2016

State Estimation of Glucose and Insulin Dynamics

Morgan Nicholas Miller

State Estimation of Glucose and Insulin Dynamics

Morgan Miller

Bachelor of Chemical Engineering

Cleveland State University

May, 2015

Submitted in partial fulfillment of requirements for the degree

MASTER OF SCIENCE IN CHEMICAL ENGINEERING

at

CLEVELAND STATE UNIVERSITY

August, 2016

We hereby approve this thesis for

Morgan Miller

Candidate for the Master of Science in Chemical Engineering degree for the

Department of Chemical and Biomedical Engineering

and the CLEVELAND STATE UNIVERSITY

College of Graduate Studies

____

Thesis Chairperson, Dr. Sridhar Ungarala

____

Department & Date

____

Thesis Committee Member, Dr. Jorge Gatica

____

Department & Date

____

Thesis Committee Member Dr. Rolf Lustig

____

Department & Date

Student's Date of Defense: 06/30/2016

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Sridhar Ungarala for his guidance, patience, and time throughout this process.

I would like to thank my thesis committee members Dr. Jorge Gatica and Dr. Rolf Lustig for their time.

I would like to thank my family for their support and for keeping me on track for the past few years.

STATE ESTIMATION OF GLUCOSE AND INSULIN DYNAMICS

MORGAN MILLER

ABSTRACT

Process simulation and state estimation have very important applications in chemical engineering as well as the biomedical field. Diabetes is a rapidly growing disease in the United States with 29 million people already diagnosed. The estimation of glucose and insulin concentration in patients is necessary in order to effectively treat diabetes. The Bergman Minimal Model is a popular process model that is used to simulate glucose and insulin dynamics. A simulation of this model was created based on estimated parameters for the model from historical data. This thesis investigated the estimation of glucose concentration, insulin concentration, and effect of active insulin using the extended Kalman filter, unscented Kalman filter, ensemble Kalman filter, and sequential Monte Carlo Particle filter. The performance of the filters was compared using root mean squared error. The filters were studied for the cases of good filter initialization, poor filter initialization, plant-model mismatch, increased measurement noise, and multiple glucose ingestions.

TABLE OF CONTENTS

ABSTRACT iv

NOMENCLATURE vii

LIST OF TABLES viii

LIST OF FIGURES ix

CHAPTER I 1

INTRODUCTION 1

1.1 Diabetes 2

1.2 Control of Diabetes 3

1.3 Motivation 5

1.4 Scope of the Thesis 5

1.5 Organization of the Thesis 6

CHAPTER II 8

GLUCOSE AND INSULIN MODEL 8

CHAPTER III 15

STATE ESTIMATION 15

3.1 Kalman Filter 15

3.2 Extended Kalman Filter 16

3.3 Unscented Kalman Filter 17

3.4 Ensemble Kalman Filter 21

3.5 Particle Filter 22

CHAPTER IV 25

SIMULATION STUDY 25

4.1 Good Filter Initialization 28

4.2 Poor Filter Initialization 33

4.3 Plant-Model Mismatch 38

4.4 Increased Measurement Noise 43

4.5 Multiple Glucose Ingestions 47

CHAPTER V 53

CONCLUSION AND FUTURE WORK 53

REFERENCES 56

APPENDIX 59

NOMENCLATURE

h(x)	measurement function
G_b	Basal glucose concentration
I_b	Basal insulin concentration
K_k	Kalman gain
Q	Process covariance
R	Measurement covariance
RMSE	Root mean squared error
S_G	Glucose effectiveness
S_I	Insulin sensitivity
x_k	State vector
y_k	Measurement

Greek Symbols

α	tuning parameter
β	tuning parameter
κ	tuning parameter
λ	tuning parameter
γ	insulin responsivity

LIST OF TABLES

Table **Page**

Table I: Typical glucose concentrations for normal, pre-diabetic, and diabetic patients [7] 3

