x_*, \boldsymbol{x}) = \begin{bmatrix} k(x_*, x_1) \\ k(x_*, x_2) \\ \vdots \\ k(x_*, x_n) \end{bmatrix} \qquad 4\text{-}6$$

The posteriori prediction is given by[10],

$$\mathrm{y}_*|x_*, \boldsymbol{X}, \boldsymbol{y} \sim N(\mu(x_*|\boldsymbol{X}, \boldsymbol{y}), k(x_*, x_*|\boldsymbol{X}, \boldsymbol{y})) \qquad 4\text{-}7$$

Where $\mu(x_*|\boldsymbol{X}, \boldsymbol{y})$ is the mean vector of the posterior distribution,

$$\mu(x_*|\boldsymbol{X}, \boldsymbol{y}) = k(x_*, \boldsymbol{x})^T k(\boldsymbol{x}, \boldsymbol{x}')^{-1} \boldsymbol{y} \qquad 4\text{-}8$$

and the variance $k(x_*, x_*|\boldsymbol{X}, \boldsymbol{y})$ is given by,

$$k(x_*, x_*|\boldsymbol{X}, \boldsymbol{y}) = k(x_*, x_*) - k(x_*, \boldsymbol{x})^T k(\boldsymbol{x}, \boldsymbol{x}')^{-1} k(x_*, \boldsymbol{x}) \qquad 4\text{-}9$$

The plots for the priori will be the same as shown previously in Figure 4, and they also shown in Figure 7. To further clarify the calculation in 4-7, 4-8, and 4-9, let the y_* for x_* which is already in the training set as given in Table 1, as it will be shown later in Figure 7, the GP derivation should capture the corresponding y_* which was given in the training data, in this derivation, the Indicator Kernel[11] will be used as an easy option for manual calculation, so let $x_* = x_1$, and $\boldsymbol{X}, \boldsymbol{y}$ are the training data given in Table 1. Equation 4-8 will be ,

$$\mu(x_* = x_1|\boldsymbol{X}, \boldsymbol{y}) = k(x_* = x_1, \boldsymbol{x})^T k(\boldsymbol{x}, \boldsymbol{x}')^{-1} \boldsymbol{y} = [1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0] \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix} \begin{bmatrix} 0.1 \\ 1.2 \\ 2.1 \\ 1.1 \\ 1.5 \\ 1.2 \end{bmatrix} = 0.1 \qquad 4\text{-}10$$

[11] This kernel is based on $k_{ij} = 1$ if $x_i = x_j$ otherwise zero

and Equation 4-9 (variance) will be,

$$k(x_* = x_1, x_* = x_1|\boldsymbol{X},\boldsymbol{y}) = k(x_*,x_*) - k(x_*,\boldsymbol{x})^T k(\boldsymbol{x},\boldsymbol{x}')^{-1} k(x_*,\boldsymbol{x})$$

$$= 1 - [1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0] \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = 0 \qquad \text{4-11}$$

So based on this calculation using the Indicator Kernel for $x_* = x_1$, the output y_* is equal to y_1 where both $(x_1, y_1) \in$ training data.

The calculation[12] using this method for the data presented in Table 1 is shown in Figure 7, while the kernel used is the Squared Exponential Kernel (Gaussian/Radial basis function kernel RBF)[13] using $\gamma = 1$. It is clear that at the training data the method captured the same output as clarified manually in Equations 4-10 and 4-11. It can also be noticed that when the data points tend to concentrate the deviation limits shrink, similar interpretation has been understood from the Bayesian estimation Figure 6. Similar results have been presented in [9] Figure 2.2 (where the grey region in that figure represents max/min bound in Figure 7). Finally the code is in the Appendix.

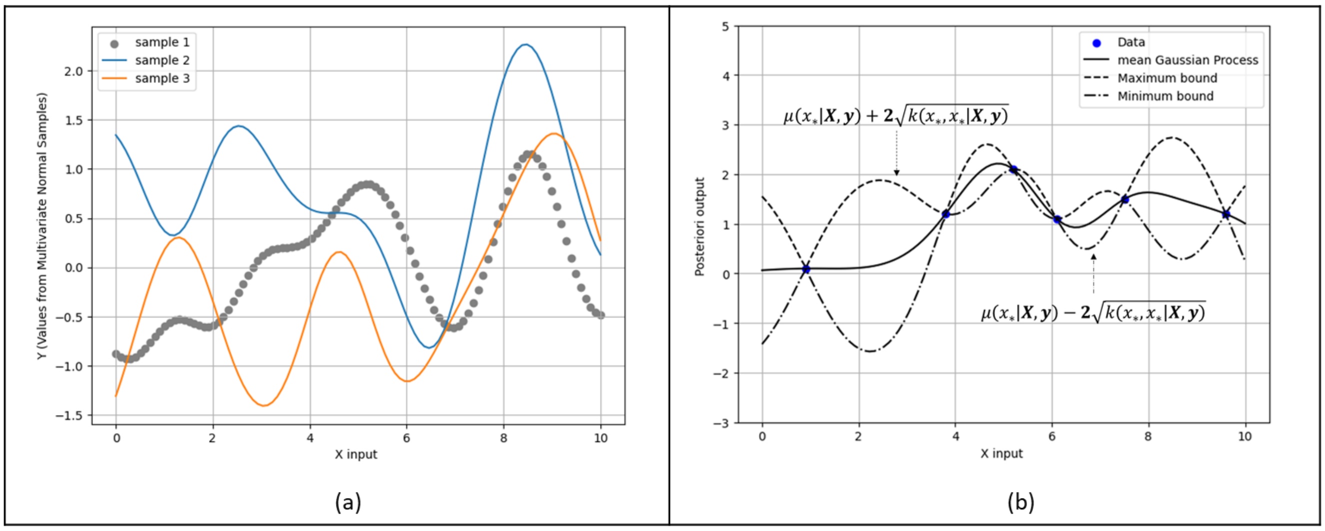


Figure 7: The plotting for the priori distribution (a) and the posterior distribution (b).

Optimization of Gaussian Process hyperparameters

The optimization for maximum likelihood is important to determine the free parameters, for example the γ in the Gaussian kernel, and when a noise parameters are used as shown in Equation 4-12. A bad selection could worsen the Gaussian process as shown in Figure 8[9], where the used kernel is given by ,

[12] After a lot of effort, the results provided in Table 3 [7] do not follow the right calculation, it may be a mistake, or they presented a kernel and followed a another kernel , I don't know where things break, but I will continue verifying the output in comparison with [9] Figure 2.2. Please if you have time check it.

[13]

$$k(x_i, x_j) = \exp\left(\frac{-(x_i - x_j)^2}{2\gamma^2}\right)$$

$$\boldsymbol{K} = k(\boldsymbol{x}, \boldsymbol{x}') + \sigma_n^2 \boldsymbol{I}, \qquad k(x_i, x_j) = \sigma_f^2 \exp\left(\frac{-(x_i - x_j)^2}{2\gamma^2}\right) \tag{4-12}$$

As shown in Figure 8 the dependence of the prediction of GP on the hyperparameters used.

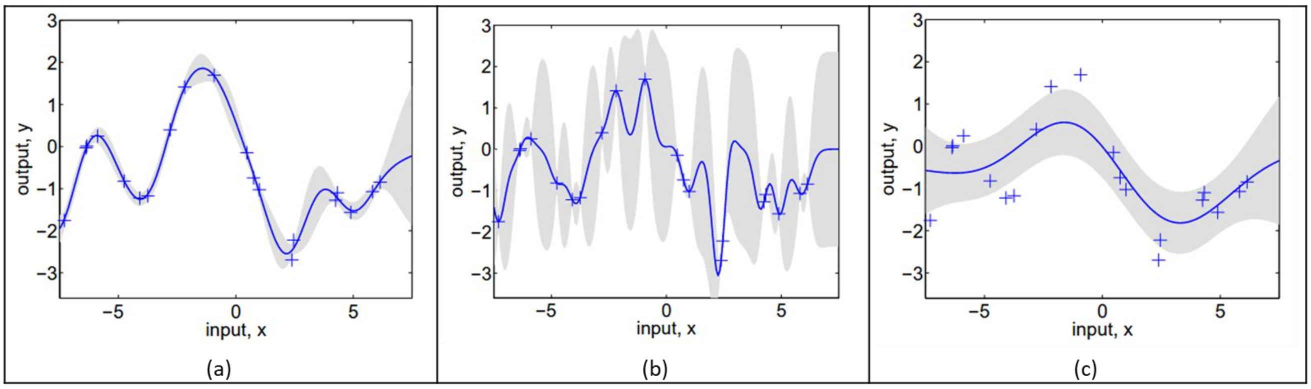


Figure 8: (a) $(\gamma, \sigma_f, \sigma_n) = (1,1,0.1), (b)(\gamma, \sigma_f, \sigma_n) = (0.3, 1.08, 0.00005), (c)(\gamma, \sigma_f, \sigma_n) = (3.0, 1.16, 0.89)$

