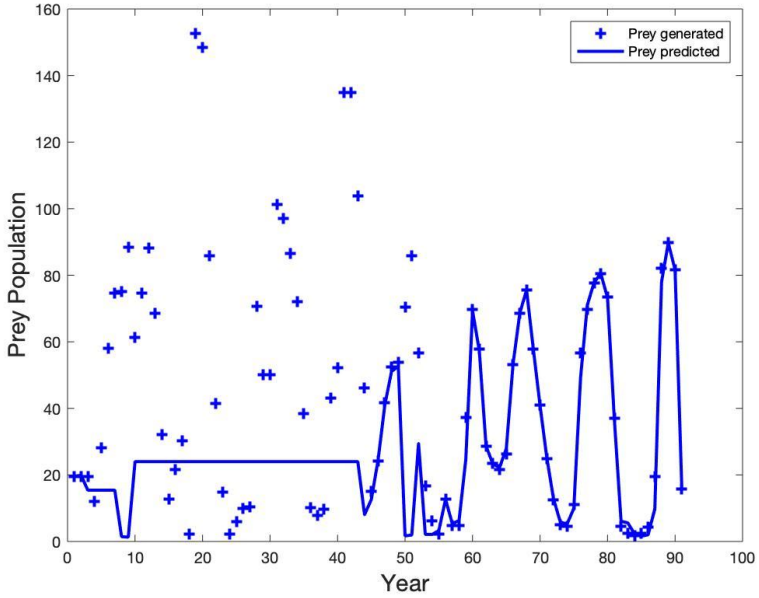
Kalman Filters
Week 4

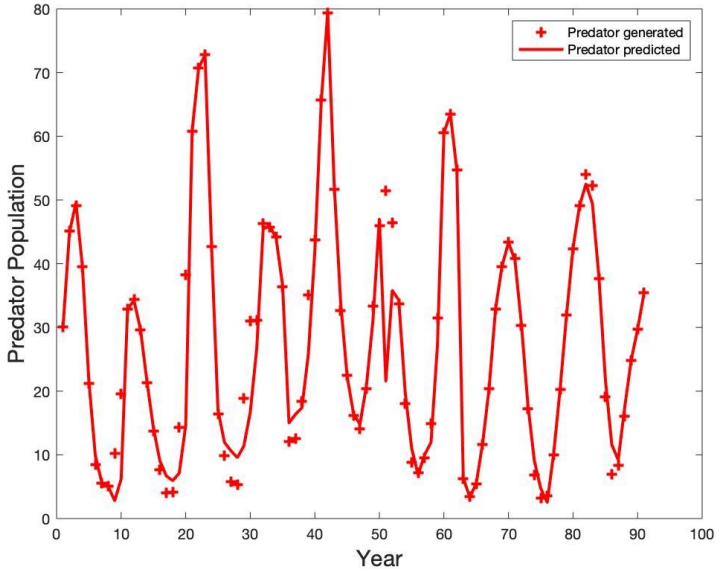
Subteam 2

Overview

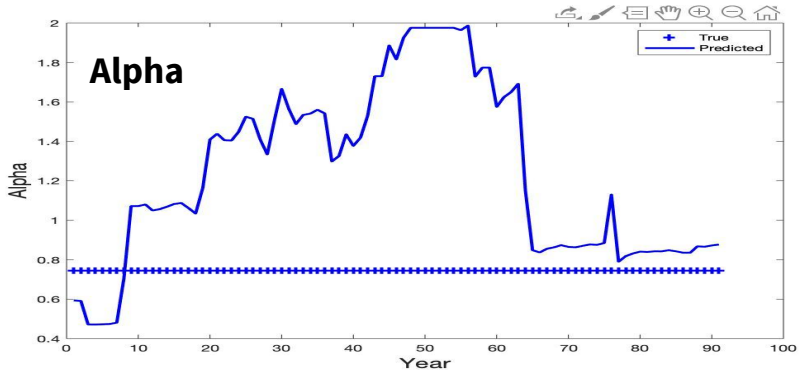
- Began with building theoretical understanding
  - Linear Kalman Filter, Unscented Kalman Filters
- Implemented UKF State Estimation
- Moved to doing parameter estimation
  - Joint UKF
  - Dual UKF
- Working implementations on Lotka-Volterra Model
- Now moving to implement on T1D model
  - Beginning first with data generated with noise

