Stability, Unscented Kalman Filters, Lorenz Transformations

Subteam 2

Stability Issues with Kalman Filters

From last time, P_k defined as $P_k^- - G_k H_k P_k^-$

Issue: may not be positive semi definite.

Solution: Choletsky factorization at each step: $P_k = P_k^{1/2} P_k^{T/2}$

Unscented Kalman Filter (UKF)

- Kalman Filter for Nonlinear Systems
- Three main applications
  - State Estimation: The UKF is used to estimate states of a system
  - Parameter Estimation: The UKF is used to estimate the parameters of a system
  - Dual Estimation: The UKF is used to estimate both the states and the parameters of the system
    - Joint Method
    - Dual Method

Optimal Recursive Estimation (Review of Last Week)

To estimate state x_k given sequence of observations Y_0^k, optimal estimate for the state is

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

We can think of this process recursively, which gives us the state at time *k*, as

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

And the observable data at time *k*, as

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

Optimal Recursive Estimation assuming Gaussian Densities

Assuming Gaussian densities, only calculations that need to be done are

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

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

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

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

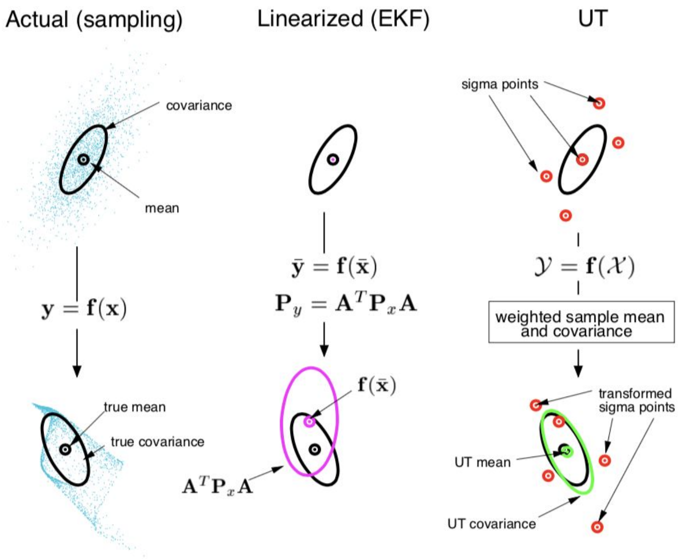
$$\hat{y}_k^- = E[H(\hat{x}_k^-, n_k)]$$

This looks reminiscent of last week!

- For linear system, easy to do with Linear Kalman Filter
- Nonlinear system
  - Can linearize locally with Taylor Expansions (EKF)
    - Does not fully capture distribution
- Solution: the Unscented Kalman Filter

Unscented Transformation (UT)

Using Sigma Points, understand the distribution of a random variable after a non-linear transformation.



Unscented Transformation

Consider the random variable x with a dimension of L and a non-linear transformation $y = f(x)$. To understand the distribution of y, we create a matrix of $2L+1$ sigma vectors as follows:

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

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

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

Unscented Transformation Parameters

Parameter $\lambda := \alpha^2(L + k) - L$.

α and k user-defined scaling parameters.

Unscented Transformation: Transformation Step

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, it is thus $H(\chi_i)$.

Unscented Transformation: Calculating mean and covariance

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):

