Imperial College London

Department of Mathematics

Learning Invariances in Dynamical System

Cheng-Cheng Lao

CID: 01353756

Supervised by Dr. Andrew Duncan and Dr. Mark van der Wilk

July 10, 2022

Submitted in partial fulfilment of the requirements for the MSc in Statistics of Imperial College London

The work contained in this thesis is my own work unless otherwise stated.

Signed: Cheng-Cheng Lao   Date: July 10, 2022

Abstract

ABSTRACT GOES HERE

Acknowledgements

ANY ACKNOWLEDGEMENTS GO HERE

1 Introduction

The introduction section goes here[1].

[1]Tip: write this section last.

2 Background

Here we will cover the background knowledge required to understand the remaining thesis, including theortical foundation of Gaussian Process, dynamical systems and invariances.

Background chapter.

2.1 Gaussian Process

Gaussian Process (GP) can be thought of a distribution over functions. Rasmussen and Williams (2006) More formally,

> GP is a collection of random variables, any finite number of which have a joint Gaussian distribution

We will be able to specify a GP on $f(\mathbf{x})$ by mean function $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$ as well as the kernel function $k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[f(\mathbf{x}) - m(\mathbf{x})(f(\mathbf{x}' - m\mathbf{x}'))]$. We then then write

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')).$$

2.2 Dynamical Systems

A dynamical system is simply a system that evolves with time under some kind of rules (Alligood et al. (2000))

2.3 Invariances

Invariances are functions of the states that describe a dynamical system, and is unchanged throughout the evolution of the system over time. An example would be from the field of physics, the conservation of energy, which will be unchanged throught out the trajectories of the system.

2.4 GPflow

GPflow is used throughout the project (Matthews et al. (2017)). It is a Python package built on TensorFlow to allow efficient and fast computation on GPUs to do GP inferences. A particular powerful feature is that it allows the choosing of hyperparameter such as lengthscales and variance of a kernel automatic by calculating the gradient and perform backprobagation. Later on, we will a lot more parameters we need to optimise with respect to maginal likelihood objective.

2.5 Related Work

2.5.1 Physics Informed Machine Learning

2.5.2 Learning Invariance using Marginal Likelihood

2.5.3 Symmetry and Invariances

2.5.4 Symbolic approach

3 Invariance Kernel

In this chapter we will start with general theoretical construction of invariance kernel given the knowlwedge of the invariance of the system. We will then apply the construction to various systems, namely linear and nonlinear system in both one and two dimensions.

3.1 General Construction

If we are given a dynamical system of dimension d with variables $\mathbf{q}, \mathbf{p}$, where $\mathbf{q} = (q_1, q_2, \ldots, q_d)$ is the vector of positional coordinates, and $\mathbf{p} = (p_1, p_2, \ldots, p_d)$ is the vector of velocity coordinates of the states of the dynamical system. We are interested to predict the future trajectories of the state of the system. Therefore, we would like to know the time derivative of these coordinates so we can update them according to Euler's integrator. These time derivatives are referred to as the dynamics of a dynamical system, which governs the evolution of the dynamical system, and is a function of the coordinates $\mathbf{q}$ and $\mathbf{p}$ For our systems, we will denote the dynamics of $\mathbf{p}$ as $\frac{d\mathbf{p}}{dt} = \mathbf{a}(\mathbf{q}, \mathbf{p})$, and that of $\mathbf{q}$ as $\frac{d\mathbf{q}}{dt} = \mathbf{v}(\mathbf{q}, \mathbf{p})$. Once we have the dynamics, we can integrate up to obtain the future trajectories. Notation wise, we will collect the two dynamics term and call them

$$\mathbf{f}(\mathbf{q}, \mathbf{p}) = \begin{pmatrix} \mathbf{a}(\mathbf{q}, \mathbf{p}) \\ \mathbf{v}(\mathbf{q}, \mathbf{p}) \end{pmatrix}$$

For simplicity of notation, the dependence on $\mathbf{q}, \mathbf{p}$ is now implicit. We will put independent GP prior on $\mathbf{a}$ and $\mathbf{v}$ since there are no reason we should assume they are correlated. For the choice of kernel, we chose the standard smooth kernel, the squared exponential kernel, or RBF, and we denote this kernel to be K_{RBF}. We will also denote any set of coordinates of length n and dimension d as

$$X = \begin{pmatrix} q_{11} & q_{21} & \cdots & q_{d1} & p_{11} & p_{21} & \cdots & p_{d1} \\ q_{12} & q_{22} & \cdots & q_{d2} & p_{12} & p_{22} & \cdots & p_{d2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ q_{1n} & q_{2n} & \cdots & q_{dn} & p_{1n} & p_{2n} & \cdots & p_{dn} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_n \end{pmatrix}.$$

Without loss of generality, we will choose to stack the dynamics vertically in $\mathbf{f}$; for example in a d dimensional system, we have

$$\mathbf{f}(X) = \begin{pmatrix} a_1(\mathbf{x}_1) \\ \vdots \\ a_1(\mathbf{x}_n) \\ \vdots \\ a_d(\mathbf{x}_1) \\ \vdots \\ a_d(\mathbf{x}_n) \\ v_1(\mathbf{x}_1) \\ \vdots \\ v_1(\mathbf{x}_n) \\ \vdots \\ v_d(\mathbf{x}_1) \\ \vdots \\ v_d(\mathbf{x}_n) \end{pmatrix}$$

For our naive independent RBF GP prior, we have

$$K = \mathrm{Cov}(\mathbf{f}(X), \mathbf{f}(X')) = \begin{pmatrix} K_{RBF,a_1}(X,X') & \dots & \dots & \dots & 0 \\ \vdots & \ddots & \dots & \ddots & \vdots \\ 0 & \dots & K_{RBF,a_d}(X,X') & \dots & 0 \\ \vdots & \ddots & \vdots & K_{RBF,v_1}(X,X') & \vdots \\ 0 & \dots & 0 & \dots & K_{RBF,v_d}(X,X') \end{pmatrix},$$

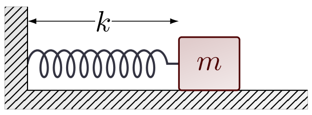
with all the off diagonal terms being zero block matrix because of the independence prior assumption. This naive kernel will be our baseline to be compared to throughout the whole project. Now we can start considering the invariance. If we assume the invariance is true throughout the input space $\mathbb{R}^{2d}$, then we can condition the GP on the a grid of points, which we referred to invariance points, X_L, and assume it has length ℓ, and we will call the invariance constraints L. The form of L will depend on the system as well as X_L, and examples will be described in the following sections. Since the invariance function will always be a linear function on the dynamics, if we apply this linear transformation L on $\mathbf{f}(X_L)$, $L(\mathbf{f}(X_L))$ will again be a GP with a transformed kernel. Therefore, we have

$$\begin{pmatrix} \mathbf{f}(X) \\ L[\mathbf{f}(X_L)] \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} 0_{2nd} \\ 0_\ell \end{pmatrix}, \begin{pmatrix} K & LK \\ KL^T & LKL^T \end{pmatrix} \right)$$

3.2 1D System

3.2.1 Linear

We will first examine one of the most simple dynamical system, an 1D simple harmonic motion (SHM). An example would be a mass spring system as shown in figure with mass m and spring constant k.



