Pomona College

Senior Thesis in Mathematics

An Exploration of Kalman Filters

Author:
Lindsey Tam

Advisor:
Dr. Blerta Shtylla

Submitted to Pomona College in Partial Fulfillment
of the Degree of Bachelor of Arts

December 11, 2019

Abstract

Kalman Filters create predictions for linear systems by recursively generating and correcting predictions according to incoming data. This process enables estimates to become progressively more accurate without relying on large amounts of initial data. Nonlinear forms of the Kalman Filter exist and include the Extended Kalman Filter, the Unscented Kalman Filter, and the Dual Unscented Kalman Filter. This paper explores the theory behind each of these filters and implements a few examples of the Unscented Kalman Filter.

Contents

Chapter 1

Introduction

A Kalman Filter, so named from its developer Rudolph Kalman, is a data prediction method that estimates the value of unknown parameters of linear dynamical systems. This is a recursive predictive-corrective process that allows Kalman Filters to continuously generate predictions about state variables without relying on large amounts of initial data. In the case of nonlinear systems, alternative forms of the Kalman Filter were developed, including the Extended Kalman Filter, the Unscented Kalman Filter, and the Dual Unscented Kalman Filter. The Extended Kalman filter linearizes the nonlinear system around the mean by taking the Jacobean of the nonlinear function. Navigation and signal processing are the main applications of the Extended Kalman Filter. Instead of linearizing the system around a single points, the Unscented Kalman Filter utilizes many points, known as Sigma points. The Sigma points then undergo a nonlinear transformation known as the Unscented Transform. The Dual Unscented Kalman Filter is similar to the Unscented Kalman Filter, but also estimates the value of some unknown parameter in the system. Applications of the Unscented and Dual Unscented Kalman Filters include modeling biological systems.

Chapter 2

Kalman Filters

The Kalman Filter (KF) generates predictions for linear dynamical systems [?]. Assuming the given data is noisy and Gaussian [?, ?], the first predictive step assumes knowledge of initial states and the model process, which takes the form of matrix F. The next step involves calculating the covariance in order to calculate the Kalman Gain, which is a measure of how much the estimate should be changed given actual measurements of the system. The corrective step utilizes the Kalman Gain and the measured system outputs to gauge whether to depend more on the model or on the data.

This process can be done recursively, allowing the model to become progressively more accurate as more data is added. Therefore, overtime, it is expected that the model will converge with the actual system measurements.

2.1 Kalman Filter Algorithm

Variables in the Kalman Filter		
Variable	Description	Dimensions
x	State variables	$d_x \times 1$
y	Output vector	$d_y \times 1$
u	System inputs	$d_u \times 1$
v	Measurement noise	$d_y \times 1$
w	Process noise	$d_x \times 1$
F	State function	$d_x \times d_x$
H	Observation function	$d_y \times d_x$
G	Input function	$d_x \times d_u$
K	Kalman Gain	$d_x \times d_y$
Q	Process noise covariance	$d_x \times d_x$
R	Measurement noise covariance	$d_y \times d_y$
P	Covariance matrix	$d_x \times d_x$

1. Initialize the state estimate (x_0) and the initial state covariance matrix (P_0)

$$\hat{x}_0 = \mathbb{E}[x_0] = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$P_0 = \begin{bmatrix} var(x_1) & \dots & cov(x_1, x_n) \\ \vdots & \ddots & \vdots \\ cov(x_n, x_1) & \dots & var(x_n) \end{bmatrix}$$

   Enough should be known about the modeled system to generate these values. In general, $\hat{x}_k$ denotes the model's estimate of the state variables. When initializing these values, we use x_0, which is the actual initial measurements of the system.

   A state covariance matrix is a symmetrical positive semi-definite square matrix whose diagonals correspond to the variance of a variable at location i and elsewhere is the covariance of the pairwise elements. In practice, covariance matrices help us better understand the spread of data. For the case of the KF, calculating the state covariance is necessary for computing Kalman Gain.

2. Generate a prediction (x_{k+1})

$$x_{k+1} = Fx_{k-1} + Gu_{k-1} + w_{k-1}$$

   State matrix F and Input matrix G is how the system is defined. Every state variable contained in x_k is defined by a linear differential equation.

