Charlie meeting 13th June

Meeting

This week, after talking with Mark, I first need to redo the calculation of degree of freedom of invariance GP on a fixed grid instead of the data, and compare similar thing with the degree of freedom on a RBF. The next thing for the week is try to extend the system to a nonlinear dynamics system as opposed to a linear system as before. A simple example will be a pendulum with largeish angle swing (not too big such that it tips over) Afterwards, we will demonstrate the kernel on a dissipative system, a prime example will be a damped oscillator, where we will not enforce strong invariance but have say a small noise tolerance, "soft invariance". We also discussed the effect of jitter and I will try to assess if it's too big by increasing it until it breaks. After discussion with Andrew, he had concerns about the grid invariance being not scalable in high dimensions and proposed the idea of active learning, where we produce a grid along the trajectory (local mesh), which will be looked into as a side exploration when dimension gets high. He also mentioned some possible parametisation techniques (something about Fourier components or polynomical or neural network) to paramertise invariances but it will be looked at only at the final phase of the project.

Pendulum

To get dataset, we need to solve the second order ODE numerically

$$\frac{d^2\theta}{dt^2} + \frac{g}{l}\sin(\theta) = 0,$$

we can rewrite it as two coupled ODEs

$$\begin{cases} \frac{d\theta}{dt} = \omega \\ \frac{d\omega}{dt} = -\frac{g}{l}\sin\theta \end{cases}$$

and then we can use (two) Runge-Kutta method (RK4) to generate data. Note that the acceleration and velocity targets will again be estimated using discrete difference since we assume the trajectory data is given. This time, we have the energy

$$E = \frac{1}{2}ml^2\dot{\theta}^2 + mgl(1-\cos(\theta)),$$

with small angle approximation, we have $1-\cos(\theta) \approx \theta^2$ so we get the original SHM form back. Now the invariance is

$$\dot{E} = ml^2\dot{\theta}a(\theta,\dot{\theta}) + mgl\sin(\theta)v(\theta,\dot{\theta}) = 0$$

Therefore, the structure of codes will not change, it is just the expression will be slightly different. We have m and l canceled out so we are left with

$$l\dot{\theta}a + g\sin(\theta)v = 0$$

For simplicity and consistency, we will label $\theta := x$ so that we can reuse most of previous works. We again have:

$$\begin{pmatrix} \mathbf{f}(X_1) \\ \mathbf{f}(X_2) \\ L_{X_g}[\mathbf{f}(X_g)] \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mathbf{0}_{2n} \\ \mathbf{0}_{2m} \\ \mathbf{0}_{\ell} \end{pmatrix}, \begin{pmatrix} K(X_1,X_1) & K(X_1,X_2) & B_1 \\ K(X_2,X_1) & K(X_2,X_2) & B_2 \\ C_1 & C_2 & D \end{pmatrix} \right),$$

This time since the invaraince is $la\dot{x} + gv\sin x = 0$, we have

$$B = \begin{pmatrix} K_a(X,X_g) \\ K_v(X,X_g) \end{pmatrix} \odot \begin{pmatrix} l\dot{\tilde{X}} \\ g\sin\tilde{X} \end{pmatrix},$$

and

$$D = K_a(X_g,X_g) \odot l^2\dot{\tilde{X}}^2 + K_v(X_g,X_g) \odot g^2 \sin\tilde{X}^2$$

Note that $\sin\tilde{X}^2 = \begin{pmatrix} \sin^2\tilde{x}_1 & \sin\tilde{x}_1\sin\tilde{x}_2 & \dots & \sin\tilde{x}_1\sin\tilde{x}_\ell \\ \vdots & \vdots & \vdots & \vdots \\ \sin\tilde{x}_\ell\sin\tilde{x}_1 & \sin\tilde{x}_\ell\sin\tilde{x}_2 & \dots & \sin^2\tilde{x}_\ell \end{pmatrix}$. For simplicity we will assume we know and set $g = 9.81$ since it is a universal constant. Need certain level of accuracy in the data to perform well, which makes sense since I assume the model, and if the data doesn't follow the model well, I will do worse.

Damped SHM

If we have a damped SHM, we have a second order ODE

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

with a damping force $F_{damp} = -cv$ and the damping coefficient is given by $\gamma = \frac{c}{2m}$. People usually write it as

$$\ddot{x} + 2\gamma\dot{x} + \omega_0^2 = 0,$$

where $\gamma = \frac{c}{2m}$ and $\omega_0 = \sqrt{\frac{k}{m}}$. We will only consider the light damping case ($\gamma << \omega_0$) since invariance definitely would not hold at all at large damping regime. Essentially, we will keep most of the machinary of original SHM but instead of conditioning on $a\dot{x} + vx = 0$, we will condition on $a\dot{x} + vx = \epsilon$, where ϵ is some small number that allows tolerance to slight violation of invariance. In this case, since $E = \frac{kx^2}{2} + \frac{m\dot{x}^2}{2}$ is still true, we have

$$\frac{dE}{dt} = \dot{x}(m\ddot{x} + kx) = -c\dot{x}^2 = -2m\gamma\dot{x}^2 < 0$$

If γ is small enough, we should expect the invariance to hold relatively okay. To produce the data, we have $x(t) = e^{-\gamma t}\sin(\omega_d t)$, and $\omega_d = \sqrt{\omega_0^2 - \gamma^2}$ $\dot{x}(t) =$