The defining equation is

$$m\frac{d^2q}{dt^2} = -kq$$

, where q is the displacement. And the analytical solution would be of the form $q = A\sin(\omega_0 t + \phi)$, where $\omega = \frac{k}{m}$ and A, ϕ depends on the initial condition, which dictates the amplitude and phase of the motion. For this case, we have

$$\mathbf{f}(X) = \begin{pmatrix} \mathbf{a}(X) \\ \mathbf{v}(X) \end{pmatrix}$$

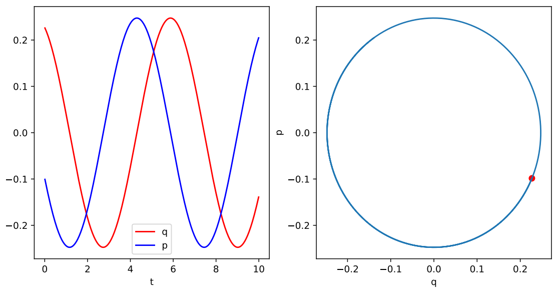


Figure 3.1: Example Trajectory of SHM

We also have the energy, $E = \frac{mp^2}{2} + \frac{kq^2}{2}$, Therefore, to obtain our invariance L, we use the conservation of energy $\frac{dE}{dt} = 0$ so we finally have

$$L(a, v) = mpa + kqv = 0$$

While L is not able to be into a matrix form, it is an linear operator as shown below. If we have

$$X_L = \begin{pmatrix} q_{L,1} & p_{L,1} \\ \vdots & \vdots \\ q_{L,\ell} & p_{L,\ell} \end{pmatrix} = \begin{pmatrix} \vdots & \vdots \\ q_L & p_L \\ \vdots & \vdots \end{pmatrix} = \begin{pmatrix} \mathbf{x}_{L,1} \\ \vdots \\ \mathbf{x}_{L,\ell} \end{pmatrix},$$

then we have

$$L([\mathbf{f}(X_L)]) = \begin{pmatrix} mp_{L,1}a(q_{L,1},p_{L,1}) + kq_{L,1}v(q_{L,1},p_{L,1}) \\ \vdots \\ mp_{L,\ell}a(q_{L,\ell},p_{L,\ell}) + kq_{L,\ell}v(q_{L,\ell},p_{L,\ell}) \end{pmatrix}.$$

Combine with original GP prior assumption, we will have

$$\begin{pmatrix} \mathbf{f}(X) \\ L([\mathbf{f}(X_L)]) \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} 0_{2n} \\ 0_\ell \end{pmatrix}, \begin{pmatrix} A & B \\ C & D \end{pmatrix} \right),$$

where

$$A = K(X,X), B = \begin{pmatrix} K_{RBF,a} \\ K_{RBF,v} \end{pmatrix} \odot \begin{pmatrix} mP_L \\ kQ_L \end{pmatrix}, C = B^T, D = K_{RBF,a} \odot m^2(p_L \otimes p_L) + K_{RBF,v} \odot k^2(q_L \otimes q_L),$$

where $\odot$ is the element wise product and $\otimes$ is the Kronecker product so that

$$P_L = \begin{pmatrix} p_{L,1} & \cdots & p_{L,\ell} \\ \vdots & \text{repeats n rows} & \vdots \\ p_{L,1} & \cdots & p_{L,\ell} \end{pmatrix}, Q_L = \begin{pmatrix} q_{L,1} & \cdots & q_{L,\ell} \\ \vdots & \text{reqeats n rows} & \vdots \\ q_{L,1} & \cdots & q_{L,\ell} \end{pmatrix}$$

and we have

$$p_L \otimes p_L = \begin{pmatrix} p_{L,1}^2 & p_{L,1}p_{L,2} & \cdots & p_{L,1}p_{L,\ell} \\ \vdots & \vdots & \vdots & \vdots \\ p_{L,\ell}p_{L,1} & p_{L,\ell}p_{L,2} & \cdots & p_{L,\ell}^2 \end{pmatrix}, q_L \otimes q_L = \begin{pmatrix} q_{L,1}^2 & q_{L,1}q_{L,2} & \cdots & q_{L,1}q_{L,\ell} \\ \vdots & \vdots & \vdots & \vdots \\ q_{L,\ell}q_{L,1} & q_{L,\ell}q_{L,2} & \cdots & q_{L,\ell}^2 \end{pmatrix},$$

These matrices can be obtained if we try to compute the covariance manually. For B, we wish to calculate

$$\begin{aligned} B_{ij} &= \mathrm{Cov}(\mathbf{f}(X), L[\mathbf{f}(X_L)])_{ij} \\ &= \mathrm{Cov}(\mathbf{f}(X)_i, L\mathbf{f}(X_L)_j) \\ &= \begin{cases} \mathrm{Cov}(a(q_i,p_i), mp_{L,j}a(q_{L,j},p_{L,j}) + kq_{L,j}v(q_{L,j},p_{L,j})) & i \leq n \\ \mathrm{Cov}(v(q_i,p_i), mp_{L,j}a(q_{L,j},p_{L,j}) + kq_{L,j}v(q_{L,j},p_{L,j})) & i > n \end{cases} \\ &= \begin{cases} K_{RBF,a}(\mathbf{x}_i, \mathbf{x}_{L,j})mp_{L,j} & i \leq n \\ K_{RBF,v}(\mathbf{x}_i, \mathbf{x}_{L,j})kq_{L,j} & i > n \end{cases}, \end{aligned}$$

and hence we have the form above. For D, we have

$$\begin{aligned}D_{ij} &= \mathrm{Cov}(L[\mathbf{f}(X_L)], L[\mathbf{f}(X_L)])_{ij}\\ &= \mathrm{Cov}(mp_{L,i}a(q_{L,i}, p_{L,i}) + kq_{L,i}v(q_{L,i}, p_{L,i}), mp_{L,i}a(q_{L,i}, p_{L,i}) + kq_{L,i}v(q_{L,i}, p_{L,i}))\\ &= m^2 p_{L,i}p_{L,j}K_{RBF,a}(\mathbf{x}_{L,i}, \mathbf{x}_{L,j}) + k^2 q_{L,i}q_{L,j}K_{RBF,v}(\mathbf{x}_{L,i}, \mathbf{x}_{L,j})\end{aligned}$$

using the bilnear property of the covariance operator and the fact that v and a are independent. Since we assume invariance on these invariance points, we will condition on $L([\mathbf{f}(X_L)]) = 0$. Now we can simply use the Gaussian conditional formula to obtain the Schur Complement

$$\mathbf{f}(X)|L[\mathbf{f}(X_L)] = 0 \sim \mathcal{N}(0_{2n}, A - BD^{-1}C),$$

we will then call the covariance part our Invariance Kernel for 1D SHM, K_L.

3.2.2 Non Linear

The story is pretty much the same for nonlinear system, it is just the fitting would be expected to be more difficult. A simple nonlinear system in every day life is a simple pendulum as shown in figure below. The governing equation is

$$\frac{d^2q}{dt^2} = -\frac{g}{\ell}\sin q,$$

where q is the angle of displacement this time. The nonlinear dynamics bit occurs because of the sine term, which will complicate things slightly. However, since $\sin x \approx x$ at small angle, this system is approximately linear under small displacement. There is no analytical solution to this nonlinear problem.

