Week 2 Summary

Subteam 2

May 2020

1 Introduction

This week, our main focus was on the Unscented Kalman Filter (UKF). Unlike the Linear Kalman Filter, UKF can be used with non-linear systems. The The UKF is a derivative free alternative to the Extended Kalman Filter (EKF), which can also be used with nonlinear systems. Since the EKF is basically a linearized version of the LKF for non-linear systems, it requires computation of Jacobian and Hessian matrices, and is usually only accurate for systems that are easily approximated by a first order linearization. However, UKF functions and is accurate up to the second order with a similar level of computational complexity to the EKF. There are three main applications of the UKF:

1. State estimation
2. Parameter estimation
3. Dual estimation

- State Estimation

  The goal of state estimation is to estimate the state of a nonlinear discrete time dynamical system described as:

  $$x_{k+1} = F(x_k, u_k, v_k) \quad (7.1)$$

  $$y_k = H(x_k, n_k) \quad (7.2)$$

  Here x_k is unknown state of the system, u_k is known exogenous (outside) input, and y_k is observed measurement (known). v_k denotes the process noise and n_k the measurement noise.

  Both F and H assumed to be known.

- Parameter Estimation

  In parameter estimation, which is also referred to as machine learning

in this context, the goal is to find nonlinear mapping:

$$y_k = G(x_k, w)$$

where x_k input, y_k output and the function G is parametrized by vector w, where x_k and y_k are both known. For example, w can specify a set of weights in a neural network. The goal in this setting is thus to determine the vector w. In order to do so, a training set of input and output pairs (x_k, d_k) is provided with the goal of minimizing the error defined as:

$$e_k = d_k - G(x_k, w)$$

The EKF can be used in this context by writing the state-space model as:

$$w_{k+1} = w_k + r_k$$

$$d_k = G(x_k, w_k) + e_k$$

where r_k is process noise and d_k is a nonlinear observation made based on w_k. The EKF can then be applied.

- Dual Estimation

  Dual estimation handles the case where x_k is unknown, and thus the parameters and the state must be estimated at the same time. To handle this consider the system:

$$x_{k+1} = F(x_k, u_k, v_k, w)$$

$$y_k = H(x_k, n_k, w)$$

  where the state and parameters must both be determined only through using the noisy observables y_k.

We will know establish what our goal is and how we will utilize the UKF to achieve it.

2 Optimal Recursive Estimation

Our goal is to estimate the state x_k given observations y_k. If Y_0^k is the sequence of observations up to time k, then the optimal estimate for the state is:

$$\hat{x}_k = E[x_k | Y_0^k]$$

In order to understand this expectation, we can think of the process recursively, since integrals at this scale are difficult to compute. Specifically, let's expression the state at time $k+1$ as:

$$x_{k+1} = F(x_k, u_k, v_k)$$

where u_k is the exogenous input and v_k is the innovations, or process, noise. Exogenous input is used to model any sort of outside forces that are known and impact the system. For example, this can be a force pushing a cart or, in the case of a diabetes model, an individual's meal plan. If, for example, the noise v_k is normally distributed with mean 0 and covariance R^v, then the distribution $p(x_k|x_{k-1})$ is normal with mean $F(x_{k-1}, u_{k-1})$ and covariance R^v.

The observable data y_k is expressed similarly through:

$$y_k = H(x_k, n_k)$$

where n_k is the measurement noise. This general idea is called Bayesian recursion. Without an assumption on the type of densities any of these terms follow, monte carlo methods are needed in order to understand the integrals needed in the calculations. However, if we assume Gaussian densities, the process is greatly simplified.

2.1 Assuming Gaussian Densities

If Gaussian densities are assumed, the only quantities that need to be explicitly calculated are $\hat{x}_k$ and P_{x_k}, which can be done recursively as follows:

$$\hat{x}_k = \hat{x}_k^- + K_k * (y_k - \hat{y}_k^-)$$

$$P_{x_k} = P_{x_k}^- - K_k P_{\tilde{y}_k} K_k^T$$

$$\hat{x}_k^- = E[F(x_{k-1}, u_{k-1}, v_{k-1})]$$

$$K_k = P_{x_k, y_k} P_{\tilde{y}_k, \tilde{y}_k}^{-1}$$

$$\hat{y}_k^- = E[H(\hat{x}_k^-, n_k)]$$

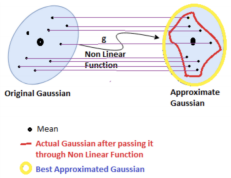
For a linear model, the regular KF can calculate all these terms, and for a nonlinear model the EKF approximates the optimal terms through "first order" approximations. These approximations can be very far off, however, which is the motivation behind using the Unscented Kalman Filter (UKF).

