Your Paper

You

August 29, 2023

Abstract

1 Introduction

Our situation is that we have obtained measurements, potentially noisy, of both the u and f. However, we do not have knowledge of the ODE parameters ϕ. This is a typical situation in for example lab work, where we have measurements and want to determine an underlying physical parameter or interpolate the data. This is often done with typical linear regression techniques.

2 Fundamentals

2.1 Gaussian Process

As defined in [7] a Gaussian Process is a collection of random variables for which every finite set of variables have a joint multivariate Gaussian distribution. For a Regression model we define a function $f(\boldsymbol{x}) : \mathcal{X} \to \mathbb{R}$ with the input set $\{\boldsymbol{X}\} = \{\boldsymbol{x_n} \in \mathcal{X}\}_{n=1}^{N}$. We now let $f : \mathcal{X} \to \mathbb{R}$ be our Gaussian Process. To fully define the GP we need to specify a mean function $m(\boldsymbol{x})$ and a covariance function or kernel $k(\boldsymbol{x}, \boldsymbol{x'}) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ for $f(\boldsymbol{x})$:

$$\begin{aligned} m(\boldsymbol{x}) &= \mathbb{E}[f(\boldsymbol{x})] \\ k(\boldsymbol{x}, \boldsymbol{x'}) &= \mathbb{E}[(f(\boldsymbol{x}) - m(\boldsymbol{x}))(f(\boldsymbol{x'}) - m(\boldsymbol{x'}))] \end{aligned} \tag{1}$$

In this work the mean function $m(\boldsymbol{x})$ of the GP is set to zero. This is quite common, mainly for notational reasons [7].
So what we are left with for $m(\boldsymbol{x})$ and $k(\boldsymbol{x}, \boldsymbol{x'})$ is:

$$\begin{aligned} m(\boldsymbol{x}) &= 0 \\ k(\boldsymbol{x}, \boldsymbol{x'}) &= \mathbb{E}[f(\boldsymbol{x})f(\boldsymbol{x'})] \end{aligned} \tag{2}$$

We now write our Gaussian Process $f(\boldsymbol{x})$ as:

$$f(\boldsymbol{x}) \sim \mathcal{GP}(0, k(\boldsymbol{x}, \boldsymbol{x'})) \tag{3}$$

It is important to note that $k(\boldsymbol{x}, \boldsymbol{x'})$ has to be positive definite and symmetric in order to be a valid kernel function. Because of the definition of the GP we can now generate a joint Gaussian distribution for a finite set of points $\boldsymbol{X_*}$:

$$p(\boldsymbol{f_*}|\boldsymbol{X_*}) = \mathcal{N}(\boldsymbol{f_*}|\boldsymbol{0}, \boldsymbol{K_*}) \tag{4}$$

with $[\boldsymbol{K}]_{i,j} = k(\boldsymbol{x_{i*}}, \boldsymbol{x_{j*}})$.

2.2 Regression

The goal of regression is given a training dataset $\mathcal{D} = \{(\boldsymbol{x_i}, y_i)|i = 1, \ldots, N\}$ we want to make predictions for new test points $\boldsymbol{x_*}$ which are not in the training set. There are several methods which can be used to tackle this kind of problem. For example the basic linear regression using least squares and specific basis functions or a neural network that does regression. In this work we make use of the Gaussian process which is a non-parametric model. We will start with a Gaussian Process prior and then use the rules for Gaussian multivariate distributions to define a posterior predictive distribution, which we can then use to make predictions for new test points.

The general setup for regression is that we make observations/measurements y of an underlying function $f(\boldsymbol{x})$ at points $\boldsymbol{x}$ which are corrupted by a zero mean Gaussian noise with variance σ_n^2:

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{X}) + \mathcal{N}(0|\sigma_n^2\boldsymbol{I}) \tag{5}$$

When we set the GP Prior onto the function f we have to add the normally distributed noise. To get the prior for the observations $\boldsymbol{y}$ we simply need to change the covariance function to $\operatorname{cov}(\boldsymbol{y_p}, \boldsymbol{y_q}) =$

