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

August 9, 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: August 9, 2022

Abstract

ABSTRACT GOES HERE

Acknowledgements

I would like to thank Dr Andrew Duncan and Dr Mark van der Wilk for being amazing supervisors, who provide guidance that I cannot complete the project without, and making time even though they are incredbily busy. I also like to thank the Statistics Department for providing the computing resources to make this project happen. Lastly, I would like the thank my family for their support.

1 Introduction

Dynamical system is one of the richest theory that models the real world with wide application across all science and engineering and therefore it is very valuable to be able to reliably predict the evolution of a system for practical purposes. Recently, data intensive machine learning methods, such as deep learning, has shown some very powerful results. However, the amount of data required to achieve reasonable performance are often enormous, will often involve lots of manual labour to label and process the data. Sometimes, the data could be very difficult to come by in biological and medical settings. Therefore, data efficiency is the key if we wish to study more complicated systems. Inductive bias, an intitial choice of hypothesis space (Baxter (2000)), is one way to achieve this goal. What is usually done is to embed the inductive bias into the model in some way so that the model "understands" this knowledge before seeing the data. Assuming the inductive bias is correct, we may expect the model to perform better, either in a data efficiency way or prediction performance way. One very powerful inductive bias is the use of symmetry and invariances, that puts strong constraints on the model to obey, which is the basis of many powerful machine learning models, such as Convolution Neural Network, utilising the translational symmetry in natural images. Marginal likelihood indicates how well the data is explained by the model as was shown to be beneficial in parameter tuning in Bayesian model and in particular in identifying the correct invariance. Wilk et al. and van der Ouderaa and van der Wilk (2021) has demonstrated using marginal likelihood as a objective is useful in learning the invariance, in particular invariance under Affine transformation. Gaussian Process (GP) was used thanks to its ability to estimate marginal likelihood accurately, and the inductive bias was embedded in the kernel function of the GP. In this report, we aim to extend on the work to learn invariance in dynamical systems, in particular the conservation of energy.

In the rest of the report, we will cover the essential theortical background to understand GP and dynamical system in section 2. We will also describe related work in the area of GP in dynamical system, physics informed machine learning as well as invariance and symmetry in machine learning in this section. In section 3, we will derive the theoretical apsects and the constructions of the invariance kernels in dynamical system. Following that, we illustrate the implemented results on various dynamical systems in section 4 along with discussions. Finally, we conclude in section 5.

2 Background

Here we will cover the background knowledge required to understand the remaining thesis, including Gaussian Process, dynamical systems as well as symmetry and invariances. This section will also cover related work in the field.

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

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

In a noise free scenario, where we observe f directly, we will have

$$\begin{bmatrix} f \\ f^* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X^*) \\ K(X^*,X) & K(X^*,X^*) \end{bmatrix}\right),$$

where we have X, f being the training data and target, with X^*, f^* the testing points and targets. If we assume the signal is not noise free such that we assume a Gaussian noise $y = f + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$. We will instead have

$$\begin{bmatrix} y \\ f^* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) + \sigma_n^2\mathbb{I} & K(X,X^*) \\ K(X^*,X) & K(X^*,X^*) \end{bmatrix}\right),$$

An important identity we use throughout the thesis is the Gaussian conditional formula; if

$$\begin{bmatrix} x \\ y \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} A & C \\ C^T & B \end{bmatrix}\right),$$

then

$$x|y \sim \mathcal{N}\left(\mu_x + CB^{-1}(y - \mu_y), A - CB^{-1}C^T\right) \tag{2.1}$$

We can therefore predict our new points with

$$f^*|X, y, X* \sim \mathcal{N}\left(\overline{f^*}, \text{cov}(f^*)\right)$$
$$\overline{f^*} = K(X^*, X)[K(X, X) + \sigma_n^2 \mathbb{I}]^{-1} y$$
$$\text{cov}(f^*) = K(X^*, X^*) - K(X^*, X)[K(X, X) + \sigma_n^2 \mathbb{I}]^{-1} K(X, X^*)$$

Below we demonstrated an example adapted from Matthews et al. (2017) to shown case the fitting procedure of a GP. Different kernels defined by their various functional form imposes different prior knowledge on the GP prior. For instance, one of the most commonly used kernel, RBF, requires smoothness; Matern requires once differentiability; Periodic kernel requires the prior to have periodic structure obviously. Below in figure 2.1 we showed 5 samples drawn from each of the kernel.

$$k_{RBF}(r) = \exp\left(-\frac{r^2}{2l^2}\right)$$
$$k_{\text{Matern}}(r) = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}r}{\ell}\right)^{\nu} K_\nu\left(\frac{\sqrt{2\nu}r}{\ell}\right)$$
$$k_{periodicRBF} = \exp\left(-\frac{2\sin^2\left(\frac{r}{2}\right)}{\ell^2}\right)$$

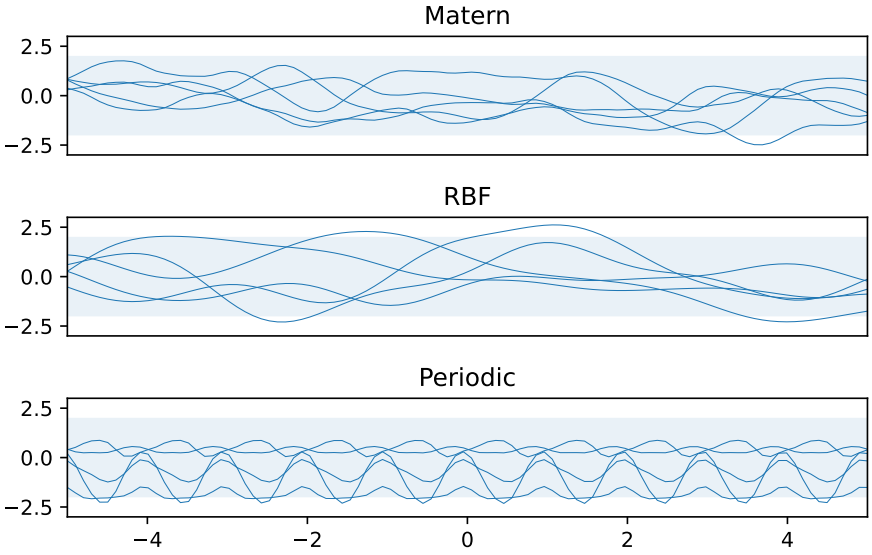


Figure 2.1: Samples from different GP priors of RBF, Matern and Periodic kernel

Now we intend to fit a GP to a function, arbitrarily defined for demonstration purpose, $(x + x^2)\sin(x)$. We randomly sampled five points and the fit is as shown below in figure 2.2. We also sampled 10 posteriors.

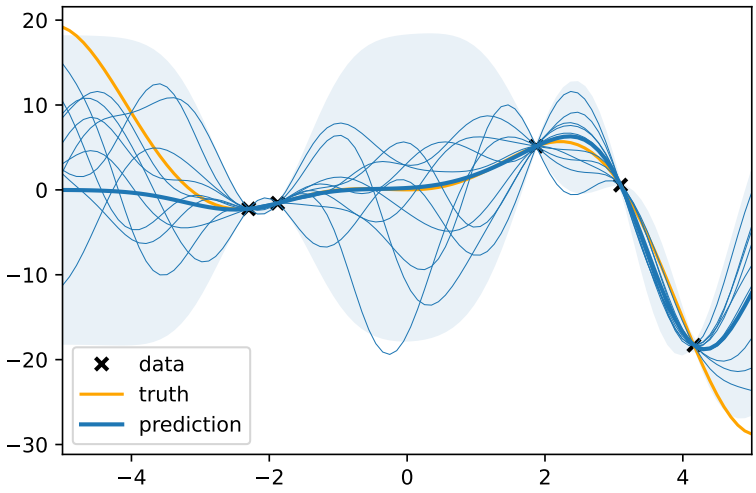


Figure 2.2: An example of fitting a GP to a function

We can see the function fit well around the data points and fall back to the GP prior (RBF in this case) away from the data.

2.1.1 GPflow

Now we can in principle build our own GP objects using basic linear algebra and statistical libraries from scratch. It is a much better idea to avoid errors and improve development speed by using a well-tested, robust and efficient library called GPflow, which 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 (using automatic differentiation feature of Tensorflow) and perform optimisation. In later sections, we will need to optimise more parameters than aforementioned ones with respect to maginal likelihood objective.

2.2 Dynamical Systems

Dynamical system, which describes the evolution of a system in time, can be broadly classified to two catergories: differential equations and difference equations, with the former being continous time and the latter being discrete time. (Strogatz (2019)). We will formalise our notion and notation using continous time (differential equation) description. There are many real world systems that can be modelled by a dynamical system, for example a huge class of problems in engineering and physics. A simple example,

which will also be explored later, is a damped pendulum, with equation of motion

$$m\frac{d^2x}{dt^2} + b\frac{dx}{dt} + kx = 0.$$

In general an n^{th} degree differential equations can be written as

$$\frac{d^nx}{dt^n} = F\left(t, x, \frac{dx}{dt}, \ldots, \frac{d^{n-1}x}{dt^{n-1}}\right),$$

(Glendinning (1994)) and we will need n initial conditions to specify a solution for the differential equations. We can also write n^{th} degree ordinary differential equations (ODEs) as n degree one coupled ODE. For example, in the above case, we can write it as

$$\begin{cases} \frac{dx}{dt} = v \\ \frac{dv}{dt} = \frac{-bv-kx}{m} \end{cases}$$

More generally, we can write

$$\begin{cases} \dot{x}_1 = f_1(x_1, \ldots, x_n) \\ \vdots \\ \dot{x}_n = f_n(x_1, \ldots, x_n) \end{cases}$$

We can then visualise the state of the system described by the n variables in the "phase space", and this system will evolve or move in the phase space as time evolves. For instance, the damped pendulum is of degree 2 so there will be two variables describing its evolution, which in this case will be x and $\frac{dx}{dt} = \dot{x}$, which is the angular displacement as well as angular velocity. Another example would be a naive model describing exponential growth of a organism, such as some sort of bacteria population in a petri dish

$$\dot{x} = rx,$$

and this time it is one-dimensional with the phase variable being the population number. Another important concept is linearity and nonlinearity.