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

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

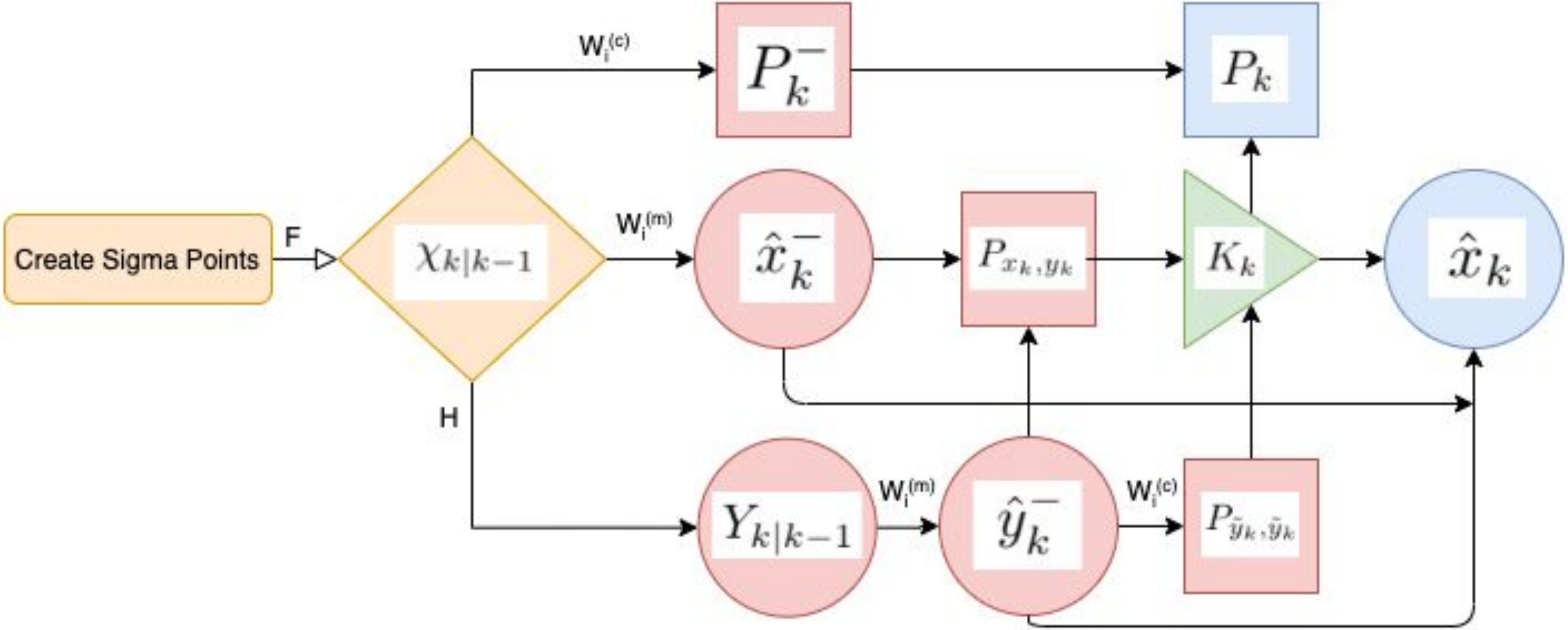
$$W_i^{(m)} = W_i^{(c)} = 1/2(L+\lambda) \ i = 1...2L$$

Where β holds prior knowledge about the distribution (for Gaussian $\beta = 2$ is used). Now the mean and covariance can be calculated:

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

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

The Entire UKF Process



UKF Initialization

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:

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

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

UKF Projection Step

Project sigma points:

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

Project states:

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

Project covariance:

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

Perform UT transformation:

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

Use transformation to predict observables:

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

UKF - Update Step

The covariance matrix for $\tilde{y}_k$:

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

Covariance between x and y:

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

The Kalman Gain:

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

Final prediction of states:

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

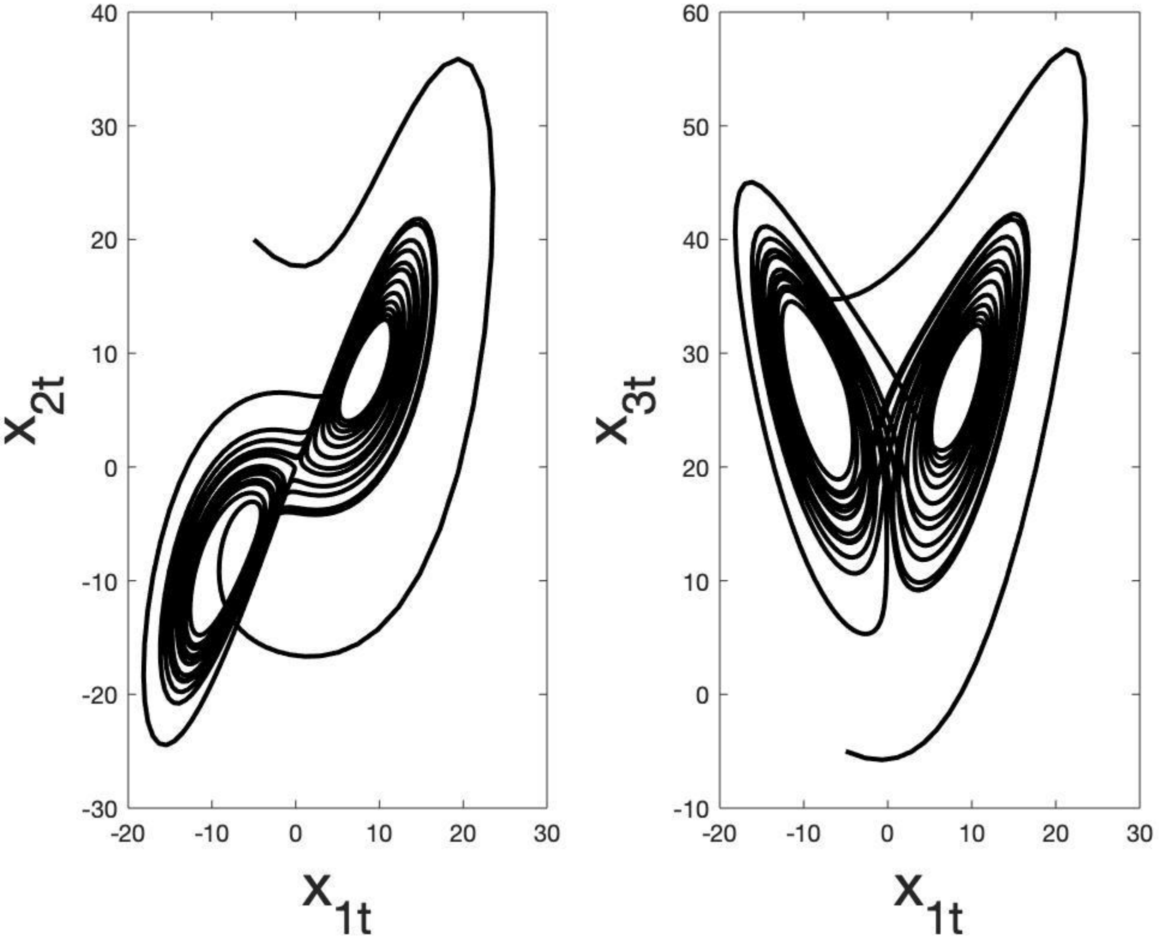
Final prediction of covariance:

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

The Lorenz System

- Butterfly effect
- Used for weather systems

$$
\begin{aligned}
\dot{x_{1t}} &= \sigma(x_{2t} - x_{1t}), \\
\dot{x_{2t}} &= \rho x_{1t} - x_{2t} - x_{1t}x_{3t} \text{ and} \\
\dot{x_{3t}} &= x_{1t}x_{2t} - \beta x_{3t},
\end{aligned}
$$


40
30
20
10
0
-10
-20
-30
-20
-10
0
10
20
30
x2t
x1t
60
50
40
30
20
10
0
-10
-20
-10
0
10
20
30
x3t
x1t

Parameters of Lorenz System

Parameters	True values
σ	10
ρ	28
β	2.667
Θ	$\text{diag}\begin{bmatrix}26\\34\\32\end{bmatrix}$

Measurement Error Covariance

The Original Code

- Taken from Chow-Ferrer 2005
- Parameter and state estimation
- First estimate ϴ using ML approach
- Then estimate vector $[x_{1t}, x_{2t}, x_{3t}, \sigma_t, \rho_t, \beta_t]$ with joint UKF