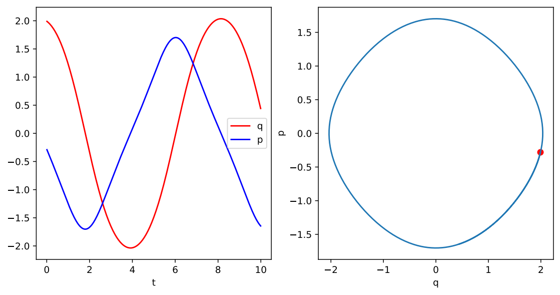


Figure 3.2: Example Trajectory of Pendulum

This time we have energy, $E = \frac{m\ell^2p^2}{2} + mg\ell(1 - \cos q)$, and by setting the time derivative to 0, we have

$$L(a, v) = \frac{dE}{dt} = m\ell^2 pa + mg\ell(\sin q)v = 0.$$

If we cancel out the common term $m\ell$ since their product cannot be zero, we have

$$L(a, v) = \ell pa + g(\sin q)v = 0$$

Most of the terms are unchanged from the linear case. However, this time,

$$B = \begin{pmatrix} K_{RBF,a} \\ K_{RBF,v} \end{pmatrix} \odot \begin{pmatrix} \ell P_L \\ g\sin(Q_L) \end{pmatrix}, D = K_{RBF,a} \odot \ell^2 (p_L \otimes p_L) + K_{RBF,v} \odot g^2 (\sin(q_L) \otimes \sin(q_L)),$$

where

$$\sin(Q_L) = g \begin{pmatrix} \sin(q_{L,1}) & \dots & \sin(q_{L,\ell}) \\ \vdots & \text{reqeats n rows} & \vdots \\ \sin(q_{L,1}) & \dots & \sin(q_{L,\ell}) \end{pmatrix},$$

$$\sin(q_L) \otimes \sin(q_L) = \begin{pmatrix} \sin(q_{L,1})^2 & \sin(q_{L,1})\sin(q_{L,2}) & \dots & \sin(q_{L,1})\sin(q_{L,\ell}) \\ \vdots & \vdots & \vdots & \vdots \\ \sin(q_{L,\ell})\sin(q_{L,1}) & \sin(q_{L,\ell})\sin(q_{L,2}) & \dots & \sin(q_{L,\ell})^2 \end{pmatrix},$$

derived in almost exactly the same way as the linear case.

3.3 Damped System

Before we have considered a perfect system, i.e. a system in an ideal world without frictions and the conservation of energy is perfectly obeyed such that $L = 0$ exactly. However, in a real world system, there will be dissipation where energy is lost in the form of heat etc. Therefore, to model that, we need to allow "approximate" invariance, such that the invariance L is noisy and not always equal to zero. To do so, it is similar in spirit to when we add noise to the underlying GP to represent noise signal, where we have $K + \sigma_n^2 \mathbb{I}$. Here, since the white noise is uncorrelated, the only place the noise will enter is D, so we simply have to add a noise term to D and replace all D with $\tilde{D} = D + \epsilon\mathbb{I}$, where ϵ is a parameter to be learnt. For the data, we will model the system as a damped SHM or pendulum with linear velocity dependent force. If we denote $\omega_0^2 = k/m$ or $\omega 0^2 = g/\ell$ We will have the equation of motion

$$\frac{d^q}{dt^2} + 2\gamma\frac{dq}{dt} + \omega_0^2 q = 0,$$

and the γ is the damping factor that controls how much frictional force there is. Note that we will only focus on underdamping case so that the system still oscillate but with gradually reduced amplitude instead of critical damped or overdamped case where the system simply slowly decays to still. A simple trajectory for damped SHM and damped pendulum is shown below.

3.4 2D System

A 2D system is not that different from an 1D system. The major difference being there are two more extra variables. We again look at two simple examples, a linear 2D SHM, and a nonlinear double pendulum.

3.4.1 Linear

A simple extension to 1D SHM to 2D is just allowing the spring to be at an angle so that the system is now in a 2D space. Again, the equations are simple that we have two equations for the two coordinates.

$$\begin{cases} \frac{d^2q_1}{dt^2} = -\frac{k}{m}q_1 \\ \frac{d^2q_2}{dt^2} = -\frac{k}{m}q_2 \end{cases}$$

The analytical solution is exactly the same, but now there are two of them with differing amplitudes and phases depending on the initial condition. And now we have

$$\mathbf{f}(X) = \begin{pmatrix} \mathbf{a_1}(X) \\ \mathbf{a_2}(X) \\ \mathbf{v_1}(X) \\ \mathbf{v_2}(X) \end{pmatrix},$$

where $\mathbf{a_1}$ and $\mathbf{a_2}$ are the dynamics or time derivative for $\mathbf{p_1}$ and $\mathbf{p_2}$ respectively; similarly $\mathbf{v_1}$ and $\mathbf{v_2}$ are the time derivative of $\mathbf{q_1}$ and $\mathbf{q_2}$. In this system, the energy is the sum of energy in the two directions so $E = \frac{m(p_1^2+p_2^2)}{2} + \frac{k(q_1^2+q_2^2)}{2}$ and so the invariance $L(a_1, a_2, v_1, v_2) = mp_1a_1 + mp_2a_2 + kq_1v_1 + kq_2v_2 = 0$. Now our naive baseline GP has the kernel

$$K(X, X') = \begin{pmatrix} K_{RBF,a_1}(X, X') & 0 & 0 & 0 \\ 0 & K_{RBF,a_2}(X, X') & 0 & 0 \\ 0 & 0 & K_{RBF,v_1}(X, X') & 0 \\ 0 & 0 & 0 & K_{RBF,v_2}(X, X') \end{pmatrix}.$$

We will then have the joint distribution of

$$\begin{pmatrix} \mathbf{f}(X) \\ L[\mathbf{f}(X_L)] \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} 0_{4n} \\ 0_\ell \end{pmatrix}, \begin{pmatrix} A & B \\ C & D \end{pmatrix} \right),$$

with

$$A = K(X, X'), B = \begin{pmatrix} K_{RBF,a_1} \\ K_{RBF,a_2} \\ K_{RBF,v_1} \\ K_{RBF,v_2} \end{pmatrix} \odot \begin{pmatrix} mP_{1L} \\ mP_{2L} \\ kQ_{1L} \\ kQ_{2L} \end{pmatrix}, C = B^T$$

$$D = K_{RBF,a_1} m^2 \odot (p_{1L} \otimes p_{1L}) + K_{RBF,a_2} m^2 \odot (p_{2L} \otimes p_{2L})$$
$$+ K_{RBF,v_1} k^2 \odot (q_{1L} \otimes q_{1L}) + K_{RBF,v_2} k^2 \odot (q_{2L} \otimes p_{2L})$$

where the terms are defined the exact same way as before, but now with respect to different coordinates, and the derivation are also the same. We will then again take the Schur Complement.