2.3 Symmetry and 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. There are other types of symmetry, such as translational symmetry, rotational symmetry, permutation symmetry, which are widely explored in geometric deep learning literature. More formally, we can describe symmetry using group theory. In physics, there is a deeper connection between symmetry and invariance, namely the Neother's Theorem. In plain words, every conservation law is associated

with an underlying symmetry. For instance, conservation of momentum is obeyed when there is a translation symmetry of the system. Similarly, rotational symmetry is associated with the conservation of angular momentum. There are more abstract conservation law, such as conservation of electrical charge is related to Gauge Symmetry, an advanced physics idea that we will not go into too much detail. Here the more relevant ones is the conservation of energy, which is related to the symmetry of the system under temporal translation.

2.4 Related Work

There are many existing literature in the field of learning invariance as well as physics informed machine learning.

2.4.1 Physics Informed Machine Learning

There are many physics informed machine learning techniques as outlined in Cuomo et al. (2022); Karniadakis et al. For example, in Physics Informed Neural Network (PINN) from Raissi et al. (2019), the authors embedded the form of partial differential equations (PDEs) that describe the physics of the problem into the loss function so that the neural network model understands the PDE; they also allow the parameterscoefficients in the PDEs to be learnt. There are many variants of such as Qian et al. (2020). In Raissi and Karniadakis (2018), it even uses a GP to model the evolution of the state by using the PDE form as a linear operator (we will see similar idea in our work) to evolve the GP; since the linear transform of GP is still a GP, we will get joint GP. However, they are still restricted to known PDE form and only learning the parameters. Furthermore, they all does not use the knowledge of conservation laws to their advantage. Moreover, for nonlinear dynamics, it has to use a linear approximation for the method to work. There are also versions of PINN that takes conservation of energy into account such as Jagtap et al. (2020). However, it still suffers from other problems described above.

Energy Conserving Neural Networks

There are other approaches like Hamiltonian Neural Network Brain et al., which enforces the law of Hamiltonian mechanics on the neural network. As a results, HNN automatically conserves the Hamiltonian or almost equivalently, the energy. There are also variants origin from this, such as the more general Lagrangian Neural Network, Cranmer et al. (2020), or other energy conserving nerual networks described in Zhong et al. (2021). However, due to their strong invariance enforcing nature, they often do not do very well with dissipative systems; Sosanya and Greydanus models the dissipative part explicitly resolves this problem. Nevertheless, these systems can only work on a very specific types of dynamical system, namely the ones that follow Sympletic form, or more simply obeys the Hamiltonian mechanics, which is only a small subset of all dynamical systems. There are a lot more general dynamical systems that has some sort of invariance property that is not able to utilise these energy conserving neural networks.

ODE approach

Or we can directly works on the ODE that describes the physics of the system. In some sense, this is similar to the PINN. Neural ODE Chen et al. is a great example of this; however, again it does not use any knowledge of the invariances and symmetries. Further, it cannot recover the law of physics.

Symbolic approach

There are many existing literatures to recover the law of physics from trajectory using various form of symbolic regression, such as Papastamatiou et al. (2022); Udrescu and Tegmark (2020) or some form of graphical representation Cranmer et al.; Mrowca et al.. In Liu and Tegmark (2021), it even able to deduce the conserved quantitiy (along with its symbolic form) from trajectory. Nevertheless, it does not include any predictive ability.

2.4.2 Symmetry and Invariance in Machine Learning

Learning Invariance using Marginal Likelihood

This paper forms the basis of this report. In Wilk et al., the author proposes learning the form of invariances of the problem by parameterising the symmetry and maximise them with respect to the marginal liklihood. The main problem studied in the paper is the MNIST dataset, and that the symmetry under consideration is all form of Affine transformation, such as rotation, shear, translation; and the invariance in this case is the label of the digit. It suggests embedding the invariance in the model (GP in this case) as an alternative to data augementation. The authors created a special kind of kernel for GP that respects the symmetry of the problem. The results are very good compare to the baseline RBF kernel, and it also is able to recover the true paramertised invariance. The logical flow and setup of this report roughly follows that of this paper as well as the HNN paper. A related paper that uses a neural network instead of GP is van der Ouderaa and van der Wilk (2021), which also optimise the marginal likelihood.

Geometric Deep Learning

Geometric deep learning proposed by Bronstein et al. (2017), is a unifying framework that aims to explain the success of several popular neural network architecture. For example, Convolution Neural Network uses the translation symmetry embedded in natural images. Another example is Graph Neural Network utilises the permutation symmetry that the graphs is invariant under permutation. Following this framework, multiple papers are proposed that aims to exploit further symmetry groups.

2.4.3 GP in dynamical systems

There are already a number of research work that uses GP to predict dynamical systems. One of them is the aforementioned Raissi and Karniadakis (2018). There is also Rath et al. (2020), that utilise the sympletic nature of Hamiltonian systems and directly

predict the flowmaps of the system; in some sense similar to HNN. There is also Heinonen et al. (2018), that also uses GP to model the ODE models in a nonparametric manner; but again it odes not uses invariance.

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

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

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 block diagonal

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

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 there are ℓ of them, 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 given in the following sections to make it clear. Since the invariance function will always be a linear function on the dynamics by the fact that it is the time derivative

$$\text{energy} = E(\theta, \mathbf{p}, \mathbf{q}); \frac{dE}{dt} = \sum_{i=1}^{d} \frac{\partial E}{\partial p_i} a_i + \sum_{i=1}^{d} \frac{\partial E}{\partial q_i} v_i$$

We can see it is always a linear combination of something independent of 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

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

We can then condition the first row on the second row using equation 2.1 to obtain

$$\mathbf{f}(X)|L[\mathbf{f}(X_L)] = 0 \sim \mathcal{N}\left(0_{2nd}, \left(K - LK(LKL^T)^{-1}KL^T\right)\right) \tag{3.1}$$

3.2 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.

3.2.1 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.

3.2.2 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.

3.3 Partially Observed System

Partially observed data can be very common in dynamical system. With examples of physical system, we can imagine not observing p and only observe q (of course we do not possess the knowledge that $\dot{q} = p$). If we have a 2D system, we might only observe 1 set of dimension due to data recording mechanism, or perhaps some situation where we project 3D objects on to the wall and we can only observe the 2D shadows. In a more general sense, we can imagine if we have a system with unquantifiable friction. We may not be able to write down the form of friction analytically due to complex interaction of the system with the enviroment (i.e. not simply a damped system as disccused above). Nevertheless, the law of physics and conservation of energy will still hold in the broad sense; in other words, if we take into account of every single force of the interactions (including the enviroment), the total energy will be conserved so the invariance holds. In practice, we can only observe the system so that the rest of the physics not taking into account is not observed, so it is again a partially observed problem. We can then set the unobserved part being a latent variable, z, and then use the invariance constraints to inference it.

4 Experiments

4.1 Data Generation

To generate the data, we use a Runge-Kutta integrator to generate data using the equation of motions for each problem. We also added some small Gaussian noise 10^{-8} to the dynamics to reflect the real world better as well as stablising the integrator. Since we aim to predict the value of dynamics at any point in the input space, we use the dynamics as targets , Y. This is done by combining forward and backward finite difference (midpoint method) to compute the gradient. We also choose the time step to be 0.01 to ensure sufficient accuracy for the integrator. For training data, X, we choose a few random starting point within a range (datails depends on the task as described later) in the input space and generate data from there for a number of timesteps (which is fixed at 10 points per trajectory in this report). To assess the performance, we generate the test set using the same input range, we will pick three different 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 of presentation. For the damping, we will choose $\gamma = 0.1$, which is fairly strongly damped as shown in figure 4.14b, 4.14a. For double pendulum, we also set $m_1 = m_2 = \ell_1 = \ell_2 = g = 1$.

4.2 Evaluation Method

We will mainly use the mean squared error (MSE) between the trajectory prediction and ground truth as our evaluation metrics. We also illustrate the amount of energy (using the exact expression) along the trajectory to demonstrate that invariance is obeyed. For learning the invariance in 1D case, we can also further compare the true invariance and the learnt invariance up to a multiplicative constant. However, it is not possible to visualise in 2D cases so we will simply assess the conservation of energy (invariance) performance.

4.3 Implementation Technicalities

We will need to add jitter, a diagnoal matrix of small value, to the LKL^T matrix in equation 3.1. This is due to the fact that we need the invert it and it will be more stable numerically if we add a little bit of jitter in the range of 10^{-6} to 10^{-4}, and its effect on the model will be explained in the sections below. Similarly, for stability, when backpropagating to optimise the hyperparameters with respect to marginal likelihood, we need to narrow range of search such that the value is more reasonable. It is particularly

important in our dynamical systems since, for example, in SHM the equation of motion is $a = -q$ and $v = p$, so that is will be a 2D plane extends to infinity; as a result, if the GP learns "correctly", the lengthscales of some directions will be very large or ∞, and that will be problematic and it would not affect the performance a lot if we limit the lengthscales for a much smaller value. Therefore, here I would set the range of both lengthscales and variable to be around the twice to three times the maximum range possible. For example, if we have maximum range of ± 3, my upper bound for search is set to 10 and lower bound is 10^{-3}. Another change we made in 2D SHM is to scale the data using MinMax scaler to squash them to be between ± 1. This is done since the polynomial behaves better around this region.

