



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

<b>1</b>	<b>Introduction</b>	<b>ii</b>
<b>2</b>	<b>Kalman Filters</b>	<b>iii</b>
2.1	Kalman Filter Algorithm . . . . .	iv
<b>3</b>	<b>Extended Kalman Filters</b>	<b>vii</b>
3.1	Extended Kalman Filter Algorithm . . . . .	viii
<b>4</b>	<b>Unscented Kalman Filters</b>	<b>x</b>
4.1	Unscented Kalman Filter Algorithm . . . . .	xi
4.2	Van Der Pol Example . . . . .	xiv
4.3	A Metabolites Example . . . . .	xv
4.4	A Diabetes Example . . . . .	xviii
<b>5</b>	<b>Dual Unscented Kalman Filters</b>	<b>xix</b>
<b>A</b>	<b>Van Der Pol Code</b>	<b>xxiii</b>
<b>B</b>	<b>Meskin Code</b>	<b>xxiv</b>
B.1	Main function . . . . .	xxiv
<b>C</b>	<b>Albers Code</b>	<b>xxviii</b>

# 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 ( $x_{k+1}$ )

$$x_{k+1} = Fx_k + Gu_k + w_k$$

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 ( $K_k$ )

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

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

$H$  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 diagonals 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 ( $K_k$ )

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

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

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

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

## Chapter 4

# Unscented Kalman Filters

The Unscented Kalman Filter (UKF) does not require using Jacobians to linearize non-linear functions. Instead, the UKF uses the Unscented Transform (UT). This process involves using sigma points, which are represented in a Sigma Point Matrix and represent the normal distribution of the data. The number of sigma points is deterministic and depends on the dimensions of the system. In general, a UKF will have  $2d_x + 1$  sigma points, where  $d_x$  represents the dimension of the state vector  $[?, ?, ?]$ . 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.

## 4.1 Unscented Kalman Filter Algorithm

Variables in the Unscented Kalman Filter		
Variable	Description	Dimensions
$\mathbf{x}$	Vector containing state variables	$d_x \times 1$
$\chi$	Sigma Point Matrix	$d_x \times 2d_x + 1$
$\mathbf{F}$	State Transition Matrix	$d_x \times d_x$
$\mathbf{H}$	Observation Matrix	$d_y \times d_x$
$\mathbf{G}$	Input Matrix	$d_x \times d_u$
$\mathbf{u}$	Input Vector	$d_u \times 1$
$\mathbf{P}$	Covariance matrix	$d_x \times d_x$
$k$	Time step	$1 \times 1$
$\mathbf{w}$	Process Noise Vector	$d_x \times 1$
$\alpha$	Controls spread of sigma points	$0 < \alpha < 1$
$\beta$	Adjust sigma point weight	$0 \leq \beta$
$\kappa$	Sigma point weighting constant	$0, 3 - d_q, 0 \leq \kappa \leq 3$
$\lambda$	Scaling parameter for sigma point weights	scalar

1. Initialize state vector and covariance

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

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

2. Calculate sigma points

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

$$\chi_{i,k-1} = \hat{x}_{k-1} + (\sqrt{(n_x + \lambda_x)P_{x_{k-1}}})_i, \quad i = 1, \dots, n_x$$

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

3. Calculate the weights for each sigma point

$$\lambda_x = \alpha_x^2(n_x + \kappa_x) - n_x$$

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

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

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

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

4. Calculate the augmented weights for each sigma point

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

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

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

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

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

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

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

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

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

7. Re-calculate sigma points

$$\chi_{i,k|k-1} = \chi_{i,k|k-1}^*, i = 0, \dots, 2n_x$$

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

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

8. Generate prediction

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

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

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

9. Calculate Kalman Gain

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

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

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

10. Update Estimate and state covariance

$$P_{x_k} = P_{x_k|k-1} - K_k P_{y_k} K_k^{-1}$$

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

## 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
$\mathbf{x}$	Vector containing state variables	$d_x \times 1$
$\chi$	Sigma Point Matrix	$d_x \times (2d_x + 1)$
$\mathbf{F}$	State Transition Matrix	$d_x \times d_x$
$\mathbf{H}$	Observation Matrix	$d_y \times d_x$
$\mathbf{G}$	Input Matrix	$d_x \times d_u$
$\mathbf{u}$	Input Vector	$d_u \times 1$
$\mathbf{P}$	Covariance matrix	$d_x \times d_x$
$k$	Time step	$1 \times 1$
$\mathbf{w}$	Parameter vector	$d_q \times 1$
$\mathbf{v}$	Process Noise Vector	$d_x \times 1$
$\alpha$	Controls spread of sigma points	1
$\beta$	Adjust sigma point weight	$0 \leq \beta$
$\kappa$	Sigma point weighting constant	$3 - d_q = -14$
$\lambda$	Scaling parameter for weights	1