$k(\boldsymbol{x_p}, \boldsymbol{x_q}) + \sigma_n^2 \delta_{pq}$.
As described before in equation 4 we can generate a Gaussian random vector for new test data $\boldsymbol{X_*}$. Both $\boldsymbol{y}$ and $\boldsymbol{f_*}$ are now multivariate Gaussians. We can therefore introduce the joint distribution of the observed data and the values of f at the test points $\boldsymbol{X_*}$ as:

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{f_*} \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} K(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 \boldsymbol{I} & K(\boldsymbol{X}, \boldsymbol{X_*}) \\ K(\boldsymbol{X_*}, \boldsymbol{X}) & K(\boldsymbol{X_*}, \boldsymbol{X_*}) \end{bmatrix}\right) \tag{6}$$

From this joint Gaussian prior we can then calculate the predictive posterior distribution by applying the rules for conditioning Gaussians. A good description for this conditioning can be found in [1] (chapter 2.3.1). The predictive distribution is then given by:

$$\begin{aligned} p(\boldsymbol{f_*}|\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{X_*}) &= \mathcal{N}(\boldsymbol{f_*}|\boldsymbol{\mu_*}, \boldsymbol{\Sigma_*}) \\ \boldsymbol{\mu_*} &= K(\boldsymbol{X_*}, \boldsymbol{X})[K(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 \boldsymbol{I}]^{-1} \boldsymbol{y} \\ \boldsymbol{\Sigma_*} &= K(\boldsymbol{X_*}, \boldsymbol{X_*}) - K(\boldsymbol{X_*}, \boldsymbol{X})[K(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 \boldsymbol{I}]^{-1} K(\boldsymbol{X}, \boldsymbol{X_*}) \end{aligned} \tag{7}$$

We now have obtained the posterior predictive mean $\boldsymbol{\mu_*}$ and the posterior covariance σ for the new test inputs $\boldsymbol{X_*}$.

2.3 Kernels

In [7] a kernel is defined as a function $k(\boldsymbol{x}, \boldsymbol{x'})$, which maps two input pairs $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{x'} \in \mathcal{X}'$ into $\mathbb{R}$. To be a valid covariance function the kernel needs to satisfy two main conditions. First it needs to be symmetric so that $k(\boldsymbol{x}, \boldsymbol{x'}) = k(\boldsymbol{x}, \boldsymbol{x'})$ and second it needs to be positive semidefinite (PSD) The general covariance function used in this work is the squared exponential covariance function (SE) as defined in [7]:

$$k(\boldsymbol{x_p}, \boldsymbol{x_q}) = \sigma_f^2 \exp(-0.5(\boldsymbol{x_p} - \boldsymbol{x_q})^\intercal M (\boldsymbol{x_p} - \boldsymbol{x_q})) \tag{8}$$

with the variance hyperparameter σ_f^2 and the matrix $M = \text{diag}(\boldsymbol{l})^{-2}$ containing the vector with the characteristic length scales l_i. The hyperparameters are collected in the vector $\boldsymbol{\theta} = (\boldsymbol{l}, \sigma_f^2)$.
For the one dimensional case the SE covariance function simply becomes:

$$k(x_p, x_q) = \sigma_f^2 \exp\left(-\frac{0.5}{l^2}(x_p - x_q)^2\right) \tag{9}$$

For a two-dimensional case with time being the second dimension, two independent length scale parameters are used for the separate space and time domains $\boldsymbol{l} = (l_x, l_t)$. For this case, we can re-write equation 7 the covariance function with $\boldsymbol{x_p} = (x_p, t_p)^\intercal$ and $\boldsymbol{x_q} = (x_q, t_q)^\intercal$ as the product of two single SE covariance functions:

$$k(\boldsymbol{x_p}, \boldsymbol{x_q}) = \sigma_f^2 \exp\left(-\frac{0.5}{l_x^2}(x_p - x_q)^2\right) \exp\left(-\frac{0.5}{l_t^2}(t_p - t_q)^2\right) \tag{10}$$

There are many kernels one can choose from. A good overview over some kernel functions, as well as how they can be modified, can be found in [2] as well as in [5]. Later we will modify the SE kernels with linear operators, more specifically with differential operators.