3.4.2 Non Linear

Now we will to introduce double pendulum, which is a fairly nonlinear system and quite complicated. It has two mass blobs m_1 and m_2 as well as two lengths for the pendlum stem ℓ_1 and ℓ_2. The defining equations are as follows:

$$\begin{cases} \frac{d^2 q_1}{dt^2} = \frac{-g(2m_1+m_2)\sin q_1 - m_2 g \sin(q_1-2q_2) - 2\sin(q_1-q_2) m_2 \left(p_2^2 l_2 + p_1^2 l_1 \cos(q_1-q_2)\right)}{l_1(2m_1+m_2-m_2\cos(2q_1-2q_2))} \\ \frac{d^2 q_2}{dt^2} = \frac{2\sin(q_1-q_2)\left(p_1^2 l_1(m_1+m_2) + g(m_1+m_2)\cos q_1 + {p_2}^2 l_2 m_2 \cos(q_1-q_2)\right)}{l_2(2m_1+m_2-m_2\cos(2q_1-2q_2))} \end{cases}$$

As we can see, it is very complicated in term of the form. We also have form of energy

$$E = -(m_1+m_2) g l_1 \cos q_1 - m_2 g l_2 \cos q_2 + \frac{m_1 l_1^2 p_1^2}{2} + \frac{m_2}{2}(l_1^2 p_1^2 + l_2^2 p_2^2 + 2 l_1 l_2 p_1 p_2 \cos(q_1 - q_2))$$

While the form is more complicated, the underlying principle to construct the invariance kernel. We again differentiate with respect to time and set it to zero to obtain the invariance equation to obtain

$$L(a_1, a_2, v_1, v_2) = \frac{dE}{dt} = (m_1 + m_2) g l_1 \sin\theta_1 v_1 + m_2 g l_2 \sin\theta_2 v_2 + m_1 l_1^2 \dot{\theta}_1 a_1 +$$
$$m_2(l_1^2 \dot{\theta}_1 a_1 + l_2^2 \dot{\theta}_2 a_2 + l_1 l_2 (\dot{\theta}_2 \cos(\theta_1 - \theta_2) a_1 + \dot{\theta}_1 \cos(\theta_1 - \theta_2) a_2 - \dot{\theta}_1 \dot{\theta}_2 \sin(\theta_1 - \theta_2)(v_1 - v_2))) = 0$$

The form of the invariance matrix is too cubersome to write down, but the derivation is as straightforward as before.

3.5 Local Invariance

Conditioning on a grid of invariance points is simply not scalable in high dimensions. If we have an invariance density of 1 in a ± 5 range for 2D SHM, then we will have 10000 points. We will easily run out of memory and also taking the matrix inverse are very difficult and time consuming. As a result, to make the approach scalable to 2D space, we will use the idea of local invariance. Which is to take samples in the local region around training and testing points and condition on these points. Hopefully that will constraint our results enough.

4 Learning Invariance

Previously, we have assumed the knowledge of the dynamics equation from Newton's Law, which allowed us to derive the energy and thereafter the invariance equation. However, for the kernel to be useful in real life, we cannot assume the equations of the system beforehand, since if we have the knowledge, then the system is pretty much solved that we can just use an ODE integrator to obtain the trajectories. Therefore, we should find a way for the system to learn the form of invariance from data directly. One way to do so is to parameterise our invariance function and allow the system to learn the coefficients by maximising the log marginal likelihood.

4.1 1D system

A simple way to parameterise an unknown function, when the problem is relatively bounded is to use polynomial basis to paramertise the unknown function $L(a, v)$. In 1D cases, for our examples, they are simple enough so that $L(a, v)$ is a sum of a function of a and p only, $f_p a$ and a different function of v and q only, $g_q(v)$ so that $L(a, v) = f_p(a) + g_q(v)$. In the 1D SHM case, $f_p(a) = mpa$ and $g_q(v) = kqv$. For simple pendulum, we have $f_p(a) = \ell pa$ and $g_q(v) = \sin qv$. We will therefore set

$$f_p(a) = \sum_{i=0}^{n_f} c_{f,i} p^i, g_q(a) = \sum_{i=0}^{n_g} c_{g,i} q^i,$$

where $c_{f,i}, c_{g,i}$ are the parameter we will optimise based on marginal likelihood and we will evaluate different degree of polynomial n_f, n_g and see which gives the best results.

4.2 2D system

In 2D, things are a bit more complicated. For example, it is very clear that in double pendulum, the invariance function is not seperable. Instead, we need to consider all combinations of polynomials; i.e. multivariate polynomial. Therefore, for 2D we have $L(a, v) = f_1 a_1 + f_2 a_2 + g_1 v_1 + g_2 v_2$. We will paramertise each of f_1, f_2, g_1, g_2 with

$$\sum_{i,j,k,l=0}^{n} c_{ijkl} p_1^i p_2^j q_1^k q_2^l$$

Therefore, there will be many more coefficients.

5 Experiments

5.1 Data Generation

To generate the data, we use RK4 integrator to generate data using the ODEs for each problem. We will also add some small Gaussian noise 10^{-8} to the dynamics to reflect the real world better. Since we aim to predict the value of dynamics at any point in the input space, we will need the dynamics as targets. This is done by combing forward and backward difference to compute the finite gradient. We also choose the time step to be 0.01 to ensure sufficient accuracy for the integrator. For training data, we will choose a few random starting point within a range(exact details depends on the task as will be explained later) in the input space and generate data from the point for a number of timesteps. For testing data, we will do the same both in the same range and outside the range, we will pick a few starting points and then average the result. Without loss of generality, we choose $m = k = 1$ and $g = \ell = 1$ for 1D and 2D SHM and simple pendulum for simplicity. For the damping, we will choose $\gamma = 0.01, 0.05, 0.1$. For double pendulum, we also let $m_1 = m_2 = \ell_1 = \ell_2 = g = 1$.

5.2 Evaluation Method

We will use a few different evaluation metrics.

1. Generate future trajectories from a random starting point and compare to true trajectories
2. Evaluate the difference between the true analytical dynamics and the predicted dynamics on a grid
3. Log marginal likelihood

If we are learning the invariance, we can also further compare the true invariance and the learnt invariance.

5.3 Implementation Technicalities

We need to add jitter to the D matrix in computation due to the fact that we need the invert it and because of numerical issues, it will be more stable if we add a little bit of jitter in the range of 10^{-6} and its effect will be explained in the sections below. Similarly, to stabilise the algorithm, when backpropagating the hyperparameters, we need to narrow range of search so it's not too crazy. For example, we wouldn't expect a lengthscale of 100 in our example with maximum range of 5; therefore, we will set for example, the maximum range of search is 10^{-3} to 5 for the length scale and similarly for the variance term.

5.4 1D

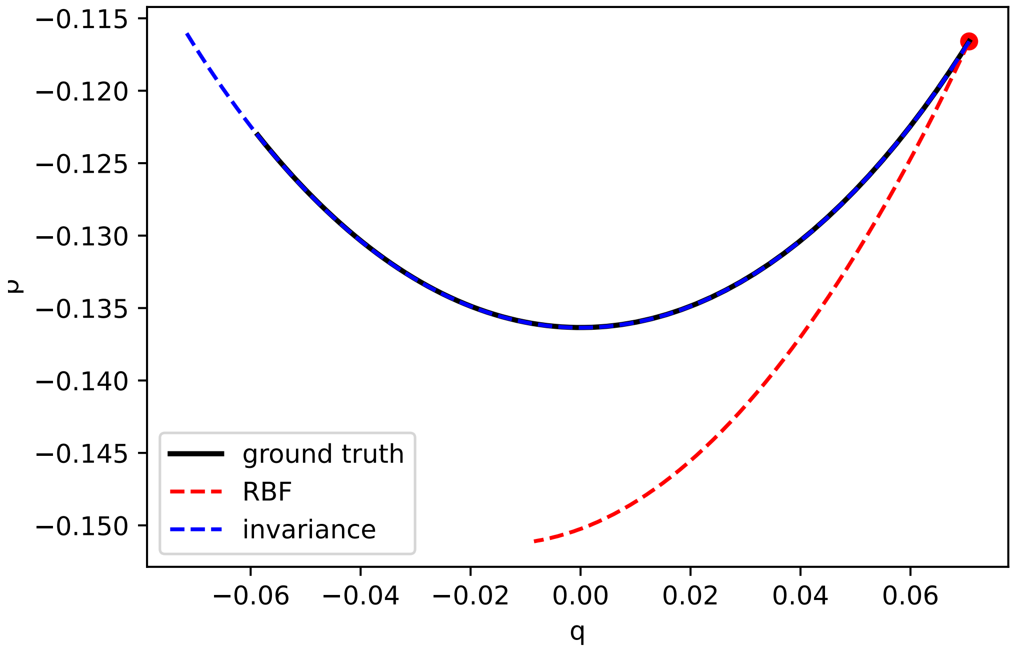
We will illustrate a lot of examples in 1D cases since the results are easy to visualise, but the principle readily extend to high dimensional spaces.