$-\gamma e^{-\gamma t}\sin(\omega_d t) + \omega_d e^{-\gamma t}\cos(\omega_d t) = e^{-\gamma t}(\omega_d \cos(\omega_d t) - \gamma \sin(\omega_d t))$. We need to modify the mean function to accomdate the small tolerance ϵ. Before we have

$$\begin{pmatrix} \mathbf{f}(X_1) \\ \mathbf{f}(X_2) \end{pmatrix} | L_{X_g}[\mathbf{f}(X_g)] = 0 \sim \mathcal{N}\left(\begin{pmatrix} \mathbf{0}_{2n} \\ \mathbf{0}_{2m} \end{pmatrix}, A - \begin{pmatrix} B_1 \\ B_2 \end{pmatrix} D^{-1} (C_1, C_2) \right)$$

and now we have

$$\begin{pmatrix} \mathbf{f}(X_1) \\ \mathbf{f}(X_2) \end{pmatrix} | L_{X_g}[\mathbf{f}(X_g)] = \epsilon \sim \mathcal{N}\left(\begin{pmatrix} \mathbf{0}_{2n} \\ \mathbf{0}_{2m} \end{pmatrix} + \begin{pmatrix} B_1 \\ B_2 \end{pmatrix} D^{-1} \begin{pmatrix} \epsilon_l \end{pmatrix}, A - \begin{pmatrix} B_1 \\ B_2 \end{pmatrix} D^{-1} (C_1, C_2) \right)$$

For the purpose of subclassing the mean function, we only need X_1 so that we have the mean function

$$\mathbf{0}_{2n} + BD^{-1}\epsilon_l$$

I make ϵ a trainable parameter, I have tried both random normal version as well as a constant ones multiplied by small ϵ. I have also made a version of mean function that uses $-2m\gamma\dot{x}^2$ as the righthand side of the invariance as opposed to using ϵ.

Damped Pendulum

It has equation

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

where γ is again some damping coefficient and ω_0^2 is $\frac{g}{l}$. Since

$$\frac{dE}{dt} = ml^2\dot{\theta}a + mgl\sin\theta v = -2\gamma ml^2\dot{\theta}^2$$

so that the invariance becomes

$$l\dot{\theta}a + g\sin\theta v = -2\gamma l\dot{\theta}^2$$

Experiments

Now I have four different systems to test, I will unify my procedure and make the data generation consistent, and send to GPU to run tests. For SHM and damped SHM, the data will be generated analytically for 30 seconds with two (initial) trajectory of amplitude 1 and 2 each. For pendulum and damped pendulum, numerical integration is used for both to generate 30 seconds of data for initial angle 90 and 150 degrees each. We will have sample one data point per second uniformly so 30 points for each trajectory. We also set all constants (k, m, g, l) to 1. For all the data, after I generate and position and velocity, I will add a random noise of $\sigma^2 = 0.1$ to each data point and then I will compute the derivative using central difference (degree 2) formula. We will test the following, the degree of freedom for SHM and pendulum for different invariance density. We will test for damped SHM and pendulum, whether epsilon method or known method is better. We will test the above for different jitter and different invariance density. We will also test different damping results.

Results

Finer grid does not lead to better performance, sometimes it's worse. Finer grid does give lower degree of freedom however. For some reason, greater jitter give better performance? (not that much difference) In general there's not a dramatic difference. Is there a problem with my data generating mechanism? Is there a problem with the way I add jitter? When damping gets stronger, the anlaytical model may be better.

General Parametisation

If we want to learn the invariance automatically by backpropagation, maybe we should try to paramertise the invariance by neural network. Say if have a general energy expression, with velocity seperated from the position (most situation), we have $E = f(\dot{x}) + g(x)$, and we also have

$$\frac{dE}{dt} = af'(\dot{x}) + vg'(x).$$

If we label $f' = f, g' = g$ for simplicity of notation, we have

$$af(\dot{x}) + vg(x) = 0.$$

In SHM case, $f = m\dot{x}, g = kx$ and for pendulum, $f = l\dot{x}, g = g\sin x$. We will then have

$$B = \begin{pmatrix} K_a(X, X_g) \\ K_v(X, X_g) \end{pmatrix} \odot \begin{pmatrix} f(\dot{\tilde{X}}) \\ g(\tilde{X}) \end{pmatrix},$$

and

$$D = K_a(X_g, X_g) \odot f(\dot{\tilde{X}})^2 + K_v(X_g, X_g) \odot g(\tilde{X})^2.$$

We can then parametise f and g with neural network. It should in principle work if we cannot seperate position and velocity, say if we have $E = V(x, \dot{x}) + T(x, \dot{x})$, we will have

$$\frac{dE}{dt} = \frac{\partial V}{\partial x}v + \frac{\partial V}{\partial \dot{x}}a + \frac{\partial T}{\partial x}v + \frac{\partial T}{\partial \dot{x}}a = (\frac{\partial V}{\partial \dot{x}} + \frac{\partial T}{\partial \dot{x}})a + (\frac{\partial V}{\partial x} + \frac{\partial T}{\partial x})v = \frac{\partial E}{\partial \dot{x}}a + \frac{\partial E}{\partial x}v = 0$$

Which we can again write as $f(x, \dot{x})a + g(x, \dot{x})v = 0$, we can then proceed as before.