1. Initialize state vector and covariance

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

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

2. Calculate sigma points

$$\chi_{0,k-1} = \hat{x}_{k-1} \quad (4.3)$$

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

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

### 3. Calculate the weights for each sigma point

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

$$\lambda_x = \alpha_x^2(n_x + \kappa_x) - n_x \quad (4.6)$$

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

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

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

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

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

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

$$(4.13)$$

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

$$\hat{w}_k = \hat{w}_{k-1} \quad (4.14)$$

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

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

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

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

### 6. Re-calculate sigma points

$$\chi_{i,k|k-1} = \chi_{i,k|k-1}^*, i = 0, \dots, 2n_x \quad (4.18)$$

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

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

7. Generate prediction

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

$$\mathcal{Y}_{k|k-1} = h(\chi_{k|k-1}, \hat{w}_k) \quad (4.22)$$

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

8. Calculate Kalman Gain

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

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

$$K_k = P_{x_k y_x} P_{y_k}^{-1} \quad (4.26)$$

9. Update Estimate and state covariance

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

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

## 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.

## Chapter 5

# Dual Unscented Kalman Filters

Used for simultaneously estimating state and parameter in dynamical systems [?, ?, ?]. Assumes parameters are identifiable (true value can be obtained from a finite number of observations) [?]. Has 2 distinct filters: one for state and the other for parameter [?, ?]

Variables in the Dual Unscented Kalman Filter		
Variable	Description	Dimensions
x	Vector containing state variables	$d_x \times 1$
$\chi$	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	$1 \times 1$
v	Process Noise Vector	$d_x \times 1$
w		

1. Initialize state vector, parameter vector, covariance, and parameter covariance

$$\hat{x}_0 = \mathbb{E}[x_0] \quad (5.1)$$

$$\hat{w}_0 = \mathbb{E}[w_0] \quad (5.2)$$

$$P_{x_0} = \mathbb{E}[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T] \quad (5.3)$$

$$P_{w_0} = \mathbb{E}[(w_0 - \hat{w}_0)(w_0 - \hat{w}_0)^T] \quad (5.4)$$

2. Calculate sigma points

$$\mathcal{W}_{0,k|k-1} = \hat{w}_{k|k-1} \quad (5.5)$$

$$\mathcal{W}_{i,k|k-1} = \hat{w}_{k|k-1} + (\sqrt{(n_w + \lambda_w)P_{w_{k|k-1}}})_i, i = 1, \dots, n_w \quad (5.6)$$

$$\mathcal{W}_{i,k|k-1} = \hat{w}_{k|k-1} - (\sqrt{(n_w + \lambda_w)P_{w_{k|k-1}}})_i, i = n_w + 1, \dots, 2n_w \quad (5.7)$$

3. Calculate the weights for each sigma point

$$\lambda_x = \alpha_x^2(n_x + \kappa_x) - n_x \quad (5.8)$$

$$W_{x,0}^{(m)} = \frac{\lambda_x}{n_x + \lambda_x} \quad (5.9)$$

$$W_{x,0}^{(c)} = \frac{\lambda_x}{n_x + \lambda_x} + (1 - \alpha_x^2 + \beta_x) \quad (5.10)$$

$$W_{x,i}^{(m)} = W_{x,i}^{(c)} = \frac{\lambda_x}{2(n_x + \lambda_x)}, i = 1, \dots, 2n_x \quad (5.11)$$

$$W_{x,0}^{\text{aug},(m)} = \frac{\lambda_x}{2n_x + \lambda_x} \quad (5.12)$$

$$W_{x,0}^{\text{aug},(c)} = \frac{\lambda_x}{2n_x + \lambda_x} + (1 - \alpha_x^2 + \beta_x) \quad (5.13)$$