These linear differential equations can be used to generate the F and G matrices. Therefore, F should be a square matrix whose dimension is equal to the number of states variables and G can be a matrix (or in some cases a vector), depending on the dimension of input vector u.

u_k is an input vector, which is a measurable value that helps define the system, but is not contained in the state vector. Depending on how the system is defined, u_k can be a constant value or it can be a value dependent on time step k.

w_k is the process noise vector at time k. Process noise can be thought of as the model's accuracy. When process noise is 0, it implies that the model is perfectly accurate and does not have to correct for incoming system measurements. On the other hand, high process noise will essentially restate the system based on incoming measurements. w_k has the same dimensions as x_k, allowing us to identify whether or not to adjust the equations for the state variables. When all values of w_k equal 0, it implies that the linear differential equations we are using to define the state variables have no error.

3. Use the state covariance matrix ($P_{k|k-1}$) to calculate Kalman Gain (K_k)

$$P_{k|k-1} = FP_{k-1}F^T + Q_{k-1}$$

$$K_k = P_{k|k-1}H_K^T(H_kP_{k|k-1}H_K^T + R_k)^{-1}$$

H is the observation matrix, which enables the state variables to be linearly transformed to match the outputs of the system. The dimensions of H reflects which state variables have measurable values in the system. It is not assumed that every state variable is measurable, so H allows us to compare the measurable state variables to x_k. Simple applications of H include creating matrices with 0's and 1's, with 1's denoting that a state variable is measurable and a 0's representing non-measurable states. In other cases, H is an integer used as a scaling factor.

The Kalman Gain is the main component of the corrective aspect of the KF. The Kalman Gain is a measure of how much to change the model based on incoming data. Low values of the Kalman Gain imply the model is accurate while higher values indicate the model should adjust based on the incoming data. Calculating Kalman Gain involves using Q, which is process noise covariance of w_k, R, which is measurement noise covariance, and H, which is also used in transforming the output vector in a later step. From this equation, one can see that balancing Q and R is critical for model performance. Larger values of Q indicate higher modeling error, which leads to a higher Kalman Gain and increased model correction. On the other hand, large values of R imply high measurement error, leading to a lower Kalman Gain and less model correction.

4. Calculate the transformed output vector ($\hat{y}_k$) and correct the prediction

$$\hat{y}_k = H_k x_k + v_k$$

$$x_k = x_{k-1} + K_k(y_k - \hat{y}_k)$$

y_k, also known as the output vector, is the actual measurements of the system and $\hat{y}_k$ is a way of transforming the prediction (using H) into a format that can be compared with y_k.The term v_k, also known as measurement noise, is also added to account for measurement error. Subtracting the actual measurements from the predicted measurements is necessary, because y_k does not always contain measurements for all variables in state vector x_K. The quantity $y_k - \hat{y}_k$ is also known as measurement residual or innovation.

5. Update the covariance matrix

$$P_k = (I - K_K H_k)P_{k|k-1}$$

P_k will be used in the next iteration of the filter.

Chapter 3

Extended Kalman Filters

The Extended Kalman Filter (EKF) is the non-linear version of the Kalman Filter. Though it can be used for non linear equations, it is important to note that it is not an optimal estimator. For the most part, the EKF Algorithm is nearly identical to the KF algorithm. The critical difference is in linearizing the state and observation function. The EKF uses the Jacobean to linearly approximate the non-linear function around the mean of the Gaussian distribution. Skipping this step would result in the transformed data being non-Gaussian; though taking the Jacobean enables the transformation to remain Gaussian, it is not exact, resulting in some generalization. Linear approximation through a single point also makes the EKF inefficient when dealing with complex, higher order systems. Because of this, the model is highly subject to error, which can be somewhat reduced by setting accurate initial values. Though these flaws exist, the EKF performs strongly with applications of real time spatial fields, including navigation and positioning systems.