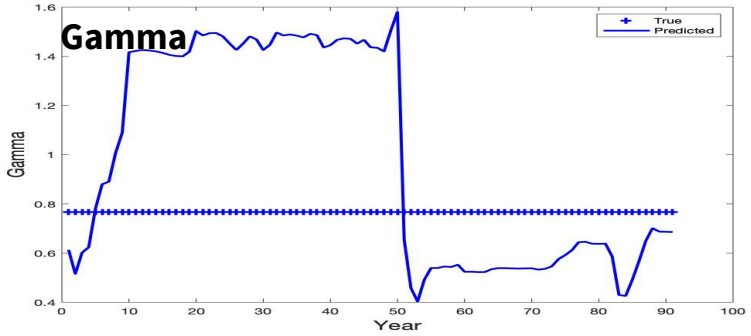
Joint UKF Last Week - States

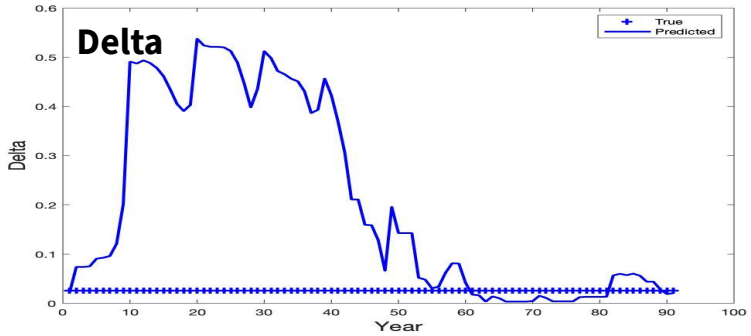


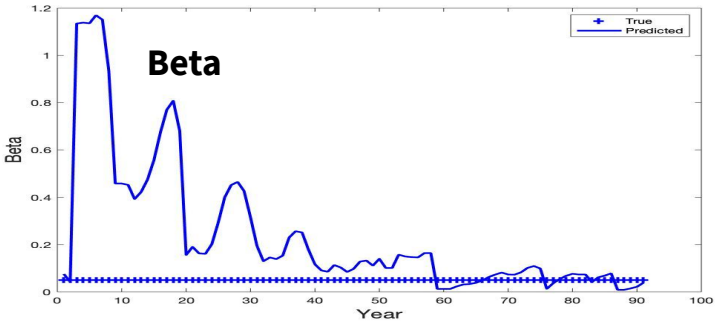


Joint UKF Last Week - Parameters





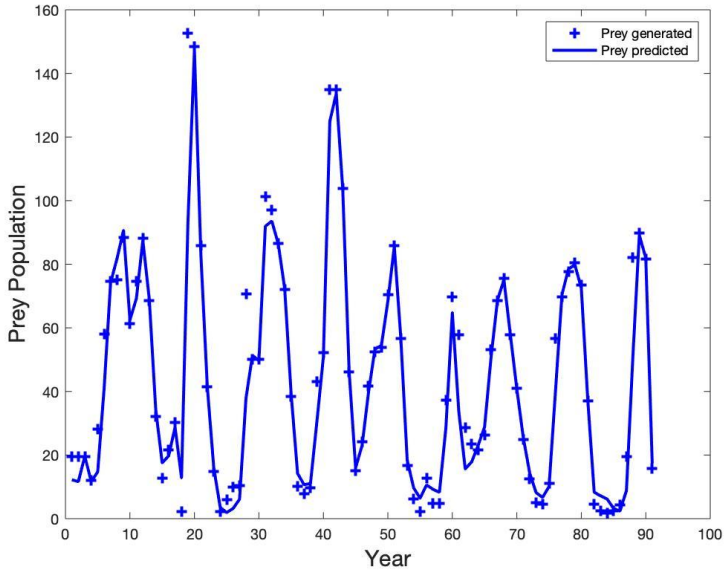


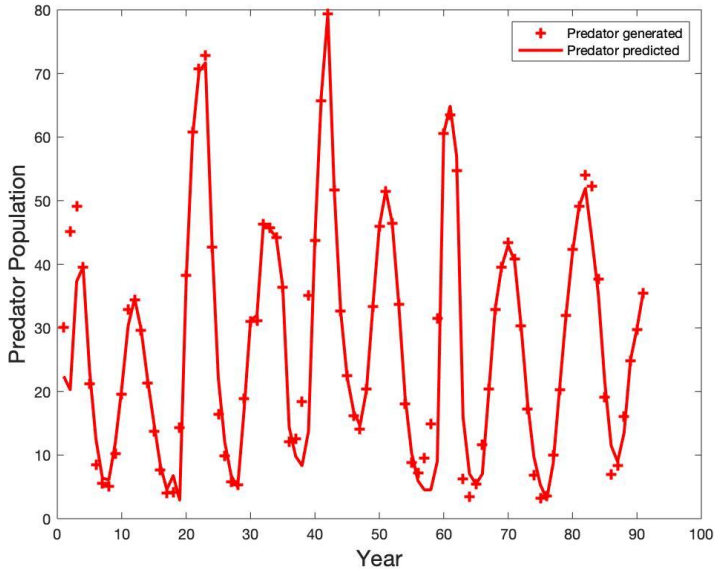


Alterations

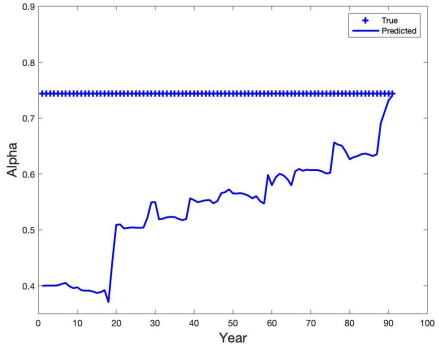
- Allowing for covariance between states and parameters
- Reduction of Process Noise

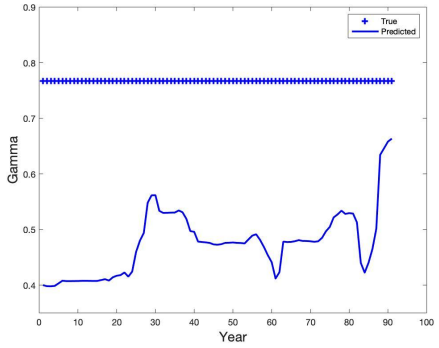
New Joint UKF Results - States

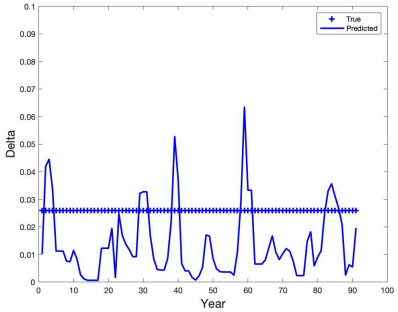


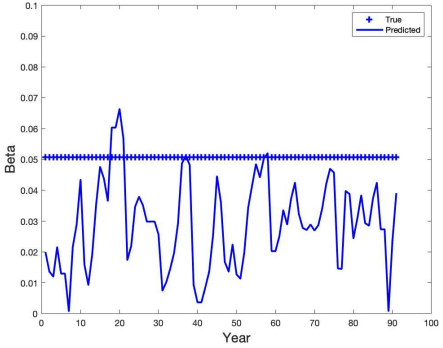


New Joint UKF Results - Parameters









Note the change in y-axis values for Beta and Delta!