$$W_{x,i}^{\text{aug},(m)} = W_{x,i}^{\text{aug},(c)} = \frac{\lambda_x}{2(2n_x + \lambda_x)}, i = 1, \dots, 4n_x \quad (5.14)$$

$$\lambda_w = \alpha_w^2(n_w + \kappa_w) - n_w \quad (5.15)$$

$$W_{w,0}^{(m)} = \frac{\lambda_w}{n_w + \lambda_w} \quad (5.16)$$

$$W_{w,0}^{(c)} = \frac{\lambda_w}{n_w + \lambda_w} + (1 - \alpha_w^2 + \beta_w) \quad (5.17)$$

$$W_{w,i}^{(m)} = W_{w,i}^{(c)} = \frac{\lambda_w}{2(n_w + \lambda_w)}, i = 1, \dots, 2n_w \quad (5.18)$$

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

$$\mathcal{Y}_{i,k|k-1} = h(f(\hat{x}_{k-1}, \mathcal{W}_{i,k|k-1}), \mathcal{W}_{i,k|k-1}) \quad (5.19)$$

$$\hat{y}_{k|k-1} = \sum_{i=0}^{2n_w} W_{w,i}^{(m)} \mathcal{Y}_{i,k|k-1} \quad (5.20)$$

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

6. Re-calculate sigma points

7. Generate prediction

8. Calculate Kalman Gain

$$P_{y_k} = \sum_{i=0}^{2n_w} W_{w,i}^{(c)} (\mathcal{Y}_{i,k|k-1} - \hat{y}_{k|k-1}) (\mathcal{Y}_{i,k|k-1} - \hat{y}_{k|k-1})^T + R_n \quad (5.21)$$

$$P_{w_k y_x} = \sum_{i=0}^{2n_w} W_{w,i}^{(c)} (\mathcal{W}_{i,k|k-1} - \hat{w}_{k|k-1}) (\mathcal{Y}_{i,k|k-1} - \hat{y}_{k|k-1})^T \quad (5.22)$$

$$K_k = P_{w_k y_x} P_{y_k}^{-1} \quad (5.23)$$

9. Update parameter covariance

$$P_{w_k} = P_{w_k|k-1} - K_k P_{y_k} K_k^{-1} \quad (5.24)$$

10. Update parameter estimate

$$\hat{w}_k = \hat{w}_{k|k-1} + K_k (y_k - \hat{y}_{k|k-1}), \quad (5.25)$$

# Bibliography



## Appendix A

### Van Der Pol Code

## Appendix B

# Meskin Code

### B.1 Main function

```
1 initialStateGuess = [4;1;3;4]; % xhat[k|k-1]
2
3 % Construct the filter
4 ukf = unscentedKalmanFilter(...
5     @MeskinStateFcn,... % State transition function
6     @MeskinMeasurementNonAddFcn,... % Measurement
7     initialStateGuess,...
8     'HasAdditiveMeasurementNoise',false)%,...
9     %'alpha', 0.9);
10
11 R = .001; % Variance of the measurement noise v[k]
12 ukf.MeasurementNoise = R;
13
14 ukf.ProcessNoise = diag([0.02 0.01 .03 .04]); %stores the
15     process noise covariance
16
17 T = 0.1; % [s] Filter sample time
18 timeVector = 0:T:5;
19 [~,xTrue]=ode45(@Meskin1,timeVector,initialStateGuess);
20
21 rng(1); % Fix the random number generator for
22     reproducible results
23 yTrue = xTrue(:,1);
24 yMeas = yTrue + (sqrt(R)*randn(size(yTrue)));
25     % sqrt(R): Standard deviation of noise
26     % randn(size(yTrue)): randomly sample 51 elements
27     from ytrue
```

```

25 %yMeas = yTrue;
26
27 %Nsteps = numel(yMeas); % Number of time steps
28 %xCorrectedUKF = zeros(Nsteps,4); % Corrected state
    estimates
29 %PCorrected = zeros(Nsteps,4,4); % Corrected state
    estimation error covariances
30 %e = zeros(Nsteps,1); % Residuals (or innovations)
31
32 for k=1:numel(yMeas)
33     % Let k denote the current time.
34
35     % Residuals (or innovations): Measured output –
        Predicted output
36
37     %e(k) = yMeas(k) – MeskinMeasurementFcn(ukf.State); %
        ukf.State is x[k|k-1] at this point
38
39     % Incorporate the measurements at time k into the
        state estimates by
40     % using the "correct" command. This updates the State
        and StateCovariance
41     % properties of the filter to contain x[k|k] and P[k|
        k]. These values
42     % are also produced as the output of the "correct"
        command.
43
44     %[xCorrectedUKF(k,:), PCorrected(k,:,:) ] = correct(
        ukf, yMeas(k));
45     [xCorrectedUKF(k,:), PCorrected(k,:,:) ] = correct(ukf
        , yMeas(k));
46     %[xCorrectedUKF(k,:,:), PCorrected(k,:,:,)] =
        correct(ukf, yMeas(k, 2));
47     %for j=1:numel(yMeas)
48     %[xCorrectedUKF(k,:), PCorrected(k,k,:)] = correct(
        ukf, yMeas(j, 1));
49     %[xCorrectedUKF(k+1,:), PCorrected(k+1,:,:) ] =
        correct(ukf, yMeas(j, 2));
50     %[xCorrectedUKF(k+2,:), PCorrected(k+2,:,:) ] =
        correct(ukf, yMeas(j, 3));
51     %[xCorrectedUKF(k+3,:), PCorrected(k+3,:,:) ] =
        correct(ukf, yMeas(j, 4));
52 %end
53
54 %xCorrectedUKF(:,1) = first col
55 %xCorrectedUKF(1,:) = first row