3.1 Extended Kalman Filter Algorithm

Variables in the Extended Kalman Filter		
Variable	Description	Dimensions
x	State variables	$d_x \times 1$
y	Output vector	$d_y \times 1$
u	System inputs	$d_u \times 1$
v	Measurement noise	$d_y \times 1$
w	Process noise	$d_x \times 1$
f	None linear state function	$d_x \times d_x$
F	State function	$d_x \times d_x$
h	Non linear observation function	$d_y \times d_x$
H	Linearized observation function	$d_y \times d_x$
K	Kalman Gain	$d_x \times d_y$
Q	Process noise covariance	$d_x \times d_x$
R	Measurement noise covariance	$d_y \times d_y$
P	Covariance matrix	$d_x \times d_x$

1. Initialize the state estimate (x_0) and the initial covariance matrix (P_0)

$$\hat{x}_0 = \mathbb{E}[x_0] = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$P_0 = \begin{bmatrix} var(x_1) & \dots & cov(x_1, x_n) \\ \vdots & \ddots & \vdots \\ cov(x_n, x_1) & \dots & var(x_n) \end{bmatrix}$$

Often, P_0 will be initialized as a diagonal matrix with the diaganals being the variance of each state variance and every other value being set to 0.

2. Calculate the Jacobean

$$F = \frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_n}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \dots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

$$H = \frac{\partial h}{\partial x} = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \dots & \frac{\partial h_m}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_m}{\partial x_1} & \dots & \frac{\partial h_{d_y}}{\partial x_n} \end{bmatrix}$$

Here, both f and h have no closed form solution. f is the non-linear state function and F is the linear approximation of f with dimension d_x. Similarly, h is the non linear observation while H is linearized observation with dimension d_y.

3. Generate a prediction

$$x_{k+1} = f(x_{k-1}, u_{k-1})$$

   This equation is different from the one used in the Kalman filter

4. Use the state covariance matrix ($P_{k|k-1}$) to calculate Kalman Gain (K_k)

$$P_{k|k-1} = FP_{k-1}F^T + Q_{k-1}$$

$$K_k = P_{k|k-1}H_K^T(H_kP_{k|k-1}H_K^T + R_k)^{-1}$$

   Recall that H and F are linear approximations of f and h. Therefore, it is assumed that there is some amount of error when calculating the Kalman Gain.

5. Correct the prediction

$$\hat{y}_k = H(x_k + v_k)$$

$$x_k = x_{k-1} + K_k(y_k - \hat{y}_k)$$

6. Update the covariance matrix

$$P_k = (I - K_KH_k)P_{k|k-1}$$

Chapter 4

Unscented Kalman Filters

The Unscented Kalman Filter (UKF) is another nonlinear version of the Kalman Filter developed to address the shortcomings of the EKF. For instance, as opposed to using the Jacobean to linearly approximate around a single point, the UKF uses the Unscented Transform (UT) to approximate around multiple points, known as sigma points. The UT is a method of approximating probability distributions that have undergone a non linear transformation using limited statistics. The UT involves using these sigma points, which are represented in a Sigma Point Matrix, to represent the normal distribution of the data. The covariance and weights of these sigma points are calculated. These sigma points undergo a non-linear transformation, resulting in a posterior distribution that is not normal [**?**, **?**] . We are able to approximate the normal distribution of the posterior distribution using the weights and covariance that were calculated prior to the transformation. This process enables the Kalman Filter to be applied to more complex non linear problems.

Unlike the Kalman Filter and the Extended Kalman Filter, the UKF also has a set of parameters. Explanations of each parameter and their default values can be found in the chart below. For the UKF, parameters are necessary for controlling the spread of sigma points. This was not needed for the EKF, since the EKF was only linearizing around the mean. expand here on how to tune parameters.

The term 'unscented' was arbitrarily coined by the developer of the UKF, Jeffrey Uhlmann. In an interview, he shares:

> "Initially I only referred to it as the “new filter.” Needing a more specific name, people in my lab began referring to it as the “Uhlmann filter,” which obviously isn’t a name that I could use, so I had to come up with an official term. One evening everyone else in the lab was at the Royal Opera House, and as I was working I noticed someone’s deodorant on a desk. The word “unscented” caught my

eye as the perfect technical term. At first people in the lab thought it was absurd—which is okay because absurdity is my guiding principle—and that it wouldn't catch on. My claim was that people simply accept technical terms as technical terms: for example, does anyone think about why a tree is called a tree?"