Goals for Modified Code

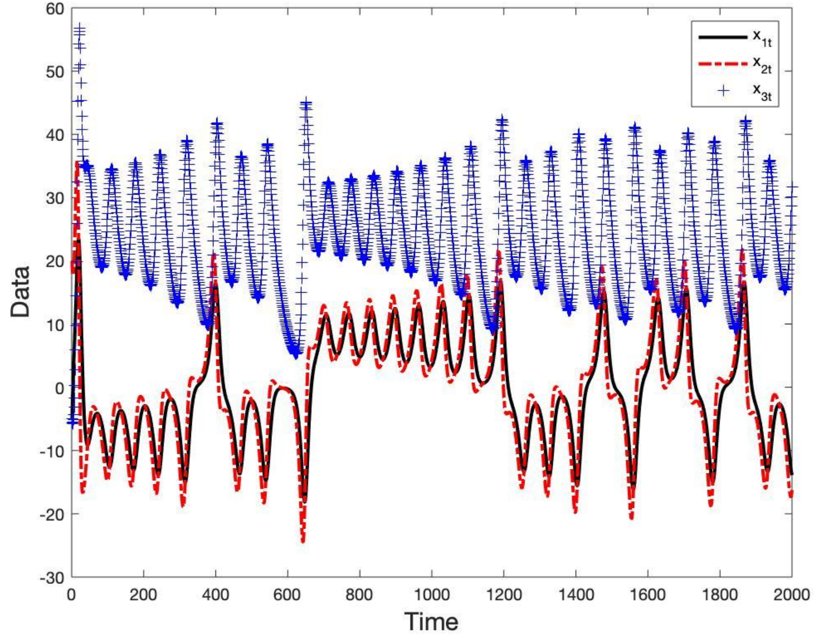
- Perform ONLY state estimation
- Keep parameter values constant
- Understand accuracy of the estimates

Pseudocode

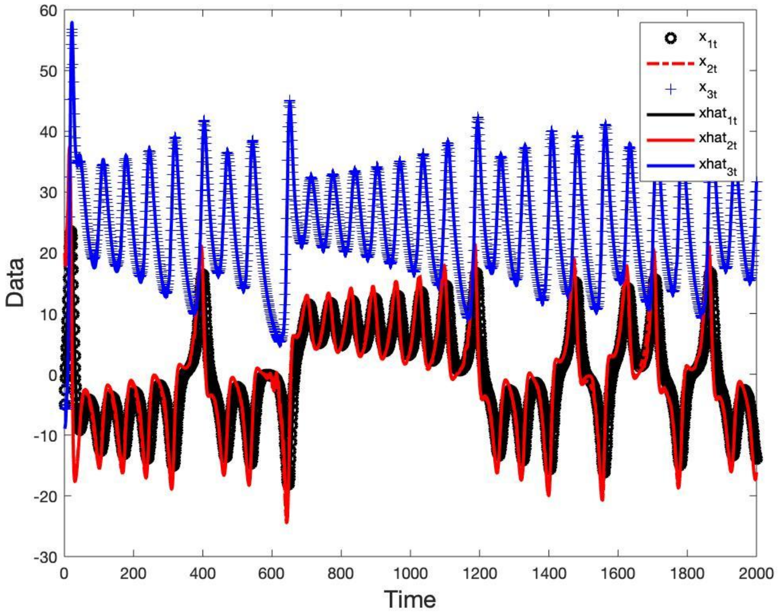
```
Set runtime and system parameters
Loop for n = 1:num_simulations
        Generate data with noise
                Use Runge-Kutta to solve ODE system
        Initialize states and covariance with guesses for t = 0
        Set model parameters as true known values
        Create InfDS, data structure 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
```