5.4.1 SHM

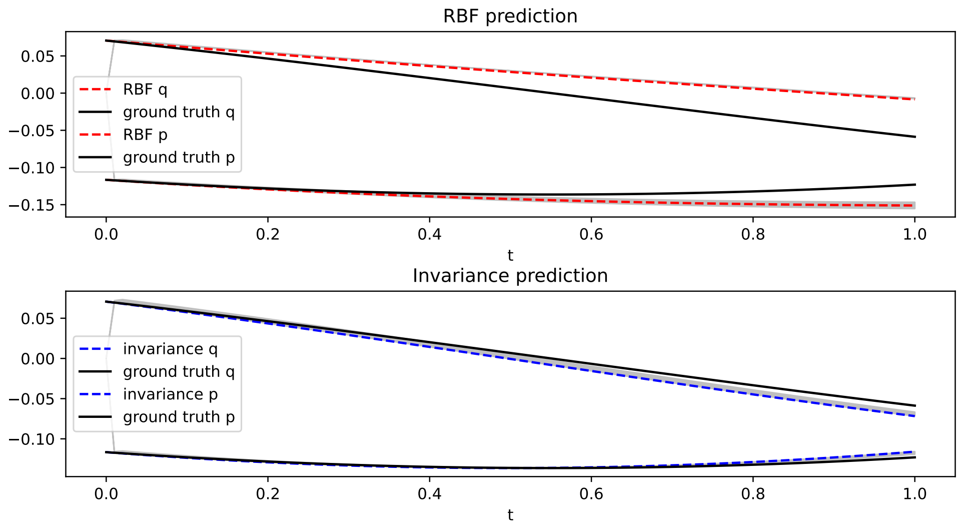
We start with the 1D SHM case, we will have only one trajectory starting randomly $-3 \leq q \leq 3$ and p to be between a fifth of that so it does not overshoot the boundary, and we allow it to run for 0.1 seconds so there will be in total 10 training points. We will then draw five test point from q between ± 3 and another five between 3 and 4, to test the performance of the future predictive power for 1 second and average over. We will choose one at random and draw the future trajectories. We will then evaluate the the recovery of ground truth dynamics $a = -q$ and $v = p$ between $q, p = \pm 4$ of 40 by 40. From figure below, we can see the prior of the naive baseline vs invariance kernel. This is using invariance points density of evenly spaced 40 by 40 grids. We can see the prior kernel as well as posterior shape as shown in figure b. The performance is summarised in table with jitter 1×10^{-5} with invariance density of 40 within ± 5.

Method	log marignal likelihood	prediction MSE	true dynamics MSE
RBF	78.03	0.4785 5.6039	
Invariance	87.49	0.0119	2.2120