```

```

56     %xCorrectedUKF(k,:) = kth row row
57     %PCorrected(k,:,:) = first row of each of the 4
        matrices
58
59     %There are 4 51x4 state covariances
60
61
62     % Predict the states at next time step, k+1. This
        updates the State and
63     % StateCovariance properties of the filter to contain
        x[k+1|k] and
64     % P[k+1|k]. These will be utilized by the filter at
        the next time step.
65     predict(ukf);
66
67 end
68
69
70 figure();
71
72 subplot(2,3,1);
73 plot(timeVector,xTrue(:,1), '—b',timeVector,xCorrectedUKF
   (:,1), '—b',...
74     timeVector,yMeas(:, ' *b',...
75     'LineWidth', 3);
76 set(gca, 'FontSize', 20);
77 legend('True','UKF estimate','Measured')
78 ylim([0 5]);
79 ylabel('x_1', 'FontSize', 20);
80
81 subplot(2,3,2);
82 plot(timeVector,xTrue(:,2), '—g', timeVector,xCorrectedUKF
   (:,2), '—g',...
83     'LineWidth', 3);
84 set(gca, 'FontSize', 20);
85 ylim([0 5]);
86 xlabel('Time [s]', 'FontSize', 20);
87 ylabel('x_2', 'FontSize', 20);
88
89 subplot(2,3,3);
90 plot(timeVector,xTrue(:,3), 'r',timeVector,xCorrectedUKF
   (:,3), '—r',...
91     'LineWidth', 3);
92 set(gca, 'FontSize', 20);
93 legend('True','UKF estimate','Measured')
94 %ylim([-2.6 2.6]);

```

```

95 ylabel('x_3', 'FontSize', 20);
96
97 subplot(2,3,4);
98 plot(timeVector,xTrue(:,4), 'c',timeVector,xCorrectedUKF
99      (:,4), '—c',...
100      'LineWidth', 3);
101 set(gca, 'FontSize', 20);
102 legend('True','UKF estimate','Measured')
103 %ylim([-2.6 2.6]);
104 ylabel('x_4', 'FontSize', 20);
105
106 subplot(2,3,5);
107 plot(timeVector,xTrue(:,1), '—b',timeVector,xCorrectedUKF
108      (:,1), '—b',...
109      timeVector,xTrue(:,2), '—g',timeVector,xCorrectedUKF
110      (:,2), '—g',...
111      timeVector,xTrue(:,3), '—r',timeVector,xCorrectedUKF
112      (:,3), '—r',...
113      timeVector,xTrue(:,4), '—c',timeVector,xCorrectedUKF
114      (:,4), '—c',...
115      'LineWidth', 3);
116 set(gca, 'FontSize', 20);
117 legend('x1 True','x1 UKF','x2 True','x2 UKF','x3 True',
118      'x3 UKF','x4 True','x4 UKF')
119 ylim([0 5]);
120 ylabel('x_4', 'FontSize', 20);

```

## Appendix C

# Albers Code