Following the previously derived optimization for maximum likelihood Equations 3-3 - 3-10, the optimization of Gaussian Process will follows the same procedure with slightly meaningful difference. For the Gaussian Process, the marginal likelihood will be optimized, marginal likelihood is used here as the Gaussian Process is non-parametric method, the parameters used are incorporated into the construction of the kernel function ; for example the general form of square exponential kernel given by[9],

$$k(\boldsymbol{X}_p, \boldsymbol{X}_q) = \sigma_f^2 \exp\left(-\frac{1}{2}(\boldsymbol{X}_p - \boldsymbol{X}_q)^T \boldsymbol{M} (\boldsymbol{X}_p - \boldsymbol{X}_q)\right) + \sigma_n^2 \delta_{pq} \tag{4-13}$$

Several formulas for M could be used, several examples are given by [9],

$$\boldsymbol{M}_1 = \gamma^{-2} \boldsymbol{I}, \qquad \boldsymbol{M}_2 = \mathrm{diag}(\boldsymbol{\gamma})^{-2}, \qquad \boldsymbol{M}_3 = \boldsymbol{\Lambda}\boldsymbol{\Lambda}^T + \mathrm{diag}(\boldsymbol{l})^{-2} \tag{4-14}$$

Thus, the set of hyperparameters are given by: $\boldsymbol{\theta} = (\{\boldsymbol{M}\}, \sigma_f^2, \sigma_n^2)$

The case of $\boldsymbol{M}_1$ is shown for example in Equation 4-12, and used with $\gamma = 1$ for generating Figure 7. For the others two cases please refer to Section 5.1[9].

The expression of the marginal likelihood function is given by,

$$\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = -\frac{1}{2}\boldsymbol{y}^T \boldsymbol{K}^{-1} \boldsymbol{y} - \frac{1}{2}\log|\boldsymbol{K}| - \frac{n}{2}\log 2\pi \tag{4-15}$$

Where $\boldsymbol{K} = k(\boldsymbol{x}, \boldsymbol{x}') + \sigma_n^2 \boldsymbol{I}$**.** The derivative of marginal likelihood function with respect to the hyperparameters is given by,

$$\begin{aligned} \frac{\partial}{\partial \theta_j} \log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) &= -\frac{1}{2}\boldsymbol{y}^T \boldsymbol{K}^{-1} \frac{\partial \boldsymbol{K}}{\partial \theta_j} \boldsymbol{K}^{-1} \boldsymbol{y} - \frac{1}{2}\mathrm{tr}\left(\boldsymbol{K}^{-1} \frac{\partial \boldsymbol{K}}{\partial \theta_j}\right) \\ &= \frac{1}{2}\mathrm{tr}\left((\boldsymbol{\alpha}\boldsymbol{\alpha}^T - \boldsymbol{K}^{-1}) \frac{\partial \boldsymbol{K}}{\partial \theta_j}\right), \text{where } \boldsymbol{\alpha} = \boldsymbol{K}^{-1}\boldsymbol{y} \end{aligned} \tag{4-16}$$

For further clarification the Squared Exponential Kernel will be used and optimized to show the differences between the optimized case and the case in Figure 7, where the hyperparameters used are $\boldsymbol{\theta} = \left(\{\boldsymbol{M}_1 = \gamma^{-2}\boldsymbol{I}\}, \sigma_f^2, \sigma_n^2\right) = (\boldsymbol{I}, 1, 0)$ **,** where the kernel function is given by Equation 4-12. Later the hyperparameter $\boldsymbol{\theta}$ will be used simply to denote the set $(\gamma, \sigma_f, \sigma_n)$.

The optimization (maximization) of Equation 4-15 using the kernel function Equation 4-12 based on the data given in Table 1, is shown in Figure *9*.

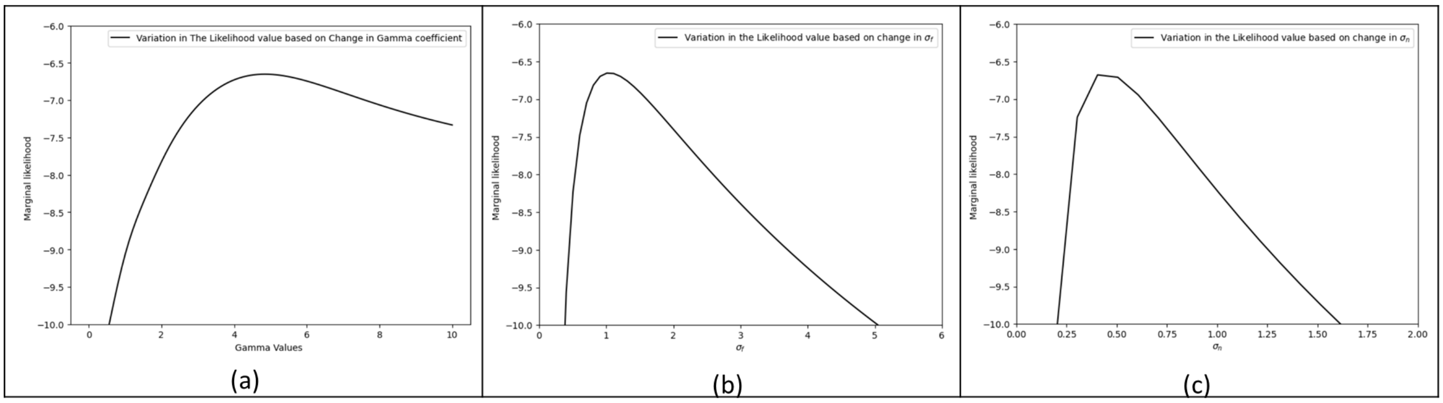


Figure 9: Optimization of Marginal Likelihood Function. The optimum values $[\gamma, \sigma_f, \sigma_n]$ are $\approx$ [4.85,1.01,0.404] . These plots shows the variation in the marginal likelihood function based on changes in one parameter while the other parameters are kept constant. The codes for showing the optimization procedure are given in the Appendix, together with code to get the optimum values.

The comparison between the implementation of Gaussian Process using a set of random parameters for the kernel function Equation 4-12, and the one using the optimum set is shown in Figure 10. It worth mentioning that the marginal likelihood function in general has several local maximum, and care should be given so the optimization does not end in bad local maximiser [9].

Gaussian Processes with Noisy Observations

Following the optimization of the hyperparameters as show in Figure *9*, the noisy term $\sigma_n^2\delta_{pq}$, which is in Equation 4-12, 4-13 will not be zero, thus, Equations 4-8, 4-9 will slightly change.

The mean for new observations, given that the noise term exists is given by [9],

$$\begin{aligned}\mu(x_*|\boldsymbol{X}, \boldsymbol{y}) &= k(x_*, \boldsymbol{x})^T (k(\boldsymbol{x}, \boldsymbol{x}') + \sigma_n^2 \boldsymbol{I}\,)^{-1} \boldsymbol{y} \\ &= k(x_*, \boldsymbol{x})^T \boldsymbol{K}^{-1} \boldsymbol{y}\end{aligned} \qquad \text{4-17}$$

And the covariance is given by [9],

$$\begin{aligned}k(x_*, x_*|\boldsymbol{X}, \boldsymbol{y}) &= k(x_*, x_*) - k(x_*, \boldsymbol{x})^T (k(\boldsymbol{x}, \boldsymbol{x}') + \sigma_n^2 \boldsymbol{I}\,)^{-1} k(x_*, \boldsymbol{x}) \\ &= k(x_*, x_*) - k(x_*, \boldsymbol{x})^T \boldsymbol{K}^{-1} k(x_*, \boldsymbol{x})\end{aligned} \qquad \text{4-18}$$

Where $\boldsymbol{K} = k(\boldsymbol{x}, \boldsymbol{x}') + \sigma_n^2 \boldsymbol{I}$, moreover the assumption of adding independent Gaussian noise ε, for the function $y = f(x) + \varepsilon$, with variance σ_n^2 , is typical for realistic modelling situations.

The comparison between the implementation of Gaussian Process using a set of random parameters for the kernel function Equation 4-12, and the one using the optimum set with noise term included is shown in Figure 10. As shown in Figure 10, the uncertainty at the input data for the case using the optimum set while the noise term is not zero is almost in range $\pm 2 \times \sigma_n \approx \pm 0.8$. It is also clear that the uncertainty for the whole data is significantly larger than that of the case where the hyperparameters used were random values, not the optimum ones. In this context, it is worth mentioning that the optimization of Gaussian Process to determine the hyperparameters mostly have significant control over the uncertainty of the model.