3 The Unscented Kalman Filter

3.1 Conceptual Introduction

The Kalman Filter that we explored last week makes assumptions about the linearity of the dynamical system it is used on that no longer hold here. Since we are now interested in nonlinear systems, meaning that F and H can now be nonlinear functions, we thus need an adaptation of the Kalman Filter. One common approach is the Extended Kalman Filter (EKF), which will not be extensively discussed here, but works by locally linearizing the system and then applying the same KF equations we have seen previously. However, by only

using a first order Taylor approximation, much of the nonlinearity of the system is effectively lost by this approach. The UKF, on the other hand, uses a distribution of points, known as sigma points, to do approximations. The sigma points are first deterministically chosen at time k and then projected to time $k+1$, as we will see in the coming sections. By using a host of points rather than a single one, the UKF more accurately captures the nonlinearity of the system, resulting in far more accurate predictions of posterior means and variances of the state.



Here, we have a visual that shows a general overview of the Kalman filter process. The sigma points are sampled, passed through the transformation to form a new distribution, and then spread of this distribution can be approximated. In this case, the yellow circle, which is the posterior Gaussian distribution after the transformation has been applied, is lost once the transformation occurs (the actual output of the transformation is in red), but can be uncovered through use of the UKF.

To begin developing an understanding of the UKF, we begin with the unscented transformation, which is the framework for how the UKF works.

3.2 Unscented Transformation

The UT is used to understand the distribution of a random variable after a nonlinear transformation. Consider RV x with dimension L and nonlinear function $y = f(x)$. Also assume mean $\bar{x}$ and covariance P_x. To understand distribution of y (i.e. the RV after the transformation is applied) create matrix χ of $2L + 1$ sigma vectors as follows:

$$\chi_0 = \bar{x}$$

$$\chi_i = \bar{x} + (\sqrt{(L+\lambda)P_x})_i \; i = 1...L$$

$$\chi_i = \bar{x} - (\sqrt{(L+\lambda)P_x})_{i-L} \; i = L+1 \; ... \; 2L+1$$

The number $2L+1$ gives us one vector at the mean, and L vectors above and below the mean. L is the number of states in our system. Here, λ is a scaling parameter defined as $\alpha^2(L+k) - L$. It is crucial to note that this is a form of **deterministic** scaling. In many sampling schemes, sampling is done **randomly** from a distribution. However, here our goal is to choose points in equal intervals

and in equal quantities of either side of the mean, $\bar{x}$. The purpose of this is so that we may be certain that sigma points accurately capture the spread of a distribution. With random sampling, if only choosing a small amount of points, no gurantees about this can be made. Thus, exact equations are given for how the sigma points should be chosen. A couple of additional points:

- The subscript i refers to the ith columns in the matrix square root of that term
- α set to small value to represent spread of points around $\bar{x}$
- k set to 0 or $3 - L$ and is secondary scaling parameter

The sigma vectors are then sent through the nonlinear function f to create vectors Y_i. This is done as follows:

$$Y_i = f(\chi_i) \quad i = 0, ..., 2L$$

In cases where the UKF is used and with the setup of the model where the measurement function is H and the transition matrix is F, as we will soon seen, the Unscented Transformation is used to both understand the prior estimate $\hat{x}^-$ (when F plays the role of f here) and the predicted observable values $\hat{y}_k^-$ (when H plays the role of f here).

Before calculating the mean and covariance of these new vectors, create the following weights ($W^{(m)}$ is the mean weight, and $W^{(c)}$ is the covariance weight):

$$W_0^{(m)} = \lambda/(L + \lambda)$$

$$W_0^{(c)} = \lambda/(L + \lambda) + (1 - \alpha^2 + \beta)$$

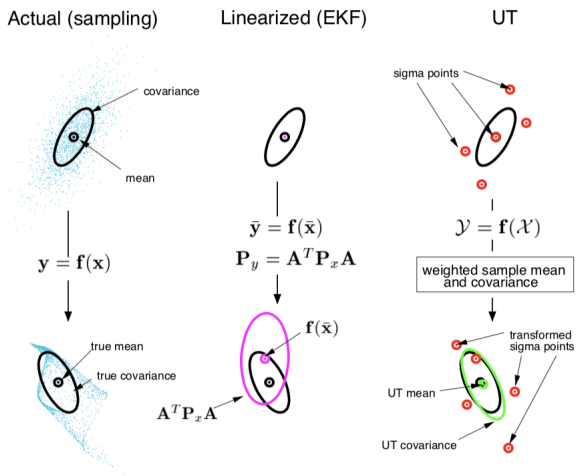
$$W_i^{(m)} = W_i^{(c)} = 1/2(L + \lambda) \; i = 1...2L$$

Where β holds prior knowledge about the distribution (for Gaussian $\beta = 2$ is used). We can see that the sigma point that represents the mean is given more weight than the others. Now the mean and covariance can be calculated:

$$\bar{y} \approx \sum_{i=0}^{2L} W_i^{(m)} Y_i$$

$$P_y \approx \sum_{i=0}^{2L} W_i^{(c)} (Y_i - \bar{y})(Y_i - \bar{y})^T$$

Understanding this mean and covariance will be useful in creating our prior estimates in the UKF method.
To see a visual of this, see the picture below from WanMere Chapter 7:



Here, on the left hand side is what the nonlinear transformation on the data x through the function $f(x)$ ideally looks like (i.e. when all the values are known). Next, in the middle column, the EKF, which linearizes through through first order Taylor approximations, is used and the mean and covariance of y is calculated as shown. Finally, using the Unscented Transformation (UT), we now instead create a sample of points from the distribution (sigma points), and apply the transformation to them. The sigma points that are chosen are indicated by the red circles. Then, using a weighting schema, we find a sample mean and covariance for the points after they have gone through the nonlinear transformation, which more closely captures the entirety of the distribution than the EKF did.

