Matlab UKF and Predator-Prey Model

Subteam 2

Reminder: The Lorenz System

- Butterfly effect
- Used for weather systems

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

Reminder: Parameters of Lorenz System

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

Measurement Error Covariance

The Base Code

- Taken from Albers T2 Diabetes Model
- State estimation using built in Matlab ukf
- Individual functions taken from Chow code
    - Lorenz System
    - LordynO (Transition function)
    - LormeasO (Measurement function)
- Parameter values taken from Chow code
    - Alpha: 10e-4
    - Beta: 2
    - Kappa: 0

Goals for Modified Code

- Perform state estimation on Lorenz System
- Compare overall results to built from scratch ukf (seen last week)
- Understand how Matlab UKF function works

Matlab UKF Function

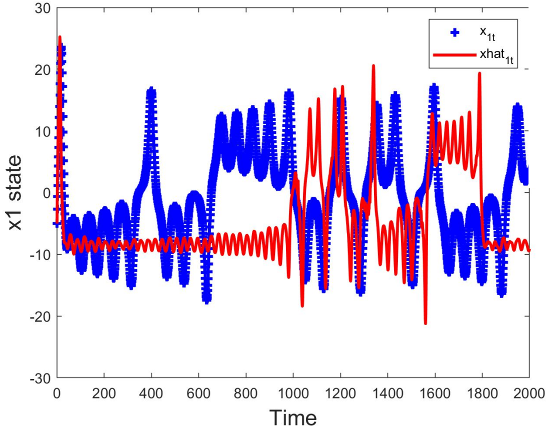
Arguments and Tunable Parameters:

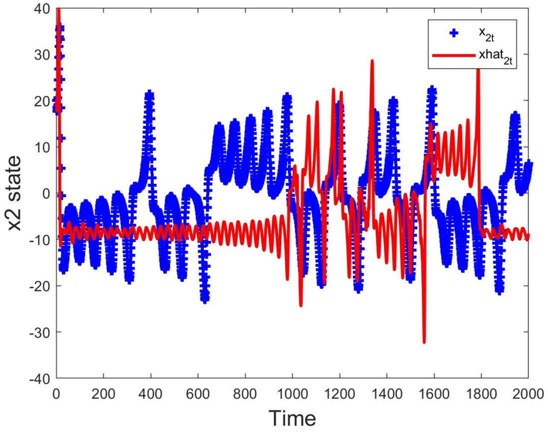
- StateTransitionFcn (taken from Chow)
- MeasurementFcn (taken from Chow)
- InitialState (taken from Chow)
- ‘HasAdditiveMeasurementNoise’ - must be true or false, in our case true
- ‘HasAdditiveProcessNoise’ - same as above
- ProcessNoise (taken from Chow)
- MeasurementNoise (taken from Chow)
- Alpha, Beta and Kappa (taken from Chow)

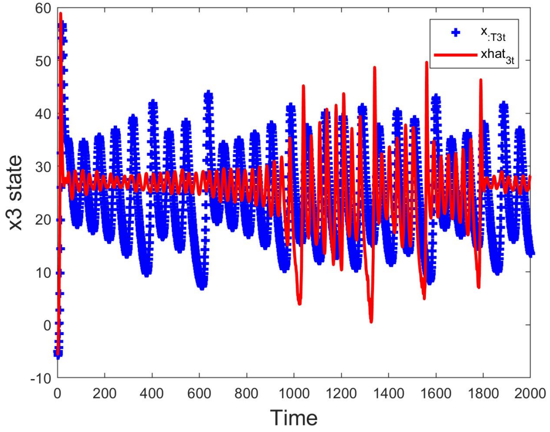
Matlab UKF Function

- Creates a ukf object
- Can find state with [objname].State
- Can find covariance matrix with [objname].StateCovariance
- Can update state using predict and correct functions
- Predict function:
    - Takes arguments: object and input (0 in our case)
- Correct Function
    - Takes arguments: object, measurement and input (0 in our case)

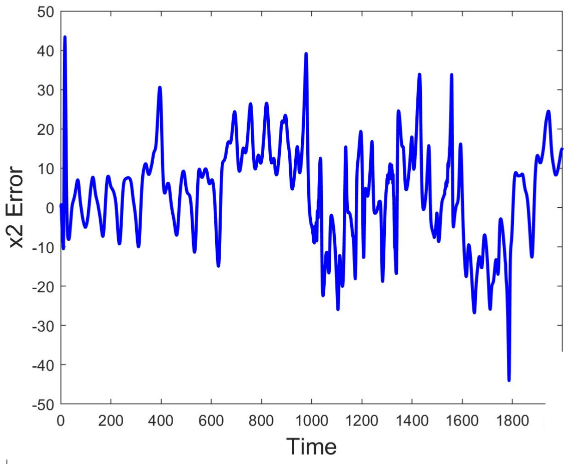
Graphs of State Estimation vs Generated data

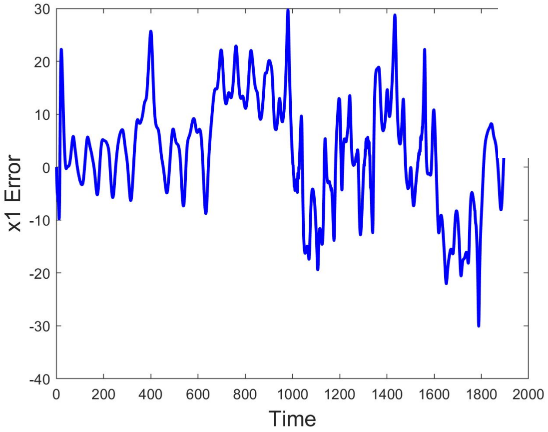


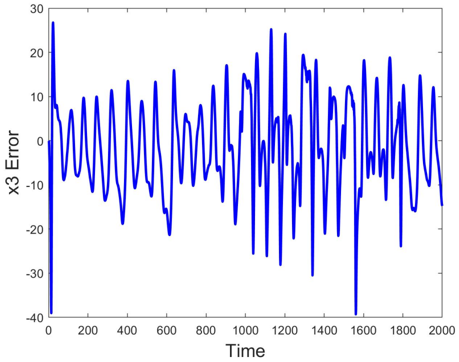




Error for State Estimation







Conclusion

- The Matlab unscented Kalman Filter function doesn’t always perform well or consistently
- It is often a better choice to use unscented kalman filter functions that you build yourself

