Charlie meeting 20th June

Meeting

In this meeting, Andrew and I discussed the result I produced last week. Everything is working, which is good. This week I will work on different type of paramtisation to paramatise the energy function, Andrew suggested a few different methods as basis functions, such as polynomial, splines and Fourier representation. We also talked about my methods of adding jitter, which might not be entire reasonable so I am fixing that too. He also talked about a way that makes this methods scalable, which is that instead of a grid of invariance, we only condition on the local area around the data points as well as the test points we wish to evaluate. If noise too small or too much data, it's not good for nonlinear nor damping. But in most reasonable cases, 30 seconds with 30 points with noise 0.1 the kernel all worked well. We have four scenarios to check. Noisy being 0.1 and not noisy being 0.01. Lots of data means 0.1 time spacing. left is naive, right is invariance

1. lots of data, noisy
2. little data, noisy
3. lots of data, not noisy
4. little, not noisy

	lots of data, noisy	little data, noisy	lots of data, not noisy	little data, not noisy
SHM	-1323, -1305	65, 81	1354, 1394	271, 311
Fixed Damped SHM	-1324, -1306; -1327, -1313; -1351, -1340	67, 88; 41, 57; 55, 68	1409, 1444; 1451, 1467; 1461, 1458	269, 305; 276, 298; 298, 313
Pendulum	-1307, -1288	61, 65	1403, 1451	232, 115
Fixed Damped Pendulum	-1330, -1311; -1307, -1292; -1290, -1278	64,74; 68, 79; 75, 83	1391, 1424; 1372, 1386; 1414, 1417	284, 276; 273, 273; 280, 278

We can see that most of the entries are good, except lots of data not noisy for fixed damping when damping is high, but dynamical damping still outperforms. A lot of problem is found in low data, low noise area; especially when it is non linear or damping is high. This make sense in some way. For the first case, since it is harder to for fixed damping to handle and predict larger damping, if there are less noise to smooth out the effect of damping, so that damping is more varying, it makes sense for fixed damping to perform worse. For the second case, I think the main reason is that the estimation of acceleration and velocity is already not so good to start with, if we have low noise, it's like we are assuming that's the truth. Therefore, our physical intuition (invariance) will misguide us since we will be expecting something else.

Local Invariance

Before we have

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

Before with the grid of invariance X_g is fixed, now they are consist of the local area around the test points and the training trajectory data. Therefore,

the only change we need to make is X_g is now a function of X_1, X_2 so that $X_g = X_g(X_1, X_2)$. As a result, now B, C, D will all be a function of X_1, X_2. Essentially, we just have to redefine our grids to be the neighbourhoods around test and training points. In our case, before we have 1600 points in a 6 by 6 area, now perhaps we can have 4 points around the 0.1 by 0.1 area around each point.

Change to damping approximate invariance

Now instead of our original method of conditioning on an either analytical form or like a fix or random vector, we will allow the invariance to be noisy.

Before we had,

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

Now it is like when we have noise on the observation, we also have noise on the invariance. Therefore, in a simliar way to $y = f + \text{noise}_{\sigma_n^2}$, wer have $\tilde{L} = L + \text{noise}_{\epsilon}$. Since all the noise are uncorrelated (Gaussian white noise), we will only modify the D matrix (signal noise will be accounted for by GPflow). Therefore, we will have $\tilde{D} = D + \epsilon I$. So we have

$$\mathbf{f}(X) | \tilde{L}_{X_g}[\mathbf{f}(X_g)] = 0 \sim \mathcal{N}\left(0, A - B\tilde{D}^{-1}C\right).$$

The rest would be the same as before. The results are good.

Conclusion of the week

Everything looks good, the local invariance, the parameterisation, the damping.Jitter is also checked. When there is damping, the correct coefficient is not picked up by the polynomial but the overall lml is still better than the baseline.

Question

1. I still don't quite understand the difference between conditioning on a random vector and add noise to the thing itself.
2. Should we try to paramertise the damping term?
3. data generation procedure (noise, quantity etc.)
4. parameterisation for more complicated cases like double pendulum or inverse square laws?
5. Can we go over my training procedure

So I have data within range -3 to 3, then condition on around the same range for the invariance. Then I evalute the test point around the same range too to make plots. Then I just take the log marginal likelihood of the data. Is it a fair way to compare? And is there other metrics I should check?

Higher Dimensions(2D)

Double pendulum data is simulated with RK4 with the following coupled first order ODEs.

$$\theta_1{}' = \omega_1$$
$$\theta_2{}' = \omega_2$$
$$\omega_1{}' = \frac{-g\left(2m_1+m_2\right)\sin\theta_1 - m_2 g\sin\left(\theta_1-2\theta_2\right) - 2\sin\left(\theta_1-\theta_2\right)m_2\left(\omega_2^2 l_2+\omega_1^2 l_1\cos\left(\theta_1-\theta_2\right)\right)}{l_1\left(2m_1+m_2-m_2\cos\left(2\theta_1-2\theta_2\right)\right)}$$
$$\omega_2{}' = \frac{2\sin\left(\theta_1-\theta_2\right)\left(\omega_1^2 l_1\left(m_1+m_2\right)+g\left(m_1+m_2\right)\cos\theta_1+{\omega_2}^2 l_2 m_2\cos\left(\theta_1-\theta_2\right)\right)}{l_2\left(2m_1+m_2-m_2\cos\left(2\theta_1-2\theta_2\right)\right)}$$