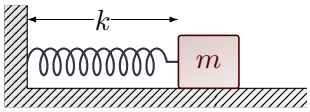
Invariance points X_L

In low dimensions, we could simply allow X_L to be a evenly spaced grid in the input space, and we will call the number of points in each dimension the **invariance density**. So that if I have an invariance density of 40, it means my X_L will contain 1600 points. However, conditioning on a grid of invariance points is simply not scalable in high dimensions. If we have an invariance density of 10 in a ± 5 range for 2D SHM (spacing of 1), then we will have 10000 points. We can 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 high dimensional space, we will use the idea of local invariance. The idea is straightforward, we simply sample uniformly (30-80 points) in the local region around the training and testing points and set them as X_L (we choose a torus shape around the data points so that we will not have invariance points too close to the data points, which will be unstable). Note that we only sampled once, so that the same local grid will be centred on every training and testing points; this is done to remove the randomness for stability and dependence on the data. This works on the assumption that only the region around testing points will be the most important to predict in terms of conditioning; and to use the same kernel, we also have to condition on the region around training points, which is also important to help maximising the marginal likelihood objective when we are learning the invariance function. Another evidence that supports the use of local invariance, which have much fewer points than using a grid, is demonstrated below where we show that there is a diminishing return in increasing the number of invariance points so that actually not that many invariance points is needed to achieve a good performance.

4.4 1D System

4.4.1 Linear SHM

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, an example trajectory is shown in figure 4.1.



The equation of motion of the system 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

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

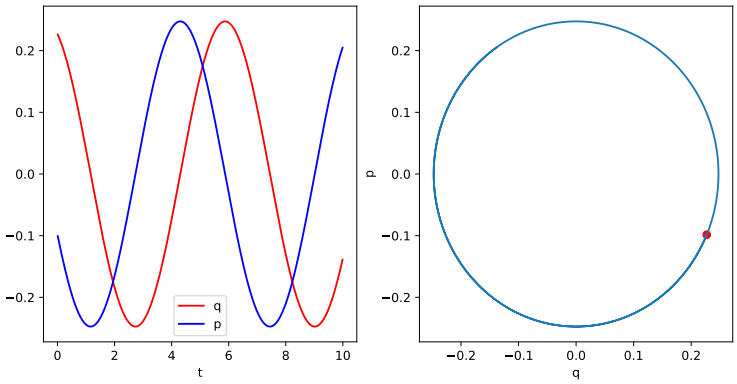


Figure 4.1: Example trajectory of SHM

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 a linear operator as shown below. If we have

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

then we have

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

which can be readily checked to be linear. Combine with original GP prior assumption, we 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 elementwise 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 matrix expressions are derived as follows by computing the covariance manually. For B, we wish to calculate

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

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

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

using the linear 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 using equation 2.1.

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

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

We start with the 1D SHM case, and for the data we will have only one trajectory starting randomly in $-3 \leq q \leq 3$ and p to be between ± 0.3. We chose p to be smaller so that the future trajectory does not overshoot the upper bound of q too much. We allow the integrator to run for 0.1 seconds so there will be in total 10 training points (10 time steps). We will then draw three test points from the same range as training, to assess the performance of the predictive power for 3 seconds (300 points) and average the MSE over the three trajectories. In figure 4.2, 4.16, we have chosen one trajectory at random to illustate the predictions as well as the energy along the trajectory. For this task, we have used a jitter amount of 10^{-5} and an invariance density of 20 between ± 3 in both p and q direction. We have also plotted the posterior distribution of using invariance kernel and compare that to using naive RBF kernel in figure 4.5. We can see the invariance has much better predictive power and generalisability. For learning the invariance, we test all combinations of polynomials of $f(p)$ and $g(q)$, each with degree 0, 1, 2, 3 (i.e. up to cubic term) so there will be 16 of them to optimise, and we choose the pair with the highest marginal likelihood. We initialise the coefficients with normal distribution with variance of 10^{-3}, we also restrict the search space to be between ± 1. We have also added a Laplace prior on the coefficients with variance 0.1 since we expect most coefficients would be zero and Laplace prior encourage sparse solution. The learnt $f(p)$ and $g(q)$ is plotted in figure 4.4. The performance is summarised in table 4.1.

Method	RBF	Known Invariance	Learnt Invariance
Log Marginal Likelihood	74.66	86.28	83.32
MSE	3.5563	0.1057	0.0927

Table 4.1: SHM performance

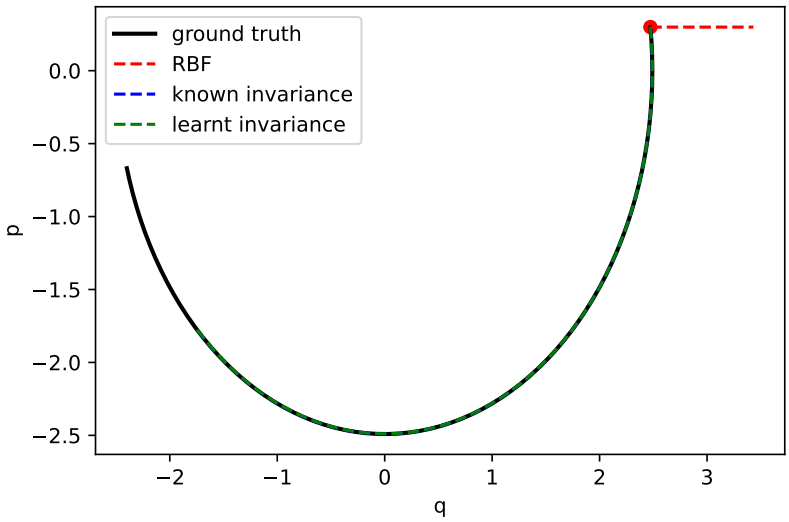


Figure 4.2: SHM predicted trajectory

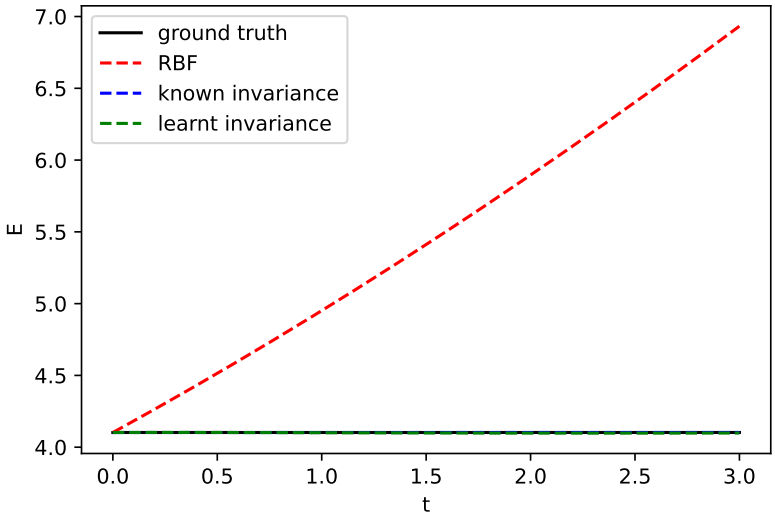


Figure 4.3: Energy conservation for SHM

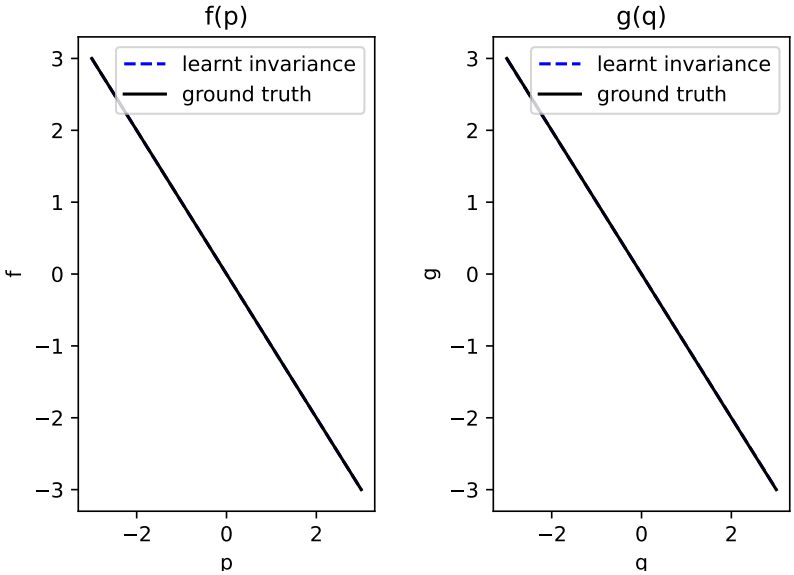
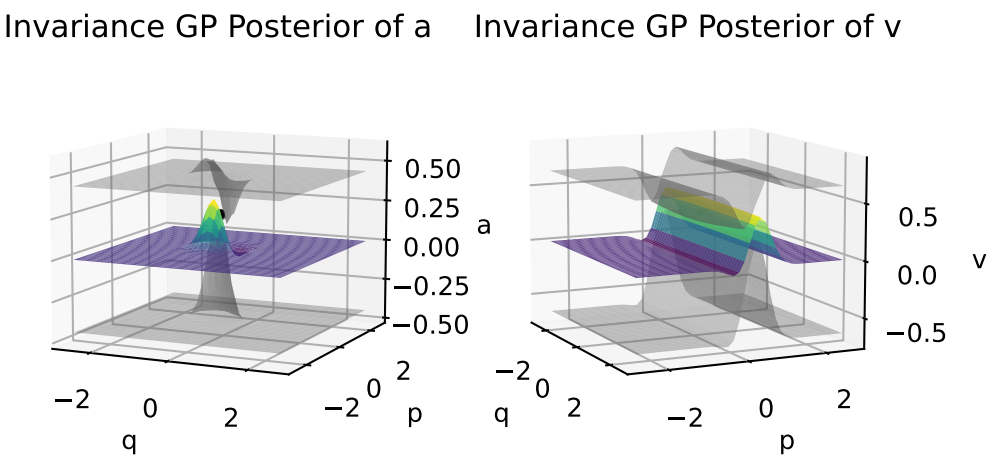
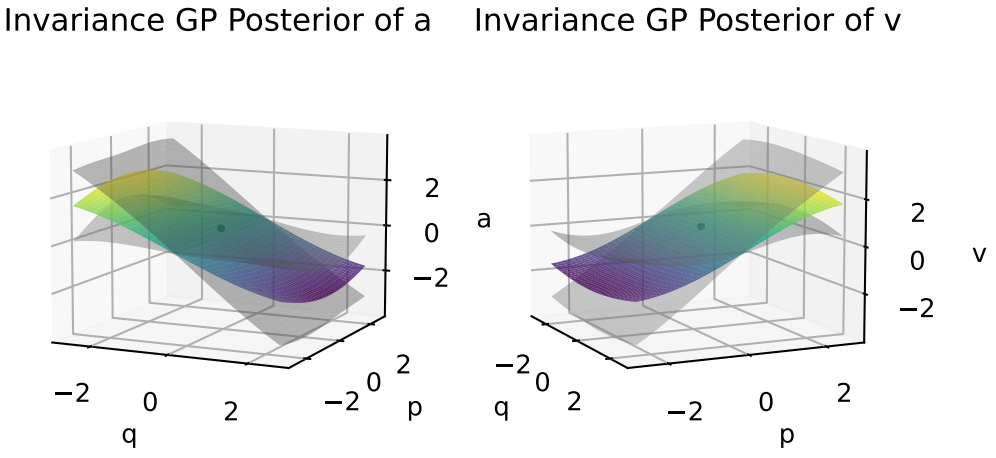