Lotka-Volterra Predator-Prey Model

The system is described by the following two differential equations:

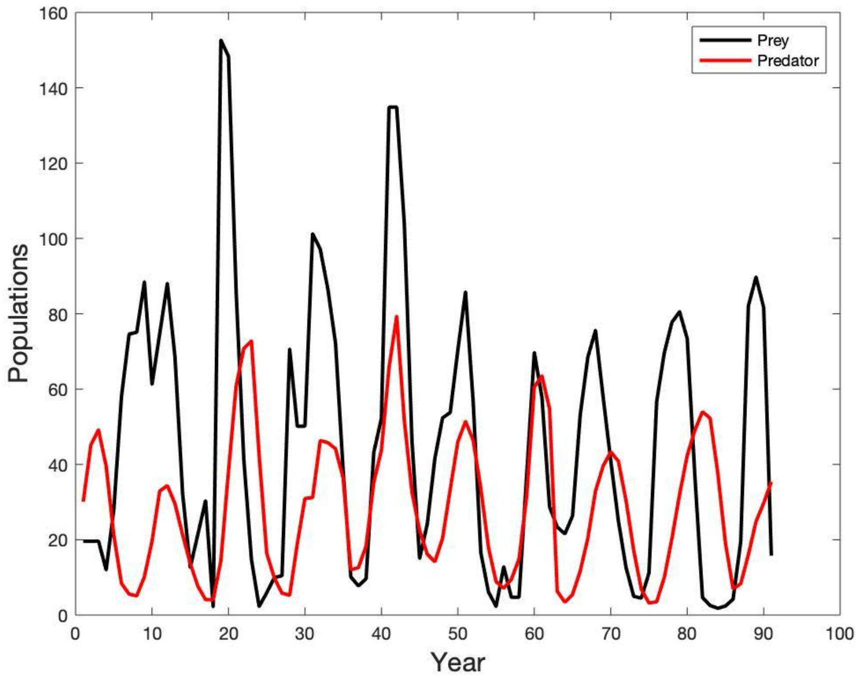
$$\dot{x}_{1t} = \alpha x_{1t} - \beta x_{1t} x_{2t}$$

$$\dot{x}_{2t} = -\gamma x_{2t} + \delta x_{1t} x_{2t}$$

Where $\alpha, \beta . \gamma, \delta$ are the model parameters.

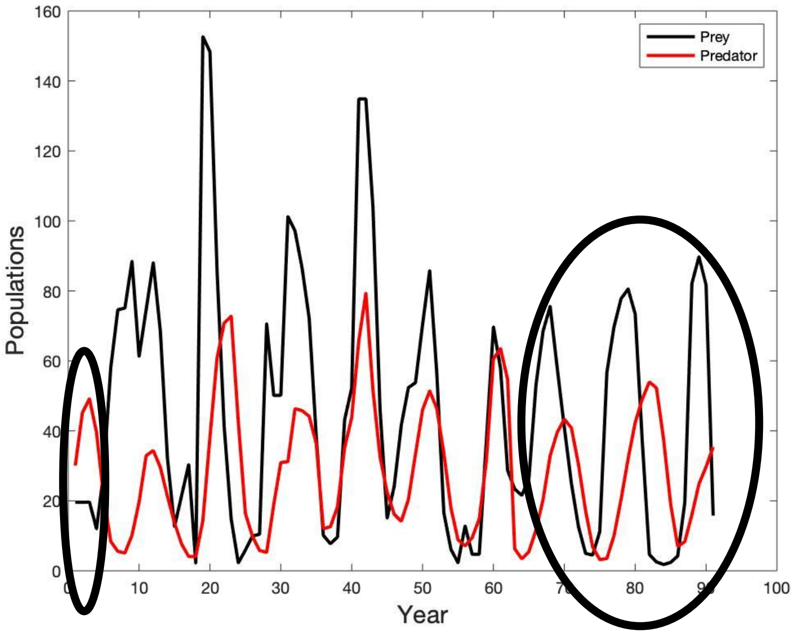
Our Data

- Hares-Lynx populations from 1845 to 1935
- Hares - prey
- Lynx - predator



Assessing Data "Niceness"

- Data set has significant noise, but predator-prey relationship is clear
- Early Prey behavior is strange
- Final ~30 years are very clean