4.1 Unscented Kalman Filter Algorithm

Variables in the Unscented Kalman Filter		
Variable	Description	Dimensions
x	Vector containing state variables	$d_x \times 1$
χ	Sigma Point Matrix	$d_x \times (2d_x+1)$
f	Nonlinear tate function	$d_x \times d_x$
h	Nonlinear observation function	$d_u \times d_x$
u	Input vector	$d_u \times 1$
P	Covariance matrix	$d_x \times d_x$
Q	Process noise covariance	$d_x \times d_x$
R	Measurement noise covariance	$d_y \times d_y$
$W^{(m)}$	Weight for state variable (mean)	scalar
$W^{(c)}$	Weight for covariance	scalar

Parameters in the Unscented Kalman Filter			
	Description	Bounds	Default
α	Controls spread of sigma points	$0 < \alpha \leq 1$	.001
β	Adjust sigma point weight	$\beta \geq 0$	2
κ	Sigma point weighting constant	$0 \leq \kappa \leq 3^*$	0

* Others use $\kappa = 3 - d_x$

1. Initialize state vector and covariance

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

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

This step is the exact same as the first step of the KF and the EKF. We need to initialize these values so the model can begin generating and correcting predictions.

2. Calculate sigma points

Sigma points characterize the distribution of the data. The number of sigma points is deterministic and depends on the dimensions of the system. In general, a UFK will have $2 \cdot d_x + 1$ sigma points, where d_x represents the dimension of the state vector [**?**, **?**, **?**]. We use the equation below to generate a scalar value that determines how spread out the sigma points are from the mean.

$$\lambda = \alpha^2(d_x + \kappa) - d_x$$

α and κ are both parameters that control for the spread of sigma points around the mean value of the state. The spread of the sigma points is proportional to α . For both α and κ, the smaller the values are, the closer the sigma points are to the mean.

β is a parameter that uses information regarding state distribution to adjust sigma points. β has a default value of 2 if the data is Gaussian.

$$\chi_{\ 0,k-1} = \hat{x}_{k-1}$$

Since the goal is to characterize the distribution, set one of these sigma points to the mean. Half of the remaining points will be smaller than the mean and the other half will be larger than the mean.

$$\chi_{\ i,k-1} = \hat{x}_{k-1} + \left(\sqrt{(d_x + \lambda)P_{x_{k-1}}}\right)_i \qquad i = 1, \ldots, d_x$$

The above equation calculates half of the sigma points that are above the mean. Note that $(\sqrt{(d_x + \lambda)P_{x_{k-1}}})$ is a matrix, and the i subscript is the i^{th} column of the matrix. Also note that $\chi_{\ i,k-1}$ is the i^{th} column of the sigma point matrix at time $k - 1$.

$$\chi_{\ i,k-1} = \hat{x}_{k-1} - \left(\sqrt{(d_x + \lambda)P_{x_{k-1}}}\right)_i \qquad i = d_x + 1, \ldots, 2d_x$$

The above equation calculates half of the sigma points that are below the mean.

Therefore, in total we achieve $(2 \cdot d_x + 1)$ sigma points that characterize the Gaussian distribution of the data.

3. Calculate the weights for each sigma point

Weights are scalars used to calculate posterior sigma points after they have undergone a nonlinear transformation. The weights are later used to approximate Gaussian mean and covariance. Weights can have positive or negative values, but will ultimately sum to 1 [**?**]. The subscript of the weight indicates which sigma point the weight is for.

$$W_0^{(m)} = \frac{\lambda}{d_x + \lambda}$$

This is the weight for the mean of the 0th sigma point.

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

This is the weight for the covariance of the 0th sigma point.

$$W_i^{(m)} = W_i^{(c)} = \frac{\lambda}{2(d_x + \lambda)} \qquad i = 1, \ldots, 2d_x$$

The rest of the weights for the means and covariances of the other $2 \cdot d_x$ sigma points is calculated above.

4. Perform nonlinear transformation on sigma points

$$\chi_{k|k-1} = f(\chi)$$

Though χ has a Gaussian distribution, $\chi_{k|k-1}$ does not because it has been transformed by the nonlinear state function f.