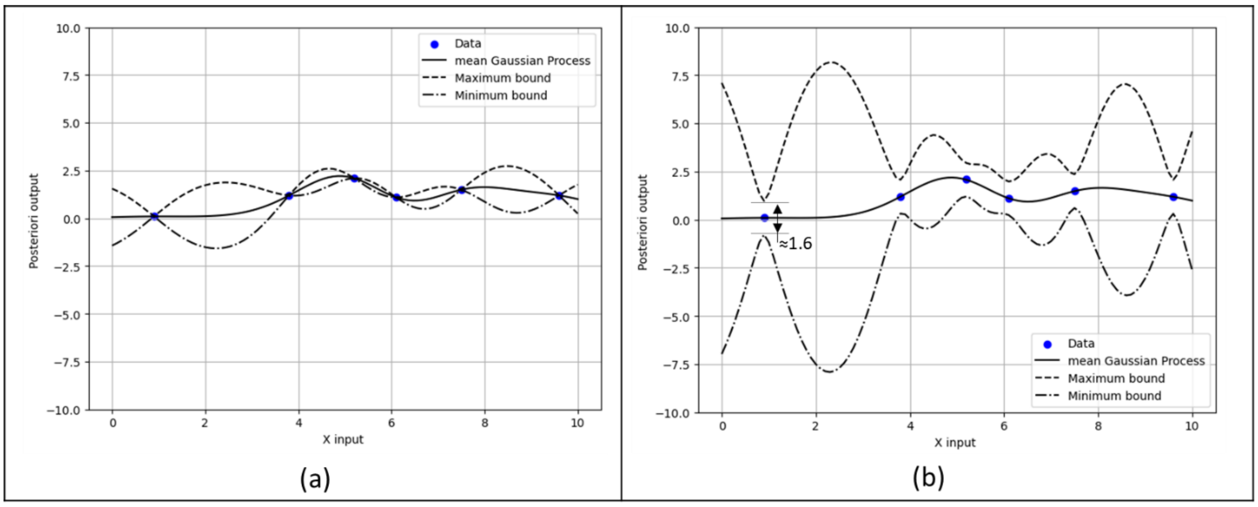


Figure 10: Comparison between (a) using the parameters for the kernel function parameters $[\gamma, \sigma_f, \sigma_n]$ randomly as [1,1,0], (b) using the optimum values are ≈ [4.85,1.01,0.404].

Gaussian Processes with Non-Zero mean Function

The use of basis functions to represent the mean function for Gaussian Process, could be used to add prior information to the model, and may also be used for several reasons[14] [9]. In contrast to Equation 4-1, the mean of the Gaussian Process is not zero, and the Gaussian Process is given by,

$$f(\boldsymbol{x}) \sim \mathcal{GP}(\mu(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')) \qquad 4\text{-}19$$

The $\boldsymbol{K}$, $k(\boldsymbol{x}, \boldsymbol{x}')$, could be used for example as in the form given in Equation 4-12.

The mean for new observations given $\mu(\boldsymbol{x}) \neq 0$, is given by [11],

$$\begin{aligned} \mu(x_* | \boldsymbol{X}, \boldsymbol{y}) &= \mu(x_*) + k(x_*, \boldsymbol{x})^T (k(\boldsymbol{x}, \boldsymbol{x}') + \sigma_n^2 \boldsymbol{I})^{-1} (\boldsymbol{y} - \mu(\boldsymbol{x})) \\ &= \mu(x_*) + k(x_*, \boldsymbol{x})^T \boldsymbol{K}^{-1} (\boldsymbol{y} - \mu(\boldsymbol{x})) \end{aligned} \qquad 4\text{-}20$$

And the covariance is given by [11],

$$\begin{aligned} k(x_*, x_* | \boldsymbol{X}, \boldsymbol{y}) &= k(x_*, x_*) - k(x_*, \boldsymbol{x})^T (k(\boldsymbol{x}, \boldsymbol{x}') + \sigma_n^2 \boldsymbol{I})^{-1} k(x_*, \boldsymbol{x}) \\ &= k(x_*, x_*) - k(x_*, \boldsymbol{x})^T \boldsymbol{K}^{-\boldsymbol{1}} k(x_*, \boldsymbol{x}) \end{aligned} \qquad 4\text{-}21$$

For optimizing the hyperparameters $\boldsymbol{\theta} = (\gamma, \sigma_f, \sigma_n)$, used for constructing the covariance function, the marginal likelihood is given by [11],

$$\log p(\boldsymbol{y} | \boldsymbol{X}, \boldsymbol{\theta}) = -\frac{1}{2} (\boldsymbol{y} - \mu(\boldsymbol{x}))^T \boldsymbol{K}^{-1} (\boldsymbol{y} - \mu(\boldsymbol{x})) - \frac{1}{2} \log|\boldsymbol{K}| - \frac{n}{2} \log 2\pi \qquad 4\text{-}22$$

[14] I think we could use it to implicitly incorporate physical model -Taylor Model- into Gaussian Process

For further clarify Equation 4-20, similar work as done in Equation 4-10, could be done here. The equation for the $\mu(\boldsymbol{x})$ could be used as a first order polynomial, where its parameters have been optimized using the least square fit section 3. The $\mu(\boldsymbol{x}) = x_1 \times w_1 + x_2 \times w_2$, where $w_1 = 0.551$, and $w_2 = 0.117$. Modifying Equation 4-10, and using it,

$$\mu(x_* = x_1|\boldsymbol{X},\boldsymbol{y}) = \mu(x_* = x_1) + k(x_* = x_1,\boldsymbol{x})^T k(\boldsymbol{x},\boldsymbol{x}')^{-1}(\boldsymbol{y} - \mu(\boldsymbol{x}))$$

$$= \mu(x_* = x_1) + [1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0] \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix} \left[\begin{bmatrix} 0.1 \\ 1.2 \\ 2.1 \\ 1.1 \\ 1.5 \\ 1.2 \end{bmatrix} - \begin{bmatrix} \mu(x_1) \\ \mu(x_2) \\ \mu(x_3) \\ \mu(x_4) \\ \mu(x_5) \\ \mu(x_6) \end{bmatrix} \right] = 0.1 \qquad \text{4-23}$$

Comparing the results for using zero mean Gaussian Process and non-zero mean Gaussian Process, where the mean function is as stated previously, and the hyperparameters are as used in Figure 10, is shown in Figure 11-the code is in the Appendix. Analysing the effect of incorporating the mean function into the Gaussian Process, could be understood by analysing the spots where the orange arrows are pointing in Figure 11. It could be understood that incorporating the mean function as a priori knowledge into the Gaussian Process has forced the Gaussian Process to follow this mean function when it is possible, but also still taking into account the data used in the model, which could make it a suitable technique to incorporate the physical model into the Gaussian Process, other techniques to do so have been developed in [11], [12].

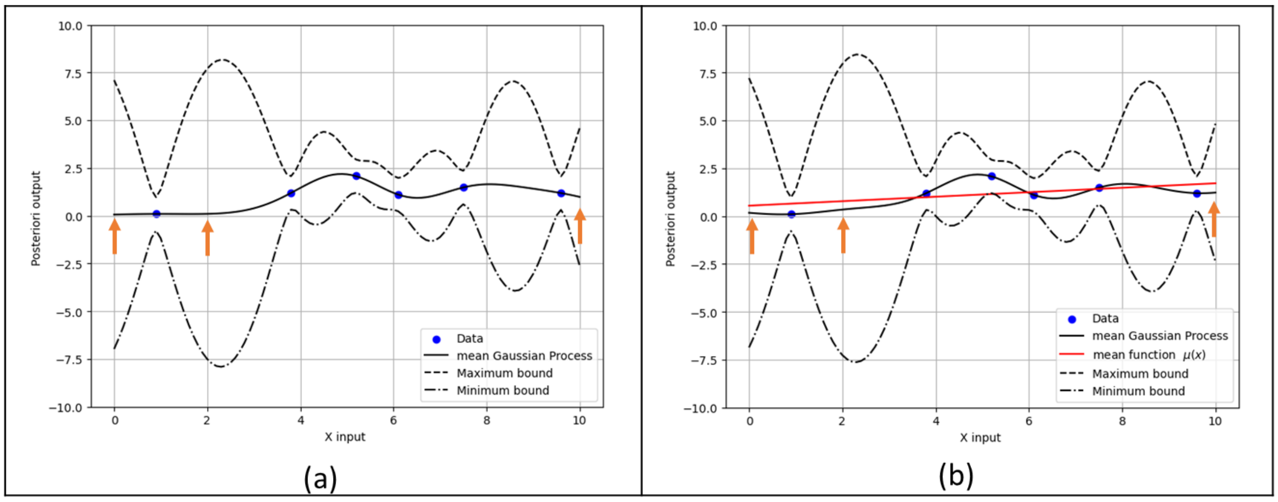


Figure 11: Comparing the results for using zero mean Gaussian Process (a) and non-zero mean Gaussian Process (b).

5. Reflection on Paper "Resource-efficient machining through physics-informed machine learning"

In this section the data presented in [13], which is used in [12] will be presented, and used to train GP. In [12] GP has been used to generated 400 tool lives based on the 12 case presented in [13], which have been used to train GP. The data is given in Table 2, where v (m/min), f (mm/rev) , and Time (min) are cutting speed , the feed rate, and the tool life, respectively.

Table 2: 12 cases presented by [13].