Table II: Minimal model parameters 25

Table IV: Good filter initialization, high confidence filter performance 30

Table V: Good filter initialization, low confidence 33

Table VI: Poor filter initialization, high confidence 35

Table VII: Poor filter initialization, low confidence 38

Table VIII: Plant-model mismatch, insulin sensitivity 40

Table IX: Elderly patient minimal model parameters 41

Table X: Plant-model mismatch, elderly patient 43

Table XI: Measurement noise 10 mg/dL 45

Table XII: Measurement noise 15 mg/dL 47

Table XIII: Multiple glucose ingestions, filter does not know 50

Table XIV: Multiple glucose ingestions, filter knows 52

LIST OF FIGURES

Figure **Page**

Figure 1: Continuous glucose monitor illustration [8] 4

Figure 2: Measured glucose and insulin concentrations vs. time [11] 9

Figure 3: Physiological representation for glucose and insulin dynamics [12] 10

Figure 4: Symmetric sigma points for the Minimal Model system [16] 18

Figure 5: Glucose concentration vs. time 26

Figure 6: Effect of active insulin vs. time 26

Figure 7: Insulin concentration vs. time 27

Figure 8: Good filter initialization, high confidence, glucose concentration vs. time .. 29

Figure 9: Good filter initialization, high confidence, effect of active insulin vs. time .. 29

Figure 10: Good filter initialization, high confidence, insulin concentration vs. time . 30

Figure 11: Good filter initialization, low confidence, glucose concentration vs. time .. 31

Figure 12: Good filter initialization, low confidence, effect of active insulin vs. time .. 32

Figure 13: Good filter initialization, low confidence, insulin concentration vs. time ... 32

Figure 14: Poor filter initialization, high confidence, glucose concentration vs. time . 34

Figure 15: Poor filter initialization, high confidence, effect of active insulin vs. time . 34

Figure 16: Poor filter initialization, high confidence, insulin concentration vs. time .. 35

Figure 17: Poor filter initialization, low confidence, glucose concentration vs. time ... 36

Figure 18: Poor filter initialization, high confidence, effect of active insulin vs. time . 37

Figure 19: Poor filter initialization, high confidence, insulin concentration vs. time .. 37

Figure 20: Plant-model mismatch: insulin sensitivity, glucose concentration vs. time 39

Figure 21: Plant-model mismatch: insulin sensitivity, effect of active insulin vs. time 39

Figure 22: Plant-model mismatch: insulin sensitivity, insulin concentration vs. time. 40

Figure 23: Plant-model mismatch: elderly patient, glucose concentration vs. time..... 41

Figure 24: Plant-model mismatch: elderly patient, effect of active insulin vs. time..... 42

Figure 25: Plant-model mismatch: elderly patient, insulin concentration vs. time...... 42

Figure 26: Increased measurement noise: 10 mg/dL, glucose concentration vs. time . 44

Figure 27: Increased measurement noise: 10 mg/dL, effect of active insulin vs. time . 44

Figure 28: Increased measurement noise: 10 mg/dL, insulin concentration vs. time .. 45

Figure 29: Increased measurement noise: 15 mg/dL, glucose concentration vs. time . 46

Figure 30: Increased measurement noise: 15 mg/dL, effect of active insulin vs. time . 46

Figure 31: Increased measurement noise: 15 mg/dL, insulin concentration vs. time .. 47

Figure 32: Multiple glucose ingestions, filter does not know, glucose concentration vs. time.. 48

Figure 33: Multiple glucose ingestions, filter does not know, effect of active insulin vs. time.. 49

Figure 34: Multiple glucose ingestions, filter does not know, insulin concentration vs. time.. 49

Figure 35: Multiple glucose ingestions, filter knows, glucose concentration vs. time .. 51

Figure 36: Multiple glucose ingestions, filter knows, effect of active insulin vs. time .. 51

Figure 37: Multiple glucose ingestions, filter knows, insulin concentration vs. time ... 52