Table 5.1: SHM Invariance Kernel Performance

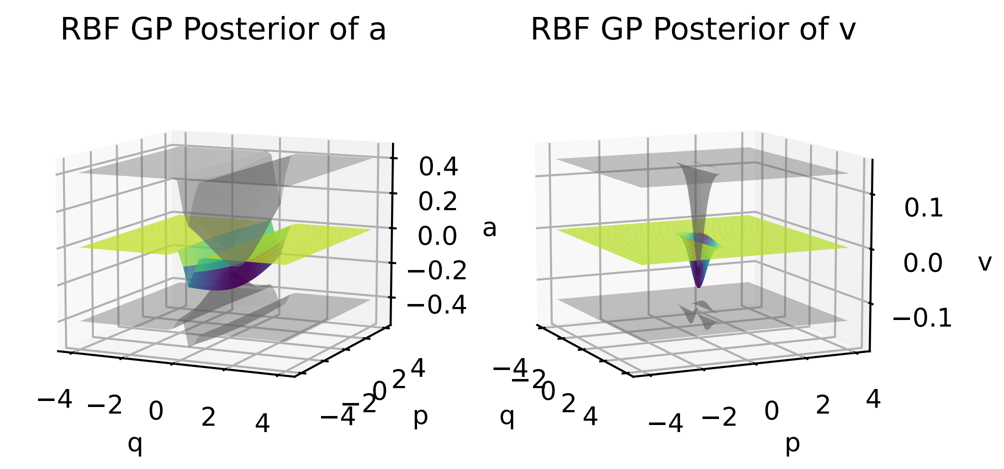


(a) SHM Predicted trajectory

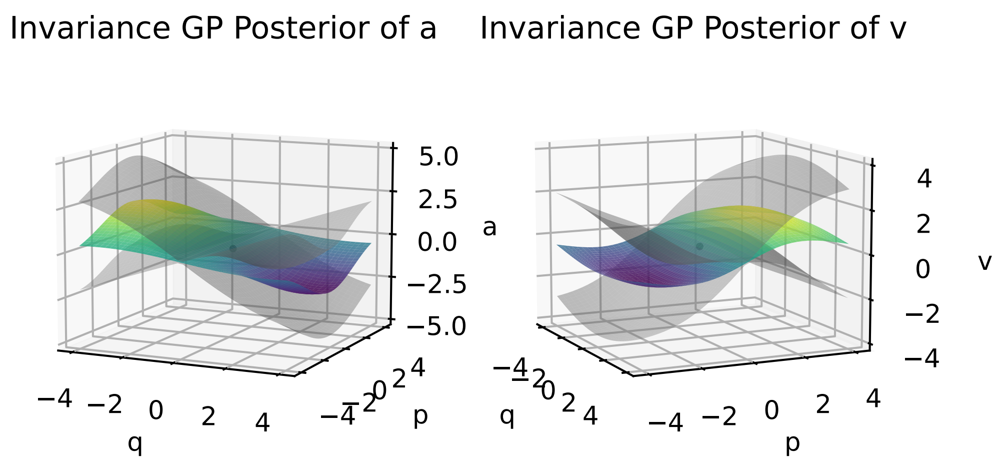


(b) Individual dynamics SHM Predicted trajectory

Figure 5.1: SHM predictions



(a) SHM RBF posterior



(b) SHM Invariance posterior

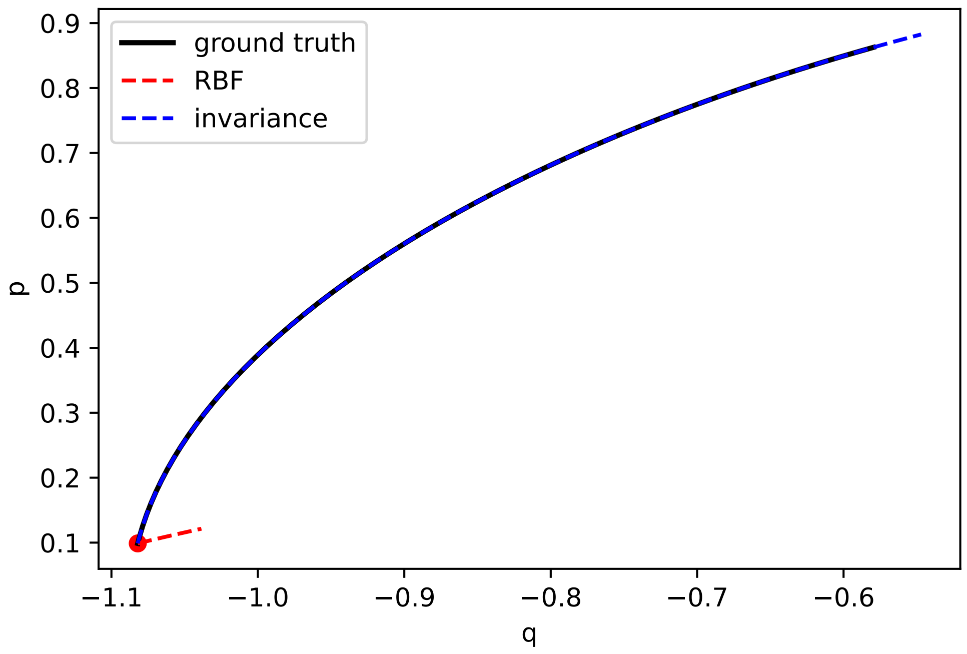
Figure 5.2: SHM predictions

5.4.2 Pendulum

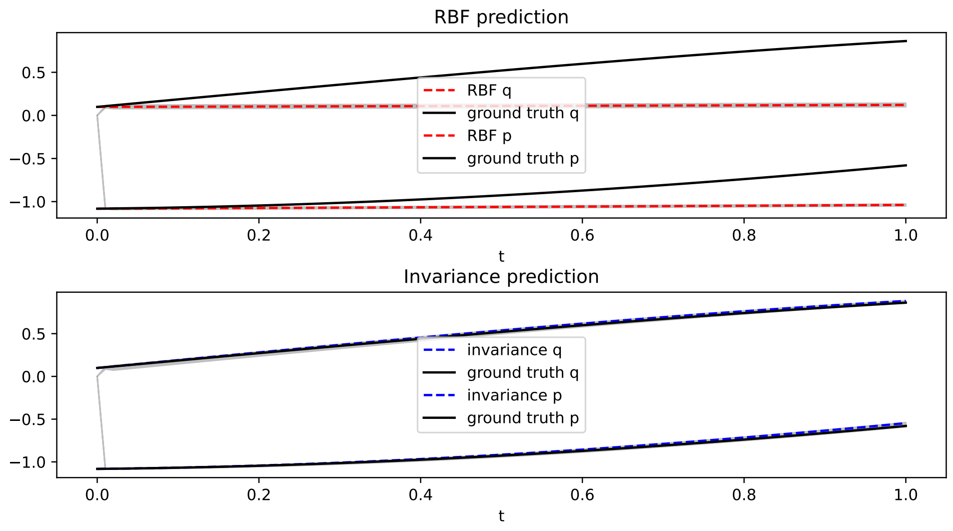
The procedure is pretty much the same here that we will draw a single training example between $q = \pm 120°$ and again one fifth the range for p. The rest is the same and we will do a test from 120° to 150° too. Again, the invariance is conditioned from ±175° with invariance density of 40 with jitter 10^{-5}. Evalute restoring dynamics on a ±150° too. Jitter is 10^{-5}

	log marignal liklihood	prediction MSE	true dynamics MSE
RBF	72.60	0.0803	1.3531
Invariance	85.38	0.0003	0.0373