Exp	v (m/min)	f (mm/rev)	Time (min)
1	105	0.15	203

2	105	0.2	223
3	105	0.3	44
4	140	0.15	102
5	140	0.2	60
6	140	0.3	3
7	170	0.15	68
8	170	0.2	35
9	170	0.3	82
10	200	0.15	48
11	200	0.2	6
12	200	0.3	0.5

The physical model used to represent the governing relation between the variables is the Taylor tool life equation given by,

$$vT^n f^m = C \qquad 5\text{-}1$$

To solve for n, m, C , least square method Equation 3-10 is used, where $\boldsymbol{X}, y$ could be generated by applying the log to Equation 5-1,

$$\log(v) + n \times \log(T) + m \times \log(f) = \log C \qquad 5\text{-}2$$

and in the matrix form ,

$$\begin{bmatrix} \log(T_1) & \log(f_1) & -1 \\ \log(T_2) & \log(f_2) & -1 \\ \vdots & \vdots & \vdots \\ \log(T_{12}) & \log(f_{12}) & -1 \end{bmatrix} \begin{bmatrix} n \\ m \\ \log C \end{bmatrix} = \begin{bmatrix} \log\left(\frac{1}{v_1}\right) \\ \log\left(\frac{1}{v_2}\right) \\ \vdots \\ \log\left(\frac{1}{v_{12}}\right) \end{bmatrix} \qquad 5\text{-}3$$

The values for n, m, C are 0.111, 0.380, and 120.684. The code for calculation is in the Appendix.

The performance of this fitting is shown in Figure 12, where the residual R has been calculated as ,

$$R = |T_{data} - T_{Taylor}| \qquad 5\text{-}4$$

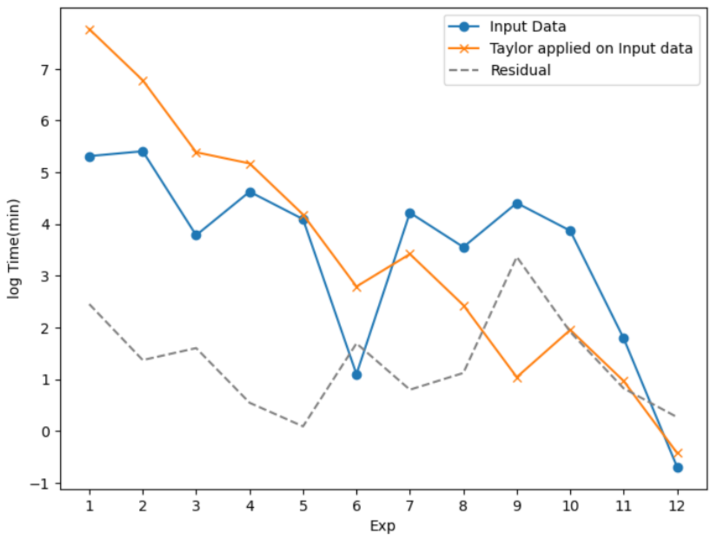


Figure 12: Studying the performance of fitting the data using Taylor model.

These results show that the absolute difference between Taylor model and the data is approximately between 0.1 and 3.4.

Using this 12 data point to train GP and then predict the output is compared with the output from Taylor model applied on the extended data and shown in Figure 13, and Figure 14.

The kernel used is Laplace Kernel Equation 5-5, and $\gamma = 1$, $\sigma_f^2 = 1$, and $\sigma_n^2 = 0$.

$$k(x_i, x_j) = \sigma_f^2 \exp\left(\frac{-|x_i - x_j|}{\gamma}\right) + \sigma_n^2 \delta_{ij} \qquad 5\text{-}5$$

As stated in [12], the 12 data point have been extended to 400 data points, the extension in the input data has been done using $v_{i+1} = v_0 + i \times \Delta v$, $v_o = 105$, $i = 0 \dots 19$, Δv is given by,

$$\Delta v = \frac{200 - 105}{20 - 1} \qquad 5\text{-}6$$

and $f_{i+1} = f_0 + i \times \Delta f$, $f_o = 0.15$, $i = 0 \dots 19$, Δf is given by,

$$\Delta f = \frac{0.3 - 0.15}{20 - 1} \qquad 5\text{-}7$$

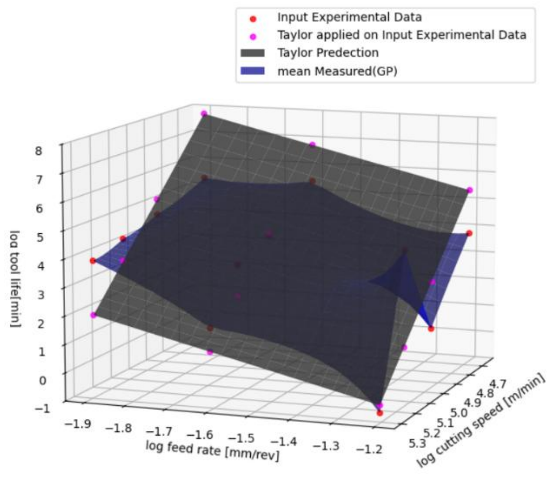


Figure 13: Studying the mean prediction of Gaussian Process and compare it to the Taylor model on generated data.

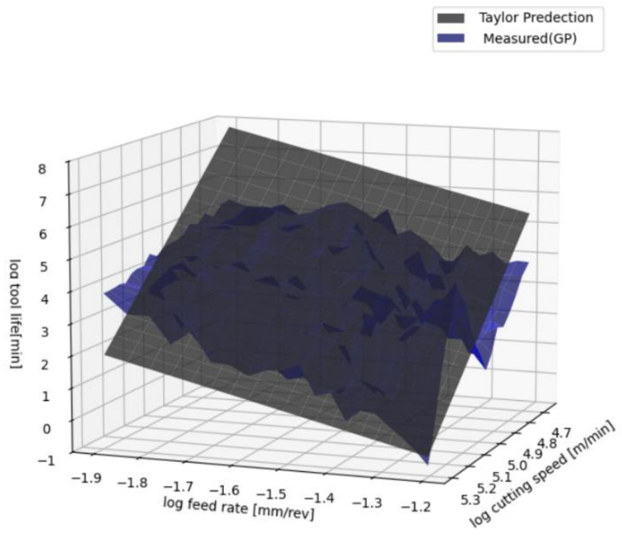


Figure 14: Studying the prediction of Gaussian Process $\sim N(\mu(x_*|\boldsymbol{X}, \boldsymbol{y}), k(x_*, x_*|\boldsymbol{X}, \boldsymbol{y}))$ and compare it to the Taylor model both applied on generated data.

In Figure 13, the blue surface which represents the mean value generated by GP have overlapped with the 12 input experimental points , which shows a primary trust in the GP implementation (still need the optimization of hyperparameters of GP to be investigated), as shown previously in Figure 7 and Figure 8. The output points from the GP is in the form clarified previously using Equation 4-7, this is plotted in Figure 14. Comparing this figure with Figure 13, show the effect of the variance, as the plotted data is following the Gaussian density, that made the surface looks wavy.

Furthermore, studying the residual between the mean prediction of GP and Taylor model is show in Figure 15 and

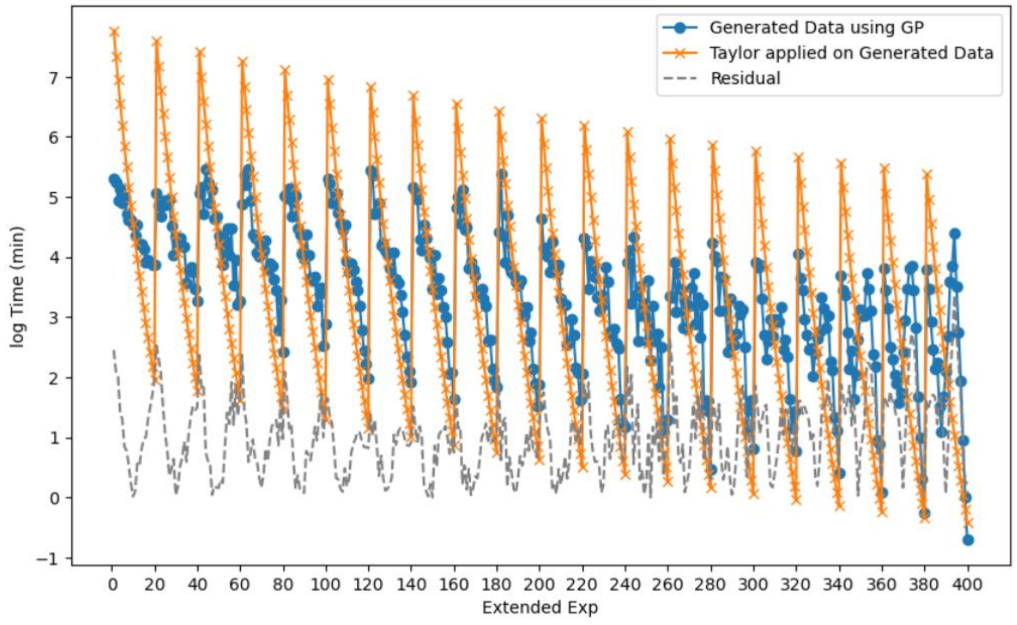


Figure 15: Studying and compare the performance of the generated data using GP and Taylor model.

In Figure 15, the results for both Taylor model and GP is presented. Comparing Figure 15 and Figure 12, it shows that both models have followed the same trend as in the initial input data - the 12 data points. Also the residual, which shown in Figure 15, is in the range approximately between 3.4 and 0.002. Finally, the residual surface is shown in Figure 16.