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

$$P_{x,k|k-1} = \sum_{i=0}^{2d_x} W_i^{(c)} (\chi_{i,k|k-1} - \hat{x}_{i,k|k-1})(\chi_{i,k|k-1} - \hat{x}_{i,k|k-1})^T + Q$$

The posterioir covariance matrix for the state variable is necessary for updating the state covariance later on (step 9)Recall that Q is process noise, which provides the error in our model f.

5. Calculate posterior sigma points

This process is necessary for interpreting the distribution. From the non linear transformation above, the outputs are not Gaussian. Using the weights calculated in step 3, we are able to approximate the Gaussian distribution of the transformed sigma points.

$$\chi_{0,k|k-1} = \hat{x}_{k|k-1}$$

$$\chi_{i,k|k-1} = \hat{x}_{k|k-1} \pm \left(\sqrt{(d_x + \lambda) P_{x_{k|k-1}}} \right)_i \qquad i = 1, \ldots, 2d_x$$

6. Calculate transformed output

$$\mathcal{Y}_{k|k-1} = h(\chi_{k|k-1}, u)$$

$$y_{k|k-1} = \sum_{i=0}^{2d_x} W_i^{(m)} \mathcal{Y}_{i,k|k-1}$$

7. Calculate Kalman Gain

$$P_{y,k|k-1} = \sum_{i=0}^{2d_x} W_i^{(c)} (\mathcal{Y}_{i,k|k-1} - \hat{y}_{i,k|k-1})(\mathcal{Y}_{i,k|k-1} - y_{i,k|k-1})^T + R$$

$$P_{x,y,k|k-1} = \sum_{i=0}^{2d_x} W_i^{(c)} (\chi_{i,k|k-1} - \hat{x}_{i,k|k-1})(\mathcal{Y}_{i,k|k-1} - y_{i,k|k-1})^T$$

$$K_k = P_{xy,k|k-1}(P_{y,k|k-1})^{-1}$$

8. Correct the prediction

$$\hat{x}_k = \hat{x}_{k|k-1} + K_k(y_k - y_{k|k-1})$$

9. Update the covariance matrix

$$P_{x,k} = P_{x,k|k-1} - K_k(P_{y,k|k-1}){K_k}^T$$

4.2 Van Der Pol Example

4.3 A Metabolites Example

A metabolite is small structure that is the byproduct of the body's metabolism. To better understand the path of metabolites, an UKF was utilized for biomedical pathway modeling. Data regarding metabolites is highly influenced by noise, which is an UKF can handle while other approaches, such as regression and annealing, fail. Since the model is complex, considering that there are many unknown parameters and many metabolites in the system, utilizing an UKF is more effective than an EKF. Initial data contained information about metabolites at various time steps.

Variables in the Unscented Kalman Filter		
Variable	Description	Dimensions
x	Vector containing state variables	$d_x \times 1$
χ	Sigma Point Matrix	$d_x \times (2d_x + 1)$
F	State Transition Matrix	$d_x \times d_x$
H	Observation Matrix	$d_y \times d_x$
G	Input Matrix	$d_x \times d_u$
u	Input Vector	$d_u \times 1$
P	Covariance matrix	$d_x \times d_x$
k	Time step	1×1
w	Parameter vector	$d_q \times 1$
v	Process Noise Vector	$d_x \times 1$
α	Controls spread of sigma points	1
β	Adjust sigma point weight	$0 \leq \beta$
κ	Sigma point weighting constant	$3 - d_q = -14$
λ	Scaling parameter for weights	1

1. Initialize state vector and covariance

$$\hat{x}_0 = \mathbb{E}[x_0] = \begin{bmatrix} 4 \\ 1 \\ 3 \\ 4 \end{bmatrix} \tag{4.1}$$

$$P_{x_0} = \mathbb{E}[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T] = \begin{bmatrix} .1 & 0 & 0 & 0 \\ 0 & .1 & 0 & 0 \\ 0 & 0 & .1 & 0 \\ 0 & 0 & 0 & .1 \end{bmatrix} \tag{4.2}$$