2.4 Marginal likelihood

For the optimization of the hyperparameters θ and ϕ the negative marginal log likelihood (*nmll*) [7] is minimized. The *nmll*

$$\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = -\frac{1}{2}\boldsymbol{y}^\intercal \boldsymbol{K_y}^{-1} \boldsymbol{y} - \frac{1}{2}\log|\boldsymbol{K_y}| - \frac{n}{2}\log(2\pi) \tag{11}$$

with $K_y = K_f + \sigma_n^2 \boldsymbol{I}$. As described in [7], the *mll* consists of three parts, which can each be understood separately. The first term $\frac{1}{2}\boldsymbol{y}^\intercal \boldsymbol{K_y}^{-1} \boldsymbol{y}$ is in charge of the data fit, the second term $\log|K_y|$ is the complexity penalty and the last term is a normalization term.

2.5 Linear operators and GPs

A useful property of Gaussian Processes is that their linear transformation is still a Gaussian Process. This main

3 Physics informed framework

We choose our problem setup similar to the one proposed in [6]. We have some sort of linear differential equation or ODE of a physical model:

$$\mathcal{L}_{\boldsymbol{x}}^{\phi} u(\boldsymbol{x}) = f(\boldsymbol{x}) \tag{12}$$

with $\mathcal{L}_{\boldsymbol{x}}^{\phi}$ being the differential operator that resembles the differential equation and ϕ being a list containing the a-priori unknown parameters which also define the differential equation. The function $f(\boldsymbol{x})$ acts as the forcing term and $u(\boldsymbol{x})$ is the solution of the differential equation.

We now start by assuming that $u(\boldsymbol{x})$ is a Gaussian Process of the form:

$$u(\boldsymbol{x}) \sim \mathcal{GP}(\mathbf{0}, k_{uu}(\boldsymbol{x}, \boldsymbol{x'}; \boldsymbol{\theta})) \tag{13}$$

with $k_{uu}(\boldsymbol{x}, \boldsymbol{x'}; \boldsymbol{\theta})$ being the kernel depending on its hyperparameters θ for the $\mathcal{GP}$ (on) $u(\boldsymbol{x})$. With the known linear relation between u and f from equation 12, we can now say that $f(\boldsymbol{x})$ must also be $\mathcal{GP}$ [7] of form:

$$f(\boldsymbol{x}) \sim \mathcal{GP}(\mathbf{0}, k_{ff}(\boldsymbol{x}, \boldsymbol{x'}; \boldsymbol{\theta})) \tag{14}$$

with now k_{ff} being the kernel defining the $\mathcal{GP}$ (on) $f(\boldsymbol{x})$. As shown in [3], [8] and [6] k_{ff} is defined by:

$$k_{ff}(\boldsymbol{x}, \boldsymbol{x'}; \boldsymbol{\theta}, \boldsymbol{\phi}) = \mathcal{L}_{\boldsymbol{x}}^{\phi} \mathcal{L}_{\boldsymbol{x'}}^{\phi} k_{uu}(\boldsymbol{x}, \boldsymbol{x'}; \boldsymbol{\theta}) \tag{15}$$

We will later need the mixed terms between $u(x)$ and $f(x)$, namely k_{uf} and k_{fu} which are formed by:

$$\begin{aligned} k_{uf}(\boldsymbol{x}, \boldsymbol{x'}; \boldsymbol{\theta}, \boldsymbol{\phi}) &= \mathcal{L}_{\boldsymbol{x'}}^{\phi} k_{uu}(\boldsymbol{x}, \boldsymbol{x'}; \boldsymbol{\theta}) \\ k_{fu}(\boldsymbol{x}, \boldsymbol{x'}; \boldsymbol{\theta}, \boldsymbol{\phi}) &= \mathcal{L}_{\boldsymbol{x}}^{\phi} k_{uu}(\boldsymbol{x}, \boldsymbol{x'}; \boldsymbol{\theta}) \end{aligned} \tag{16}$$