Visualizing State Estimation

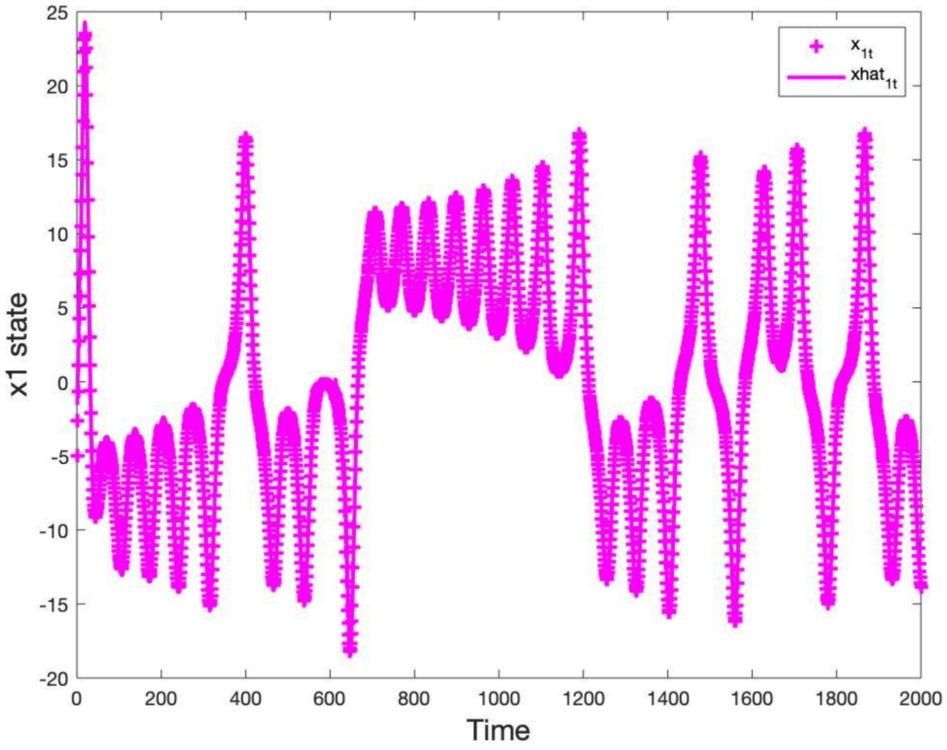
Generated Data



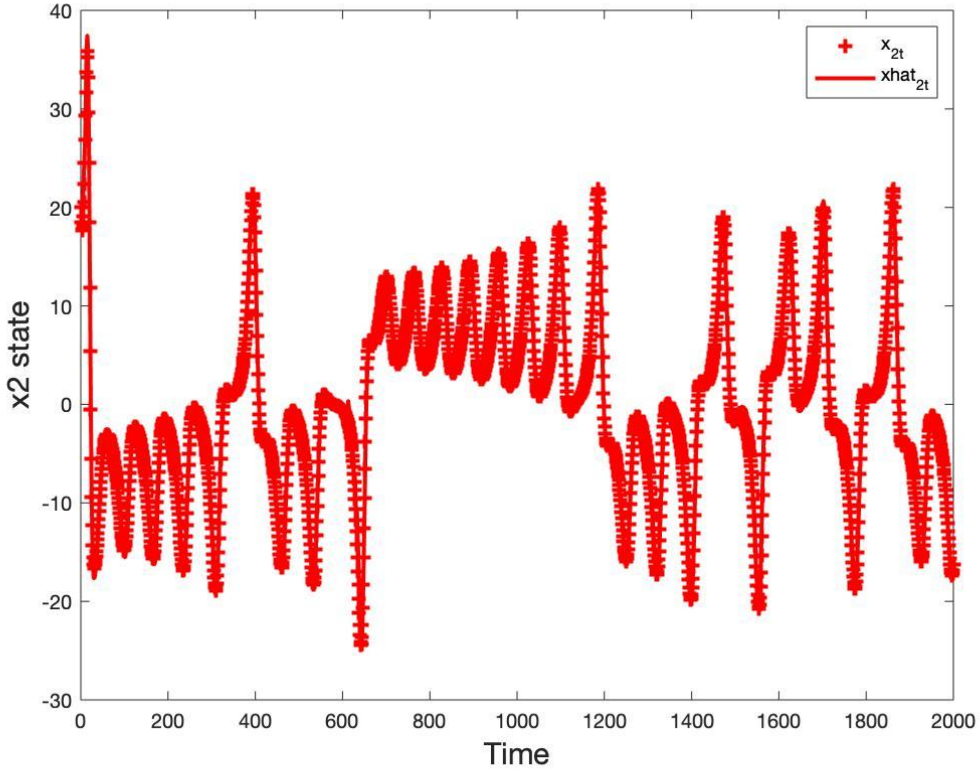
Generated + Predicted Data



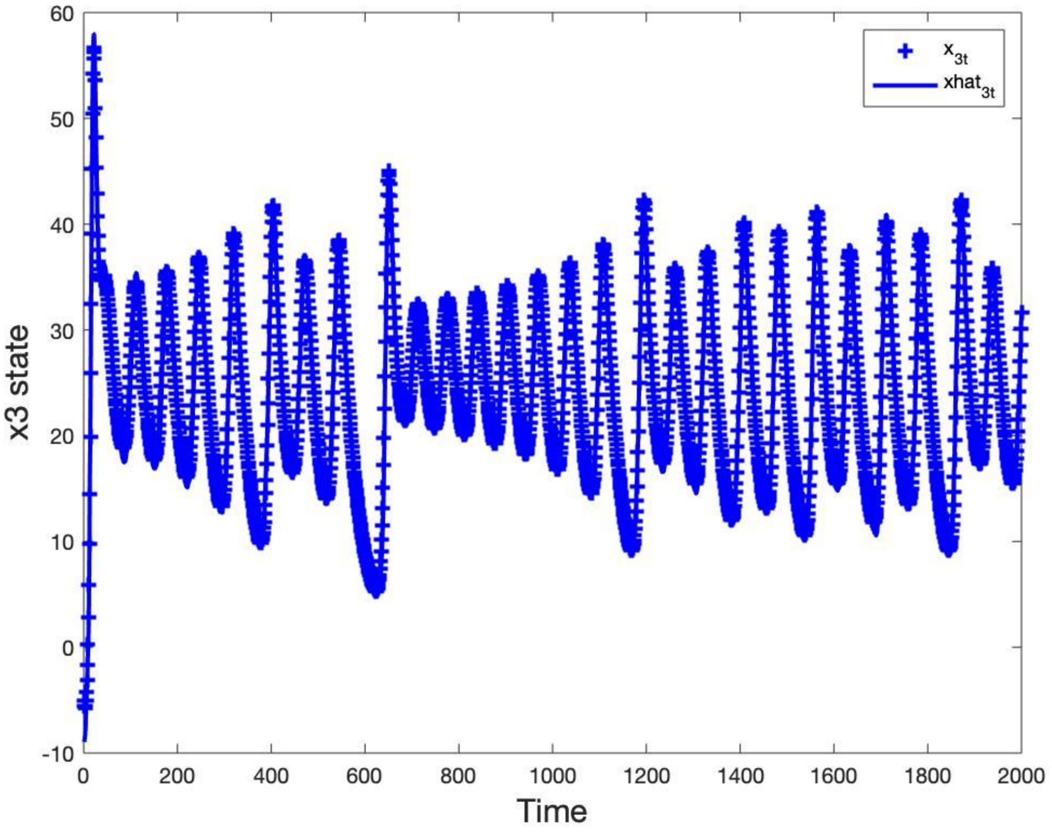