CHAPTER I

INTRODUCTION

Process simulation allows for the testing of experiments that would ordinarily be impractical to be performed physically as a result of financial, safety, or legal constraints. Instead of a physical experiment, a model is developed for the process and tested using computer simulation. Most variables that are desired to be known about a process cannot be measured directly and are correlated to other measurements from sensors such as current or voltage. The methods and techniques for state estimation have existed for several decades but have gained recent interest in the chemical engineering field. This is due to the need of discrete analytical models to solve engineering problems where accurate theoretical models either do not exist or cannot be solved practically [1]. Measurements in a chemical process are subject to errors and noise to the point where the laws of conservation of mass and energy are not observed from these noisy measurements [2]. This is a major problem that can affect the monitoring and control of a process [3]. Measurements need to be filtered to reduce the impact of the noise in order to have a better estimate for the true state of a process. In the area of chemical process control, state estimation can be applied to estimate state variables in distillation columns, biomedical devices, or continuously stirred tank reactors.

A main area of interest in state estimation are biomedical applications. One of these biomedical applications is the estimation of blood glucose and insulin dynamics due to prevalence of diabetes in the population.

1.1 Diabetes

Diabetes is a rapidly growing disease in the United States. 29 million people in the United States are diabetic and 86 million people are pre-diabetic, which means they are very likely to become diabetic unless radical changes are made to their diet and exercise routine [4]. The combined amount of people that are diabetic or pre-diabetic accounts for nearly one third of the population in the United States. In type 1 diabetes, the pancreas cannot produce enough insulin to control the glucose concentration in the blood stream [4]. In type 2 diabetes, the body has become resistant to the insulin that is produced by the pancreas [4]. In both cases of the disease, the body is unable to effectively regulate the glucose concentration in the blood. This can have serious medical consequences. If the glucose concentrations in the blood are too high over long periods of time, there can be damage to the eyes including blindness, damage to the kidneys, nerve damage, heart disease can result, and the amputation of a limb may be required [5].

One of the techniques that is used for diagnosing diabetes is the glucose tolerance test. Patients fast for at least 8 hours before the test and have a blood sample taken initially. This sample represents the fasting or basal glucose concentration. This can be thought of as a steady state glucose concentration that the body always returns to after a meal. The patients are then given a concentrated sugar solution or an injection of glucose and their blood is drawn at pre-determined intervals over a period of approximately 3 hours [6]. Table 1 below summarizes the typical ranges of concentrations for individuals who are healthy, pre-diabetic, or diabetic.

	Fasting (mg/dL)	Immediately after meal (mg/dL)	2-3 hours after eating (mg/dL)
Normal	80-100	170-200	120-140
Pre-Diabetic	101-125	190-230	140-160
Diabetic	>125	220-300	>200

Table I: Typical glucose concentrations for normal, pre-diabetic, and diabetic patients *[7]*

1.2 Control of Diabetes

The most common treatments for diabetes involve the injection of insulin through needles or insulin pumps. Insulin injections are given at predetermined times throughout the day. Insulin pumps deliver a continuous dosage throughout the day known as the basal dosage which is determined by a physician. At meal times, the dosage is changed to a bolus dosage which is higher than the basal dosage since it needs to compensate for the increased glucose levels in the blood as a result of the meal. These dosages are based on a “guess” of the proper insulin dosage that would respond to what the glucose concentrations will rise to and is adjusted by the patient based on the resulting glucose measurements from either a continuous glucose monitor or a glucose test strip. Insulin concentrations cannot be measured in a nonclinical setting. A glucose test strip involves drawing a small sample of blood from the fingertip. A continuous glucose monitor samples interstitial tissue fluid from an electrode implanted underneath the skin [8].