Creating 3 Datasets of Interest

1. Full dataset
2. Years 1908-1935 (the very clean data)
3. Years 1850 - 1935 (eliminates early prey behavior)

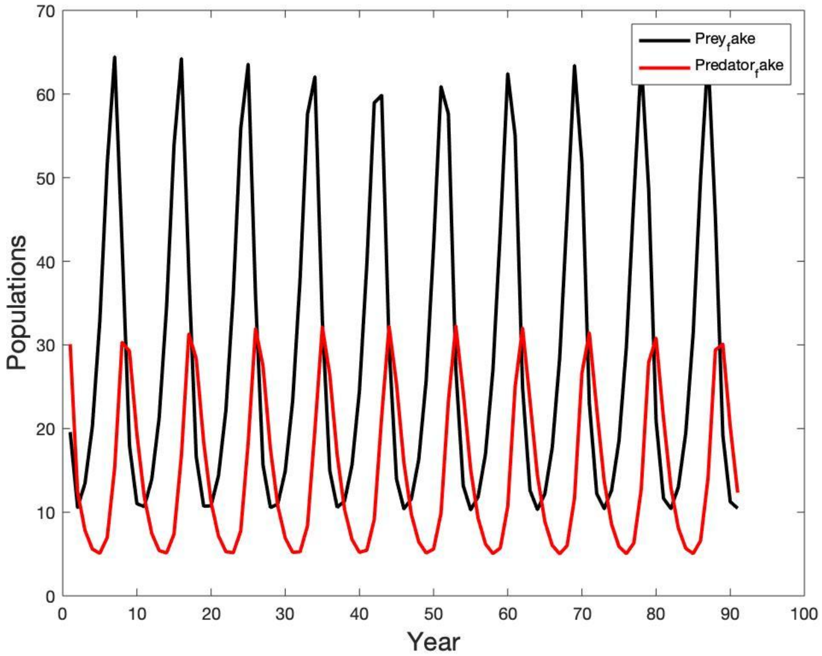
Model Parameters

- Estimated using Particle Swarm Least Square approach for each subset

	Full Dataset	1908-1935	1850-1935
α	.7436	.5969	.6259
β	.0507	.0413	.1067
δ	.7669	.6967	.7229
γ	.0259	.0202	.0158