The Dual UKF

- Run two UKF's simultaneously: one for the populations and another for parameters
- For all time points do the following:

1. Create better estimate for parameters using observable at time t
2. Use estimated parameters to predict state at time t

The State Filter

- Regular state estimation UKF
- Input: current best estimate of parameters, estimate for state at time t, and observable at time t + 1
- Output: estimate for state at time t + 1

The Parameter Filter

- State kept constant
- Create sigma points for parameters
- Input: previous estimate of parameters, estimate of states at time t, observable at time t + 1
- Output: new estimate for parameters

Process Noise

Parameters

$$w_k = w_{k-1} + v_{k-1} \quad (1)$$

Transition Function

Measurement Noise

Observables

$$y_k = h(f(\hat{x}_{k-1}; w_k), w_k) + \epsilon_k \quad (2)$$

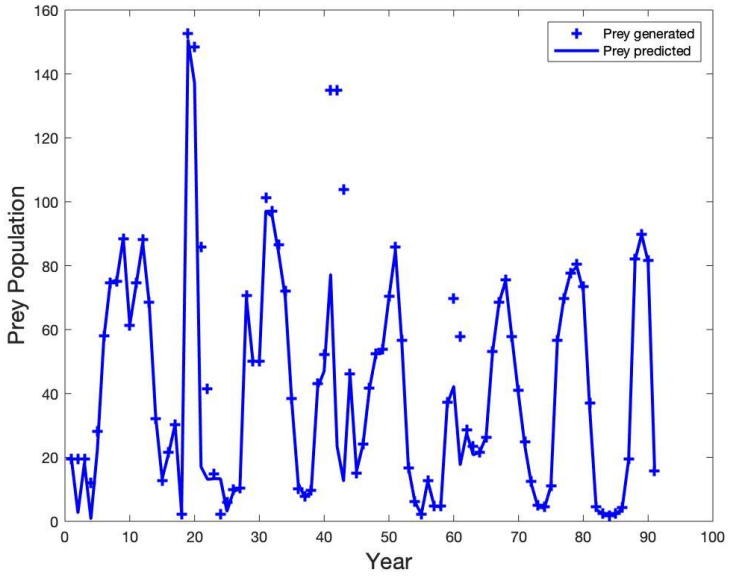
Measurement Function

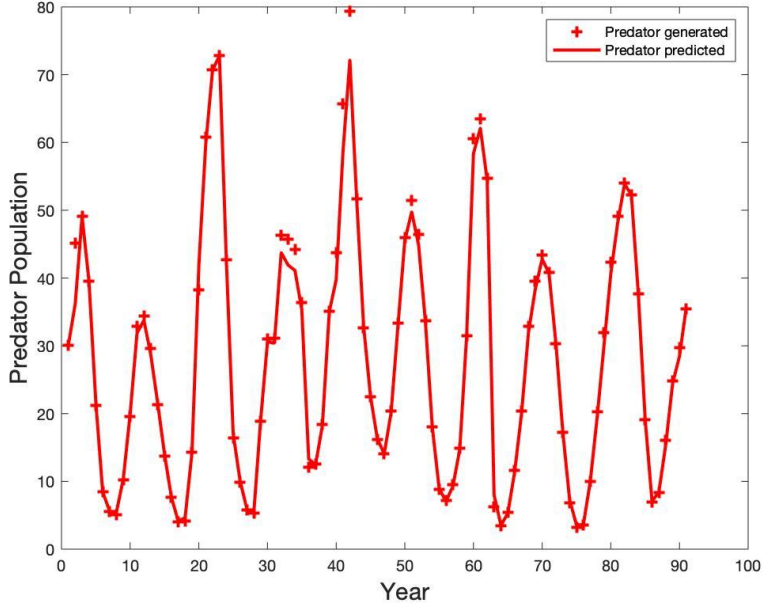
Dual UKF Parameters

- State Filter Parameters
  - P_x_0 - covariance between states
  - Q_x - state process noise covariance
  - R_x - state measurement noise covariance
- Parameter Filter Parameters
  - P_param_0 - covariance between parameters
  - Q_p - parameter process noise covariance
  - R_p - parameter measurement noise covariance