Figure 4.4: Learnt invariance for SHM



(a) SHM RBF posterior



(b) SHM invariance posterior

Figure 4.5: SHM posteriors

We can clearly see the naive RBF kernel failed to learn at all as it does not have enough example data to genealise, as we can see from 4.5a, so that it is only able to predict well in the region where the data are. On the other hand, from 4.5b and the prediction performance, we can see the invariance kernel has much better generalisation ability. From the energy plot in figure 4.3, we again see both the known and learnt invariance obeys the invariance very well while the RBF kernel does not at all. In this case, the learnt invariance performs almost exactly as the known form, as we can also see in figure 4.4, and has successfully recovered the physics.

4.4.2 Nonlinear Pendulum

The idea is pretty much the same for nonlinear system compared to linear; however, the fitting is 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 and we have example trajectory in 4.6, where we can see the effect of nonlinearity on the shape. The nonlinear dynamics arises from the sine term above, which will complicate the derivation for invariance kernel 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 in general, besides the small displacement case, which will be 1D SHM.

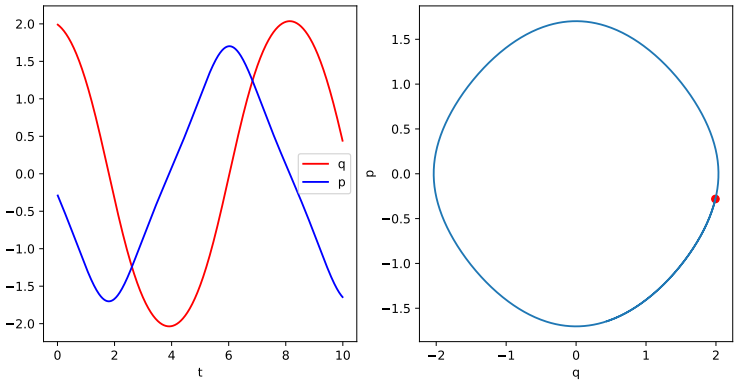


Figure 4.6: Example trajectory of pendulum

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

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

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,

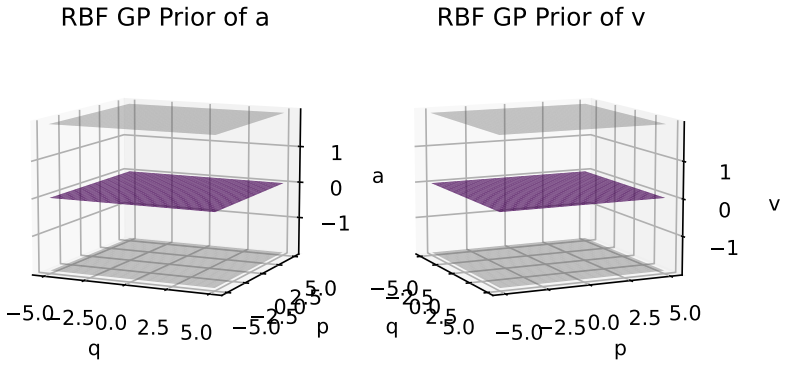
$$B = \begin{pmatrix} K_{RBF,a} \\ K_{RBF,v} \end{pmatrix} \odot \begin{pmatrix} \ell P_L \\ g \sin(Q_L) \end{pmatrix}, D = K_{RBF,a} \odot \ell^2 (p_L \otimes p_L) + K_{RBF,v} \odot g^2 (\sin(q_L) \otimes \sin(q_L)),$$

where

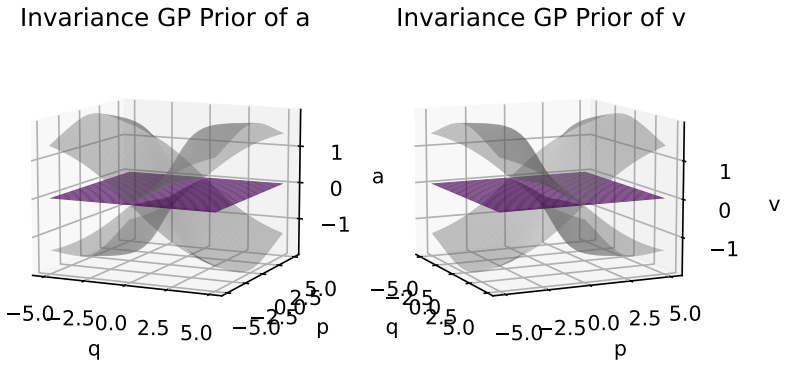
$$\sin(Q_L) = g \begin{pmatrix} \sin(q_{L,1}) & \dots & \sin(q_{L,\ell}) \\ \vdots & \text{reqeats n rows} & \vdots \\ \sin(q_{L,1}) & \dots & \sin(q_{L,\ell}) \end{pmatrix},$$

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

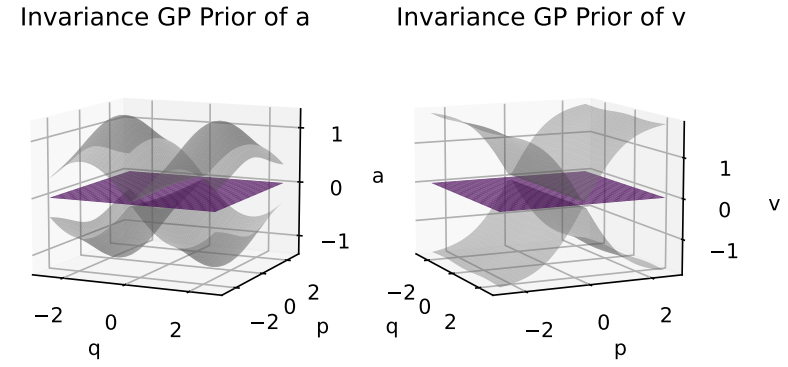
derived in exactly the same way as the linear case. We can look at the different priors in figure 4.7, which showcases the baseline kernel RBF, SHM invariance kernel, as well as the pendulum invariance kernel. We can see while the mean remains zero, the uncertainty parts reflects the form of invariance; such as the plane of SHM and the sinusoidal nature of pendulum invariance in 4.7c (left).



(a) RBF prior



(b) SHM invariance prior



(c) Pendulum invariance prior

Figure 4.7: Priors

In this task, the experiment setup is essentially exactly the same as the SHM case. However, since this time the dynamics is more complicated, we will draw three different starting points for training between $q = \pm 150°$ and $\pm 10°$ for p, so in total 30 points. This time, since it is a harder task, we will use an invariance density of 30 between $\pm 150°$. We again have the prediction performance summarised in table 4.2 as well as figures 4.8, 4.9 and the posteriors in 4.11. The learnt polynomial is in 4.10.