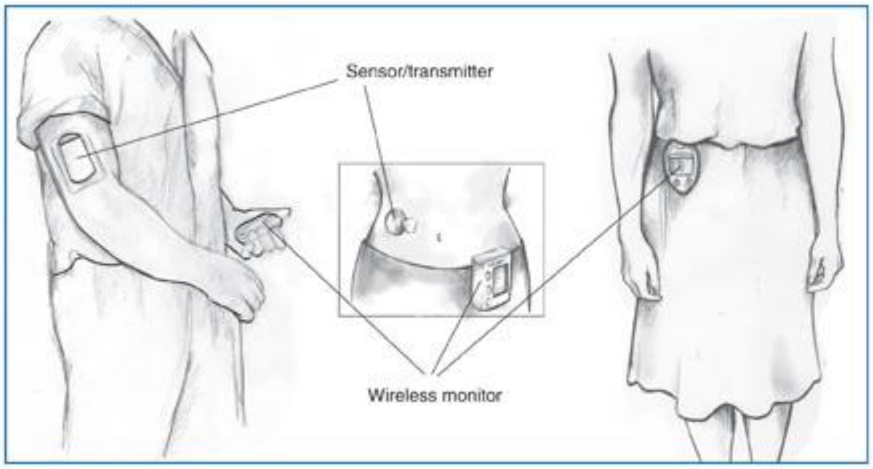

Figure 1: Continuous glucose monitor illustration *[8]*

While there are insulin pumps available on the market that have been coupled with continuous glucose monitors, they do not function as feedback process controllers [9]. Rather, the insulin pump takes action only when the glucose concentration is too low or too high to avoid periods of hypoglycemia or hyperglycemia. These conditions can result in confusion, falls, seizures, coma, or death [10]. This is not an optimal design since the insulin dosage being provided to the patient may not accurately reflect the actual dosage that is required to reach their normal glucose levels which could result in frequent oscillations in glucose concentration along with wasted insulin.

1.3 Motivation

In order to ensure the proper dosage of insulin is being administered at all times, a feedback loop needs to be created where measurements from a glucose monitor are used to estimate the true glucose and insulin concentrations. This feedback loop process controller can replace the function of the pancreas in a diabetic and may be referred to as an artificial pancreas.

1.4 Scope of the Thesis

This thesis investigates the use of the extended Kalman filter, unscented Kalman filter, ensemble Kalman filter, and particle filter to estimate the glucose concentration, insulin concentration, and effect of active insulin in the human body.

The performance of the extended Kalman filter, unscented Kalman filter, ensemble Kalman filter, and particle filter is investigated for the following cases:

- Good filter initialization: The performance of the filters is compared when the initial estimate of the state is very close to the actual state. This is done for cases of both high and low confidence in the initial estimate which is reflected in adjusting the value for the initial covariance matrix for the filter.
- Poor filter initialization: The performance of the filters is compared for the scenario when the initial estimate of the state is distant from the actual state. This is also done for the cases of both high and low confidence in the initial estimate which is reflected in the initial covariance matrix.

- Plant-Model mismatch: The performance of the filters is compared for the scenario where the model for the system used in the state estimator does not accurately describe the true state. There are 2 cases of plant-model mismatch. The 1st case is where the value for insulin sensitivity for the estimator is not equal to the actual insulin sensitivity of the patient. The 2nd case where the system is changed to an elderly patient while the estimator uses the model parameter values for a normal patient.
- Large measurement noise: The performance is compared for 3 levels of signal noise from the glucose measurement.
- Multiple glucose ingestions: The performance of the filters is compared for the scenario where a 2nd glucose ingestion occurs 60 minutes into the simulation. The filters are compared for the case when the filter knows the ingestion has taken place and when the filter does not know the ingestion has taken place.

1.5 Organization of the Thesis

This thesis has been organized in the following manner. The mathematical model for glucose and insulin dynamics is discussed in Chapter 2. Chapter 3 explains the different types of filters used for state estimation. This includes the different variants of the Kalman filter along with the Monte Carlo based particle filter. The results of the simulation study and comparison of the filters are presented in Chapter 4. Chapter 5 presents the conclusions and provides the direction for future work.