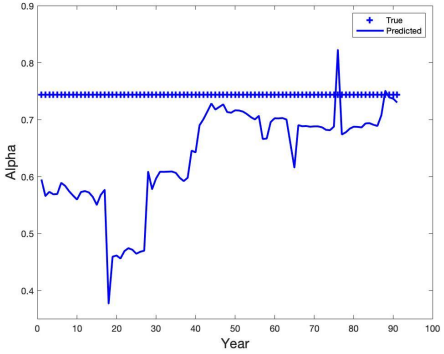
Note: $R_x = R_p$

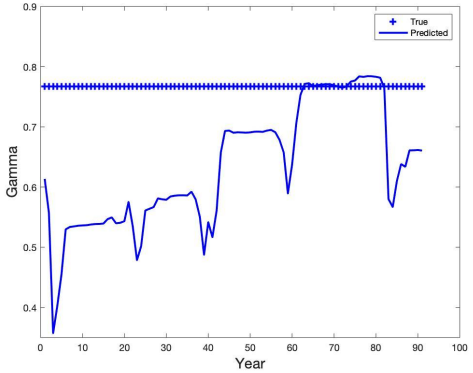
Dual UKF Results - States

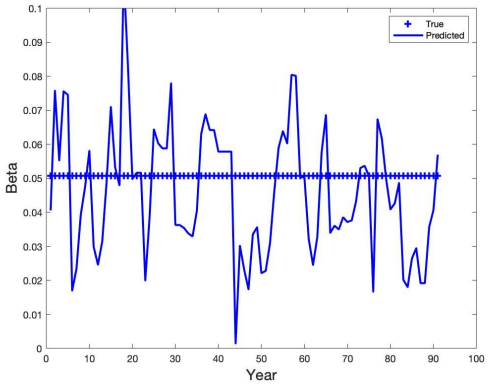


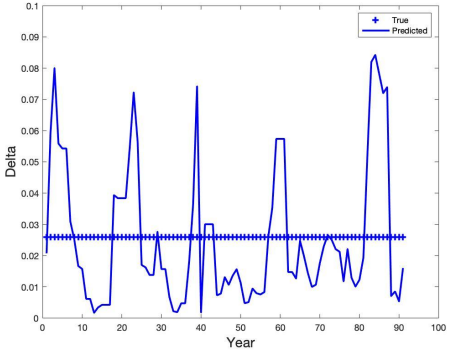


Dual UKF Results - Parameters





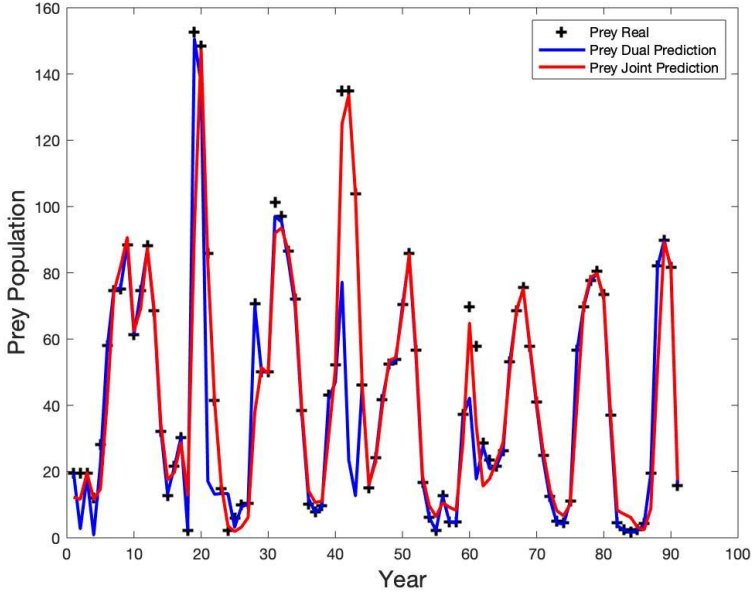


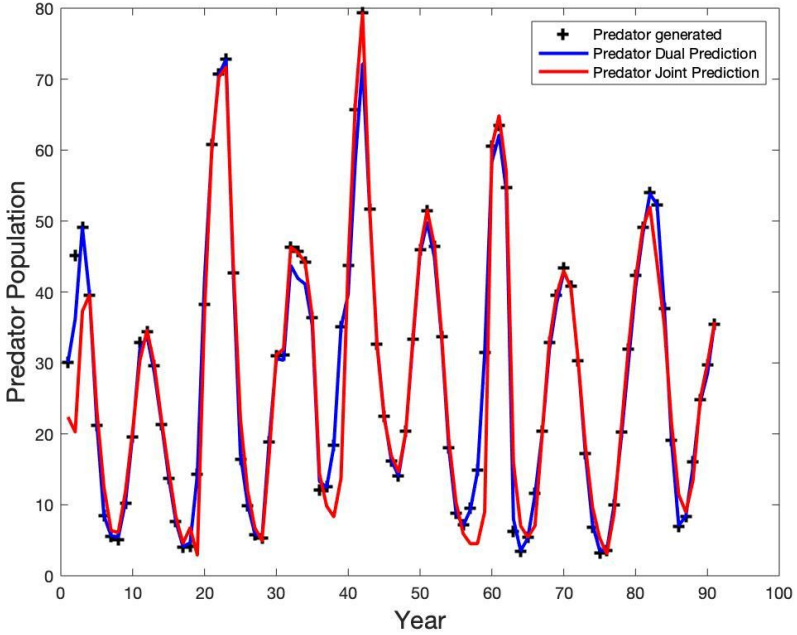


Tuning State Process Noise (Q)

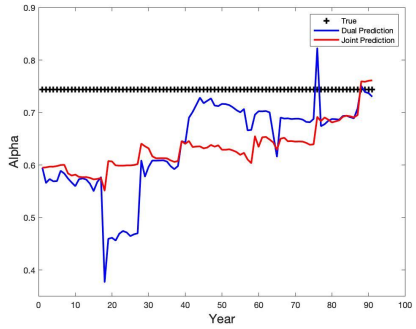
```
HLData = load('HaresLynxData.mat');  %Load dataset
rawData = HLData.Lotka_Volterra_Data;
x(1:2,:) = rawData(:, 2:3)'; %The real data

T = 91;
x_fake = zeros(Nx,T);
x_fake(:,1) = [x(1,1); x(2,1)];
tspan = [0 T];
sol = ode45(@(t, y) Lotka_Volterra_Model(t, y, truepar), tspan, x_fake(:,1)); %Use ODE solver
for t=2:T %Generate data for each time point
  x_fake(:,t) = deval(sol, t); %Get ODE solution for time t
end

prey_noise = x(1,:) - x_fake(1,:); %Difference between real and ODE
predator_noise = x(2,:) - x_fake(2,:);
cov(prey_noise, predator_noise) %Print covariance matrix
```