The energy is given by

$$E = -(m_1+m_2)gl_1\cos\theta_1 - m_2 g l_2\cos\theta_2 + \frac{m_1 l_1^2\dot{\theta}_1^2}{2} + \frac{m_2}{2}(l_1^2\dot{\theta}_1^2 + l_2^2\dot{\theta}_2^2 + 2l_1 l_2\dot{\theta}_1\dot{\theta}_2\cos(\theta_1-\theta_2))$$

We then have the invariance equation

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

For simplicty, we set all the constants to 1 and we have

$$(2\sin\theta_1 - \dot{\theta}_1\dot{\theta}_2\sin(\theta_1-\theta_2))v_1 + (\sin\theta_2 + \dot{\theta}_1\dot{\theta}_2\sin(\theta_1-\theta_2))v_2 + (2\dot{\theta}_1 + \dot{\theta}_2\cos(\theta_1-\theta_2))a_1 + (\dot{\theta_2} + \dot{\theta}_1\cos(\theta_1-\theta_2))a_2 = 0$$

Perhaps we should start with 2D SHM. The equation of motion is $\ddot{\mathbf{r}} = -\frac{k}{m}\mathbf{r}$, where $\mathbf{r} = (x_1, x_2)$. We have energy $E = \frac{m(\dot{x_1^2}+\dot{x_2^2})}{2} + \frac{k(x_1^2+x_2^2)}{2}$ so that

$$L = \frac{dE}{dt} = k(x_1 v_1 + x_2 v_2) + m(\dot{x}_1 a_1 + \dot{x}_2 a_2) = 0$$

If we set the constants to be 1, we again have

$$\dot{x}_1 a_1 + \dot{x}_2 a_2 + x_1 v_1 + x_2 v_2 = 0$$

If we want to do 2D space, we need generalise our kernel to four dimensional space. Like before, but now we have

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

with $\mathbf{x} = (x_{1i}, x_{2i}, \dot{x}_{1i}, \dot{x}_{2i})$. And we again stack them up vertically in the order of a_1, a_2, v_1, v_2 so that we have

$$\mathbf{f}(X) = \begin{pmatrix} a_1(x_{11}, x_{21}, \dot{x}_{11}, \dot{x}_{21}) \\ a_1(x_{12}, x_{22}, \dot{x}_{12}, \dot{x}_{22}) \\ \vdots \\ a_1(x_{1n}, x_{2n}, \dot{x}_{1n}, \dot{x}_{2n}) \\ a_2(x_{11}, x_{21}, \dot{x}_{11}, \dot{x}_{21}) \\ a_2(x_{12}, x_{22}, \dot{x}_{12}, \dot{x}_{22}) \\ \vdots \\ a_2(x_{1n}, x_{2n}, \dot{x}_{1n}, \dot{x}_{2n}) \\ v_1(x_{11}, x_{21}, \dot{x}_{11}, \dot{x}_{21}) \\ v_1(x_{12}, x_{22}, \dot{x}_{12}, \dot{x}_{22}) \\ \vdots \\ v_1(x_{1n}, x_{2n}, \dot{x}_{1n}, \dot{x}_{2n}) \\ v_2(x_{11}, x_{21}, \dot{x}_{11}, \dot{x}_{21}) \\ v_2(x_{12}, x_{22}, \dot{x}_{12}, \dot{x}_{22}) \\ \vdots \\ v_2(x_{1n}, x_{2n}, \dot{x}_{1n}, \dot{x}_{2n}) \end{pmatrix}$$

So we have the naive MOI kernel

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

For 2D SHM invariance kernel, we have

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

We have $A = K(X, X)$,

$$B = \begin{pmatrix} K_{a_1}(X, X_g) \\ K_{a_2}(X, X_g) \\ K_{v_1}(X, X_g) \\ K_{v_2}(X, X_g) \end{pmatrix} \odot \begin{pmatrix} \dot{\tilde{X}}_1 \\ \dot{\tilde{X}}_2 \\ \tilde{X}_1 \\ \tilde{X}_2 \end{pmatrix},$$

where $\odot$ is elementwise product and $\dot{\tilde{X}} = \begin{pmatrix} \dot{\tilde{x}}_1 & \dots & \dot{\tilde{x}}_\ell \\ \vdots & \text{n rows} & \vdots \\ \dot{\tilde{x}}_1 & \dots & \dot{\tilde{x}}_\ell \end{pmatrix}$ and similarly for $\tilde{X}$, $C = B^T$,

$$D = K_{a_1}(X_g, X_g) \odot \dot{\tilde{X}}_1^2 + K_{a_2}(X_g, X_g) \odot \dot{\tilde{X}}_2^2 + K_{v_1}(X_g, X_g) \odot \tilde{X}_1^2 + K_{v_2}(X_g, X_g) \odot \tilde{X}_2^2$$

, where $\tilde{X}^2 = \begin{pmatrix} \tilde{x}_1^2 & \tilde{x}_1\tilde{x}_2 & \dots & \tilde{x}_1\tilde{x}_\ell \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{x}_\ell\tilde{x}_1 & \tilde{x}_\ell\tilde{x}_2 & \dots & \tilde{x}_\ell^2 \end{pmatrix}$, and similarly for $\dot{\tilde{X}}^2$. And also, similar to before, we will have

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

Problem! Parameterisation has mixing and simple polynomial probably will not do! Neural Nets?