Method	RBF	Known Invariance	Learnt Invariance
Log Marginal Likelihood	233.89	262.59	255.76
MSE	0.0500	0.0025	0.0020

Table 4.2: Pendulum performance

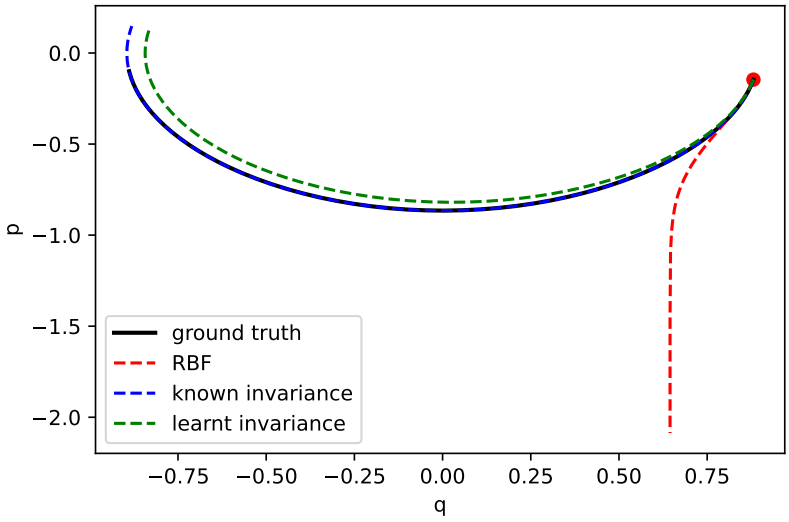


Figure 4.8: pendulum predicted trajectory

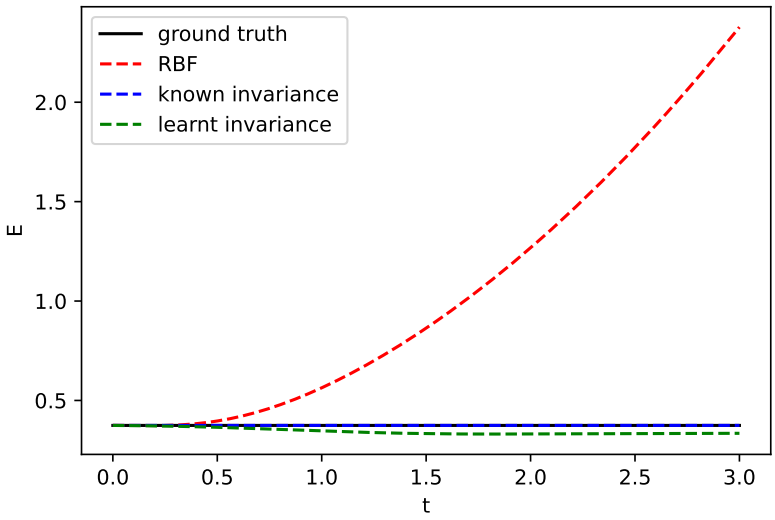


Figure 4.9: Energy conservation for pendulum

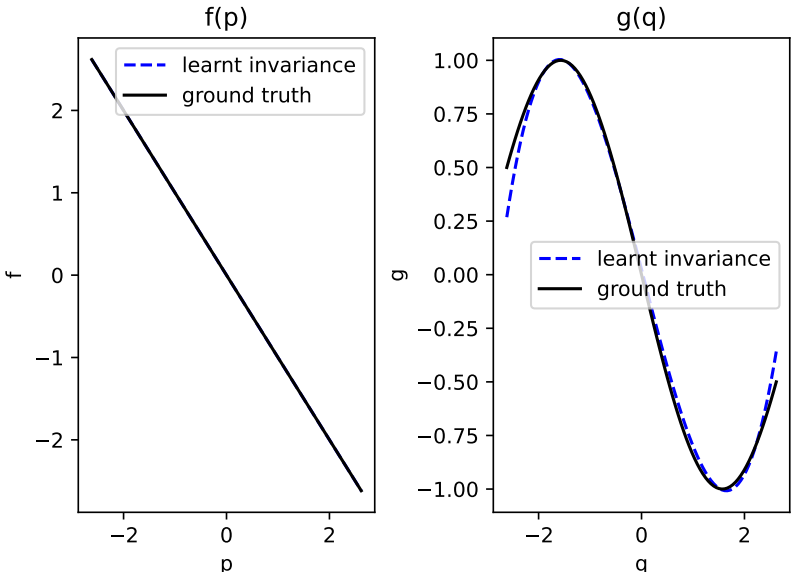
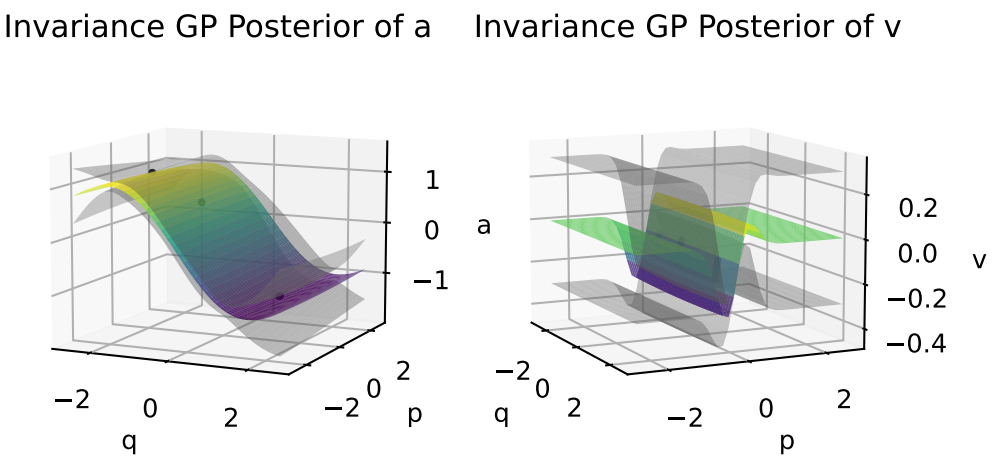
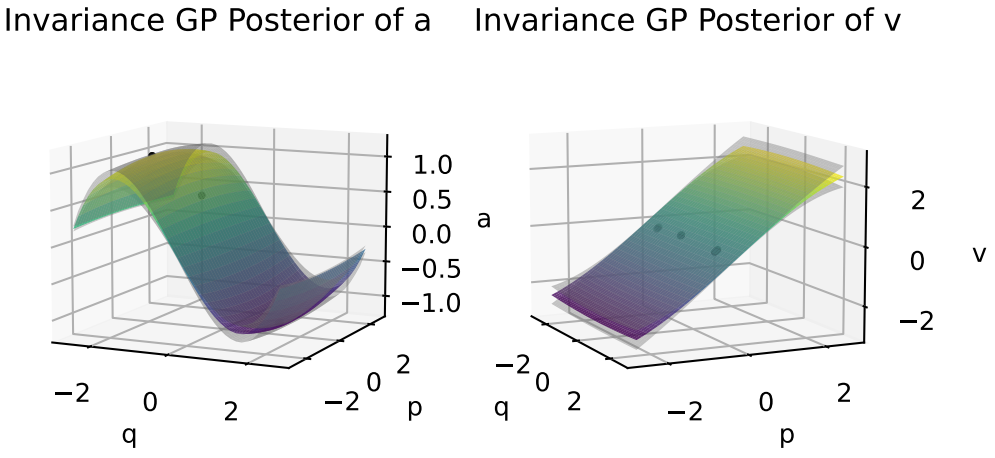


Figure 4.10: Learnt invariance for pendulum



(a) pendulum RBF posterior



(b) pendulum invariance posterior

Figure 4.11: pendulum posteriors

From table 4.2, figure 4.8, 4.9. This time, again the RBF kernel performs poorly. The known kernel performs very well again, almost exactly the same as the ground truth. However, this time, the learnt invariance is slightly worse; this is expected since we are approximating a sine function with a cubic polynomial, and so some level of deviation is expected. As we can see from figure 4.10, the learning is pretty accurate. From the posteriors in 4.11, we see RBF generalise much better due to extra data points, but it is still very far from giving a reasonable prediction.

4.5 Invariance Density

Here we will to explore how different invariance density affect the performance of the kernel as well as the degree of freedom. Since 1D SHM is trival to learn, we test it on the nonlinear pendulum case, which will illustrate the point better. We will use different invariance density using the same training data and three different trajectories of testing data. We again calculate the MSE of trajectory prediction as well as the percentage of energy deviation from the starting point of three different trajectories and average over. The results is shown in figure 4.12.

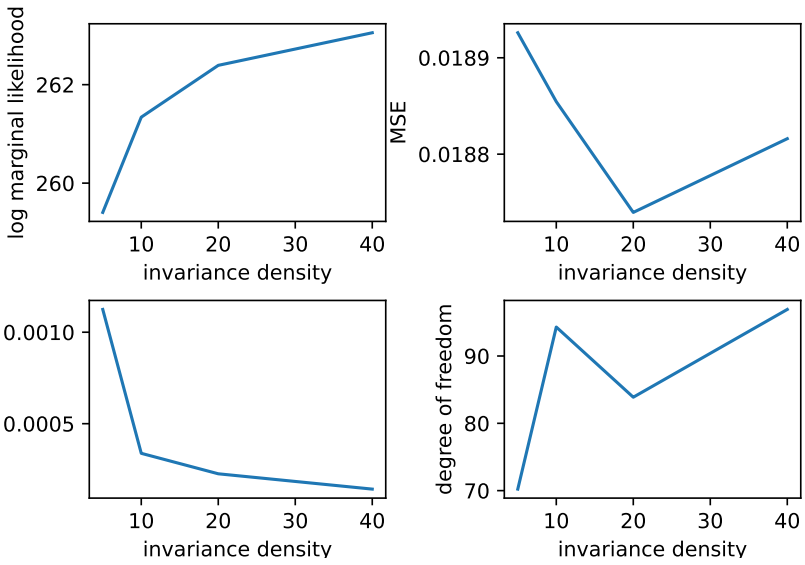


Figure 4.12: Effect of varying invariance density

We can see that for the marginal likelihood, it increases as the number of invariance points increases, which is expected since we have more information/evidence to support the data. However, MSE first decreases then increases. The first decrease makes sense since with more invariance points, we should in principle predict better. However, the increase is counter-intuitive. We conclude that there is actually not that big a difference and it is possibly due to the numerical instability as we increase the size of the matrix

with increasing invariance density. We also see the deviation of energy decreases as we increase the number of invariance points, which makes sense because we are imposing "more" invariance in the input space by increasing the invariance density. Lastly, we see the degree of freedom does not vary much either, while in theory it should decrease with increasing invariance density due to more constraints on the function.