To simplify the notation, the dependence on the hyperparameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ is not written explicitly anymore. With this construction we now have two Gaussian Processes. As described in [3] we can now create a joint Gaussian process of u and f in the form: (not sure if I understood this correctly)

$$\begin{bmatrix} u \\ f \end{bmatrix} \sim \mathcal{GP}\left(\mathbf{0}, \begin{bmatrix} k_{uu} & k_{uf} \\ k_{fu} & k_{ff} \end{bmatrix}\right) \tag{17}$$

with $k_{uu} : \mathcal{U} \times \mathcal{U}$, $k_{uf} : \mathcal{U} \times \mathcal{F}$, $k_{fu} : \mathcal{F} \times \mathcal{U}$ and $k_{ff} : \mathcal{F} \times \mathcal{F}$ We now look at noisy observations of both u and f,with noise σ_u and σ_f, of the form:

$$\begin{aligned} y_{i,u} &= u_i(\boldsymbol{x_{i,u}}) + \mathcal{N}(0|\sigma_u^2) \\ y_{i,f} &= u_i(\boldsymbol{x_{i,f}}) + \mathcal{N}(0|\sigma_f^2) \end{aligned} \tag{18}$$

with which we create two data sets $\{\boldsymbol{X_u}, \boldsymbol{y_u}\}$ and $\{\boldsymbol{X_f}, \boldsymbol{y_f}\}$. Similarly, as we did before for the test points in equation 6 we can now write (create) a joint Gaussian distribution of the observations $\boldsymbol{y_u}$ and $\boldsymbol{y_f}$. We collect $\boldsymbol{X_u}, \boldsymbol{X_f}$ in $\boldsymbol{X}$ and $\boldsymbol{y_u}, \boldsymbol{y_f}$ in $\boldsymbol{y}$. With the joint GP at equation 17, the joint distribution will take the following form:

$$p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}, \sigma_{n_u}^2, \sigma_{f_u}^2, \boldsymbol{\phi}) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} k_{uu}(\boldsymbol{X_u}, \boldsymbol{X_u}) + \sigma_{n_u}^2 \boldsymbol{I} & k_{uf}(\boldsymbol{X_u}, \boldsymbol{X_f}) \\ k_{fu}(\boldsymbol{X_f}, \boldsymbol{X_u}) & k_{ff}(\boldsymbol{X_f}, \boldsymbol{X_f}) + \sigma_{n_f}^2 \boldsymbol{I} \end{bmatrix}\right) \tag{19}$$

We can directly use $\boldsymbol{K}$ to compute the mag log likelihood and train the model to obtain the optimized hyperparameters $\boldsymbol{\theta_*}$ and $\boldsymbol{\phi_*}$. More information about the optimization process will be provided in the next chapter. Here we now continue with the optimized hyperparameters.

3.1 Optimization/Finding of the hyperparameters

A very, if not the most important part of the process is the optimization of the hyperparameters. The optimization is done by maximizing the marginal log likelihood (mll) of the joint GP with respect to the hyperparameter. The mll is defined as:

$$\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}, \sigma_{n_u}^2, \sigma_{f_u}^2, \boldsymbol{\phi}) = -\frac{1}{2}\boldsymbol{y}^\intercal \boldsymbol{K}^{-1}\boldsymbol{y} - \frac{1}{2}\log|\boldsymbol{K}| - \frac{n}{2}\log(2\pi) \tag{20}$$

with $\boldsymbol{K}$ being the covariance matrix of the joint GP in equation 19. For the optimization we used different gradient based optimizers, aswell as a heuristic search method, namely the Nelder-Mead method. All the optimizers used came from the scipy.optimize.minimize function from the scipy library [?]. The optimization of the hyperparameters deemed to be harder than expected. Different optimizers were used. Mention that the results varie (numerical instabilities, etc.) but with sufficient number of optimization runs the values always reach nearly the same number of magintude. gradient based optimizers fail with increasing number of training points. However, the the Nelder-Mead optimizer still performs well with a high number of training points.

3.2 Predictions with the Posterior

With the obtained hyperparameters $\boldsymbol{\theta_*}$ and $\boldsymbol{\phi_*}$ we can make new predictions at a test location $\boldsymbol{x_*}$. Similar to the construction in chapter 2.2 we can now formulate the predictive posterior distribution for $u(\boldsymbol{x_*})$ and also for $f(\boldsymbol{x_*})$ as done in [6]:

$$\begin{aligned} p(u(\boldsymbol{x_*})|\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{\theta}_*, \boldsymbol{\phi}_*) &= \mathcal{N}(\mu_{u*}, \Sigma_u) \\ p(u(\boldsymbol{x_*})|\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{\theta}_*, \boldsymbol{\phi}_*) &= \mathcal{N}(\mu_{f*}, \Sigma_f) \end{aligned} \tag{21}$$