CHAPTER II

GLUCOSE AND INSULIN MODEL

In 1986, Richard Bergman and Giovanni Pacini developed a model to describe glucose and insulin dynamics known as the Minimal Model [11]. The Minimal Model can be used to describe the glucose and insulin dynamics of healthy people not affected by diabetes as well as diabetics. This is accomplished by adjusting the parameter values introduced in the model to fit recorded patient data. Since the Minimal Model was first developed, there have been over 100 technical reports published regarding the model [12]. The accuracy of the model was tested by performing intravenous glucose tolerance tests on humans. The patients were given an initial injection of 0.3 g/kg of glucose. This means that the dosages were normalized to account for the different weights of patients in the study. Their glucose and insulin levels were then measured for approximately 3 hours to obtain information on the dynamic behavior of the glucose and insulin system. Frequent samples of blood were drawn from the patients to obtain measurements of the glucose and insulin concentrations. This was done in order to establish the shape of the concentration profiles over time. The measured data along with the model that was fit to the data can be seen in Figure 2. The basal glucose and insulin concentrations are plotted as dashed lines. It can be seen that after the initial peak from the glucose injection, the levels undershoot the basal levels and then oscillate around the basal levels. The insulin concentration is reported in units of micro units per milliliter which is a standardized unit of measure in the medical field.