Overall, we see there is not that big an advantage to increase our invariance density beyond a certain point due to diminishing return in performance and N^6 computational cost. Therefore, from this point we will choose the smallest invariance density that give reasonable performance.

4.6 Effect of Jitter

Since we need a little bit of jitter to stabilise the computation of our invariance kernel as explained above, and that adding jitter is the same as adding noise to the invariance (since we are adding it on the diagonal of LKL^T submatrix), we need to make sure it is not too large and too noisy, which will destroy the point of having invariance to start with. So we will again assess the performance in the similar way as above using different amount of jitter. The results is shown in figure 4.13.

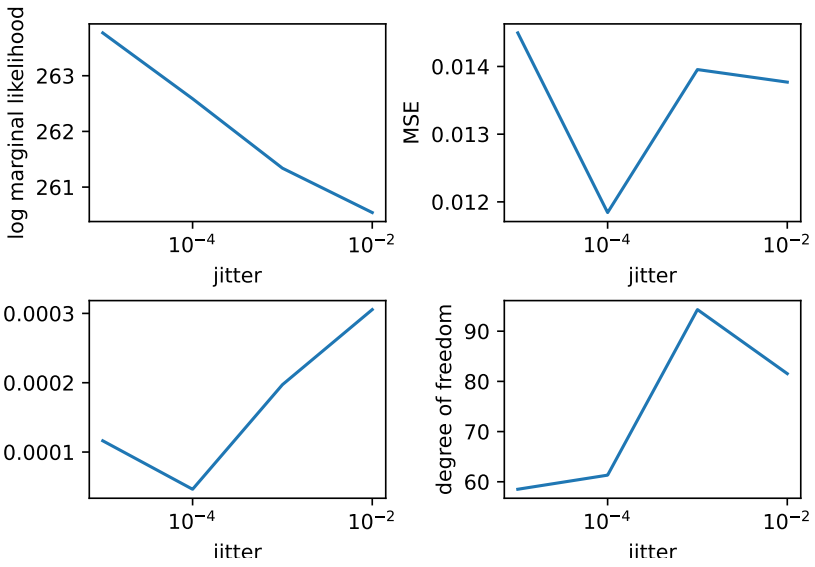


Figure 4.13: Effect of varying jitter

For marginal likelihood, we see it decreases as jitter increases as expected because then the invariance is not as strong a contraint. For MSE, we expect to see steady decrease in performance. However, here we see a dip initially. Again, there is not a big difference, and that the lowest jitter might not be enough to stablise the model so that the worse performance could be due to numerical instability. We see a similar pattern in energy

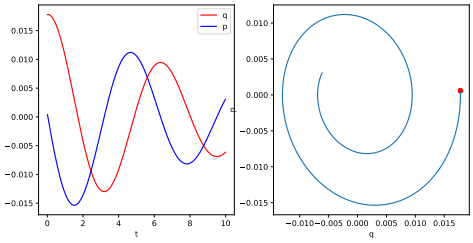
conservation, where we see energy consevation is not as enforced when we have noisy invariance. Degree of freedom increases with jitter as expected since we are relaxing the constraints. While the lower the jitter, the higher the performance should be in theory; the instability of matrix inversion and numerical errors could cause a drop in performance. As a result, we will also choose the smallest amount of jitter that still gives stable performance. Note that, in both cases of varying jitter and invariance density; all of them are much better than the RBF in performance anyways.

4.7 Damped System

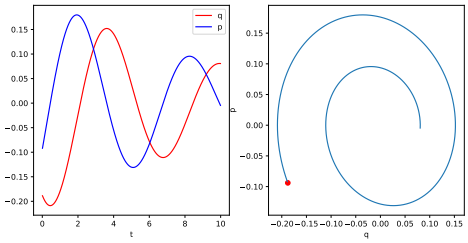
In previous sections, we have considered a perfect system, i.e. ideal world without frictions and the conservation of energy is completely obeyed such that $L = 0$ exactly. However, in a real system, there will be dissipation where energy is lost in the form of heat etc. Therefore, to model that, we need to allow "approximate" invariance, such that the invariance L is noisy and not always equal to zero, i.e. soft invariance. To account for that, in a similar spirit to when we add noise to the underlying latent GP to represent noise signal in equation, 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 for SHM and pendulum respectively

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

and the γ is the damping factor that controls how much frictional force there is. Note that we will only focus on underdamping case ($\gamma < \omega_0^2$)so that the system still oscillate but with gradually reduced amplitude instead of critical damped ($\gamma = \omega_0^2$) or overdamped ($\gamma > \omega_0^2$) case where the system simply slowly decays to still. An example trajectory for damped SHM and damped pendulum is shown below in figure 4.14.



(a) Example trajectory of damped SHM



(b) Example trajectory of damped pendulum

Figure 4.14: Example trajectories of damped systmes

4.7.1 Damped SHM

For damped system, we have exactly the same setup as the original undamped SHM. The results is again summarised in table 4.3 and figure 4.15, 4.16. The learnt $f(p)$ and $g(q)$ is shown in figure 4.17.

Method	RBF	Known Invariance	Learnt Invariance
Log Marginal Likelihood	67.69	74.84	77.43
MSE	0.9226	0.4772	0.2405

Table 4.3: Damped SHM performance

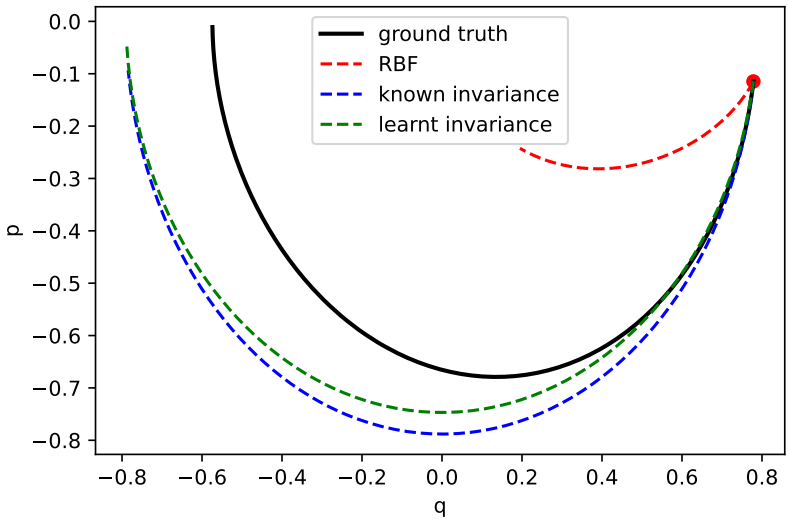


Figure 4.15: Damped SHM predicted trajectory

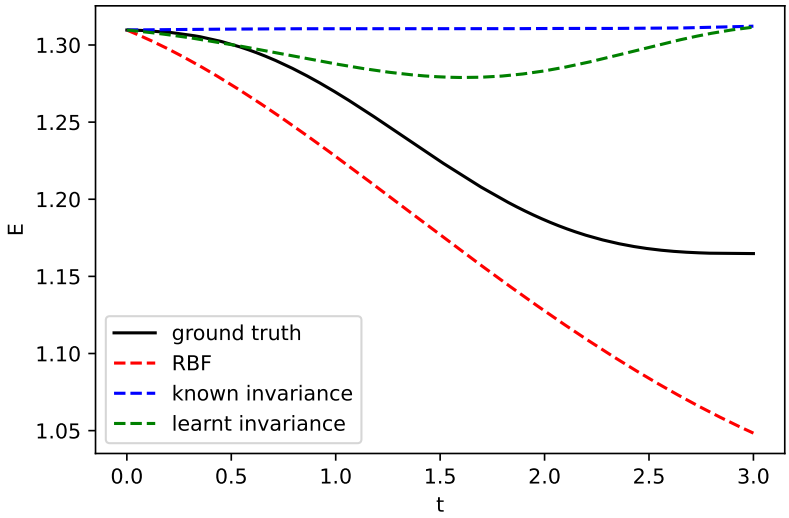


Figure 4.16: Energy conservation for damped SHM

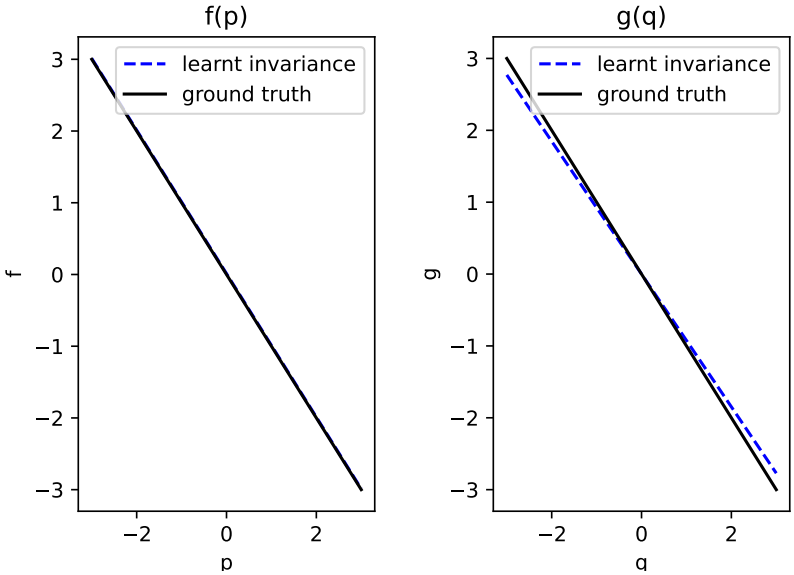


Figure 4.17: Learnt invariance for damped SHM

We can see that the RBF performance is still poor as before. However, this time the invariance kernel does not perform as well as before. This is expected since we still enforce some degree of invariance, even though it is noisy. However, it is clear that it is making use of the knowledge of the invariance since the performance is fairly good. For energy consevation, it is reassuring that the invariance kernel is still remains invariant, while the learnt invariance might deviate a little since the extra parameters in the polynomial might fit to the damped data; but overall it conserves energy relatively well compare to the ground truth. Also, when we are learning the invariance, it again successfully recover the physics even under the effect of damping.