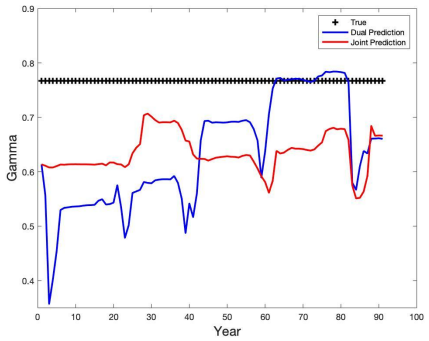
Joint versus Dual - States

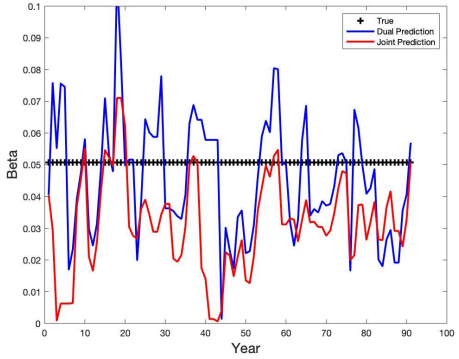


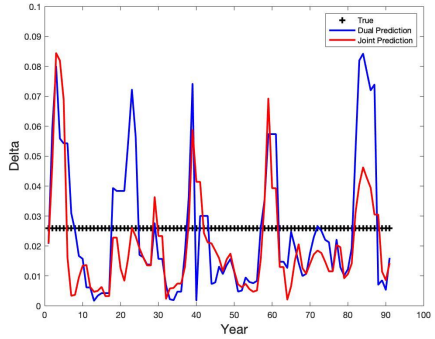


Joint versus Dual - Parameters









Joint versus Dual - Error Comparison

	Dual UKF	Joint UKF
Final 50% Prey MSE*	50.7984	51.3580
Final 50% Predator MSE*	0.6321	19.2451
Alpha Error	0.0134	0.0177
Beta Error	.1064	0.0001
Delta Error	.0062	0.0116
Gamma Error	.0099	0.1005

* We use final 50% to allow UKF time to learn before judging various approaches

Joint versus Dual UKF Takeaways

- Currently, more faith in the Dual
- Benefits of Joint
  - Allows for covariances between states and parameters
- Benefits of Dual
  - **Separates out noise of states and parameters** (can tune independently)
- Right now, benefit of Dual more valuable

T1D State Estimation: Unscented Kalman Filter

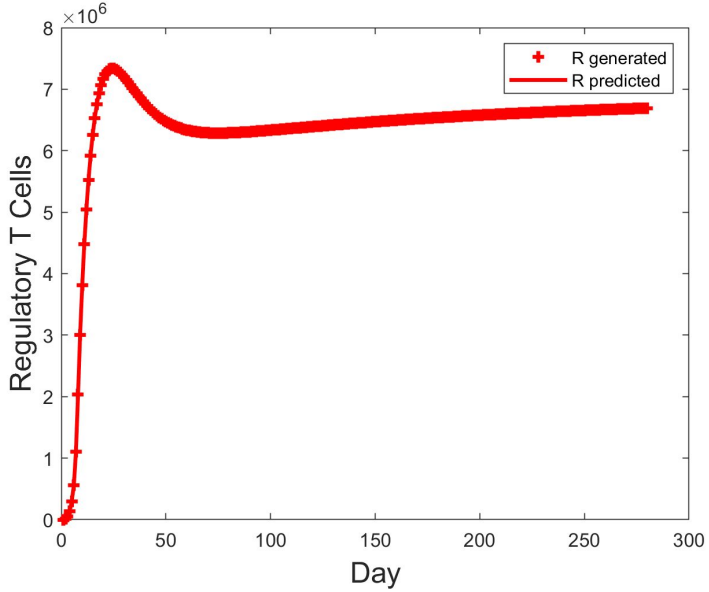
Reminders:

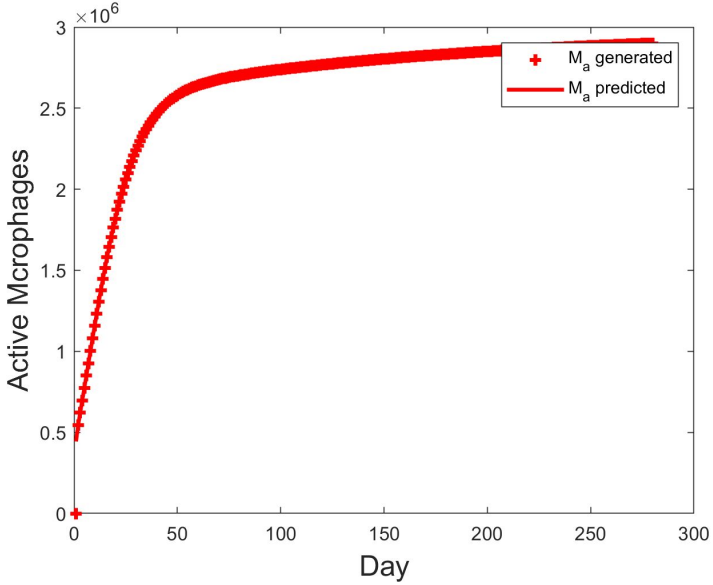
- State Estimation is where you keep the parameters constant and try to estimate the next state using the previous estimate and the data
- The Unscented Kalman Filter uses the unscented transformation to transform a number of sigma points in order to estimate the distribution

T1D State Estimation

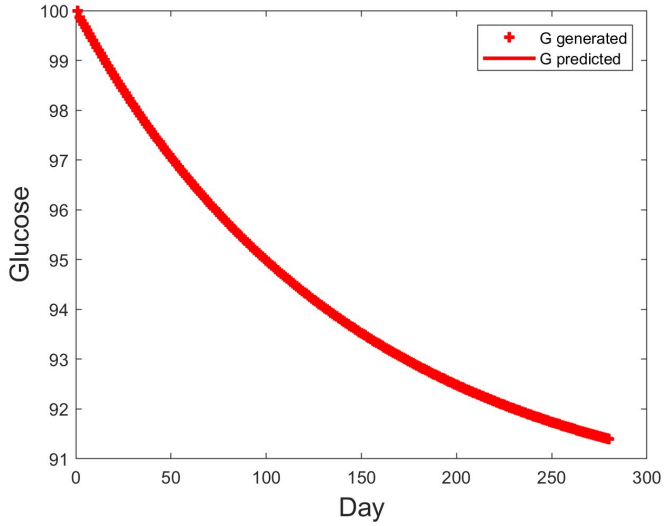
- Used Chow State Estimation
- ODE and Parameters taken from T1D code
- 12 States (from the 12 DEs)
- 1 Observable (Glucose)
- Ran Four Times
  - Wild-type without wave
  - Wild-type with wave
  - NOD without wave
  - NOD with wave