with

$$\begin{aligned} \mu_u &= \boldsymbol{\kappa_u}\boldsymbol{K}^{-1}\boldsymbol{y},\ \Sigma_* = k_{uu}(\boldsymbol{x_*}, \boldsymbol{x_*}) - \boldsymbol{\kappa_u}\boldsymbol{K}^{-1}\boldsymbol{\kappa_u}^\intercal \\ \mu_f &= \boldsymbol{\kappa_f}\boldsymbol{K}^{-1}\boldsymbol{y},\ \Sigma_* = k_{ff}(\boldsymbol{x_*}, \boldsymbol{x_*}) - \boldsymbol{\kappa_f}\boldsymbol{K}^{-1}\boldsymbol{\kappa_f}^\intercal \end{aligned} \tag{22}$$

with $\boldsymbol{\kappa_u}$ and $\boldsymbol{\kappa_f}$ being:

$$\begin{aligned} \boldsymbol{\kappa_u} &= [k_{uu}(\boldsymbol{x_*}, \boldsymbol{X_u})k_{uf}(\boldsymbol{x_*}, \boldsymbol{X_f})] \\ \boldsymbol{\kappa_f} &= [k_{fu}(\boldsymbol{x_*}, \boldsymbol{X_u})k_{ff}(\boldsymbol{x_*}, \boldsymbol{X_f})] \end{aligned} \tag{23}$$

4 Results

In the following we will call the physics informed model which is used here the informed model and the vanilla model the model without the physics informed framework and only a general SE kernel.

For testing the framework we will use ODEs and PDEs of physical models. More specifically, we will look at the three main types of PDE, namely Elliptic, Hyperbolic, and Parabolic. The numerically obtained solutions will be used as the ground truth. For obtaining $u(\boldsymbol{x})$, we will fully define the differential equation with its initial- and boundary conditions and then use mathematicas NDSolve (numerical solution) or NSolve (analytical solution) to compute the solution $\{(\boldsymbol{x}); u(\boldsymbol{x}), f(\boldsymbol{x})\}$. The training and validation points will be sampled from the solution domain $[\boldsymbol{x}_{\min}, \boldsymbol{x}_{\max}]$ using a sobol sequence[cite]. We will add some noise to the samples for the training and validation points, to analyse the impact onto the results and to simulate real world data. Furthermore, often the incoporation of noise improves stability in the optimization and prediction process. This will be done by creating two data sets $\{(\boldsymbol{x_u}); \boldsymbol{y_u}\}$ and $\{(\boldsymbol{x_f}); \boldsymbol{y_f}\}$ with $\boldsymbol{y_i} = u(\boldsymbol{x_i}) + \mathcal{N}(\boldsymbol{0}|\sigma_{ni}^2 I);\ i = u, f$. The implementation of a sobol sequence helps with evenly spaced training and validation points, which is important for the training process of the model.
Through the optimization of the mll, we will obtain the parameters of the underlying equation ϕ, as well as the hyperparameters θ of the kernel. With these parameters we will then calculate the predictive mean and covariance and then compare the ground truth with the computed mean function. To benchmark the performance we will look at the marginal log likelihood and at the mean squared error of the validation set with the prediction. As another way of looking at the performance of the informed model we will compare it to a vanilla model with a general SE kernel, as described in 2.2.

n	5	10	20	50
MSE_u				
MSE_f				

Table 1: MSEs for the informed model with different number of training points.

For this we will use the GPy library [4]. However, it is important to acknowledge that the vanilla model with the simple SE kernel only has two hyperparameters, namely the length scale l and the variance σ_f^2. Additionaly, the informed model takes advantage of the joint GP. This means that the informed model has more parameters to optimize and additional information from the joint GP, and therefore it is not a fair comparison. However, it is still interesting to see how the informed model performs compared to the vanilla model and to have a general benchmark. To further investiage this we will give one exmaple with the same amount of hyperparameters, namely the Poisson equation and compare the results of the two models.