Benchmarking with Fake Data

- Used full dataset parameters

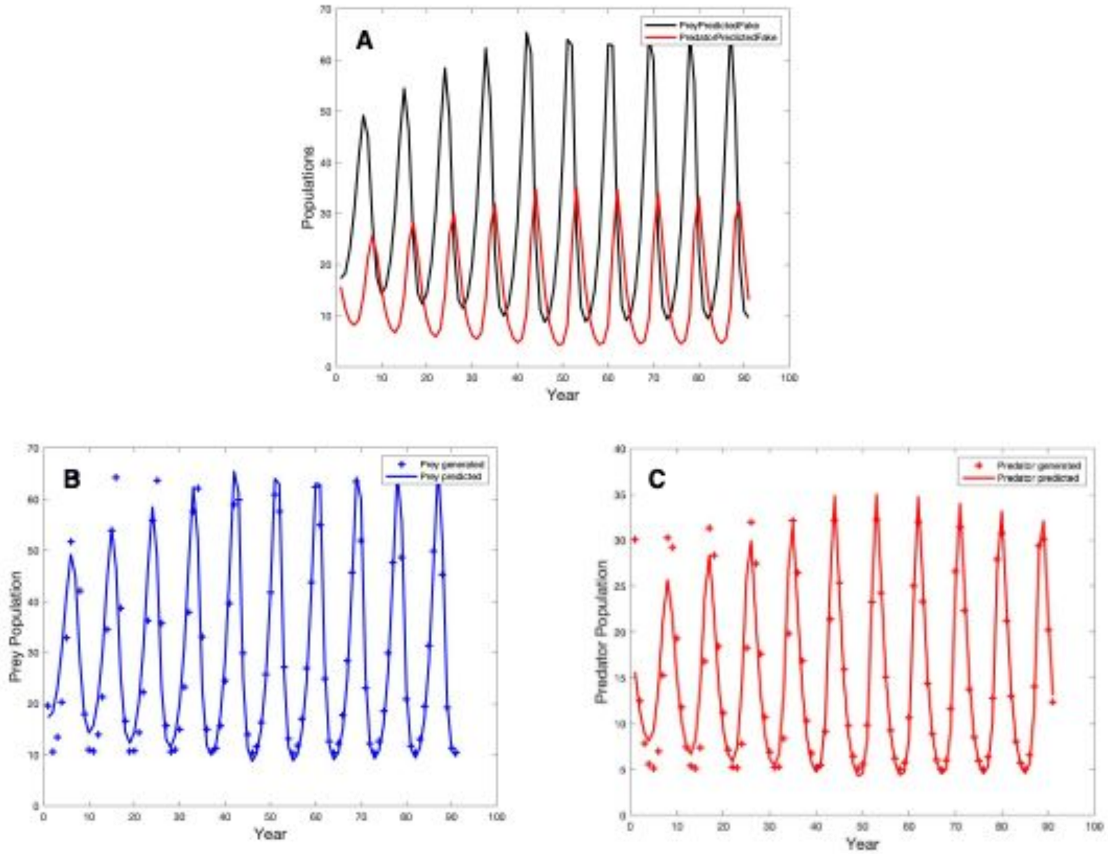


Fake Data Results

A - Predictions

B - Prey prediction overlay

C - predator prediction overlay

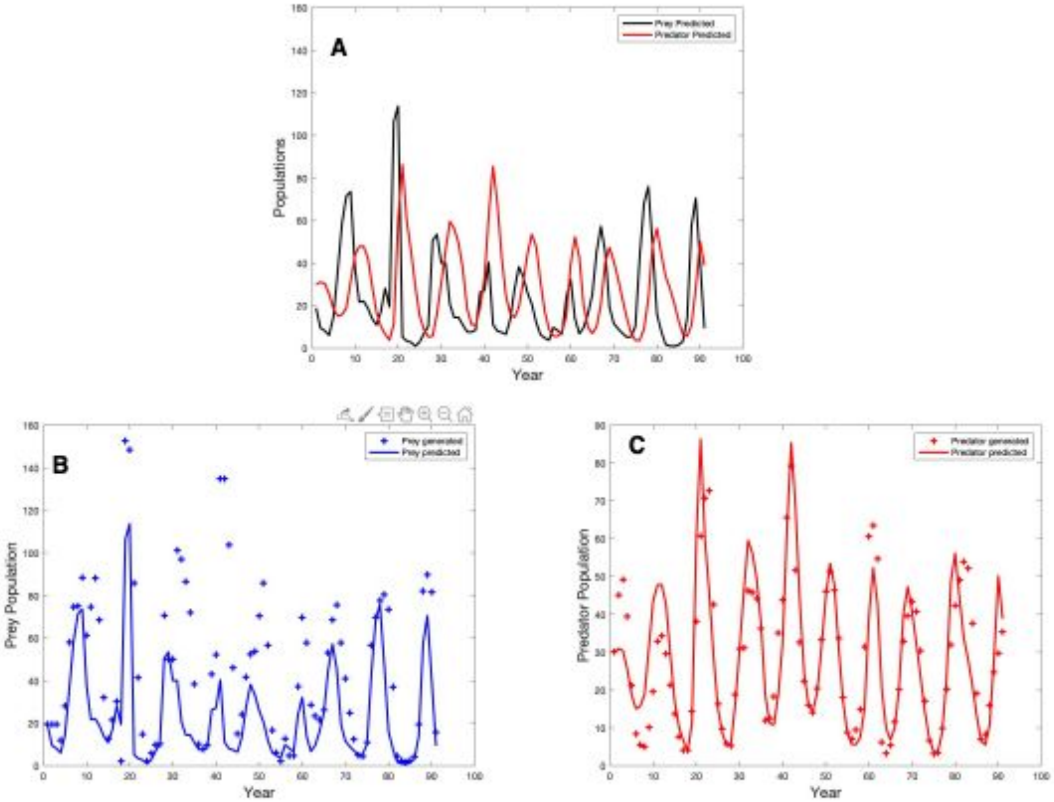


Full Dataset Results

A - Predictions

B - Prey prediction overlay

C - predator prediction overlay

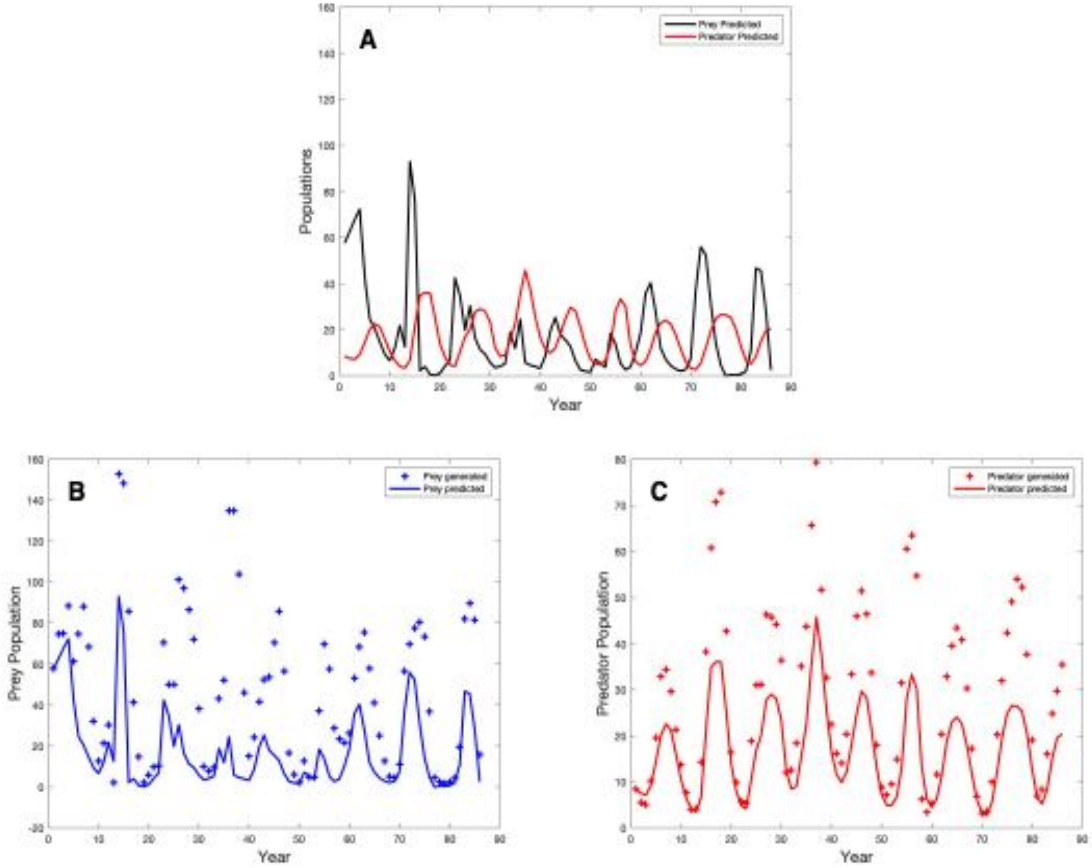


1850-1935 Results

A - Predictions

B - Prey prediction overlay

C - predator prediction overlay

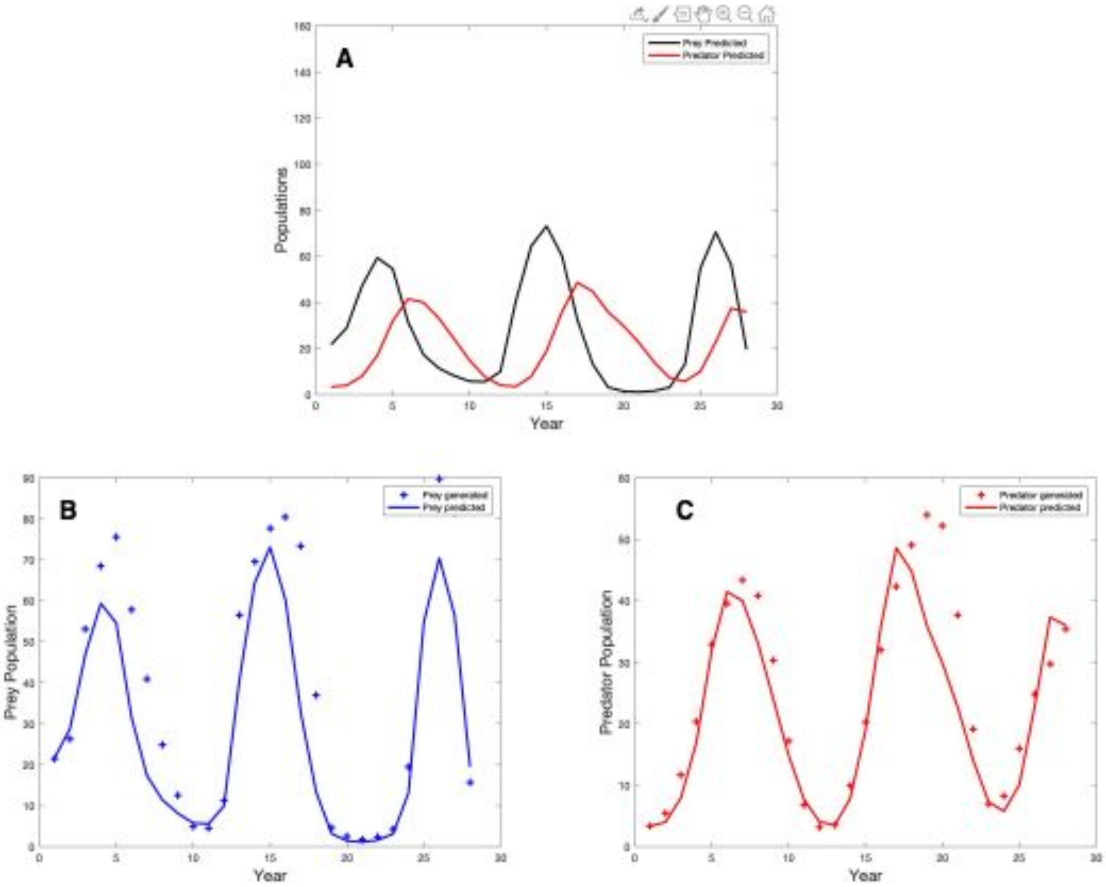


1908-1935 Results

A - Predictions

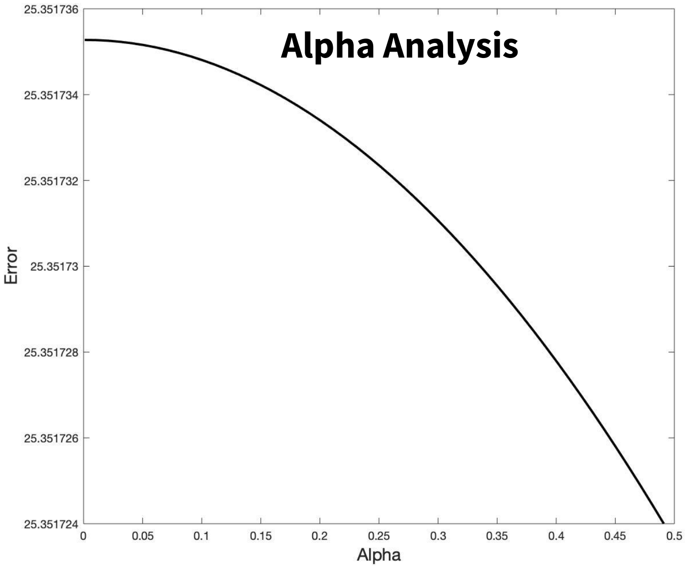
B - Prey prediction overlay

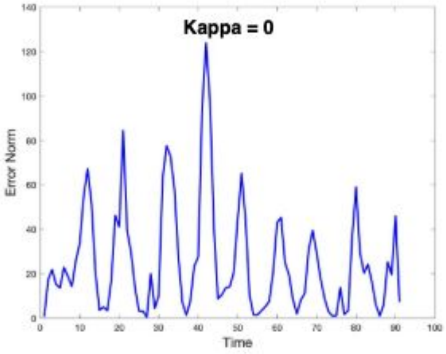
C - predator prediction overlay

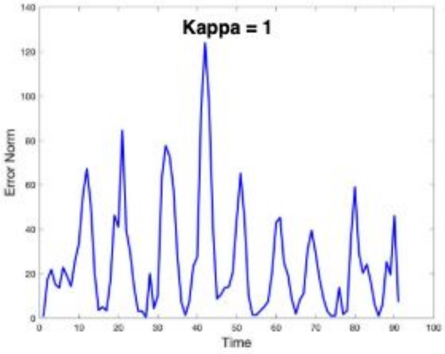


UKF Parameters and Sensitivity Analysis

- κ - from literature, is 0 or 3 - L where L is number of states
- α - [10e-4, 1] scaling parameter







Joint Parameter Estimation

- Joint Parameter Estimation is a form of dual estimation where both the states and the parameters are estimated
- Joint Parameter estimation functions by placing both states and parameters in one large state vector and running it through an Kalman Filter, in our case an Unscented Kalman Filter

$$\begin{bmatrix} \mathbf{x}_{k+1} \\ \mathbf{w}_{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{F}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k) \\ \mathbf{I}\mathbf{w}_k \end{bmatrix} + \begin{bmatrix} \mathbf{B}v_k \\ \mathbf{r}_k \end{bmatrix}. \tag{7.94}$$

$$y_k = [1 \quad 0 \quad \dots \quad 0] \begin{bmatrix} \mathbf{x}_k \\ \mathbf{w}_k \end{bmatrix} + n_k, \tag{7.95}$$

The State Vector

Assume we want to estimate 3 states and 2 parameters, then the state vector would look as follows:

$$[x_1 \ x_2 \ x_3 \ p_1 \ p_2]$$

Thus, the parameters are now thought of as STATES, meaning they will be predicted OVER TIME.