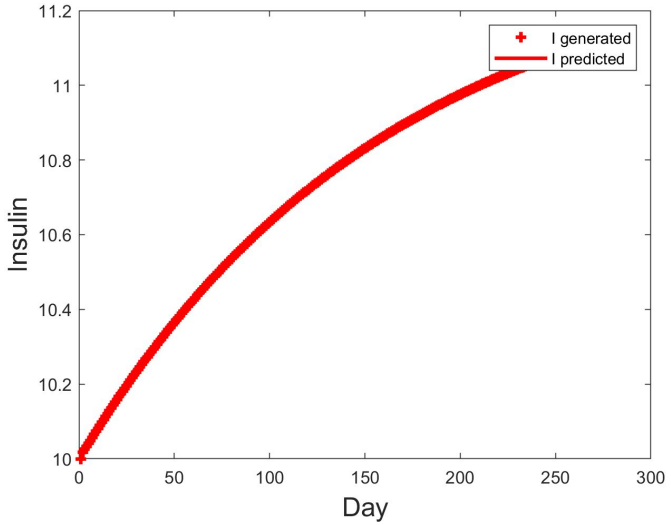
Results - Wildtype Mouse Without Wave



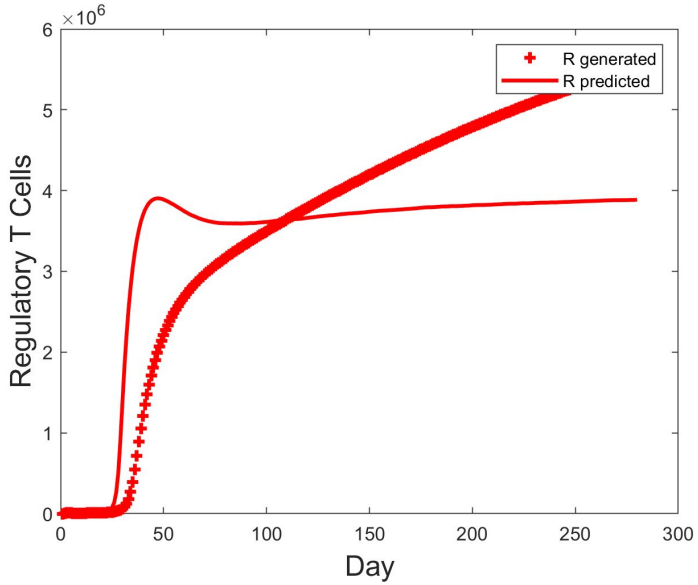


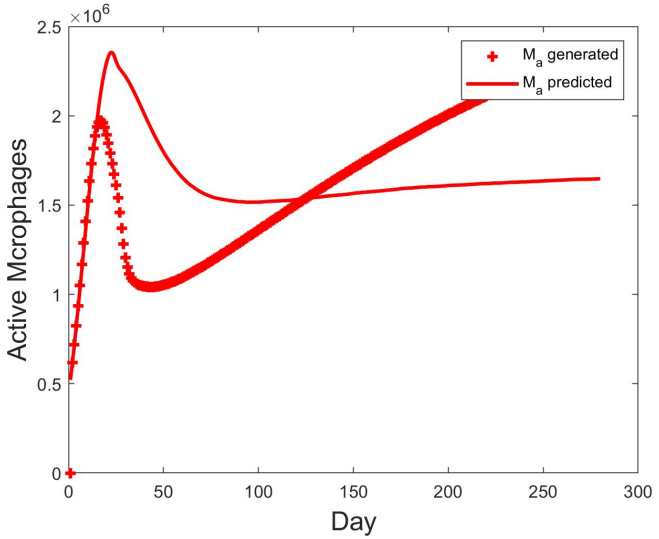
Results - Wildtype Mouse Without Wave



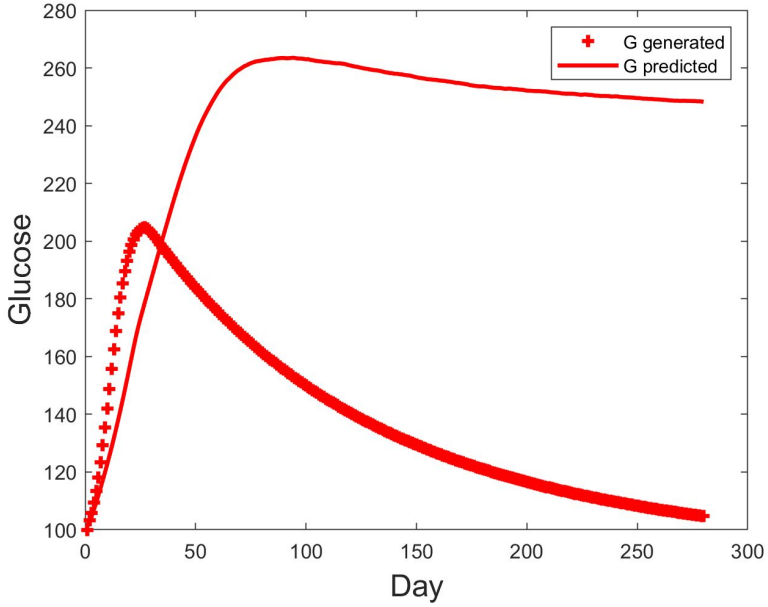


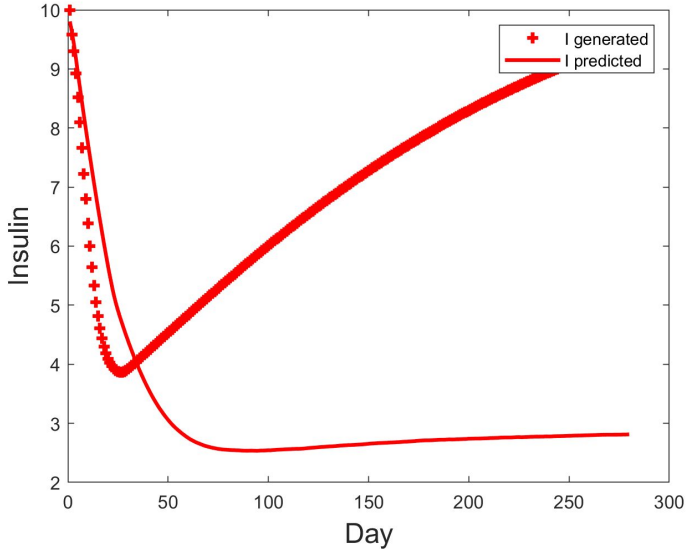
Results - Wildtype Mouse With Wave



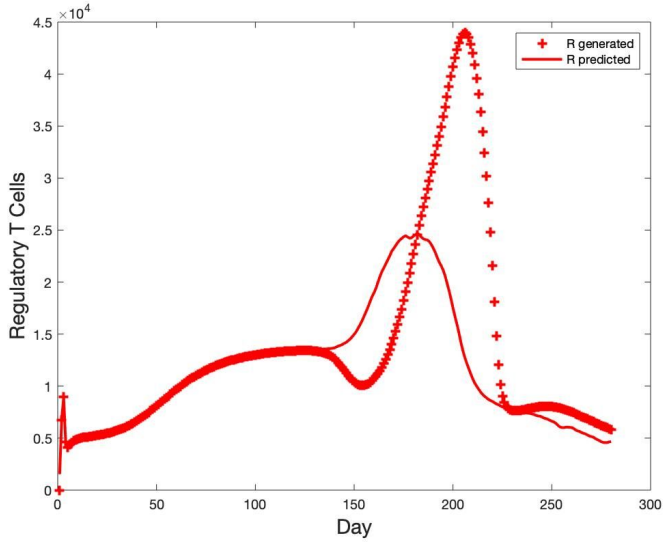


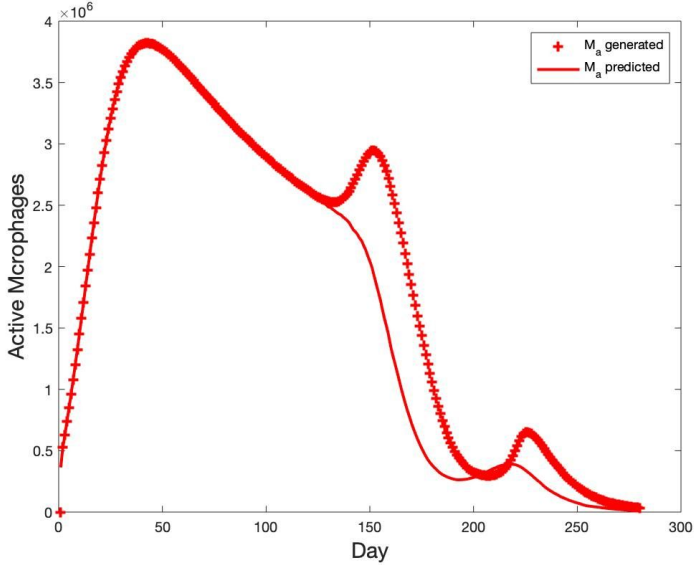
Results - Wildtype Mouse With Wave



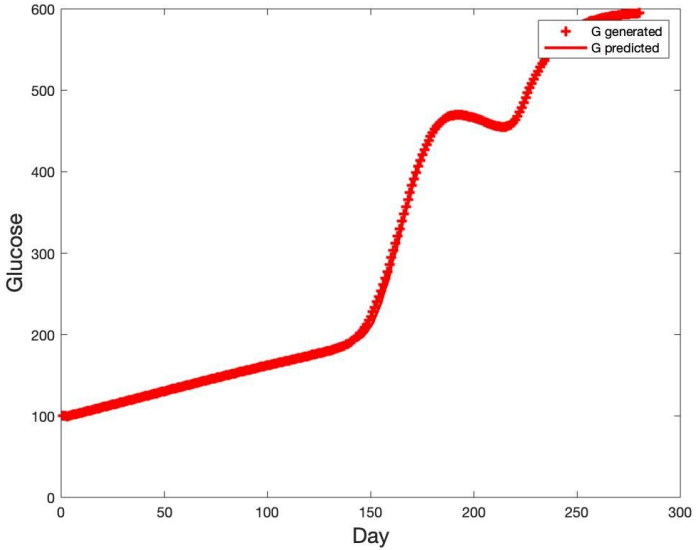