4.1 Damped oscillator

As a first example we will look at a one dimensional differential equation of a damped oscillator with mass m, damping γ and force constant k. The differential equation is given by:

$$\begin{aligned} m\frac{\partial^2 u(t)}{\partial t^2} + \gamma\frac{\partial u(t)}{\partial t} + ku(t) &= f(t) \\ u(0) = 0,\ \frac{\partial u(t)}{\partial t} &= 0 \end{aligned} \tag{24}$$

The first data set was generated with $m = 1, \gamma = 0.1, k = 1$ and a simple sinusoidal forcing term $f(t) = 4\sin(2\pi t)$. For the added noise we used $\sigma_{nu} = 10^{-8}$ and $\sigma_{nf} = 10^{-8}$ which can be seen as noise free data. However, adding some noise imporoves stability in some cases. We calculated the result in the time interval $t \in [0, 3]$ from with we sample the training points. The model learns the hyperparameters ϕ (m, γ, k) to be $(3, 3, 3)$. These results are very accurate considering we are only using 9 training points for a model with 5 hyperparameters. As elaborated before, the exact value of the final values after optimizing varies. The predictive distribution for the general solution $u(t)$ and the forcing term $f(t)$ can be seen in figure 1. Looking at the results for the informed model we can see that the prediction for $u(t)$ and for $f(t)$ are very precise. The posterior mean overlines with the validation points and the uncertanty band is so small that is not even visible. The mean squared error of $u(t)$ and $f(t)$ are $MSE_u =$ and $MSE_f =$. The predictive mean of the vanilla model can also be seen in figure 1. The MSEs for the vanilla model are $MSE_{u,va}$ and $MSE_{f,va}$. The MSE for the vanilla model is 3 to 4 orders of magnitude higher than the MSE of the informed model which gets very clear when looking at figure 1. The prediction from the vanilla model are not as precise as the ones from the informed model. The uncertanty band is much larger and the mean does not overline with the validation points in many reagions. This is also reflected in the MSEs. The informed model even manages to accuratly predict between the training points without an uncertanty band, as well as the edges of the domain.
To benchmark the performance further we looked at cases with more noise and with more training points. Table of the mses with n training points

4.2 Heat Equation

The one dimensional time dependent heat equation with a forcing term is:

$$\begin{aligned} \frac{\partial u(x,t)}{\partial t} - \alpha\frac{\partial^2 u(x,t)}{\partial x^2} &= f(x,t) \\ u[0, x] &= \\ u(t, 0) =; u(t, L) &= \end{aligned} \tag{25}$$

with α as the thermal diffusivity.