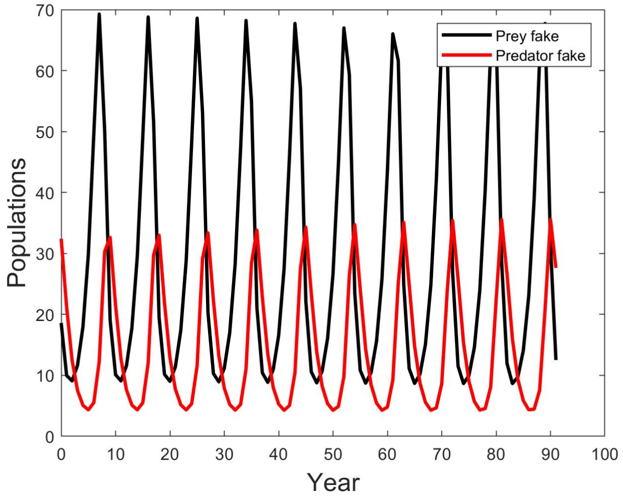
Starting Code

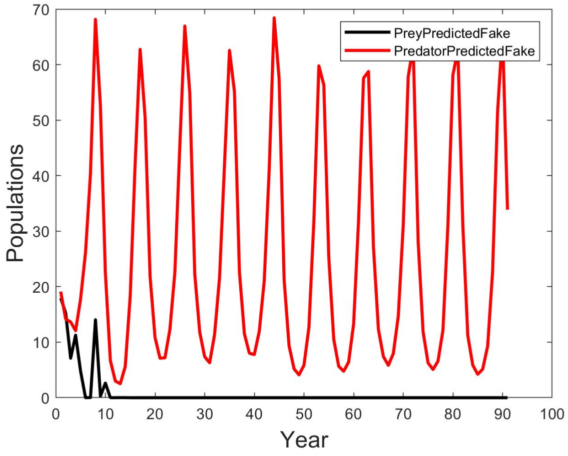
1. Chow Base
    a. Modify code from Predator-Prey state estimation
2. Albers Base
    a. Code originally from Albers for joint estimation
    b. Has been worked on by Prof’s Shtylla and de Pillis for T2D

Issues with Albers Code Base

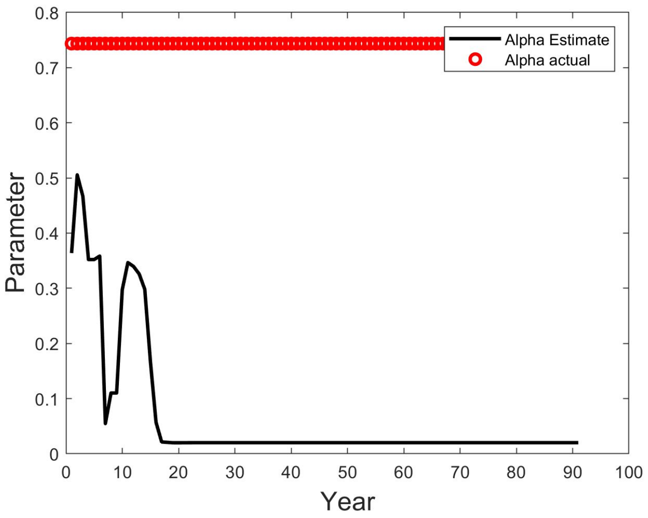
- The states had a tendency to go negative (not possible with real data) which would cause the program to crash -we fixed this by manually checking for the negativity in the state vector - after this, we were able to get the code to run, but we would ideally like to fix the underlying problem. We also gave the parameters a ceiling
- The predator estimate is mirroring what the prey should look like and the predators are just converging to zero
- Currently have only run it with the fake generated data, not the real data set
- Possible things we can change to affect the results:
    - Filter parameters - alpha, beta, kappa
    - Initial guesses
    - Noise

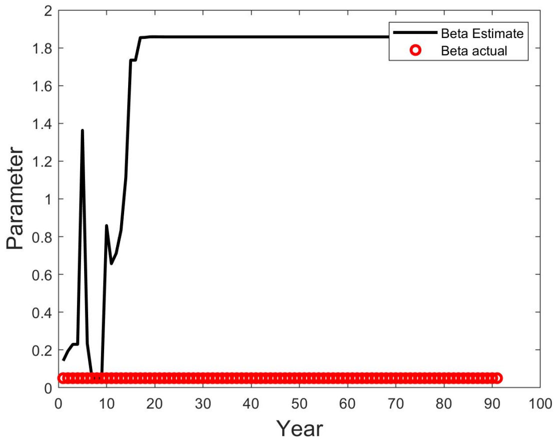
Albers Base Current Results - State Estimation, Fake Data



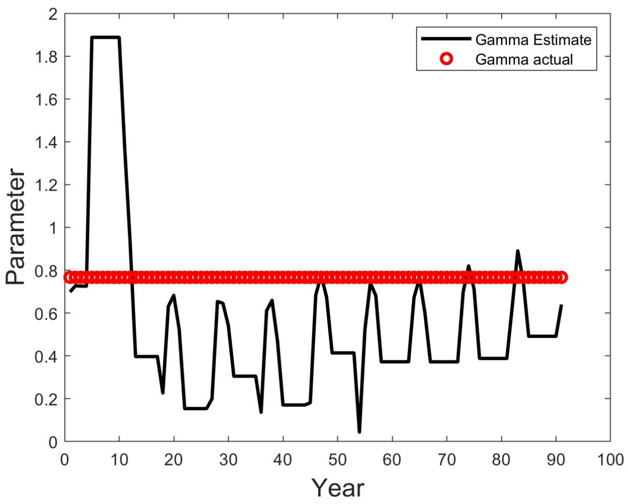


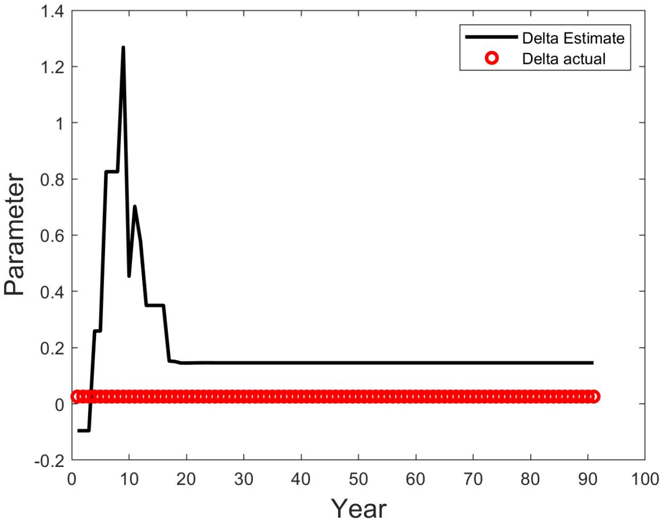
Albers Base Current Results - Fake Data Parameter Estimation