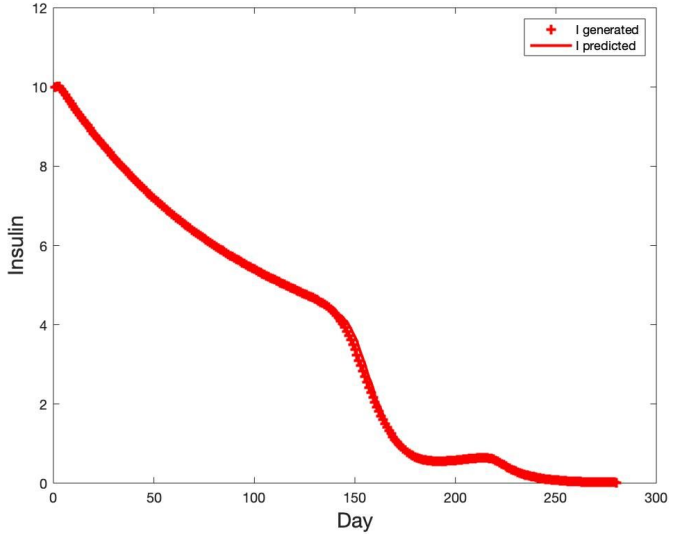
Results - NOD Mouse Without Wave



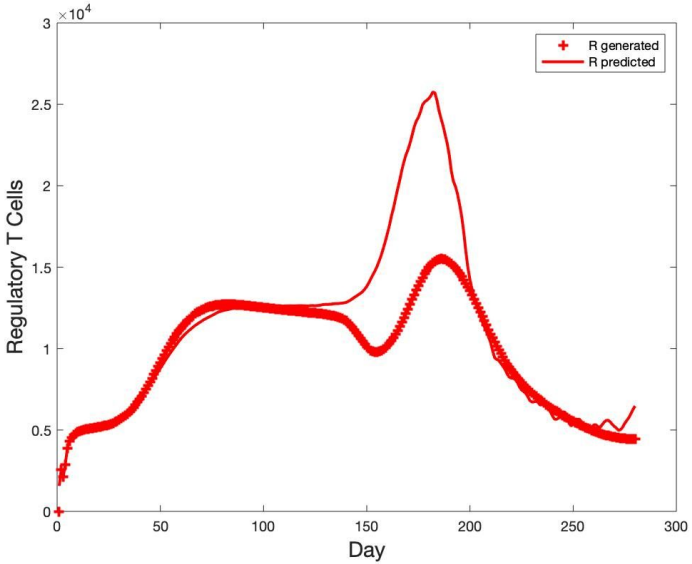


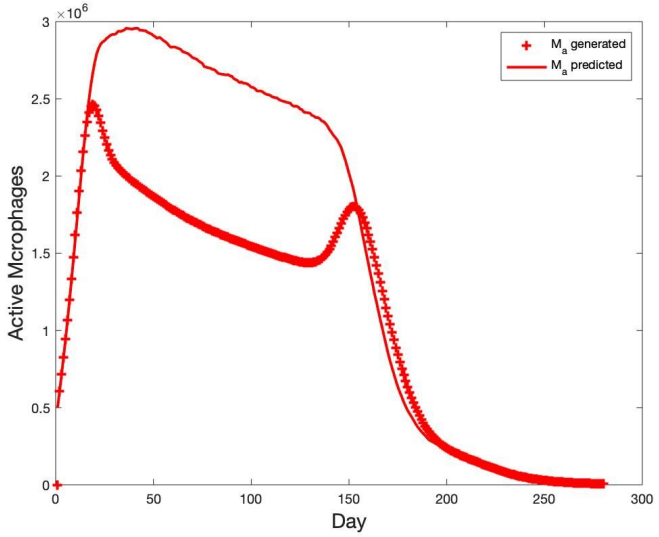
Results - NOD Mouse Without Wave



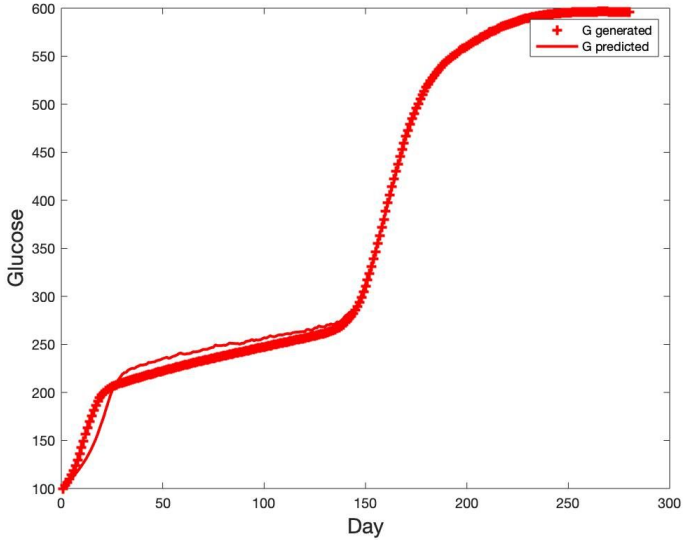


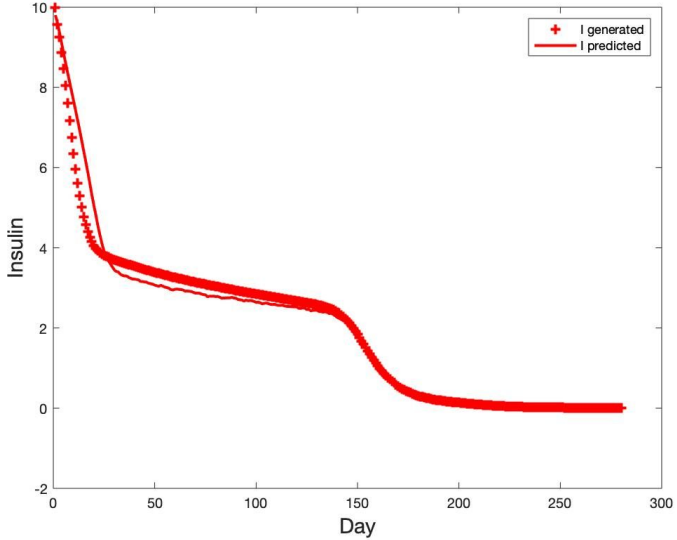
Results - NOD Mouse With Wave





Results - NOD Mouse With Wave





Takeaways

- In the absence of a major shift in the states such as the apoptotic wave, state estimation works slightly better with a more stable system like the wild-type mouse as opposed to a more volatile system such as the NOD mouse
- When something such as the apoptotic wave is introduced, it causes some states to shift quickly; if the filter doesn’t catch this it can reach a different steady state. The sharper the shift, the more likely the filter is to miss it and reach a different steady state

Next Steps

- Set up Joint and Dual Filters for parameter fitting
- Run filter on actual data