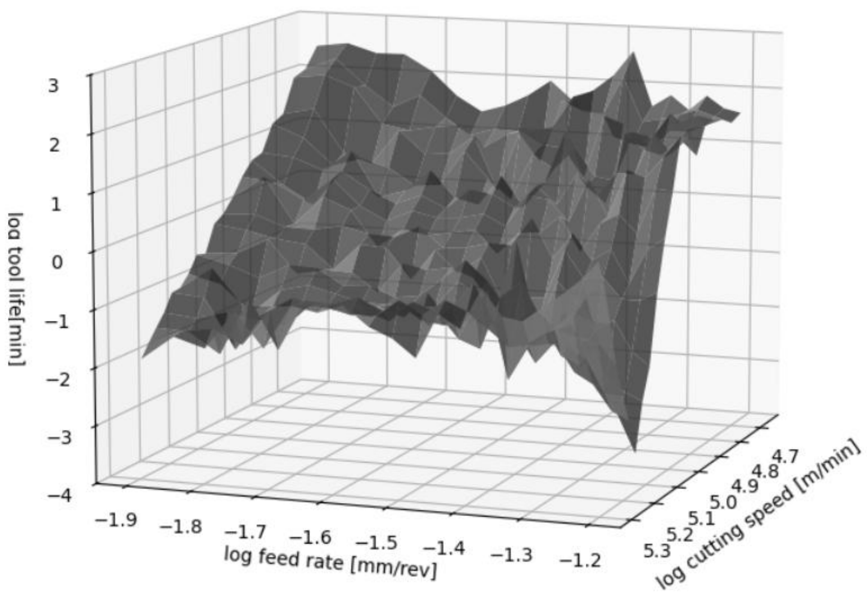


Figure 16: The residual surface between the Taylor model and GP plotted in Figure 14.

The code of this part as it is long is submitted separately under the name "Taylor_GP-submit-final".

Comments on the disagreement between this work and the work presented in [12]:

As shown in figure 1 [12] the absolute of residual is in range with maximum = 1.5 but even if the implementation of GP is different, the GP model should (as I understand) capture the initial experimental data as the variance at these data points should be almost zero. Also, the maximum of range of absolute residual between initial experimental data and Taylor model based on my calculation shown in Figure 12, is = 3.4 , that leaves a question about the calculation of the coefficients of Taylor model, as stated in the same paper they have used ordinary least squares fit, so this point is a must to be checked.

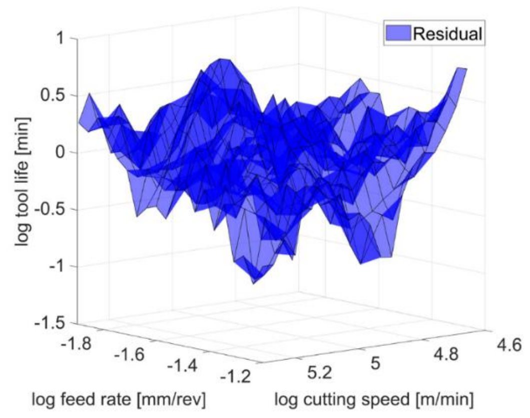


Fig. 1: Measured data, Taylor's prediction, and the residual surface

Figure 17: Residual surface presented in figure1 [12]

6. **References:**

[1] G. E. P. Box and G. C. Tiao, *Bayesian Inference in Statistical Analysis*. Addison-Wesley Publishing Company, 1973.

[2] R. O. Duda, “Pattern Classification,” 2001.

[3] J. Sayir, “Probability and Statistics”.

[4] C. M. Bishop, *Pattern recognition and machine learning*. in Information science and statistics. New York: Springer, 2006.

[5] D. B. Scribe, V. Narayanan, M. Koval, and P. Parashar, “1 Applications of Gaussian Processes”, [Online]. Available: https://www.cs.cmu.edu/~16831-f12/notes/F12/16831_lecture20_venkatrn.pdf

[6] M. Neumann, “Lecture 8: Gaussian Processes,” spring 2024, [Online]. Available: https://classes.cec.wustl.edu/~SEAS-SVC-CSE517A/lecturenotes/08_lecturenote_GPs.pdf

[7] E. Schulz, M. Speekenbrink, and A. Krause, “A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions,” *J. Math. Psychol.*, vol. 85, pp. 1–16, Aug. 2018, doi: 10.1016/j.jmp.2018.03.001.

[8] “cs229-gaussian_processes.pdf.” Accessed: Mar. 04, 2025. [Online]. Available: https://cs229.stanford.edu/section/cs229-gaussian_processes.pdf

[9] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*. in Adaptive computation and machine learning. Cambridge, Mass: MIT Press, 2006.

[10] J. Gonzalvez, E. Lezmi, T. Roncalli, and J. Xu, “Financial Applications of Gaussian Processes and Bayesian Optimization,” Mar. 12, 2019, *arXiv*: arXiv:1903.04841. doi: 10.48550/arXiv.1903.04841.

[11] S. Pfingstl and M. Zimmermann, “On integrating prior knowledge into Gaussian processes for prognostic health monitoring,” *Mech. Syst. Signal Process.*, vol. 171, p. 108917, May 2022, doi: 10.1016/j.ymssp.2022.108917.

[12] M. Tóth, A. Brown, E. Cross, T. Rogers, and N. D. Sims, “Resource-efficient machining through physics-informed machine learning,” *Procedia CIRP*, vol. 117, pp. 347–352, 2023, doi: 10.1016/j.procir.2023.03.059.

[13] M. E. Kabaso, “Experimental Investigation of PCBN Cutting Tool Insert When Hard Turning Hardened 42CrMo4 Steel,” *IOSR J. Eng.*, vol. 4, no. 8, pp. 47–61, Aug. 2014, doi: 10.9790/3021-04814761.

7. Appendix:

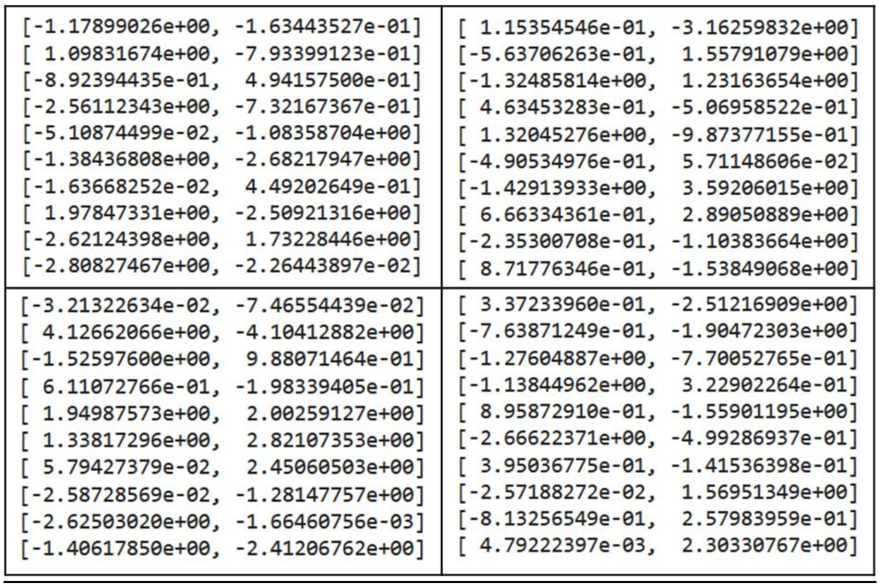
Code for Figure 3

```
import numpy as np
from scipy.stats import multivariate_normal
import matplotlib.pyplot as plt
# Define mean vector (2-dimensional)
mu = [0, 0]
# Define covariance matrix (2x2 identity matrix)
sigma =np.zeros((2, 2))
sigma[0,0]=2
sigma[1,1]=2
# Generate 40 random samplesp
samples = multivariate_normal(mu, sigma).rvs(size=40)
# Generate a meshgrid
# Create a grid of points
x_plot, y_plot = np.meshgrid( samples[:,0], samples[:,1] )
# Define zero matrix for plotting the probability
Probability = np.zeros_like(x_plot)
# Calculating the probability
for i in range(x_plot.shape[0]):
   for j in range(x_plot.shape[1]):
       Probability[i, j] = multivariate_normal.pdf(( x_plot[i,j],  y_plot[i,j]) ,mu,sigma)
# Create a figure for the plot
fig = plt.figure(figsize=(10, 8))
# Create a 3D axis object
ax = fig.add_subplot(111, projection='3d')
# Plot the first surface
#Create grid and multivariate normal for the surface normal distribution
x = np.linspace(-4,4,40)
y = np.linspace(-4,4,40)
# Generate a meshgrid
x_plot_ND,y_plot_ND=np.meshgrid( x, y )
Probability_surface = np.zeros_like(x_plot_ND)# Calculating the probability
for i in range(x_plot_ND.shape[0]):
      for j in range(x_plot_ND.shape[1]):
            Probability_surface[i, j] = multivariate_normal.pdf(( x_plot_ND[i,j],  y_plot_ND[i,j]) ,mu,sigma)
ax.plot_surface(x_plot_ND, y_plot_ND, Probability_surface ,cmap='viridis', alpha=0.7,label = 'Multivariate Normal Distribution Surface')
ax.scatter(x_plot, y_plot, Probability, cmap='viridis', alpha=0.8, label = ' Samples from Multivariate Normal Distribution ')
# Add labels to the axes
ax.set_zlabel('Probability')
ax.set_xlabel('x1')
ax.set_ylabel('x2')
ax.view_init(elev=10, azim=10)
ax.legend()
# Show the plot
plt.show()
```