Albers Base Current Results - Fake Data Parameter Estimation





Albers Base - Real Data

- Currently when I try to run my filter on the actual data, it crashes due to a non-positive covariance matrix
- My next step in this case is to figure out how this is happening and how I can fix it
- One possible way to fix it is to make sure that during the propagation of sigma points, none of them are becoming negative

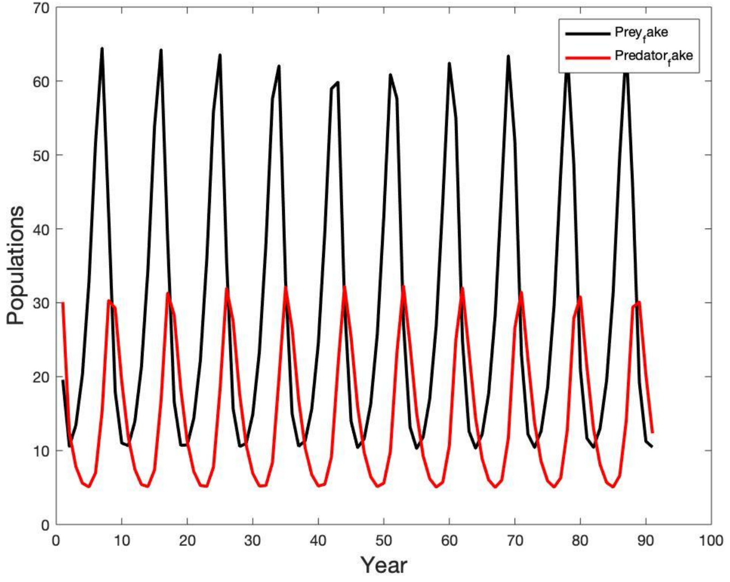
Chow Code Base Joint Estimation Summary

- Problem: The filter is prone to crashing early
    - When parameters < 0 → :(
- Solution: Introduce constraints based on prior knowledge of parameter ranges
    - Would like to eliminate this

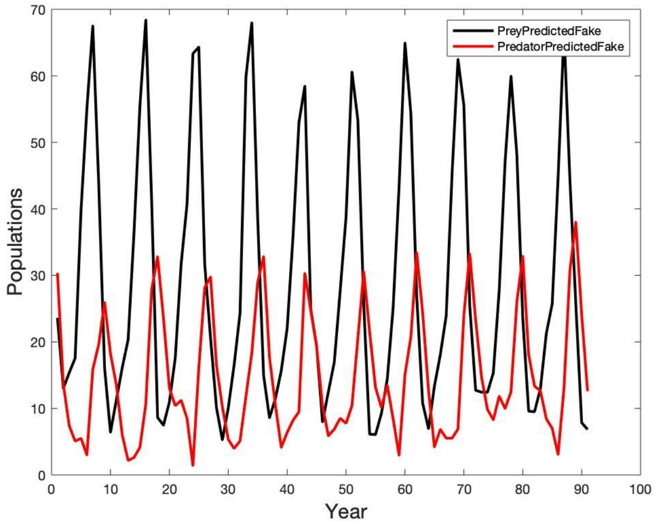
- Problem: Filter performs better on predator populations
- Solution (so far): Introduce measurement noise
    - Making sure the filter is aware the data is noisy