3.3 Unscented Kalman Filter

The set up of our system for the Unscented Kalman Filter is the same as those described for x_{k+1} and y_k in section two, **Optimal Recursive Estimation**. It is important to note that the system has two different forms of noise. The vector v_k describes the process noise and has covariance matrix R^v. Similarly, n_k describes the measurement noise and has covariance matrix R^n. Both of these covariance matrices will play a role in the UKF calculations.

In the general case of the UKF, where no assumptions are made on the additivity of the noise, the state random variable is seen as a concatenation between the original state variables, x_k, the process noise, v_k and the measurement noise n_k, and defined as

$$x_k^a = [x_k^T \ v_k^T \ n_k^T]^T$$

Similarly, the sigma pint selection scheme is applied to all of these variables to calculate the sigma points, giving us:

$$\chi^a = [(\chi^x)^T \ (\chi^v)^T (\chi^n)^T]^T$$

This is why, in the following set of equations, there are sometimes reference to only a particular portion of the sigma points, such as χ^x. The UKF process in this scenario is outlined in the table below, but our major focus has been on the UKF approach under additive noise assumptions

The following equations are needed to use the general UKF:

Initialize with:

$$\hat{\mathbf{x}}_0 = \mathbb{E}[\mathbf{x}_0] \tag{7.35}$$

$$\mathbf{P}_0 = \mathbb{E}[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T] \tag{7.36}$$

$$\hat{\mathbf{x}}_0^a = \mathbb{E}[\mathbf{x}^a] = [\hat{\mathbf{x}}_0^T \ \mathbf{0} \ \mathbf{0}]^T \tag{7.37}$$

$$\mathbf{P}_0^a = \mathbb{E}[(\mathbf{x}_0^a - \hat{\mathbf{x}}_0^a)(\mathbf{x}_0^a - \hat{\mathbf{x}}_0^a)^T] = \begin{bmatrix} \mathbf{P}_0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}^{\mathbf{v}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R}^{\mathbf{n}} \end{bmatrix} \tag{7.38}$$

For $k \in \{1, \ldots, \infty\}$,

Calculate sigma points:

$$\boldsymbol{\mathcal{X}}_{k-1}^a = \left[\hat{\mathbf{x}}_{k-1}^a \quad \hat{\mathbf{x}}_{k-1}^a + \gamma\sqrt{\mathbf{P}_{k-1}^a} \quad \hat{\mathbf{x}}_{k-1}^a - \gamma\sqrt{\mathbf{P}_{k-1}^a}\right] \tag{7.39}$$

Time update:

$$\boldsymbol{\mathcal{X}}_{k|k-1}^x = \mathbf{F}[\boldsymbol{\mathcal{X}}_{k-1}^x, \mathbf{u}_{k-1}, \boldsymbol{\mathcal{X}}_{k-1}^v] \tag{7.40}$$

$$\hat{\mathbf{x}}_k^- = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{X}_{i,k|k-1}^x \tag{7.41}$$

$$\mathbf{P}_k^- = \sum_{i=0}^{2L} W_i^{(c)} [\mathcal{X}_{i,k|k-1}^x - \hat{\mathbf{x}}_k^-][\mathcal{X}_{i,k|k-1}^x - \hat{\mathbf{x}}_k^-]^T \tag{7.42}$$

$$\boldsymbol{\mathcal{Y}}_{k|k-1} = \mathbf{H}[\boldsymbol{\mathcal{X}}_{k|k-1}^x, \boldsymbol{\mathcal{X}}_{k-1}^n] \tag{7.43}$$

$$\hat{\mathbf{y}}_k^- = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{Y}_{i,k|k-1} \tag{7.44}$$

Measurement update equations:

$$\mathbf{P}_{\tilde{\mathbf{y}}_k \tilde{\mathbf{y}}_k} = \sum_{i=0}^{2L} W_i^{(c)} [\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-][\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-]^T \tag{7.45}$$

$$\mathbf{P}_{\mathbf{x}_k \mathbf{y}_k} = \sum_{i=0}^{2L} W_i^{(c)} [\mathcal{X}_{i,k|k-1} - \hat{\mathbf{x}}_k^-][\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-]^T \tag{7.46}$$

$$\mathcal{K}_k = \mathbf{P}_{\mathbf{x}_k \mathbf{y}_k} \mathbf{P}_{\tilde{\mathbf{y}}_k \tilde{\mathbf{y}}_k}^{-1} \tag{7.47}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathcal{K}_k(\mathbf{y}_k - \hat{\mathbf{y}}_k^-) \tag{7.48}$$