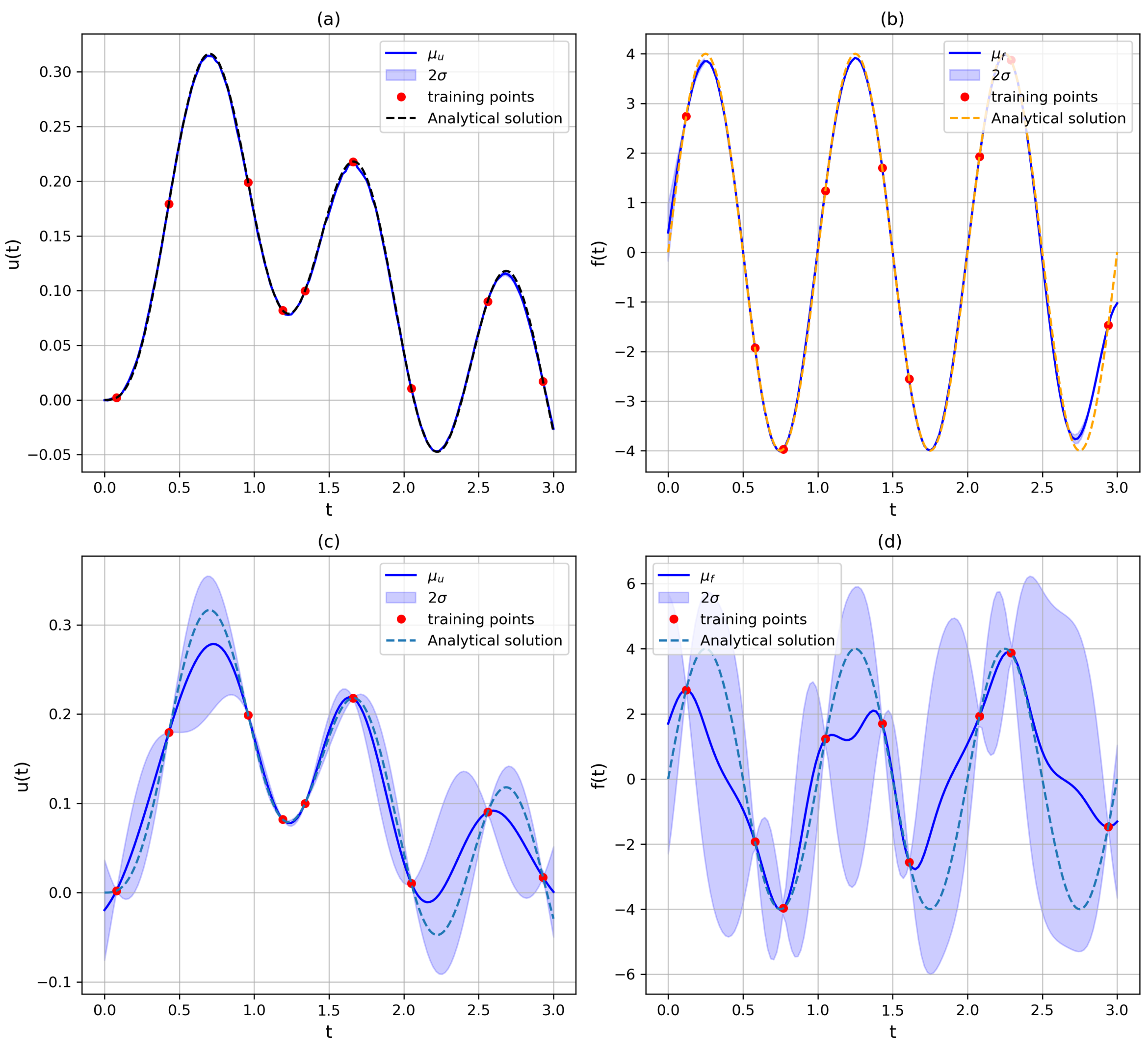

Figure 1: (a) & (b) predictive mean μ_u and μ_f for the informed model, two σ uncertanty band, training and validation points sampled from $\{\boldsymbol{x_u}, u(\boldsymbol{y_u})\}$ and $\{\boldsymbol{x_f}, u(\boldsymbol{y_f})\}$ respectivly. (c) & (d) predictive mean μ_u and μ_f for the vanilla model, two σ uncertanty band, training and validation points sampled from $\{\boldsymbol{x_u}, u(\boldsymbol{y_u})\}$ and $\{\boldsymbol{x_f}, u(\boldsymbol{y_f})\}$ respectivly. The noise σ_{nu} and σ_{nf} were both set to 10^{-8}. The MSEs of the informed model are $MSE_u =$ and $MSE_f =$. The MSEs for the vanilla model are $MSE_{u,va}$ and $MSE_{f,va}$.

4.3 Wave Equation

$$\begin{aligned} \frac{1}{c^2}\frac{\partial^2 u(x,t)}{\partial t^2} - \frac{\partial^2 u(x,t)}{\partial x^2} &= f(x,t) \\ u[0,x] &= \\ u(t,0) =; u(t,L) &= \end{aligned} \tag{26}$$

4.4 Poisson Equation

Here we will look at the Poisson equation in two spatial dimensions with a forcing term and with dirichlet boundary conditions:

$$\begin{aligned} \nabla^2 u(x,y) &= f(x,y) \\ u[0,x] &= \\ u(t,0) =; u(t,L) &= \end{aligned} \tag{27}$$

References

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning.* 2006.

[2] D. Duvenaud. Automatic model construction with gaussian processes. 2014.

[3] Roman Garnett. *Bayesian Optimization.* Cambridge University Press, 2023.

[4] GPy. GPy: A gaussian process framework in python. http://github.com/SheffieldML/GPy, since 2012.

[5] Jochen Görtler, Rebecca Kehlbeck, and Oliver Deussen. A visual exploration of gaussian processes. *Distill*, 2019. https://distill.pub/2019/visual-exploration-gaussian-processes.

[6] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Machine learning of linear differential equations using gaussian processes. *Journal of Computational Physics*, 348:683–693, 2017.

[7] Carl Edward Rasmussen. *Gaussian processes for machine learning.* Adaptive computation and machine learning. 2006.

[8] Simo Särkkä. Linear operators and stochastic partial differential equations in gaussian process regression. pages 151–158, 06 2011.