2. Calculate sigma points

$$\chi_{\ 0,k-1} = \hat{x}_{k-1} \tag{4.3}$$

$$\chi_{\ i,k-1} = \hat{x}_{k-1} + (\sqrt{(n_x + \lambda_x) P_{x_{k-1}}})_i, \qquad i = 1, \ldots, n_x \tag{4.4}$$

$$\chi_{\ i,k-1} = \hat{x}_{k-1} - (\sqrt{(n_x + \lambda_x) P_{x_{k-1}}})_i, \qquad i = n_x + 1, \ldots, 2n_x, \tag{4.5}$$

3. Calculate the weights for each sigma point

   Weights can have positive or negative values, but will ultimately sum to 1 [**?**]

$$\lambda_x = \alpha_x^2(n_x + \kappa_x) - n_x \tag{4.6}$$

$$W_{x,0}^{(m)} = \frac{\lambda_x}{n_x + \lambda_x} \tag{4.7}$$

$$W_{x,0}^{(c)} = \frac{\lambda_x}{n_x + \lambda_x} + (1 - \alpha_x^2 + \beta_x) \tag{4.8}$$

$$W_{x,i}^{(m)} = W_{x,i}^{(c)} = \frac{\lambda_x}{2(n_x + \lambda_x)}, i = 1, \ldots, 2n_x \tag{4.9}$$

$$W_{x,0}^{\text{aug},(m)} = \frac{\lambda_x}{2n_x + \lambda_x} \tag{4.10}$$

$$W_{x,0}^{\text{aug},(c)} = \frac{\lambda_x}{2n_x + \lambda_x} + (1 - \alpha_x^2 + \beta_x) \tag{4.11}$$

$$W_{x,i}^{\text{aug},(m)} = W_{x,i}^{\text{aug},(c)} = \frac{\lambda_x}{2(2n_x + \lambda_x)}, i = 1, \ldots, 4n_x \tag{4.12}$$

$$\tag{4.13}$$

4. Perform a non-linear transformation on the sigma points

$$\hat{w}_k = \hat{w}_{k-1} \tag{4.14}$$

$$\chi^*_{i,k|k-1} = f(\chi_{i,k-1}, \hat{w}_{k-1}) \tag{4.15}$$

5. Calculate the mean and covariance of the transformed sigma points

$$\hat{x}^*_{k|k-1} = \sum_{i=0}^{2n_x} W_{x,i}^{(m)} \chi^*_{i,k|k-1} \tag{4.16}$$

$$P_{x_{k|k-1}} = \sum_{i=0}^{2n_x} W_{x,i}^{(c)} (\chi^*_{i,k|k-1} - \hat{x}^*_{k|k-1})(\chi^*_{i,k|k-1} - \hat{x}^*_{k|k-1})^T + R_v \tag{4.17}$$

6. Re-calculate sigma points

$$\chi_{i,k|k-1} = \chi^*_{i,k|k-1}, i = 0, ..., 2n_x \tag{4.18}$$

$$\chi_{i,k|k-1} = \chi^*_{0,k|k-1} + (\sqrt{(n_x + \lambda_x)P_{x_{k|k-1}}})_i, i = 2n_x + 1, \ldots, 3n_x \tag{4.19}$$

$$\chi_{i,k|k-1} = \chi^*_{0,k|k-1} - (\sqrt{(n_x + \lambda_x)P_{x_{k|k-1}}})_i, i = 3n_x + 1, \ldots, 4n_x \tag{4.20}$$

7. Generate prediction

$$\hat{x}_{k|k-1} = \sum_{i=0}^{4n_x} W_{x,i}^{\text{aug},(m)} \chi_{\ i,k|k-1} \tag{4.21}$$

$$\mathcal{Y}_{k|k-1} = h(\chi_{k|k-1}, \hat{w_k}) \tag{4.22}$$

$$\hat{y}_{k|k-1} = \sum_{i=0}^{4n_x} W_{x,i}^{\text{aug},(m)} \mathcal{Y}_{\ i,k|k-1} \tag{4.23}$$