4.7.2 Damped Pendulum

Again, the setting of damped pendulum is the same as the undamped pendulum. The results is summarised in table 4.4 and figure 4.18, 4.19. With the learnt polynomial in 4.20.

Method	RBF	Known Invariance	Learnt Invariance
Log Marginal Likelihood	229.96	243.41	243.48
MSE	0.5983	0.0323	0.0877

Table 4.4: Damped Pendulum Performance

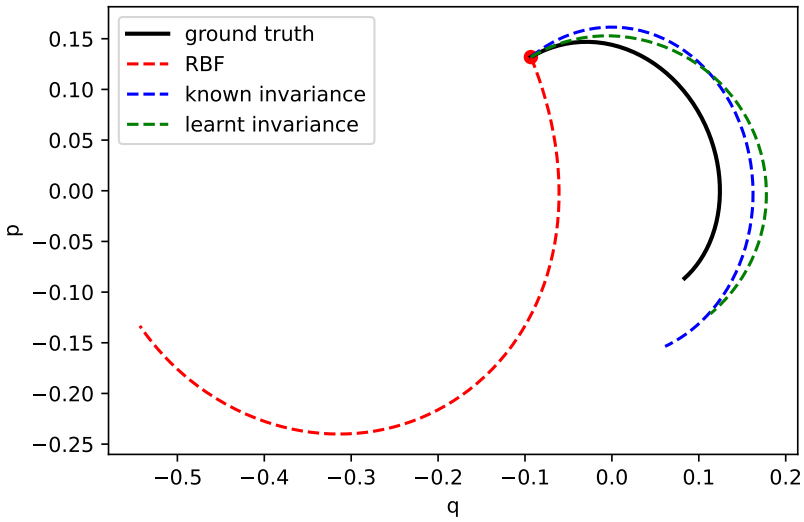


Figure 4.18: damped pendulum predicted trajectory

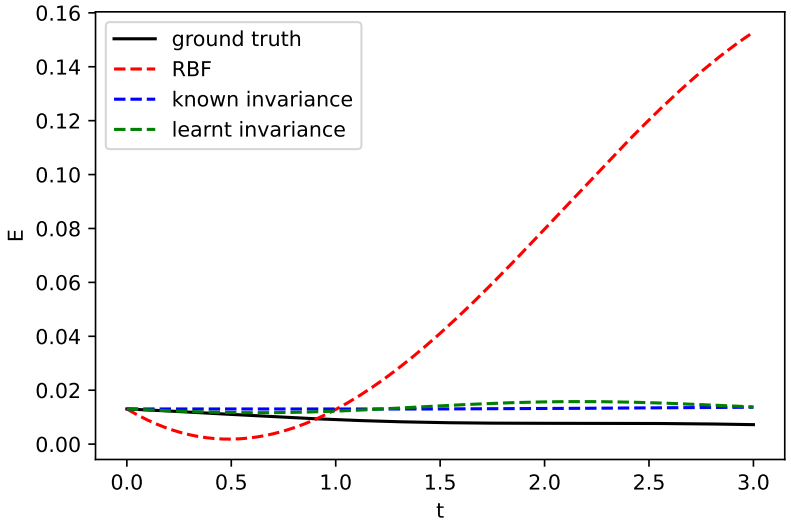


Figure 4.19: Energy conservation for damped pendulum

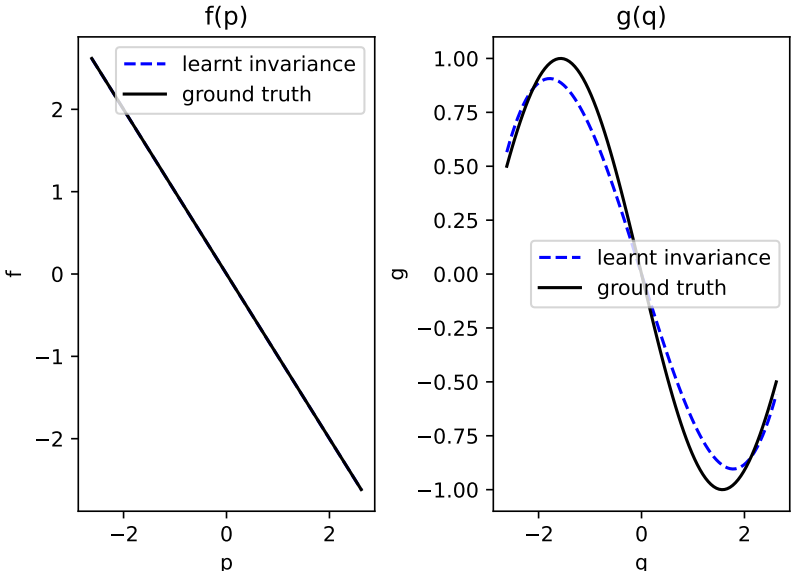


Figure 4.20: Learnt invariance for damped pendulum

We see similar performance as in damped SHM, with invariance kernels performs slightly worse than undamped version but still much better than the RBF. The energy conservation still holds mostly for invariance kernels. However, this time, the learnt polynomial is not as good as the undamped case with much more deviation; this is expected due to the damping.

4.8 2D System

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

4.8.1 Linear SHM

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 $L(a_1, a_2, v_1, v_2) = \frac{dE}{dt} = mp_1a_1 + mp_2a_2 + kq_1v_1 + kq_2v_2 = 0$. 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.

Here we again use the same set of input space as the 1D space, but double the dimensions. Here we actaully have the input space of q_1, q_2 being between ± 5 and p_1, p_2 between ± 0.5. This is done so that when we scale the data with MinMax scaler, the points will be less clustered. We use 30 points in this setting. **One very important note** is that when we are learning invariance, we initialise the hyperparameters of the base RBF kernel (lengthscales and variance local grids as well as likelihood variance) at the values trained by the known invariance. We also intialise the polynomial coefficients at the true theoretical values. These are done due to the fact the optimiser is often stuck at local minima. We intialise the values there so that if it is indeed the true invariance,

the optimiser should regonise it and not optimise too much since that is roughly the true minimum. Therefore, we would expect to get similar performance and marginal likelihood.

The results are summarised in table 4.5 and figure 4.21, 4.22. Since it is difficult to visualise 4 dimensional space, we will plot them individually.

Method	RBF	Known Invariance	Learnt Invariance
Log Marginal Likelihood	387.08	441.60	438.42
MSE	1.1252	0.7374	0.7359

Table 4.5: 2D SHM Invariance performance

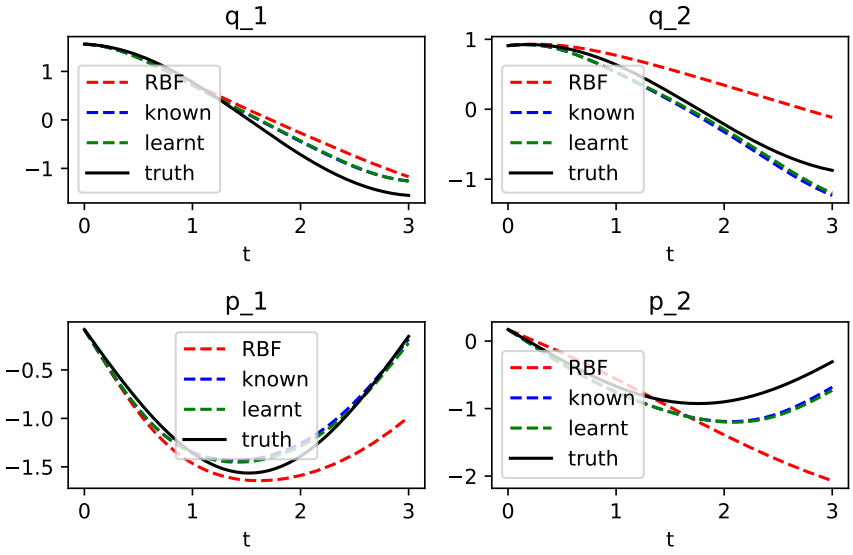


Figure 4.21: 2D SHM prediction

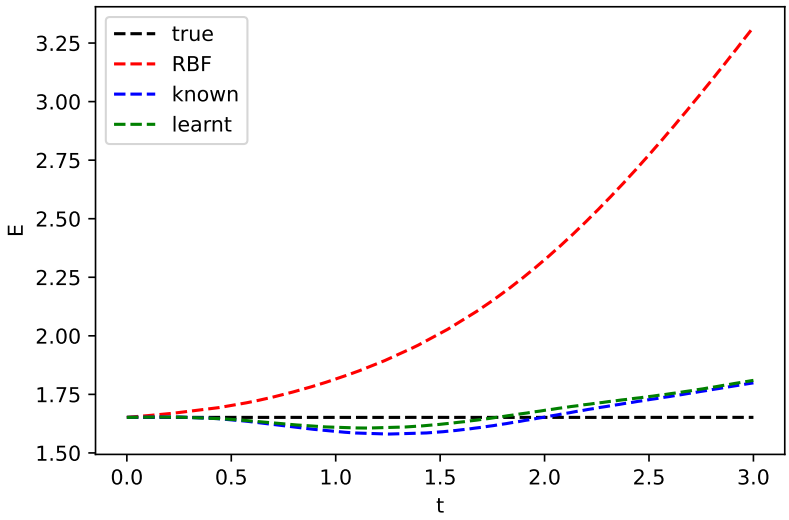


Figure 4.22: 2D SHM energy

Firstly, we can see that the invariance kernel still performs very well compared to RBF, although not as good as the 1D case as expected since higher dimensional problem is deemed to be more difficult. We can see the learnt invariance again is almost exactly the same as the known form, which is reassuring. Also, the conservation of energy is again pretty well obeyed.