Current Results - Fake Data with Noise

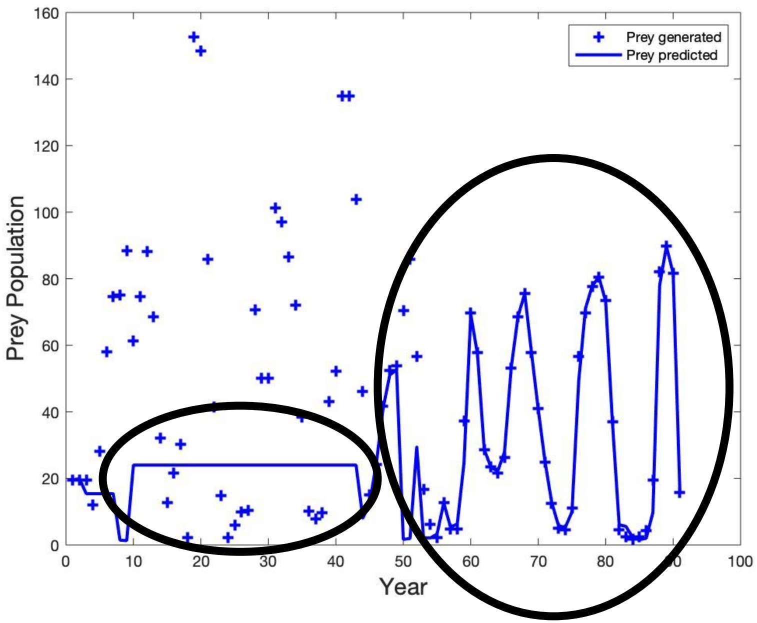
Generated Data

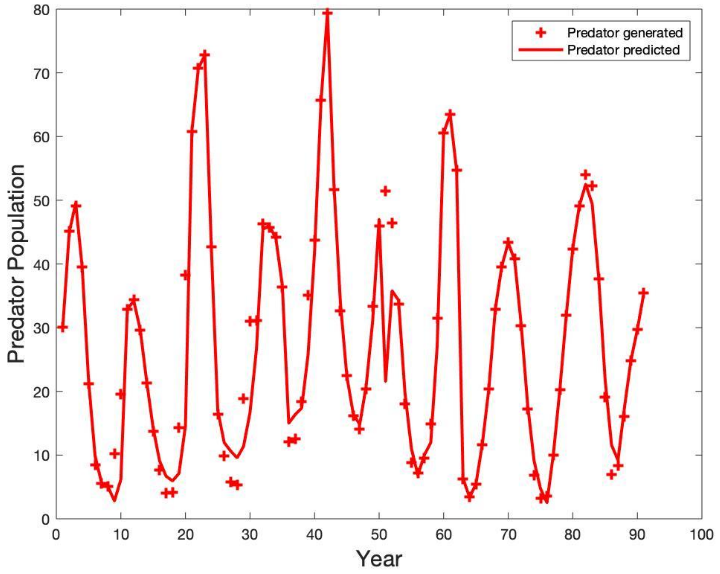


Estimated Data

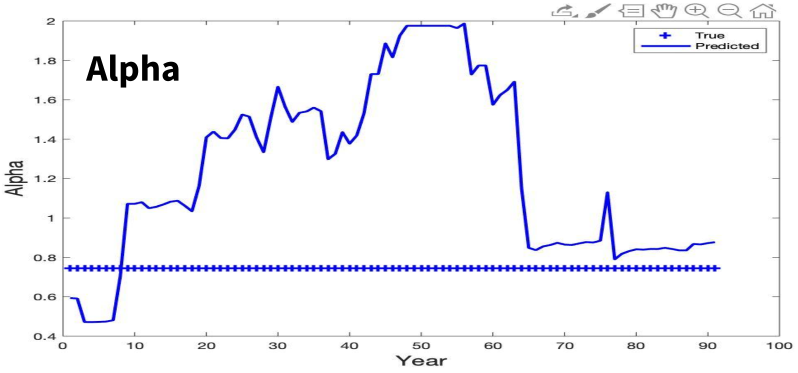


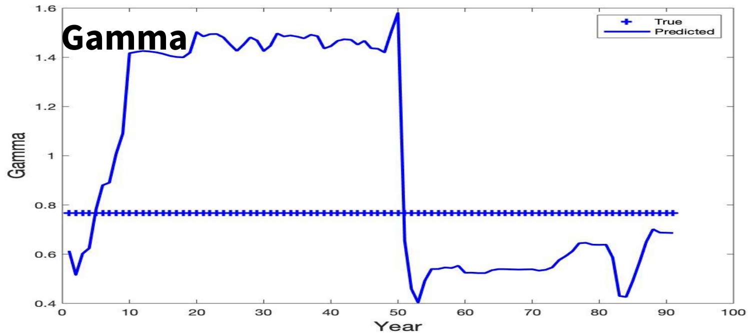
Current Results - Real Data States

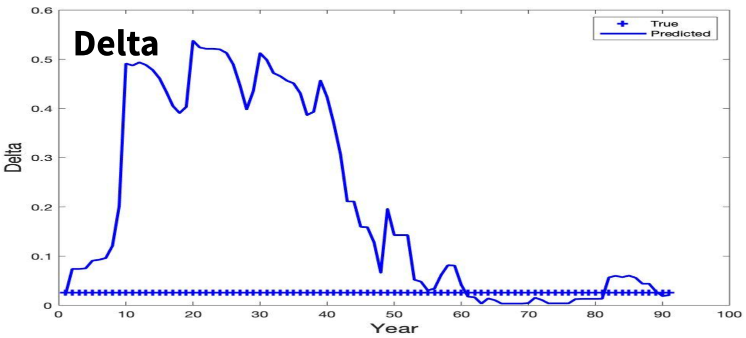


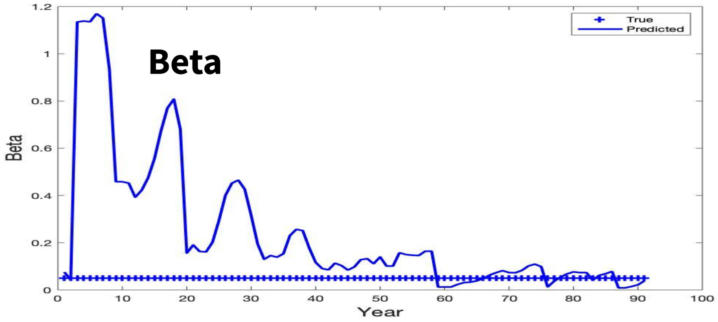


Current Results - Real Data Parameters



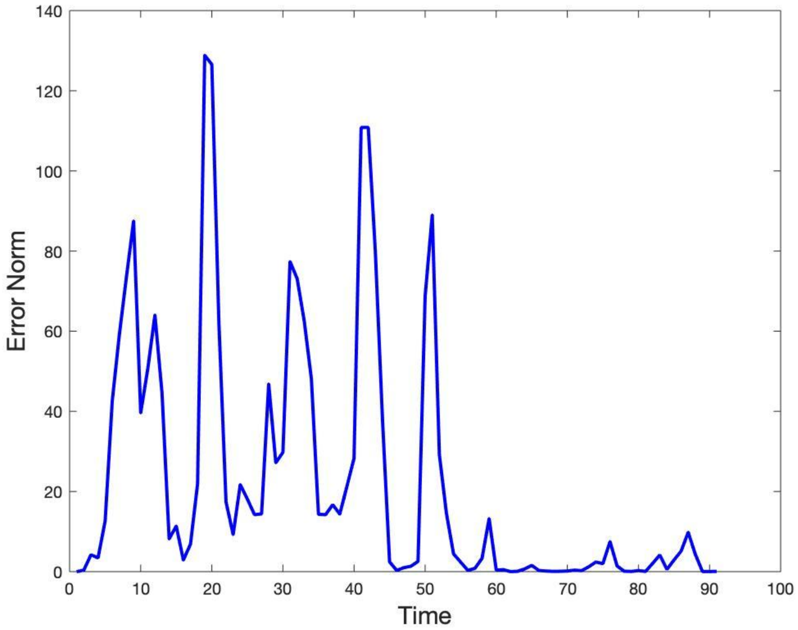






Error

Using our definition of error from last time (using the norm), can produce:



Conclusions

- Without given parameters, filter takes longer to learn the correct behavior
- However, once it does, the joint filter outperforms the simple state estimation UKF
- Currently, the Chow code base is outperforming the Albers Code base

Next Steps

- Improve performance of Joint UKF
- Move to a Dual UKF approach
    - Here, two SEPARATE UKF’s run simultaneously, one for states and one for parameters