8. Calculate Kalman Gain

$$P_{y_k} = \sum_{i=0}^{4n_x} W_{x,i}^{\text{aug},(c)} (\mathcal{Y}_{i,k|k-1} - \hat{y}_{k|k-1})(\mathcal{Y}_{i,k|k-1} - \hat{y}_{k|k-1})^T + R_n \tag{4.24}$$

$$P_{x_k y_x} = \sum_{i=0}^{4n_x} W_{x,i}^{\text{aug},(c)} (\chi_{\ i,k|k-1} - \hat{x}^*{}_{k|k-1})(\mathcal{Y}_{i,k|k-1} - \hat{y}_{k|k-1})^T \tag{4.25}$$

$$K_k = P_{x_k y_x} P_{y_k}^{-1} \tag{4.26}$$

9. Update Estimate and state covariance

$$P_{x_k} = P_{x_{k|k-1}} - K_k P_{y_k} K_k^{-1} \tag{4.27}$$

$$\hat{x}_k = \hat{x}_{k|k-1} + K_k(y_k - \hat{y}_{k|k-1}), \tag{4.28}$$

4.4 A Diabetes Example

Data assimilation (DA) uses mechanistic and empirical data to train machine learning models to make predictions. In the context of type 2 diabetes, DA was effective in providing recommendations on maintaining certain glucose thresholds, smoothing gaps in sparse finger-prick data, and estimating the state of patient phenotypes through constant estimation of physiological parameters. In this thesis, we explore how DA utilizes Bayesian Inference, a process of gauging the probability of an event, in this case glucose levels, given the known probabilities of events from the empirical data. Accuracy also depends on the estimation of relevant physiological processes; the mechanistic model uses differential equations to represent these processes by monitoring 3 variables (remote insulin, plasma insulin, and glucose) in estimating exchange rates. DA allows empirical and physiological data to work in tandem, resulting prognoses that are not only more accurate than other models, but also personalized for the patient and requiring less data. Further research in implementing DA in biomedicine has the potential to provide better patient care.

Bibliography

[1] J. Gove and DY Hollinger. Application of a dual unscented kalman filter for simultaneous state and parameter estimation in problems of surface-atmosphere exchange. *Journal of Geophysical Research*, 111:21 PP.–21 PP., 04 2006.

[2] Matthew Rhudy, Roger Salguero, and Keaton Holappa. A kalman filtering tutorial for undergraduate students. *International Journal of Computer Science & Engineering Survey*, 08:01–18, 02 2017.

[3] Esra Saatci and Aydin Akan. *Dual Unscented Kalman Filter and Its Applications to Respiratory System Modeling*, page 4. intechopen, 04 2009.

[4] Eric Wan and Ronell Merwe. The unscented kalman filter for nonlinear estimation. In *The Unscented Kalman Filter for Nonlinear Estimation*, volume 153-158, pages 153 – 158, 02 2000.

[5] Eric A. Wan and Rudolph Van Der Merwe. The unscented kalman filter. In *Kalman Filtering and Neural Networks*, pages 221–280. Wiley, 2001.