Calculation for equation 2-17

```
n=40
m1=np.sum(samples[:,0])/n
m2=np.sum(samples[:,1])/n
X1=(samples[:,0]-m1).reshape(n,1)
X2=(samples[:,1]-m2).reshape(n,1)
data=np.column_stack((X1,X2))
sum=np.zeros((2,2))
for i in range (n):
  sum = sum + data[i,:]* data[i,:].reshape(2,1)
sum = sum/n
sum
```

Samples data for Figure 3



```
[-1.17899026e+00, -1.63443527e-01]
[ 1.09831674e+00, -7.93399123e-01]
[-8.92394435e-01,  4.94157500e-01]
[-2.56112343e+00, -7.32167367e-01]
[-5.10874499e-02, -1.08358704e+00]
[-1.38436808e+00, -2.68217947e+00]
[-1.63668252e-02,  4.49202649e-01]
[ 1.97847331e+00, -2.50921316e+00]
[-2.62124398e+00,  1.73228446e+00]
[-2.80827467e+00, -2.26443897e-02]
```

```
[ 1.15354546e-01, -3.16259832e+00]
[-5.63706263e-01,  1.55791079e+00]
[-1.32485814e+00,  1.23163654e+00]
[ 4.63453283e-01, -5.06958522e-01]
[ 1.32045276e+00, -9.87377155e-01]
[-4.90534976e-01,  5.71148606e-02]
[-1.42913933e+00,  3.59206015e+00]
[ 6.66334361e-01,  2.89050889e+00]
[-2.35300708e-01, -1.10383664e+00]
[ 8.71776346e-01, -1.53849068e+00]
```

```
[-3.21322634e-02, -7.46554439e-02]
[ 4.12662066e+00, -4.10412882e+00]
[-1.52597600e+00,  9.88071464e-01]
[ 6.11072766e-01, -1.98339405e-01]
[ 1.94987573e+00,  2.00259127e+00]
[ 1.33817296e+00,  2.82107353e+00]
[ 5.79427379e-02,  2.45060503e+00]
[-2.58728569e-02, -1.28147757e+00]
[-2.62503020e+00, -1.66460756e-03]
[-1.40617850e+00, -2.41206762e+00]
```

```
[ 3.37233960e-01, -2.51216909e+00]
[-7.63871249e-01, -1.90472303e+00]
[-1.27604887e+00, -7.70052765e-01]
[-1.13844962e+00,  3.22902264e-01]
[ 8.95872910e-01, -1.55901195e+00]
[-2.66622371e+00, -4.99286937e-01]
[ 3.95036775e-01, -1.41536398e-01]
[-2.57188272e-02,  1.56951349e+00]
[-8.13256549e-01,  2.57983959e-01]
[ 4.79222397e-03,  2.30330767e+00]
```

Code for Figure 4 (a)

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import multivariate_normal
import math
n=11
m=5 #samples
x = np.linspace(0, 1, n)
# Define mean vector (11-dimensional)
mu = np.linspace(0, 0, n)
gamma=0.5 # this the constant in the kerenl
# Define covariance matrix (11x11 identity matrix)
sigma = np.zeros((n,n))
for i in range (n):
  for j in range (n):
    sigma[i,j]=math.exp((-(x[i]-x[j])**2)/(2*gamma**2))
# Generate 11 random samplesp
samples = multivariate_normal(mu, sigma, allow_singular=True).rvs(size=m)
# Plot each column of the samples against x
plt.figure(figsize=(8, 6))
x_s = np.linspace(0, 1, n)
for i in range(m):
   plt.plot(x_s, samples[i,:],'-' ,label=f'sample {i+1}')
plt.xlabel('X (0 to 1)')
plt.ylabel('Y (Values from Multivariate Normal Samples)')
plt.legend()
plt.grid(True)
plt.show()
```

Code for Figure 4 (b)

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import multivariate_normal
import math
n=11
m=5 #samples
x = np.linspace(0, 1, n)
# Define mean vector (11-dimensional)
mu = np.linspace(0, 0, n)
gamma=0.2 # this the constant in the kerenl
# Define covariance matrix (11x11 identity matrix)
```

```
sigma = np.zeros((n,n))
for i in range (n):
  for j in range (n):
    sigma[i,j]=math.exp((-np.abs(x[i]-x[j]))/(gamma))
# Generate 11 random samplesp
samples = multivariate_normal(mu, sigma, allow_singular=True).rvs(size=m)
# Plot each column of the samples against x
plt.figure(figsize=(8, 6))
x_s = np.linspace(0, 1, n)
for i in range(m):
   plt.plot(x_s, samples[i,:],'-' ,label=f'sample {i+1}')
plt.xlabel('X (0 to 1)')
plt.ylabel('Y (Values from Multivariate Normal Samples)')
plt.legend()
plt.grid(True)
plt.show()
```

Code for Figure 4 (c)

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import multivariate_normal
import math
n=11
m=5 #samples
x = np.linspace(0, 1, n)
# Define mean vector (11-dimensional)
mu = np.linspace(0, 0, n)
# Define covariance matrix (11x11 identity matrix)
sigma = np.eye(n,n)
# Generate 11 random samplesp
samples = multivariate_normal(mu, sigma, allow_singular=True).rvs(size=m)
# Plot each column of the samples against x
plt.figure(figsize=(8, 6))
x_s = np.linspace(0, 1, n)
for i in range(m):
   plt.plot(x_s, samples[i,:],'-' ,label=f'sample {i+1}')
plt.xlabel('X (0 to 1)')
plt.ylabel('Y (Values from Multivariate Normal Samples)')
#plt.title('Multivariate Normal Samples (11 Dimensions)')
plt.legend()
plt.grid(True)
plt.show()
```

Code for Figure 4 (d)

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import multivariate_normal
import math
n=11
m=5 #samples
x = np.linspace(0, 1, n)
# Define mean vector (11-dimensional)
mu = np.linspace(0, 0, n)
# Define covariance matrix (11x11 identity matrix)
for i in range (n):
  for j in range (n):
    sigma[i,j]=x[i]*x[j]
# Generate 11 random samplesp
samples = multivariate_normal(mu, sigma, allow_singular=True).rvs(size=m)
# Plot each column of the samples against x
plt.figure(figsize=(8, 6))
x_s = np.linspace(0, 1, n)
for i in range(m):
   plt.plot(x_s, samples[i,:],'-' ,label=f'sample {i+1}')
plt.xlabel('X (0 to 1)')
plt.ylabel('Y (Values from Multivariate Normal Samples)')
plt.legend()
plt.grid(True)
plt.show()
```

Code for Maximum likelihood, least squares and Figure 5

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import multivariate_normal
import math
x_1 = np.ones(6)
x_2 = np.array([0.9, 3.8, 5.2, 6.1, 7.5, 9.6])
y   = np.array([0.1, 1.2, 2.1, 1.1, 1.5, 1.2]).reshape(6,1)
M = np.column_stack((x_1 , x_2))
w=np.linalg.solve(np.dot(M.T,M),np.dot(M.T,y) )
variance=np.sum((y-np.dot(M,w))**2)/6
sigma = np.sqrt(variance)
plt.figure(figsize=(8, 6))
m=10 # just to show some data points
x_2_data=np.linspace(0.9,9.6,10)
x_1_data=np.ones(10)
M_data = np.column_stack((x_1_data , x_2_data))
s=np.random.normal(0, variance, 10)
plt.scatter(x_2_data,(np.dot(M_data,w).T+s).reshape(10),color='blue', label='Data from (linear fit + noise)')
for i in range(m):
    s=np.random.normal(0, variance, 10)
    plt.scatter(x_2_data,(np.dot(M_data,w).T+s).reshape(10),color='blue')

plt.scatter(x_2,y ,color='black',marker='s', label='Input Data')
plt.plot(x_2,np.dot(M,w)  ,'-k',label='Maximum Likelihood Linear Fit')
plt.plot(x_2,np.dot(M,w)+2*sigma  ,'--k',label='Maximum bound')
plt.plot(x_2,np.dot(M,w)-2*sigma  ,'-.k',label='Minimum bound')