4.8.2 Non Linear Double Pendulum

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:

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

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

$$E = -(m_1+m_2)gl_1\cos q_1 - m_2 g l_2 \cos q_2 + \frac{m_1 l_1^2 p_1^2}{2} + \frac{m_2}{2}(l_1^2 p_1^2 + l_2^2 p_2^2 + 2l_1 l_2 p_1 p_2 \cos(q_1 - q_2))$$

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

$$L(a_1, a_2, v_1, v_2) = \frac{dE}{dt} = (m_1 + m_2)gl_1 \sin\theta_1 v_1 + m_2 g l_2 \sin\theta_2 v_2 + m_1 l_1^2 \dot{\theta}_1 a_1 +$$
$$m_2(l_1^2 \dot{\theta}_1 a_1 + l_2^2 \dot{\theta}_2 a_2 + l_1 l_2(\dot{\theta}_2 \cos(\theta_1 - \theta_2)a_1 + \dot{\theta}_1 \cos(\theta_1 - \theta_2)a_2 - \dot{\theta}_1 \dot{\theta}_2 \sin(\theta_1 - \theta_2)(v_1 - v_2))) = 0$$

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

With double pendulum, we have angles being between $\pm 60°$ and angular velocity between $\pm 10°$. The results are summarised in table 4.6 and figure 4.23, 4.24. Again we used 30 points to train, but this time we do not use scaling since perhaps due to its nonlinear nature, it made the performance much worse by possibly modifying its underlying distribution.

Method	RBF	Known Invariance	Learnt Invariance
Log Marginal Likelihood	455.89	471.01	494.37
MSE	0.3879	0.0286	0.0362

Table 4.6: Double pendulum invariance performance

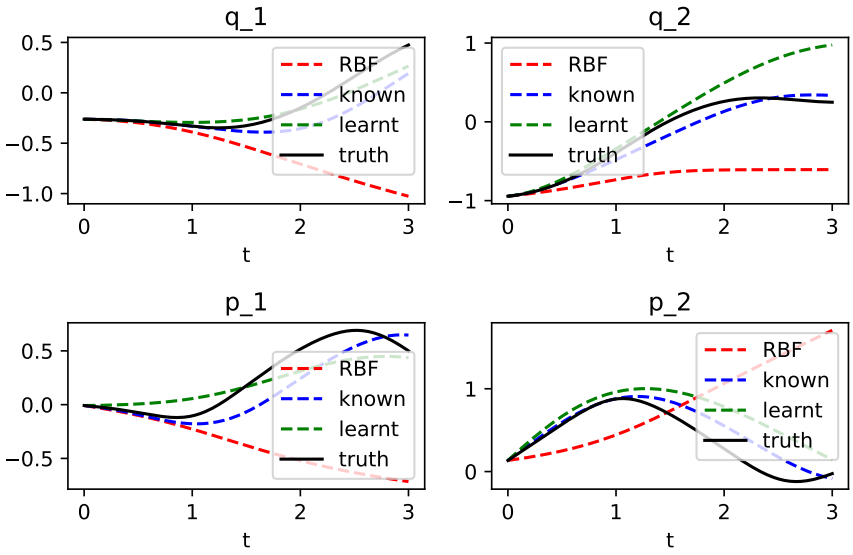


Figure 4.23: Double pendulum prediction

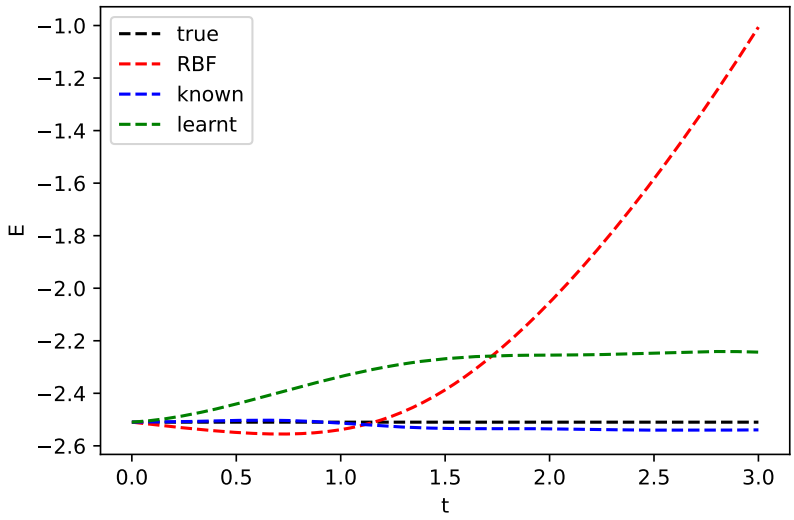


Figure 4.24: Double pendulum energy

Now it is interesting to obsesrve that the marginal likelihood of the learnt form is even better than known form. This is likely due to "overfitting" of the large number of parameters from the polynomials (140 in total). Also, we are only expanding up to the cubic term, therefore we do not expect very good learning for the invariance. However, in our case, the approximation is fairly good, possibly due to relatively small value of q, p so the polynomial approximation is good. Nevertheless, it is clear that there is still some errors with learning invariance from figure 4.24 since it only flattens toward the end of the trajectory.

5 Conclusion

Conclusion goes here.

References

Jonathan Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.

Sam Greydanus Google Brain, Misko Dzamba PetCube, and Jason Yosinski. Hamiltonian neural networks.

Michael M. Bronstein, Joan Bruna, Yann Lecun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34:18–42, 7 2017. ISSN 10535888. doi: 10.1109/MSP.2017.2693418.

Ricky T Q Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations.

Miles Cranmer, Sam Greydanus, Stephan Hoyer, Google Research, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. 2020.

Miles D Cranmer, Rui Xu, Peter Battaglia, and Shirley Ho. Learning symbolic physics with graph networks.

Salvatore Cuomo, Vincenzo Schiano, Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific machine learning through physics-informed neural networks: Where we are and what's next. 2022.

Paul Glendinning. *Stability, instability and Chaos: An introduction to the theory of nonlinear differential equations.* Cambridge University Press, 1994.

Markus Heinonen, C ¸ Agatay Yıldız, Henrik Mannerström, Jukka Intosalmi, and Harri Lähdesmäki. Learning unknown ode models with gaussian processes. 2018. URL `http://www.`

Ameya D. Jagtap, Ehsan Kharazmi, and George Em Karniadakis. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 365, 6 2020. ISSN 00457825. doi: 10.1016/J.CMA.2020.113028.

George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*. doi: 10.1038/s42254-021-00314-5. URL `www.nature.com/natrevphys`.

Ziming Liu and Max Tegmark. Machine learning conservation laws from trajectories. 2021. doi: 10.1103/PhysRevLett.126.180604.

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`.

Damian Mrowca, Chengxu Zhuang, Elias Wang, Nick Haber, Li Fei-Fei, Joshua B Tenenbaum, and Daniel L K Yamins. Flexible neural representation for physics prediction.

Konstantinos Papastamatiou, Filippos Sofos, and Theodoros E Karakasidis. Machine learning symbolic equations for diffusion with physics-based descriptions articles you may be interested in machine learning symbolic equations for diffusion with physics-based descriptions. *AIP Advances*, 12:25004, 2022. doi: 10.1063/5.0082147. URL `http://creativecommons.org/licenses/by/4.0/`.

Elizabeth Qian, Boris Kramer, Benjamin Peherstorfer, and Karen Willcox. Lift & learn: Physics-informed machine learning for large-scale nonlinear dynamical systems. *Physica D: Nonlinear Phenomena*, 406, 5 2020. ISSN 01672789. doi: 10.1016/J.PHYSD.2020.132401.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2 2019. ISSN 0021-9991. doi: 10.1016/J.JCP.2018.10.045.

Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357: 125–141, 3 2018. ISSN 10902716. doi: 10.1016/J.JCP.2017.11.039.

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

Katharina Rath, Christopher G Albert, Bernd Bischl, and Udo Von Toussaint. Symplectic gaussian process regression of hamiltonian flow maps symplectic gaussian process regression of hamiltonian flow maps symplectic gaussian process regression of hamiltonian flow maps. 2020.

Andrew Sosanya and Sam Greydanus. Dissipative hamiltonian neural networks: Learning dissipative and conservative dynamics separately.

Steven Strogatz. *Nonlinear Dynamics and Chaos: With applications to physics, Biology, Chemistry and Engineering*. CRC Press, 2019.

Silviu Marian Udrescu and Max Tegmark. Ai feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6, 4 2020. ISSN 23752548. doi: 10.1126/SCIADV.AAY2631/ASSET/DEA58C72-32B7-4207-BD40-91D257F0D857/ASSETS/GRAPHIC/AAY2631-F2.JPEG. URL `https://www.science.org/doi/10.1126/sciadv.aay2631`.

Tycho FA van der Ouderaa and Mark van der Wilk. Learning invariant weights in neural networks. 2021.

Mark Van Der Wilk, Matthias Bauer, S T John, and James Hensman. Learning invariances using the marginal likelihood.

Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Benchmarking energy-conserving neural networks for learning dynamics from data. *Proceedings of Machine Learning Research*, 144:1–12, 2021.