X_1 State Estimation Overlay



X_2 State Estimation Overlay



X_3 State Estimation Overlay



Quantifying Error of Estimates

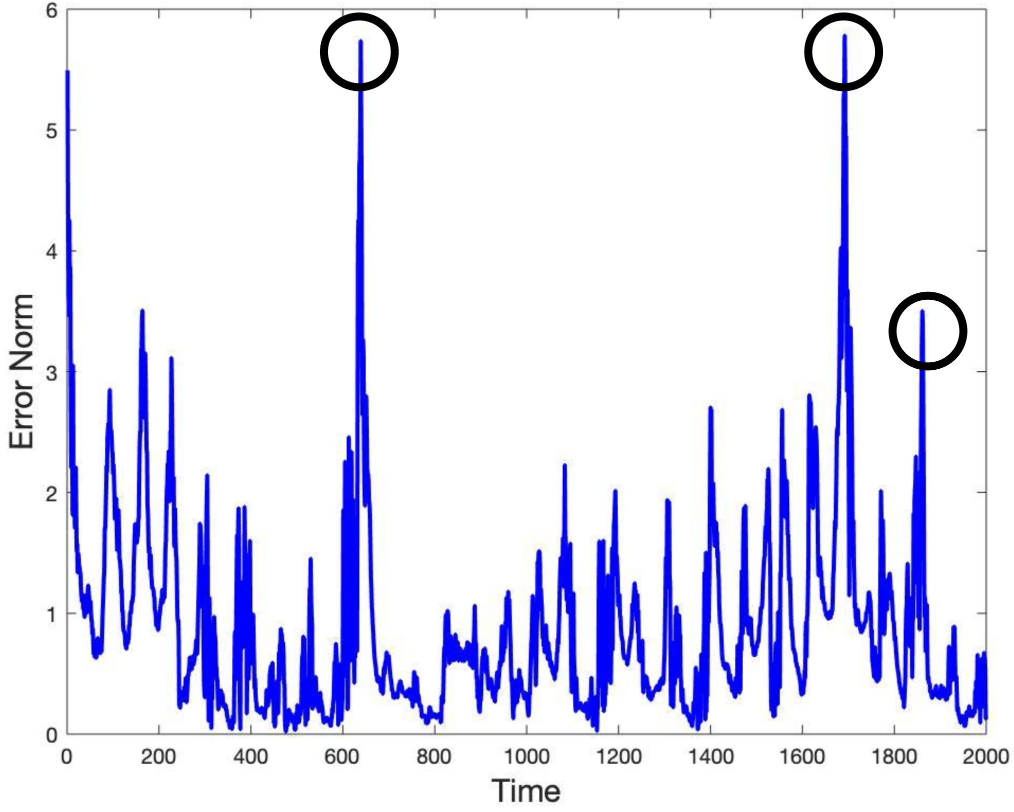
From last time, quantify error as $\tilde{x} = x - \hat{x}$