Table 5.2: Pendulum

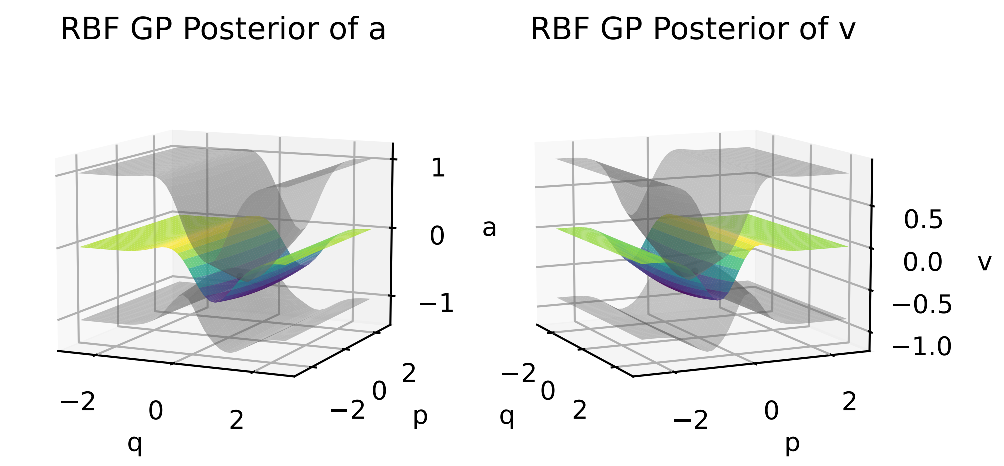


(a) Pendulum Predicted trajectory

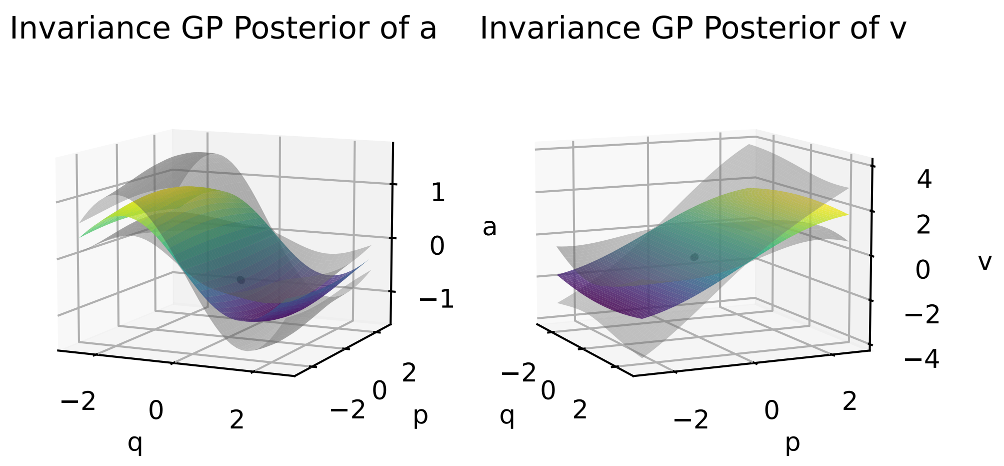


(b) Individual dynamics Pendulum Predicted trajectory

Figure 5.3: Pendulum predictions

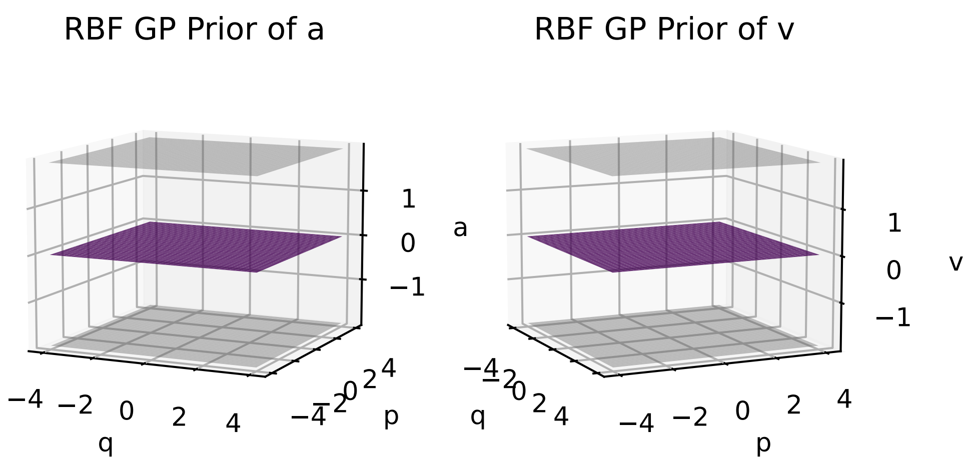


(a) Pendulum RBF Posterior

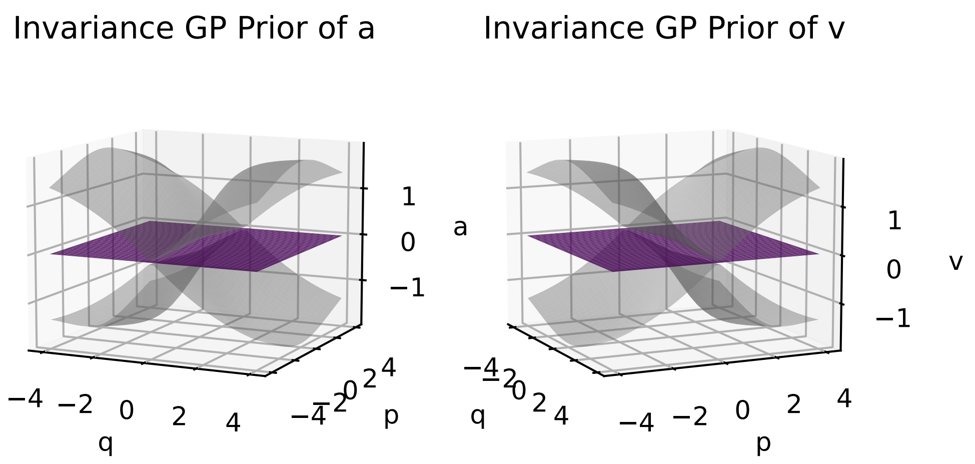


(b) Pendulum Invariance Posterior

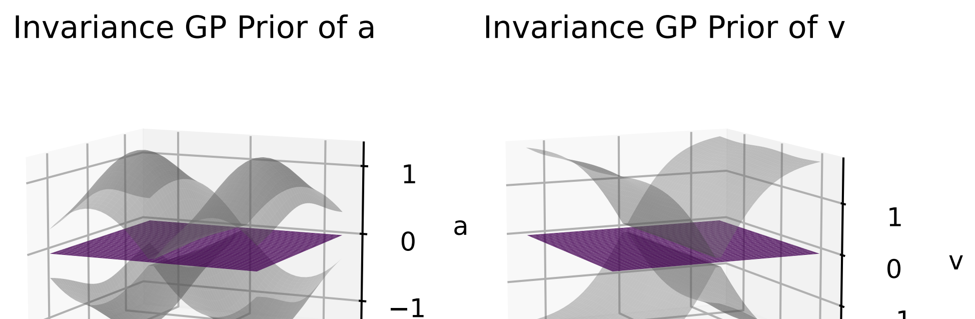
Figure 5.4: pendulum predictions



(a) SHM Predicted trajectory



(b) Individual dynamics SHM Predicted trajectory



5.4.3 Damped SHM

The results is the same, and the learnt ϵ is summarised in table. Learnt epsilon 1.004×10^{-5}.

	log marignal likelihood	prediction MSE	true dynamics MSE
Baseline RBF	84.25	0.5499	5.7187
Invariance Kernel	89.51	0.0248	3.1687

Table 5.3: Damped SHM performance

5.4.4 Damped Pendulum

The results is the same, and the learnt ϵ is summarised in table. With setting exactly the same as the pendulum case and $\gamma = 0.1$ Learnt epsilon is $8x10^{-4}$

	log marignal liklihood	prediction MSE	true dynamics MSE
Baseline RBF	71.44	0.3531	1.8513
Invariance Kernel	77.72	0.0161	0.6379

Table 5.4: Damped Pendulum Performance

5.5 Invariance Density

Here we will to explore how the change of invariance density could affect the performance of the kernel as well as degree of freedom. Since 1D SHM is rather trival, I will test it on the nonlinear pendulum case. Between -160 and 160, fix jitter at 1e-5 recover dynamics between 160.

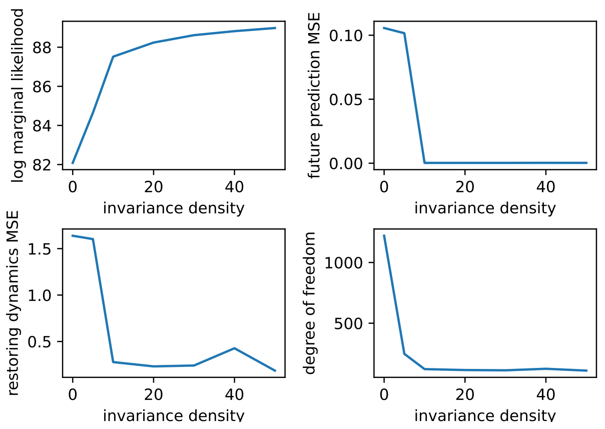


Figure 5.6: Effect of varying invariance density

polynomial degree	(1, 1)	(1, 2)	(1, 3)	(1, 4)	(2, 1)	(2, 2)	(2, 3)	(2, 4)
log marginal likelihood	140.61	140.83	139.45	140.99	-34.20	156.16	150.16	-58.81
polynomial degree	(3, 1)	(3, 2)	(3, 3)	(3, 4)	(4, 1)	(4, 2)	(4, 3)	(4, 4)
log marginal likelihood	136.18	-34.33	149.97	-34.29	140.84	nan	-34.17	-34.12

Table 5.5: Log marginal likelihood for polynomial of different degree

5.6 Effect of Jitter

Since we need a little bit of jitter to stabilise the computation of our invariance kernel, since adding jitter is the same as adding noise to the invariance (since we are adding it on the D submatrix) so that we need to make sure it is not too much. So we will again assess the performance

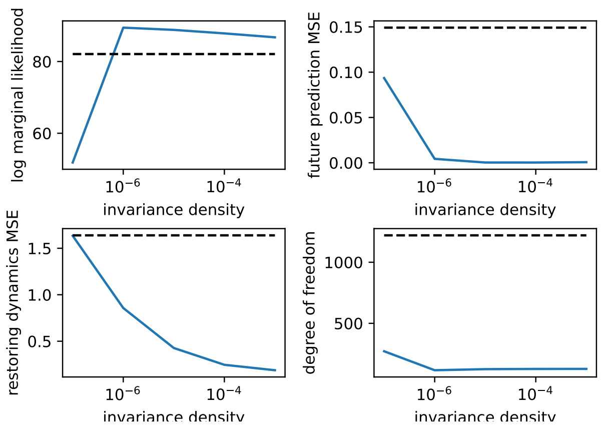


Figure 5.7: Effect of varying jitter

5.7 2D

5.8 Learning 1D invariance

5.8.1 SHM

We will test all combination of polynomial of degree 0, 1, 2, 3 so there will be 16 of them, and we will choose the ones with highest marginal likelihood. We put a prior of Laplace distribution of all the coefficients with variance 0.01 to encourage sparsity and we initialise the coefficients with normal distribution with variance of 10^{-3}, we also restrict the search between ± 1. This is done by 20 training points (2 random start points) since we need more data to learn parameters. We have jitter 5×10^{-6} and invariance density of 20. Data is again between ± 3 so is the invariance conditioning. The baseline marginal likelihood is 141.16 and known invariance is 168.44.