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathcal{K}_k \mathbf{P}_{\tilde{\mathbf{y}}_k \tilde{\mathbf{y}}_k} \mathcal{K}_k^T \tag{7.49}$$

where, $\mathbf{x}^a = [\mathbf{x}^T \ \mathbf{v}^T \ \mathbf{n}^T]^T$, $\boldsymbol{\mathcal{X}}^a = [(\boldsymbol{\mathcal{X}}^x)^T \ (\boldsymbol{\mathcal{X}}^v)^T \ (\boldsymbol{\mathcal{X}}^n)^T]^T$, $\gamma = \sqrt{(L+\lambda)}$, λ=composite scaling parameter, L=dimension of augmented state, $\mathbf{R}^{\mathbf{v}}$=process noise cov., $\mathbf{R}^{\mathbf{n}}$=measurement noise cov., W_i=weights as calculated in Eqn. 7.34.

Table 7.3.1: Unscented Kalman Filter (UKF) equations

Our focus, however, will be on a simpler scenario where the noise is assumed to be purely additive. Here, the complexity of the UKF reduces greatly. In particular, there is no longer a need for the x_k^a vector, only the x_k. Under this

framework, both the dimensions and the number of calculations that we need to make are reduced.

To begin, initialization is done:

$$\hat{x}_0 = E[x_0]$$

$$P_0 = E[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T]$$

Then, Sigma Points can be calculated:

$$\chi_{k-1} = [\hat{x}_{k-1} \;\; \hat{x}_{k-1} + \gamma\sqrt{P_{k-1}} \;\; \hat{x}_{k-1} - \gamma\sqrt{P_{k-1}}]$$

where $\gamma = \sqrt{L + \lambda}$.

Now, we proceed to do the Projection and Update portions of the algorithm. Beginning with the Projection step, the following equations are needed:

$$\chi_{k|k-1} = F[\chi_{k-1}, \; u_{k-1}]$$

Which projects the sigma points forward to time k and takes into account any input u_k.

$$\hat{x}_k^- = \sum_{i=0}^{2L} W_i^{(m)} \chi_{i,k|k-1}$$

Which gets a prior prediction for x using a weighted average of the sigma points.

$$P_k^- = \sum_{i=0}^{2L} W_i^{(c)} [\chi_{i,k|k-1} - \hat{x}_k^-][\chi_{i,k|k-1} - \hat{x}_k^-]^T + R^v$$

Which gets a prior prediction for P using a weighted covariance along with some process noise, described by R^v. We can see that the calculation of $\hat{x}_k^-$ and P_k^- are both applications of the Unscented Transformation.

$$Y_{k|k-1} = H[\chi_{k|k-1}]$$

Which is the UT transformation to understand the observable values.

$$\hat{y}_k^- = \sum_{i=0}^{2L} W_i^{(m)} Y_{i,k|k-1}$$

Which creates a prior prediction for the observables using a weighted average of Y. Once again, the UT has been applied to understand the distribution. Next is the update steps where the observed data is brought in:

$$P_{\tilde{y}_k, \tilde{y}_k} = \sum_{i=0}^{2L} W_i^{(c)} [Y_{i,k|k-1} - \hat{y}_k^-][Y_{i,k|k-1} - \hat{y}_k^-]^T + R^n$$

This is the variance matrix for $\tilde{y}_k$, which takes into account the measurement noise through R^n.

$$P_{x_k,y_k} = \sum_{i=0}^{2L} W_i^{(c)}[\chi_{i,k|k-1} - \hat{x}_k^-][Y_{i,k|k-1} - \hat{y}_k^-]^T$$

This is the covariance between x and y.

$$K_k = P_{x_k,y_k} P_{\tilde{y}_k,\tilde{y}_k}^{-1}$$

Which calculates the Kalman Gain.

$$\hat{x}_k = \hat{x}_k^- + K_k(y_k - \hat{y}_k^-)$$

This is the posterior prediction of x after now incorporating the observable data.

