Charlie meeting 6th May

My previous implementation is incorrect since I treated the velocity and acceleration seperately (independently) after conditioning on the invariance. However, since invariance "intertwined" the acceleration and velocity, they cannot be independent after conditioning. Therefore, we need to consider them all together. Going back to the original eqautions

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} a(\mathbf{x}) \\ v(\mathbf{x}) \end{pmatrix} = \dot{\mathbf{x}}.$$

So we map from 2n to 2n. We will choose to stack these n data points vertically such that

$$\mathbf{f}(X) = \begin{pmatrix} a(x_1, \dot{x}_1) \\ a(x_2, \dot{x}_2) \\ \vdots \\ a(x_n, \dot{x}_n) \\ v(x_1, \dot{x}_1) \\ \\ v(x_2, \dot{x}_2) \\ \vdots \\ v(x_n, \dot{x}_n) \end{pmatrix}$$

And we put independent prior between $a(\mathbf{x}), v(\mathbf{x})$ on $\mathbf{f}$ so that $\mathrm{Cov}(\mathbf{f}(X), \mathbf{f}(X')) = K(X, X') = \begin{pmatrix} K_a(X, X') & 0 \\ 0 & K_v(X, X') \end{pmatrix}$. We then condition on the invariance $L(\mathbf{f}) = 0$, where L is the appropriate invariance function. In the case of simple harmonic oscillator, with have $a\dot{x} + vx = 0$ so for a single invaraiance point, $L = (\dot{x}, x)$, a row vector. The form of condition depends on the grid of invariance X_g, which we assume has ℓ points, so

$$X_g = \begin{pmatrix} \tilde{x}_1 & \dot{\tilde{x}}_1 \\ \tilde{x}_2 & \dot{\tilde{x}}_2 \\ \vdots & \vdots \\ \tilde{x}_\ell & \dot{\tilde{x}}_\ell \end{pmatrix} \text{ and } \mathbf{f}(X_g) = \begin{pmatrix} a(\tilde{x}_1, \dot{\tilde{x}}_1) \\ \vdots \\ a(\tilde{x}_\ell, \dot{\tilde{x}}_\ell) \\ v(\tilde{x}_1, \dot{\tilde{x}}_1) \\ \vdots \\ v(\tilde{x}_\ell, \dot{\tilde{x}}_\ell) \end{pmatrix}.$$

In this case

$$L_{X_g}[\mathbf{f}(X_g)] = \begin{pmatrix} \dot{\tilde{x}}_1 a(\tilde{x}_1, \dot{\tilde{x}}_1) + \tilde{x}_1 v(\tilde{x}_1, \dot{\tilde{x}}_1) \\ \dot{\tilde{x}}_2 a(\tilde{x}_2, \dot{\tilde{x}}_2) + \tilde{x}_2 v(\tilde{x}_2, \dot{\tilde{x}}_2) \\ \vdots \\ \dot{\tilde{x}}_\ell a(\tilde{x}_\ell, \dot{\tilde{x}}_\ell) + \tilde{x}_\ell v(\tilde{x}_\ell, \dot{\tilde{x}}_\ell) \end{pmatrix}.$$

We can check L_{X_g} is a linear transfomration. As a result, we have joint distribution

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

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

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

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 $\tilde{X}$, $C = B^T$,

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

, 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 $\dot{\tilde{X}}^2$.

Using the Gaussian condition formula, we have

$$\mathbf{f}(X)|L_{X_g}[\mathbf{f}(X_g)] = 0 \sim \mathcal{N}\left(0, A - BD^{-1}C\right).$$

If we want to write a GPflow kernel with it, we need an expression of K_{inv}, and we will need to compute the value of $K_{inv}(X_1, X_2) = \text{Cov}(\mathbf{f}(X_1), \mathbf{f}(X_2))$, while the above expression if for $X_1 = X_2$. Consider two inputs X_1 and X_2, with dimension 2n and 2m, so that we have

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

We will call $A = \begin{pmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{pmatrix}$, and similar to before $B_1 = \begin{pmatrix} K_a(X_1, X_g) \\ K_v(X_1, X_g) \end{pmatrix} \odot \begin{pmatrix} \dot{\tilde{X}} \\ \tilde{X} \end{pmatrix}$, and similarly for B_2. We again have $C_1 = B_1^T, C_2 = B_2^T$ and D is the same as before. Finally, we have

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

So our $K_{inv}(X_1, X_2)$ will be the top right quadrant of the above invariance matrix. we have K_a, K_v being independent 2D RBF kernel with input space $x, \dot{x}$. I have also restrict the lengthscales as well as variance to be between 10. and 0.01, or else it will just simply maximise the lengthscales and become a flat plane (which is what we want, but it is not very useful here as we wouldn't know that as prior). I have also add a small (10^{-5}) white kernel to the individual RBF kernel to increase stability under optimisation so it can invert the matrix properly. I impose the invariance between -5 and 5, and I vary the density of invariance points manually. The key finding is that I achieve a lower training

loss and a much higher log marginal likelihood by imposing invariance. By increasing density, it slightly increases the log marginal likelihood, but beyond the point, the improvement is marginal. I have also tried to calculate the degree of freedom by considering Trace$(K(K + \sigma_n^2 I)^{-1})$, which counts the number of eigenvectors not eliminated (ML for GP book on GP regression, by considering the GP regression as a smoother on the target). There is a huge decrease in degree of freedom from free GP to invariance GP, but there seem not to be that much difference from coarse grids to fine grids of invariance (I tested from 10×10 to 60×60).

Now I am trying to not telling the invaraince what the mass, m, and spring constant k is, which I previously assumed to be one. (the data will still fix at one, but the model should learn it by maximising log marginal likelihood).