The best coefficients are$-5.687\times10^{-9}+-1.7864\times10^{-2}x$ $-4.5534\times10^{-9}-1.7885\times10^{-2}x$ We will evaluate the peformance again choosing from ±3 and restoring the dynamics on a grid of 40 by 40 within ±3.

	log marginal liklihood	prediction MSE	true dynamics MSE
Baseline RBF	141.16	0.1581	0.8303
Invariance Kernel	168.44	0.0052	0.0417
Parameterised Kernel	155.91	0.0303	0.3750

Table 5.6: Learnt SHM Invariance performance

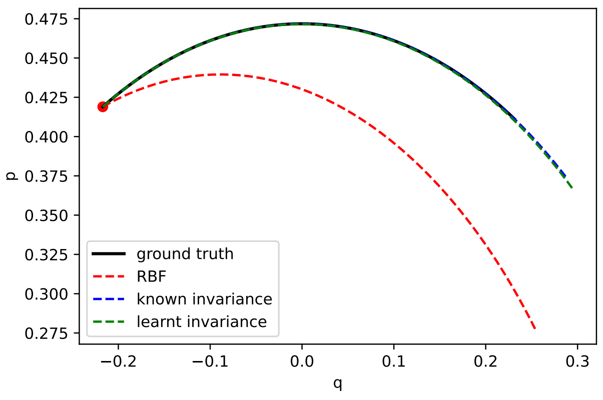


Figure 5.8: Comparison of learnt SHM invariance trajectory prediction

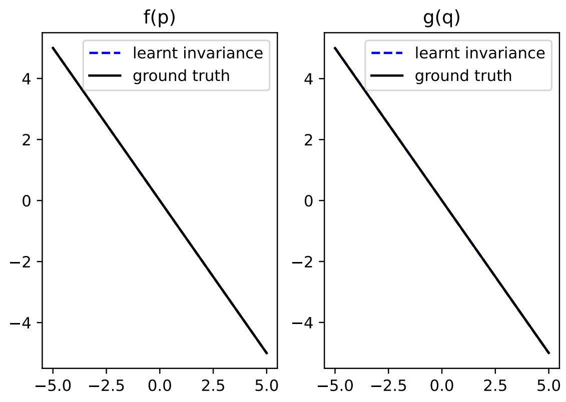


Figure 5.9: Learnt SHM polynomial

5.8.2 Pendulum

We will test all combination of polynomial of degree 0, 1, 2, 3, 4 so there will be 25 of them, and we will choose the ones with highest marginal likelihood. We put a prior of Laplace distribution of all the coefficients with variance 0.1 to encourage sparsity and we initialise the coefficients with normal distribution with variance of 10^{-3}, we also restrict the search between ± 1. Since this time, we canot force such strong prior on the coefficients to be zero. This is done by 30 training points (3 random start points) since we need more data to learn parameters. We have jitter 5×10^{-6} and invariance density of 40. Data is again between $\pm 150°$ so is the invariance conditioning. The baseline marginal likelihood is 234.99 and known invariance is 265.41.

polynomial degree	(1, 1)	(1, 2)	(1, 3)	(1, 4)	(2, 1)	(2, 2)	(2, 3)	(2, 4)
log marginal likelihood	234.11	203.26	203.05	199.71	234.03	243.55	249.73	nan
polynomial degree	(3, 1)	(3, 2)	(3, 3)	(3, 4)	(4, 1)	(4, 2)	(4, 3)	(4, 4)
log marginal likelihood	220.67	206.17	250.61	256.79	236.49	225.86	253.86	nan

Table 5.7: Log marginal likelihood for polynomial of different degree for pendulum

The best coefficients are $1.698 \times 10^{-3} + x - 1.77 \times 10^{-2}x^2$ and $-1.073 \times 10^{-3} + x + 1.331 \times 10^{-3}x^2 - 0.1488x^3$

We will evaluate the peformance again choosing from ± 3 and restoring the dynamics on a grid of 40 by 40 within ± 3.

	log marignal liklihood	prediction MSE	true dynamics MSE
Baseline RBF	234.99	0.0047	0.2398
Invariance Kernel	265.41	0.0008	0.0052
Parameterised Kernel	257.59	0.0161	0.1385

Table 5.8: Learnt Pendulum Invariance performance

The learnt invariance function is as follows

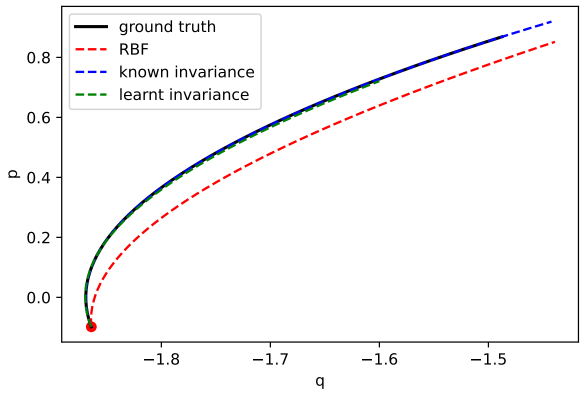


Figure 5.10: Comparison of learnt pendulum invariance trajectory prediction

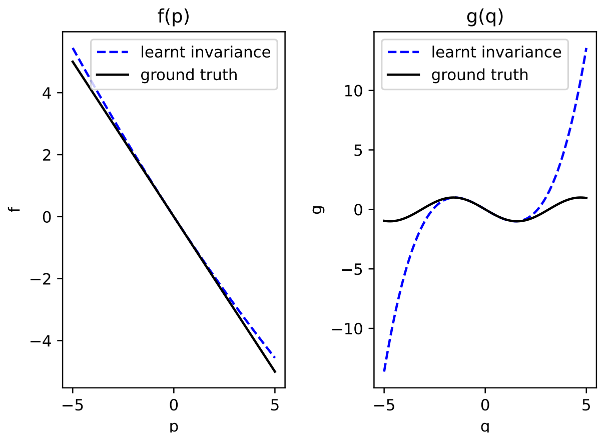


Figure 5.11: Learnt pendulum polynomial

5.9 Learning 2D invariance

6 Conclusion

Conclusion goes here.

References

Kathleen T. Alligood, Tim Sauer, and James A. Yorke. *Chaos: An introduction to dynamical systems*. Springer, 2000.

Alexander G. de G. Matthews, Mark van der Wilk, Tom Nickson, Keisuke. Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman. GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 18(40):1–6, apr 2017. URL `http://jmlr.org/papers/v18/16-537.html`.

Carl Edward Rasmussen and Christopher K I Williams. *Gaussian process for machine learning*. The MIT Press, 2006.