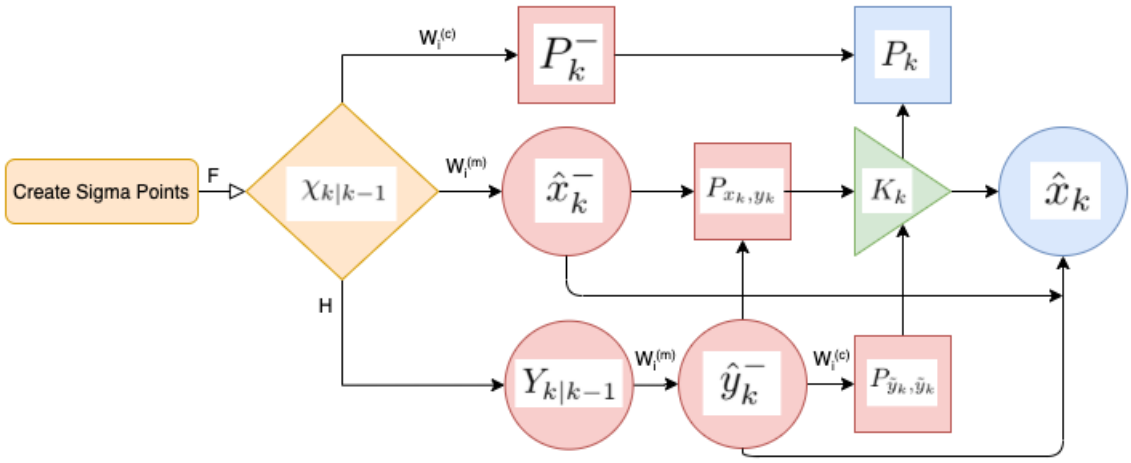
$$P_k = P_k^- - K_k P_{\tilde{y}_k,\tilde{y}_k} K_k^T$$

Which is the posterior variance calculated using the Kalman Gain.

To clear up some notation used here:

- γ - $\sqrt{L + \lambda)}$
- λ - scaling factor
- L - dimension of state vector (i.e. number of states)
- R^v - covariance of process noise
- R^n - covariance of measurement noise
- W_i - the weights calculated earlier
- F is the transition matrix to project the sigma points forward
- H is the measurement matrix to calculate estimates for Y based on the projected forward sigma points
- K is the Kalman Gain Matrix

A flow chart depicting the entire UKF process is below. To summarize, our goal is to begin with a set of sigma points chosen in a deterministic fashion and arrive at the blue shapes, which are posterior estimates for the states and covariance. The portions in between, shaded in red, describe all of the intermediary steps that are needed to achieve this goal. Finally, the Kalman Gain, due to its significance, is shaded in green.



3.4 State Estimation Example

As an example, consider a "double inverted pendulum control system". The system has the following states associated with it:

- Position x
- Velocity $\dot{x}$
- Top pendulum angle θ_1
- Top pendulum angular velocity $\dot{\theta}_1$
- Bottom pendulum angle θ_2
- Bottom pendulum angular velocity $\dot{\theta}_2$
- The states are all in vector x

as well as the following parameters:

- l_1 - length of pendulum 1
- l_2 - length of pendulum 2
- m_1 - mass of pendulum 1
- m_2 - mass of pendulum 2
- M - mass of the cart
- The parameters are all in vector w

The following equations relate elements in the system to one another:

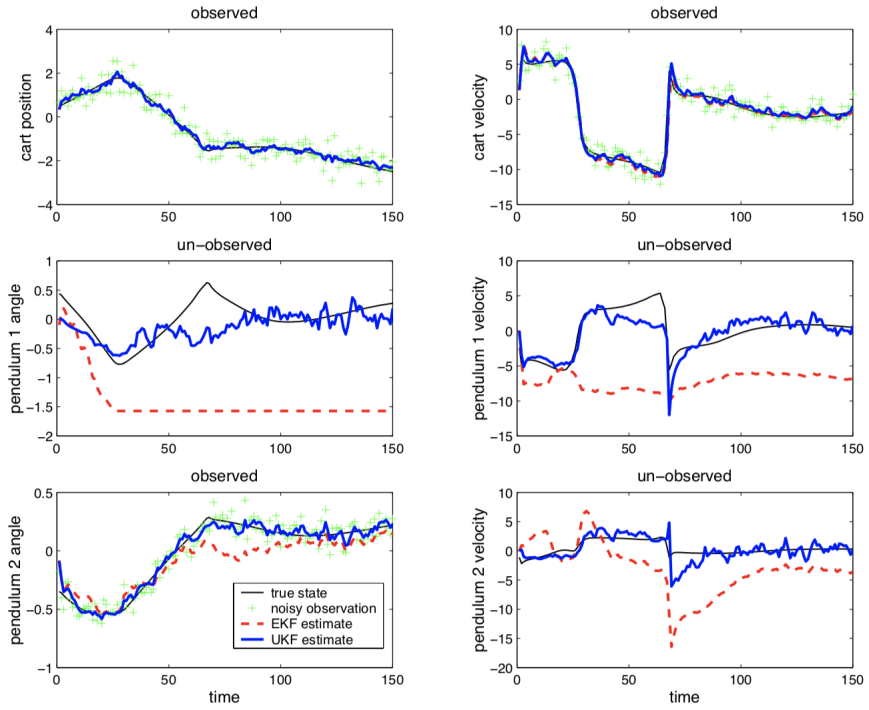
$$\begin{aligned}(M + m_1 + m_2)\ddot{x} - (m_1 + 2m_2)l_1\ddot{\theta_1}\cos\theta_1 - m_2 l_2\ddot{\theta_2}\cos\theta_2 \\ = u + (m_1 + 2m_2)l_1\dot{\theta_1}^2\sin\theta_1 + m_2 l_2\dot{\theta_2}^2\sin\theta_2\end{aligned}$$