Appendix A

Van Der Pol Code

Appendix B

Meskin Code

B.1 Main function

```
initialStateGuess = [4;1;3;4]; % xhat[k|k-1]

% Construct the filter
ukf = unscentedKalmanFilter(...
    @MeskinStateFcn,... % State transition function
    @MeskinMeasurementNonAddFcn,... % Measurement
       function
    initialStateGuess,...
    'HasAdditiveMeasurementNoise',false)%,...
    %'alpha', 0.9);

R = .001; % Variance of the measurement noise v[k]
ukf.MeasurementNoise = R;

ukf.ProcessNoise = diag([0.02 0.01 .03 .04]); %stores the
    process noise covariance

T = 0.1; % [s] Filter sample time
timeVector = 0:T:5;
[~,xTrue]=ode45(@Meskin1,timeVector,initialStateGuess);

rng(1); % Fix the random number generator for
   reproducible results
yTrue = xTrue(:,1);
yMeas = yTrue + (sqrt(R)*randn(size(yTrue)));
    % sqrt(R): Standard deviation of noise
    % randn(size(yTrue)): randomly sample 51 elements
       from ytrue
```

```
%yMeas = yTrue;

%Nsteps = numel(yMeas); % Number of time steps
%xCorrectedUKF = zeros(Nsteps,4); % Corrected state
    estimates
%PCorrected = zeros(Nsteps,4,4); % Corrected state
    estimation error covariances
%e = zeros(Nsteps,1); % Residuals (or innovations)

for k=1:numel(yMeas)
    % Let k denote the current time.

    % Residuals (or innovations): Measured output -
        Predicted output

    %e(k) = yMeas(k) - MeskinMeasurementFcn(ukf.State); %
         ukf.State is x[k|k-1] at this point

    % Incorporate the measurements at time k into the
        state estimates by
    % using the "correct" command. This updates the State
         and StateCovariance
    % properties of the filter to contain x[k|k] and P[k|
        k]. These values
    % are also produced as the output of the "correct"
        command.

    %[xCorrectedUKF(k,:), PCorrected(k,:,:)] = correct(
        ukf, yMeas(k));
     [xCorrectedUKF(k,:), PCorrected(k,:,:)] = correct(ukf
        , yMeas(k));
    %[xCorrectedUKF(k,:,:), PCorrected(k,:,:,:)] =
        correct(ukf, yMeas(k, 2));
    %for j=1:numel(yMeas)
    %[xCorrectedUKF(k,:), PCorrected(k,k,:)] = correct(
        ukf, yMeas(j, 1));
    %[xCorrectedUKF(k+1,:), PCorrected(k+1,:,:)] =
        correct(ukf, yMeas(j, 2));
    %[xCorrectedUKF(k+2,:), PCorrected(k+2,:,:)] =
        correct(ukf, yMeas(j, 3));
    %[xCorrectedUKF(k+3,:), PCorrected(k+3,:,:)] =
        correct(ukf, yMeas(j, 4));
    %end

    %xCorrectedUKF(:,1) = first col
    %xCorrectedUKF(1,:) = first row
    %xCorrectedUKF(k,:) = kth row row
    %PCorrected(k,:,:) = first row of each of the 4
        matrices

    %There are 4 51x4 state covariances


    % Predict the states at next time step, k+1. This
        updates the State and
    % StateCovariance properties of the filter to contain
         x[k+1|k] and
    % P[k+1|k]. These will be utilized by the filter at
        the next time step.
    predict(ukf);

end


figure();

subplot(2,3,1);
plot(timeVector,xTrue(:,1), '-b',timeVector,xCorrectedUKF
    (:,1), '--b',...
    timeVector,yMeas(:), '*b',...
    'LineWidth', 3);
set(gca, 'FontSize', 20);
legend('True','UKF estimate','Measured')
ylim([0 5]);
ylabel('x_1', 'FontSize', 20);

subplot(2,3,2);
plot(timeVector,xTrue(:,2),'-g', timeVector,xCorrectedUKF
    (:,2), '--g',...
    'LineWidth', 3);
set(gca, 'FontSize', 20);
ylim([0 5]);
xlabel('Time [s]', 'FontSize', 20);
ylabel('x_2', 'FontSize', 20);

subplot(2,3,3);
plot(timeVector,xTrue(:,3), 'r',timeVector,xCorrectedUKF
    (:,3), '--r',...
    'LineWidth', 3);
set(gca, 'FontSize', 20);
legend('True','UKF estimate','Measured')
%ylim([-2.6 2.6]);
```

```
ylabel('x_3', 'FontSize', 20);

subplot(2,3,4);
plot(timeVector,xTrue(:,4), 'c',timeVector,xCorrectedUKF(:,4), '--c',...
    'LineWidth', 3);
set(gca, 'FontSize', 20);
legend('True','UKF estimate','Measured')
%ylim([-2.6 2.6]);
ylabel('x_4', 'FontSize', 20);

subplot(2,3,5);
plot(timeVector,xTrue(:,1), '-b',timeVector,xCorrectedUKF(:,1), '--b',...
     timeVector,xTrue(:,2), '-g',timeVector,xCorrectedUKF(:,2), '--g',...
     timeVector,xTrue(:,3), '-r',timeVector,xCorrectedUKF(:,3), '--r',...
     timeVector,xTrue(:,4), '-c',timeVector,xCorrectedUKF(:,4), '--c',...
    'LineWidth', 3);
set(gca, 'FontSize', 20);
legend('x1 True','x1 UKF','x2 True', 'x2 UKF', 'x3 True', 'x3 UKF','x4 True', 'x4 UKF')
ylim([0 5]);
ylabel('x_4', 'FontSize', 20);
```

Appendix C

Albers Code