plt.ylim(-1, 3)
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid(True)
plt.show()
```

Code for Bayesian linear regression and Figure 6

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import multivariate_normal
import math
x_1 = np.ones(6)
x_2 = np.array([0.9, 3.8, 5.2, 6.1, 7.5, 9.6])
y   = np.array([0.1, 1.2, 2.1, 1.1, 1.5, 1.2]).reshape(6,1)
M = np.column_stack((x_1 , x_2))
w=np.linalg.solve(np.dot(M.T,M),np.dot(M.T,y) )
sigma_2=1
X = np.column_stack((x_1 , x_2))
A = np.dot(X.T,X)+np.eye(2,2)
A_inv=np.linalg.inv(A)
plt.figure(figsize=(8, 6))
m=2
x_2_data=np.linspace(0.9,9.6,10)
x_1_data=np.ones(10)
M_data = np.column_stack((x_1_data , x_2_data))
variance_data=np.zeros(10)
noise_data=np.zeros(10)
for i in range(10):
   variance_data[i]= np.dot(M_data[i,:],np.dot(A_inv,M_data[i,:].T))+sigma_2
   noise_data[i]= np.random.normal(0, variance_data[i], 1)[0]
```

```
plt.scatter(x_2_data, np.dot(M_data,np.dot(A_inv,np.dot(X.T,y))).reshape(10)+noise_data ,color='blue', label='Data from (Bayesian Estimation +
noise)')
for i in range(m):
    for i in range(10):
      variance_data[i]= np.dot(M_data[i,:],np.dot(A_inv,M_data[i,:].T))+sigma_2
      noise_data[i]= np.random.normal(0, variance_data[i], 1)[0]
    plt.scatter(x_2_data, np.dot(M_data,np.dot(A_inv,np.dot(X.T,y))).reshape(10)+noise_data,color='blue')

variance_input_data=np.zeros(6)
sqrt_variance_input_data=np.zeros(6)
for i in range(6):
   variance_input_data[i]= np.dot(X[i,:],np.dot(A_inv,X[i,:].T))+sigma_2
   sqrt_variance_input_data[i]=np.sqrt(variance_input_data[i])
plt.scatter(x_2,y ,color='black',marker='s', label='Input Data')
plt.plot(x_2,np.dot(X,np.dot(A_inv,np.dot(X.T,y)))  ,'-k',label='Bayesian Estimation')
plt.plot(x_2,np.dot(M,w)  ,'-r',label='Maximum Likelihood Linear Fit')
plt.plot(x_2,np.dot(X,np.dot(A_inv,np.dot(X.T,y))).reshape(6)     +2*sqrt_variance_input_data     ,'--k',label='Maximum bound using Bayesian
Estimation')
plt.plot(x_2,np.dot(X,np.dot(A_inv,np.dot(X.T,y))).reshape(6)      -2*sqrt_variance_input_data    ,'-.k',label='Minimum bound using Bayesian
Estimation')

plt.ylim(-5, 5)
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid(True)
plt.show()
```

Code for Gaussian priori Figure 7 (a)

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import multivariate_normal
import math
n=100
x = np.linspace(0, 10, n)
mu = np.linspace(0, 0, n)

# covariance matrix
gamma=1
sigma = np.zeros((n,n))
for i in range (n):
   for j in range (n):
     sigma[i,j]=math.exp((-0.5*(x[i]-x[j])**2))

# Generate 4 random samplesp for priori
samples = np.random.multivariate_normal(mu, sigma, size=3).T

# Plot each column of the samples against x
plt.figure(figsize=(8, 6))

plt.scatter(x , samples[:, 0],color='gray', label=f'sample {1}')

for i in range(1,3):
    plt.plot(x , samples[:, i], label=f'sample {i+1}')
plt.xlabel('X input ')
plt.ylabel('Y (Values from Multivariate Normal Samples)')
#plt.title('Multivariate Normal Samples (11 Dimensions)')
plt.legend()
plt.grid(True)
plt.show()
```

Code for Gaussian Posterior Figure 7 (b)

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import multivariate_normal
```

```
import math
n=100 # inputs
n_data= 6
x = np.linspace(0, 10, n) # input
mu_input        = np.linspace(0, 0, n) # for posteriori
variance_input   = np.linspace(0, 0, n) # for posteriori
deviation_input  = np.linspace(0, 0, n) # for posteriori
x_data = np.array([0.9, 3.8, 5.2, 6.1, 7.5, 9.6]).reshape(6,1)
y   = np.array([0.1, 1.2, 2.1, 1.1, 1.5, 1.2]).reshape(6,1)
# covariance matrix
K = np.zeros((n_data,n_data))
for i in range (n_data):
  for j in range (n_data):
    K[i,j]=math.exp((-0.5*(x_data [i,0]-x_data [j,0])**2))
k_inv=np.linalg.inv(K)
K_inv_y=np.dot(k_inv ,y)
# k for each x
k_x= np.zeros(n_data)
for j in range (n):
  for i in range (n_data):
    k_x[i]= math.exp((-0.5*(x[j]-x_data[i,0])**2))
  mu_input [j]= np.dot(k_x.reshape(1,n_data),K_inv_y)[0,0]
  variance_input[j]=1.0 - np.dot(k_x.reshape(1,n_data) , np.dot(k_inv ,k_x.reshape(n_data,1)))[0,0]
deviation_input = np.sqrt(variance_input)
plt.figure(figsize=(8, 6))
plt.scatter(x_data ,y , color='blue', label = 'Data')
plt.plot(x ,mu_input,'-k' ,label ='mean Gaussian Process'  )
plt.plot(x ,mu_input + 2*deviation_input,'--k', label='Maximum bound' )
plt.plot(x ,mu_input - 2*deviation_input , '-.k',label='Minimum bound' )
plt.xlabel('X input ')
plt.ylabel('Posteriori output ')
plt.legend()
plt.ylim(-3, 5)
plt.grid(True)
plt.show()
```

Code For Figure 9(a)

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import multivariate_normal
import math
n_data= 6
x   = np.array([0.9, 3.8, 5.2, 6.1, 7.5, 9.6]).reshape(6,1)
y   = np.array([0.1, 1.2, 2.1, 1.1, 1.5, 1.2]).reshape(6,1)
# Hyper_parameters _just for example
gamma   = 1 # ceta[0]
sigma_f = 1 # ceta[1]
sigma_n = 0 # ceta[2]
# ceta has the hyperparameters (gamma, sigma_f^2, sigma_n^2)

ceta=[1,1,0]
# covariance matrix
def covaraince(x,ceta):
        n_data = len(x)
        K = np.zeros((n_data,n_data))
        for i in range (n_data):
          for j in range (n_data):
            if i==j:
              K[i,j]=(ceta[1]**2)*math.exp(-(0.5/(ceta[0]**2))*(x[i,0]-x[j,0])**2) +(ceta[2]**2)
            else:
              K[i,j]=(ceta[1]**2)*math.exp(-(0.5/(ceta[0]**2))*(x[i,0]-x[j,0])**2)
        return K

def Liklihood_fun(x,y,ceta):
    n_data = len(x)
    K      = covaraince(x,ceta)
    K_inv  = np.linalg.inv(K)
    K_det  = np.linalg.det(K)
    likelihood = (-1/2)*np.dot(y.T, np.dot(K_inv,y ))-(1/2)*math.log(K_det )-(n_data/2)*math.log(2*math.pi)
    return likelihood
n=100
c = np.linspace(0.0000001, 10, n)
likelihood=np.zeros(n)
ceta=[4.84711844, 1.045064 ,  0.44071752]
# variang over gamma
for i in range(n):
   ceta[0]=c[i]
   likelihood[i]=Liklihood_fun(x,y,ceta)[0,0]
plt.figure(figsize=(8, 6))
plt.plot(c  ,likelihood,'-k' ,label ='Variation in The Likelihood value based on Change in Gamma coefficient'  )
plt.ylim(-10,-6)
plt.xlabel('Gamma Values ')
plt.ylabel('Marginal likelihood  ')
plt.legend()
```

Code For Figure 9(b)

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import multivariate_normal
import math
n_data= 6
x   = np.array([0.9, 3.8, 5.2, 6.1, 7.5, 9.6]).reshape(6,1)
y   = np.array([0.1, 1.2, 2.1, 1.1, 1.5, 1.2]).reshape(6,1)
# Hyper_parameters _just for example
gamma   = 1 # ceta[0]
sigma_f = 1 # ceta[1]
sigma_n = 0 # ceta[2]
# ceta has the hyperparameters (gamma, sigma_f^2, sigma_n^2)

ceta=[1,1,0]
# covariance matrix
def covaraince(x,ceta):
      n_data = len(x)
      K = np.zeros((n_data,n_data))
      for i in range (n_data):
         for j in range (n_data):
            if i==j:
               K[i,j]=(ceta[1]**2)*math.exp(-(0.5/(ceta[0]**2))*(x[i,0]-x[j,0])**2) +(ceta[2]**2)
            else:
               K[i,j]=(ceta[1]**2)*math.exp(-(0.5/(ceta[0]**2))*(x[i,0]-x[j,0])**2)
      return K

def Liklihood_fun(x,y,ceta):
    n_data = len(x)
    K      = covaraince(x,ceta)
    K_inv  = np.linalg.inv(K)
    K_det  = np.linalg.det(K)
    likelihood = (-1/2)*np.dot(y.T, np.dot(K_inv,y ))-(1/2)*math.log(K_det )-(n_data/2)*math.log(2*math.pi)
    return likelihood
n=100
c = np.linspace(0.0000001, 10, n)
likelihood=np.zeros(n)
ceta=[4.84711844, 1.045064 ,  0.44071752]
# variang over gamma
for i in range(n):
   ceta[1]=c[i]
   likelihood[i]=Liklihood_fun(x,y,ceta)[0,0]
plt.figure(figsize=(8, 6))
plt.plot(c, likelihood, '-k', label=r'Variation in the Likelihood value based on change in $\sigma_f$')

plt.ylim(-10,-6)
plt.xlim(0,6)
plt.xlabel(r'$\sigma_f$')
plt.ylabel('Marginal likelihood  ')
plt.legend()
```