$$\begin{aligned}-(m_1 + 2m_2)l_1\ddot{x}\cos\theta_1 + 4(\frac{m_1}{3} + m_2){l_1}^2\ddot{\theta_1} + 2m_2 l_1 l_2\ddot{\theta_2}\cos(\theta_2 - \theta_1) \\ = (m1 + 2m_2)gl_1\sin\theta_1 + 2m_2 l_1 l_2\dot{\theta_2}^2\sin(\theta_2 - \theta_1)\end{aligned}$$

$$\begin{aligned}-m_2\ddot{x}l_2\cos\theta_2 + 2m_2 l_1 l_2\ddot{\theta_1}\cos(\theta_2 - \theta_1) + \frac{4}{3}m_2{l_2}^2\ddot{\theta_2} \\ = m_2 g l_2\sin\theta_2 - 2m_2 l_1 l_2\dot{\theta_1}^2\sin(\theta_2 - \theta_1)\end{aligned}$$

To discretize the system, measurements are taken every 0.02 seconds. An outside force, u, which appears in the equations, is also applied to the cart. The noisy observations that are made, which would be the y_k, are of the cart position, cart velocity, and angle of pendulum two. This leaves the pendulum one angle, pendulum one velocity, and pendulum two velocity as the unobserved states.

For initialization, the pendulums are placed in a jackknife position (+25/-24 degrees) and the cart is offset by 0.5 meters. A figure depicting the ability of the EKF and UKF to estimate the states, particularly the unknown ones, is below. Clearly, the UKF performs much better at predicting the unknown states.



4 The Lorenz System

The Lorenz System is a well known dynamical system which results in data that gives it its common name, the "butterfly effect". The system is made up of three state variables, x_1, x_2, and x_3 modeled by the following equations:

$$\dot{x}_{1t} = \sigma(x_{2t} - x_{1t})$$

$$\dot{x}_{2t} = \rho x_{1t} - x_{2t} - x_{1t}x_{3t}$$

$$\dot{x}_{3t} = x_{1t}x_{2t} - \beta x_{3t}$$

where σ, ρ, and β are our model parameters. The measurement error covariance is an additional parameter and is contained within Θ. Our goal is to apply the UKF in order to do state estimation for the Lorenz System

4.1 Starting Code

We began with code from Chow-Ferrer's 2005 UKF paper. This MATLAB code performed both state and parameter estimation through the following two steps:

- The Θ parameters are estimated using approximate-ML with prediction error decomposition

- Using the final estimates of Θ from above, the joint vector of states and parameters, defined as $x_t = [x_{1t}, x_{2t}, x_{3t}, \sigma_t, \rho_t, \beta_t]^T$ was estimated using the UKF

Our goal was to modify this code to use fixed, known parameter values throughout and instead focus only on state estimation.

4.2 Code Modification

In order to modify the existing code, the following modifications needed to be made:

- The vectors and matrices which represented the quantities to be estimated needed to be changed from being based on 6 values (3 states and 3 parameters) to only 3 values (the 3 states)
- Parameter values now needed to be held constant. This means that the true parameter values listed in Chow were input into the code and used throughout the state estimates, instead of having them vary over time and be themselves estimated
- Code related to parameter fitting was removed in order to increase readibility

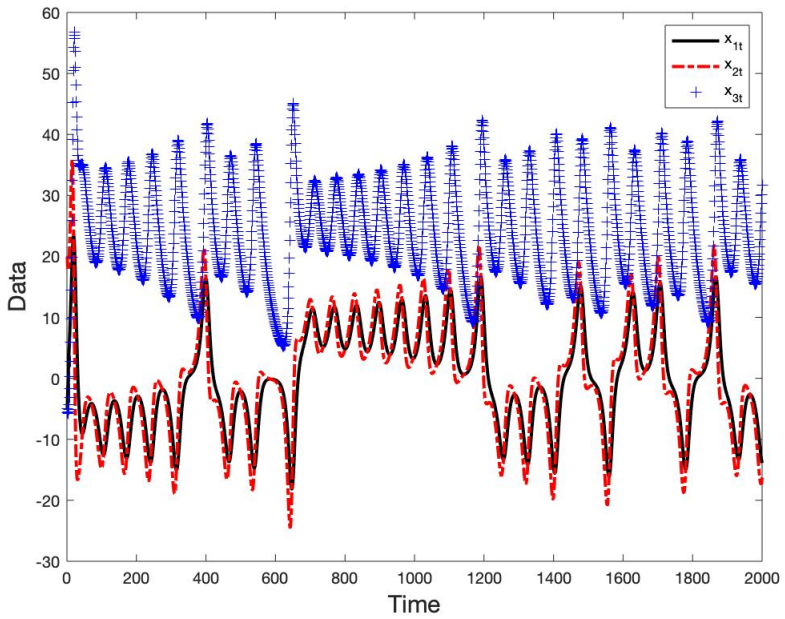
The following pseudocode resembles the general process that the code now caries out:

```
Set runtime and system parameters
Loop for n = 1:num_simulations
    Generate data with noise
        Use Runge-Kutta to solve ODE system
    Initialize states and covariance with guesses for t = 0
    Set model parameters as true known values
    Create InfDS, data structure of model information
    Loop UKF for t = 1:2000
        Perform time update
        Perform measurement update
    End
    Create figures for estimates
    Calculate error
    Calculate norm of error
    Plot error
End
```