To understand error, first calculate $\tilde{x}$

Problem: $\tilde{x}$ is a **vector**. Solution: Quantify error as $\mathbf{norm}(\tilde{x})$. Which is to say:

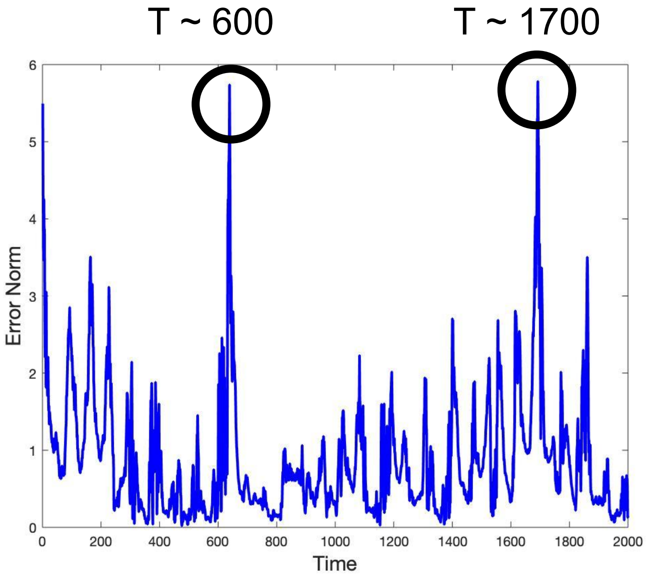
$$error = \sqrt{\tilde{x}_1^2 + \tilde{x}_2^2 + \tilde{x}_3^2}$$

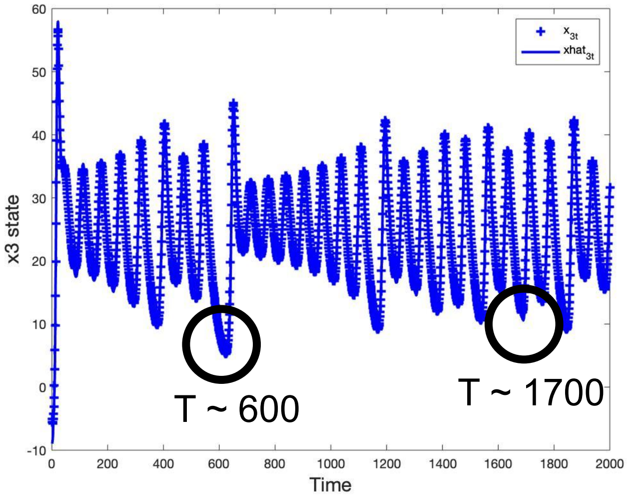
If we do this calculation for each time, we can generate:



Analyzing the Error Trends

- Overall downward trend → more data reduces error
- Spikes in error at times of state value derivative sign changes





Guiding Questions

- How can the UKF predict changes in sign of slope of the states?
- How can this code be applied to a Predator Prey model?
- How will parameter estimation be done at the same time?
- How can we apply these codes to a diabetes model?