Code For Figure 9(c)

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import multivariate_normal
import math
n_data= 6
x   = np.array([0.9, 3.8, 5.2, 6.1, 7.5, 9.6]).reshape(6,1)
y   = np.array([0.1, 1.2, 2.1, 1.1, 1.5, 1.2]).reshape(6,1)
# Hyper_parameters _just for example
gamma   = 1 # ceta[0]
sigma_f = 1 # ceta[1]
sigma_n = 0 # ceta[2]
# ceta has the hyperparameters (gamma, sigma_f^2, sigma_n^2)

ceta=[1,1,0]
# covariance matrix
def covaraince(x,ceta):
      n_data = len(x)
      K = np.zeros((n_data,n_data))
      for i in range (n_data):
        for j in range (n_data):
           if i==j:
             K[i,j]=(ceta[1]**2)*math.exp(-(0.5/(ceta[0]**2))*(x[i,0]-x[j,0])**2) +(ceta[2]**2)
           else:
             K[i,j]=(ceta[1]**2)*math.exp(-(0.5/(ceta[0]**2))*(x[i,0]-x[j,0])**2)
      return K

def Liklihood_fun(x,y,ceta):
    n_data = len(x)
    K      = covaraince(x,ceta)
    K_inv  = np.linalg.inv(K)
    K_det  = np.linalg.det(K)
    likelihood = (-1/2)*np.dot(y.T, np.dot(K_inv,y ))-(1/2)*math.log(K_det )-(n_data/2)*math.log(2*math.pi)
    return likelihood
n=100
c = np.linspace(0.0000001, 10, n)
likelihood=np.zeros(n)
ceta=[4.84711844 ,1.045064 ,  0.44071752]
# variang over gamma
for i in range(n):
   ceta[2]=c[i]
   likelihood[i]=Liklihood_fun(x,y,ceta)[0,0]
plt.figure(figsize=(8, 6))
plt.plot(c, likelihood, '-k', label=r'Variation in the Likelihood value based on change in $\sigma_n$')

plt.xlim(0, 2)
plt.ylim(-10,-6)

plt.xlabel(r'$\sigma_n$')
plt.ylabel('Marginal likelihood  ')
plt.legend()
```

Code For Optimization

```
from scipy.optimize import minimize

# Original function you want to maximize
def Liklihood_fun_opt(ceta):
    x   = np.array([0.9, 3.8, 5.2, 6.1, 7.5, 9.6]).reshape(6,1)
    y   = np.array([0.1, 1.2, 2.1, 1.1, 1.5, 1.2]).reshape(6,1)
    n_data = len(x)
    K      = covaraince(x,ceta)
    K_inv  = np.linalg.inv(K)
    K_det  = np.linalg.det(K)
    likelihood = (-1/2)*np.dot(y.T, np.dot(K_inv,y ))-(1/2)*math.log(K_det )-(n_data/2)*math.log(2*math.pi)
    return likelihood

# Define the negative for minimization
def neg_f(ceta):
   return -Liklihood_fun_opt(ceta)
ceta=[4.74,1.01,0.404]
# Run optimization
result = minimize(neg_f, ceta)  # ceta is the initial guess

# Output the result
max_x = result.x
max_value = Liklihood_fun_opt(max_x)

print("Maximum value:", max_value)
print("At x =", max_x)
```

Code for Figure 10

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import multivariate_normal
import math
n=101 # inputs
n_data= 6
x = np.linspace(0, 10, n) # input
mu_input        = np.linspace(0, 0, n) # for posteriori
variance_input   = np.linspace(0, 0, n) # for posteriori
deviation_input  = np.linspace(0, 0, n) # for Posterior
x_data = np.array([0.9, 3.8, 5.2, 6.1, 7.5, 9.6]).reshape(6,1)
y   = np.array([0.1, 1.2, 2.1, 1.1, 1.5, 1.2]).reshape(6,1)
sigma_n =0.44071752
sigma_f =4.84711844
gamma = 1.045064
# covariance matrix
K = np.zeros((n_data,n_data))
for i in range (n_data):
  for j in range (n_data):
     if i==j :
        K[i,j]=(sigma_f**2)*math.exp(-(0.5/(gamma**2))*(x_data[i,0]-x_data[j,0])**2) + (sigma_n**2)

     else:
        K[i,j]=(sigma_f**2)*math.exp(-(0.5/(gamma**2))*(x_data[i,0]-x_data[j,0])**2)
k_inv=np.linalg.inv(K)
K_inv_y=np.dot(k_inv ,y)

# k for each x
k_x= np.zeros(n_data)
for j in range (n):
  for i in range (n_data):

              k_x[i]= (sigma_f**2)*math.exp( -(0.5/(gamma**2))*(x[j]-x_data[i,0])**2)


  mu_input [j]= np.dot(k_x.reshape(1,n_data),K_inv_y)[0,0]
  variance_input[j]=abs((sigma_f**2) - np.dot(k_x.reshape(1,n_data) , np.dot(k_inv ,k_x.reshape(n_data,1)))[0,0])
deviation_input = np.sqrt(variance_input)

#+(sigma_n**2)
plt.figure(figsize=(8, 6))

plt.scatter(x_data ,y , color='blue', label = 'Data')
plt.plot(x ,mu_input,'-k' ,label ='mean Gaussian Process'  )
plt.plot(x ,mu_input + 2*deviation_input,'--k', label='Maximum bound' )
plt.plot(x ,mu_input - 2*deviation_input , '-.k',label='Minimum bound' )
plt.xlabel('X input ')
plt.ylabel('Posteriori output ')
plt.legend()
plt.ylim(-10, 10)

plt.grid(True)
plt.show()
```

Code For Figure 11

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import multivariate_normal
import math
w1=0.551
w2=0.117
n=101 # inputs
n_data= 6
x = np.linspace(0, 10, n) # input
mu_input        = np.linspace(0, 0, n) # for posteriori
variance_input   = np.linspace(0, 0, n) # for posteriori
deviation_input  = np.linspace(0, 0, n) # for Posterior
x_data = np.array([0.9, 3.8, 5.2, 6.1, 7.5, 9.6]).reshape(6,1)
y   = np.array([0.1, 1.2, 2.1, 1.1, 1.5, 1.2]).reshape(6,1)
y_m=y-(w1+w2*x_data)
sigma_n =0.44071752
sigma_f =4.84711844
gamma = 1.045064
# covariance matrix
K = np.zeros((n_data,n_data))
for i in range (n_data):
   for j in range (n_data):
      if i==j :
         K[i,j]=(sigma_f**2)*math.exp(-(0.5/(gamma**2))*(x_data[i,0]-x_data[j,0])**2) + (sigma_n**2)

      else:
         K[i,j]=(sigma_f**2)*math.exp(-(0.5/(gamma**2))*(x_data[i,0]-x_data[j,0])**2)
k_inv=np.linalg.inv(K)
K_inv_y=np.dot(k_inv ,y_m)

# k for each x
k_x= np.zeros(n_data)
for j in range (n):
  for i in range (n_data):

            k_x[i]= (sigma_f**2)*math.exp( -(0.5/(gamma**2))*(x[j]-x_data[i,0])**2)


  mu_input [j]= w1+w2*x[j] + np.dot(k_x.reshape(1,n_data),K_inv_y)[0,0]
  variance_input[j]=abs((sigma_f**2) - np.dot(k_x.reshape(1,n_data) , np.dot(k_inv ,k_x.reshape(n_data,1)))[0,0])
deviation_input = np.sqrt(variance_input)

#+(sigma_n**2)
plt.figure(figsize=(8, 6))

plt.scatter(x_data ,y , color='blue', label = 'Data')
plt.plot(x ,mu_input,'-k' ,label ='mean Gaussian Process'  )
plt.plot(x ,0.551+0.117*x,'-r' ,label =r'mean function  $\mu(x)$'  )
plt.plot(x ,mu_input + 2*deviation_input,'--k', label='Maximum bound' )
plt.plot(x ,mu_input - 2*deviation_input , '-.k',label='Minimum bound' )
plt.xlabel('X input ')
plt.ylabel('Posteriori output ')
plt.legend()
plt.ylim(-10, 10)
plt.grid(True)
plt.show()
```

Code for Equation 5-3

```
import numpy as np
import pandas as pd
import math
df = pd.read_csv('main_data_n.csv')
df.head()  # Display
log_v=np.log(df['v (m/min)'].to_numpy())
log_T=np.log(df['Time min)'].to_numpy())
log_f=np.log(df['f (mm/rev)'].to_numpy())
cons = np.ones(12)
y = np.zeros((12, 1))
X = np.column_stack((log_T ,log_f,  -cons, ))
y=-log_v.reshape(12,1)
x=np.linalg.solve(np.dot(X.T,X),np.dot(X.T,y) )
x=x.reshape(3)
c=math.exp(x[2])
n=x[0]
m=x[1]
c,n,m
```