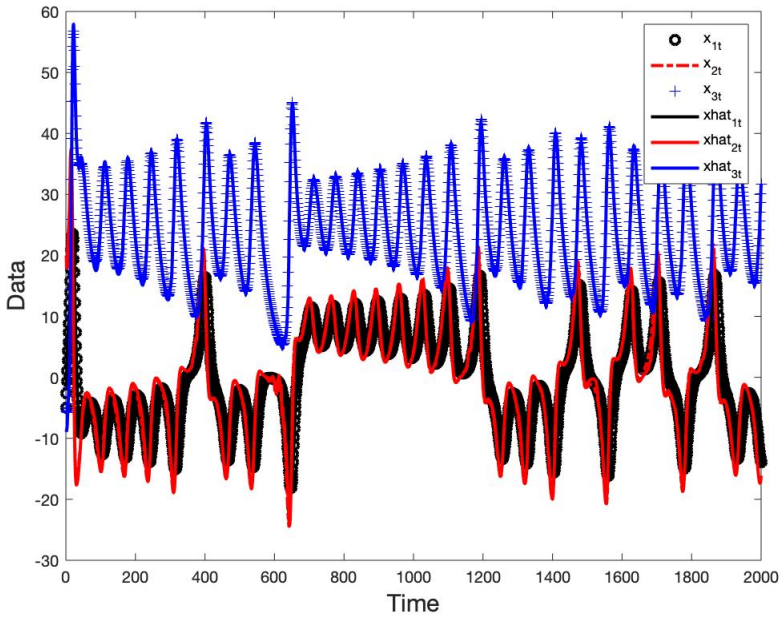
We can now look at how well this code does at doing state estimation.

4.3 Analysis

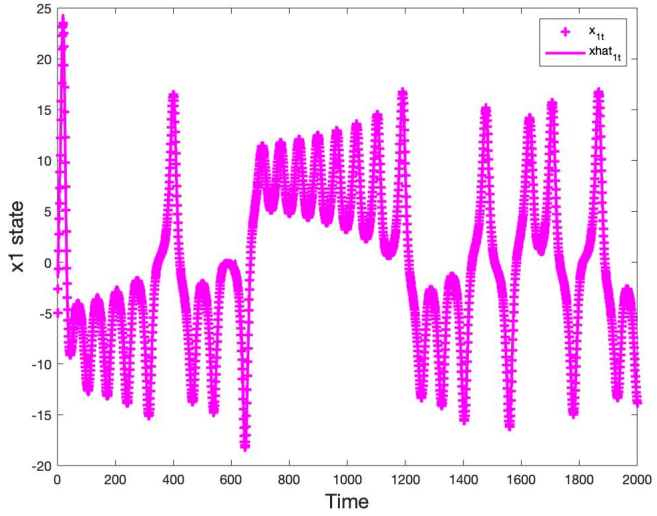
Our goal is to do state estimation that smooths out noise (i.e. creates predictions that are less jaggedy)***** and produces estimates that closely resemble the generated data. To begin, let's consider the generated data, which is created using the Runge-Kutta approach:

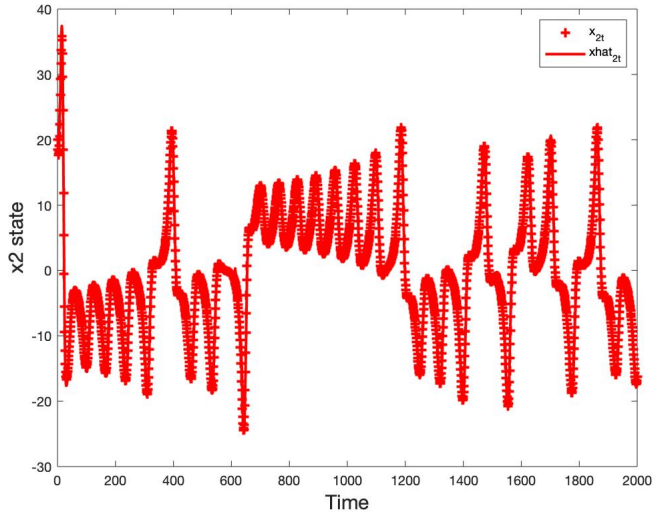


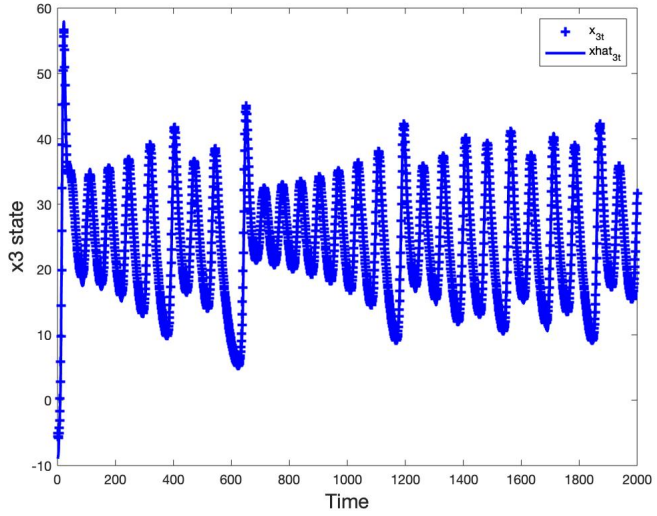
Now, here we see simply the generated data, without any state estimation being done. To visualize the effectiveness of our estimates, we can now overlay the estimates as lines over this data to produce the following figure:



Here we can see that the estimation from the UKF method very closely lines up with the generated data. If we wish to zoom in on a specific state specifically, we can graph the generated and estimated data for a single state as well:







These graphics validate that our code is functioning properly and is making feasible estimates. However, it is now useful to be able to quantify precisely how well the code is doing by looking at the error, $\tilde{x} = x - \hat{x}$. We will then, in the future, be able to compare this code to other UKF implementations.

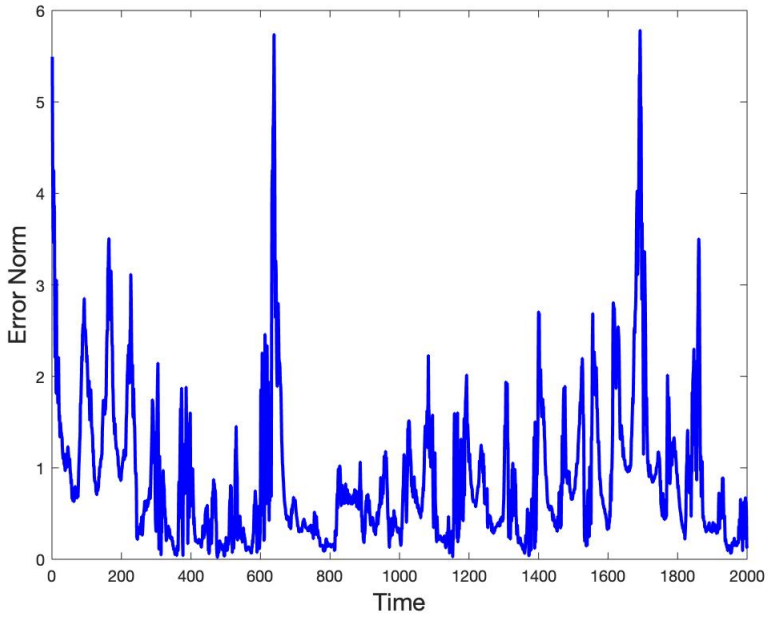
4.4 Quantifying Error

To understand the error, we look at $\tilde{x}$. In this context, this will be a matrix where, for each time point, the column will be the error for states 1, 2, and 3 respectively. However, having three separate values of error is not ideal. Instead,

we'd like to have a single value to quantify the error at each time point. In order to achieve this, we use the norm of the $\tilde{x}_t$ vector, calculated as:

$$norm_t = \sqrt{\tilde{x}_1^2 + \tilde{x}_2^2 + \tilde{x}_3^2}$$

This now gives us a single value for each time that represents the error of our estimates. Ideally, this error should decrease over time. To visualize the error, consider the following figure:



Here, we can see a general downward trend in error. However, there are various peaks, most notably at $Time \approx 700$ and $Time \approx 1700$. To understand why these are occurring, it is useful to look back at the figures which showed the overlay of the raw data and the UKF estimates. There, it is evident that the times at which the error spikes map to the times when the general direction of the states change (i.e. transitions from increasing to decreasing or visa versa). Thus, these are the points where the UKF is currently struggling the most. The algorithm appears to expect the trajectory to stay the same and is unable to accurately predict when it no longer will. Decreasing the error at these points will be crucial moving forward.

5 Moving Forward

This week, our main focus has been on understanding and utilizing the Unscented Kalman Filter for state estimation. We have successfully modified Chow-Ferrer's code to do purely state estimation on the Lorenz system, and

will now seek to apply this code in order to do state estimation for a Predator-Prey model. Our hope is that doing state estimation on a Predator-Prey model moves us closer to our eventual goal of applying UKF's to the T1D model. Moreover, we will now work to benchmark how well our current state estimation code is working by comparing its performance to the built in Matlab UKF subroutine. This will give us both a better understanding of the UKF as well as understand in what areas we need to make improvements to our code.