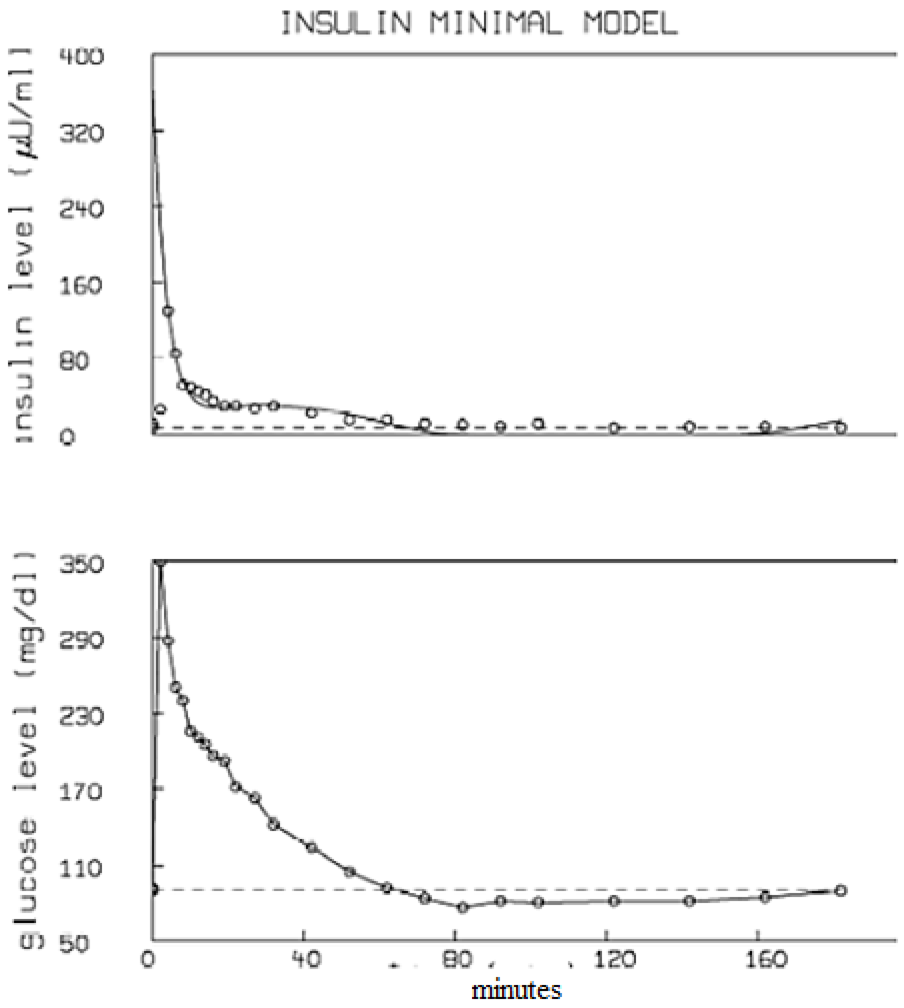

Figure 2: Measured glucose and insulin concentrations vs. time *[11]*

The model assumed that glucose returns to its basal concentration due to the effect of glucose to regulate its concentration by itself and due to the effect of insulin. Figure 3 shows a physiological representation of the model proposed by Bergman.

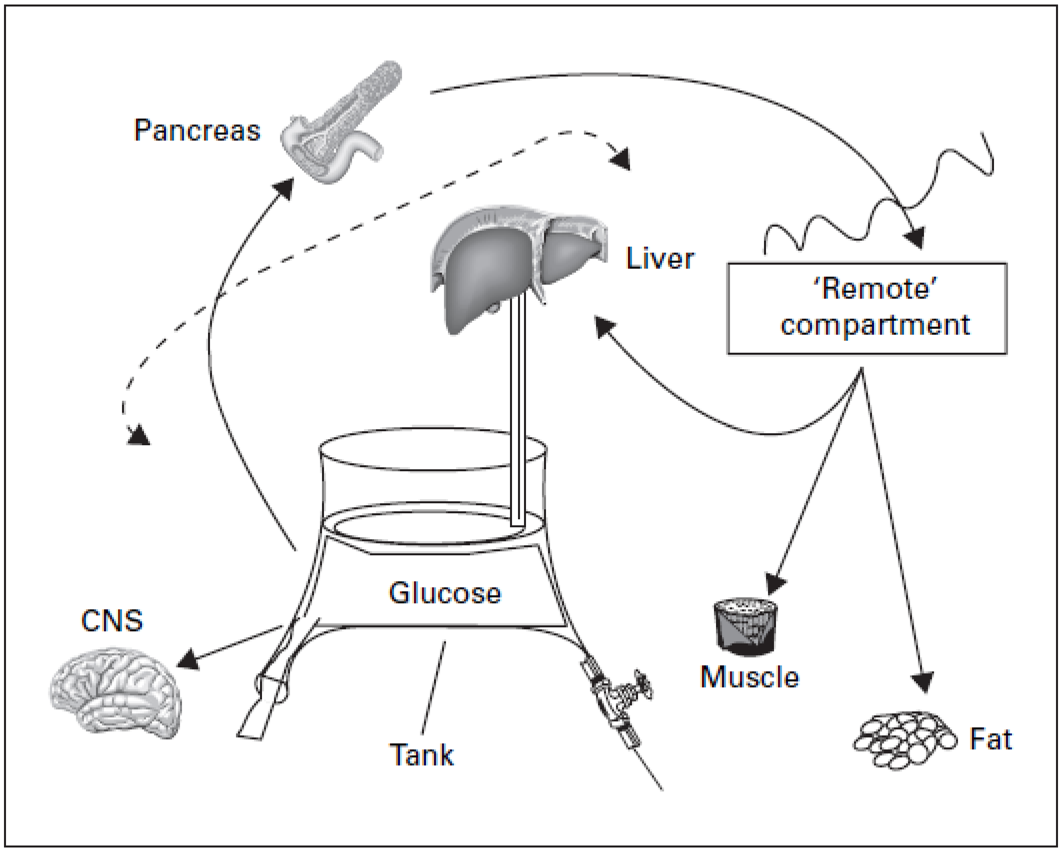

Figure 3: Physiological representation for glucose and insulin dynamics *[12]*

The glucose concentration is represented by the level in the tank in figure 3. The level can change due to production by the liver or utilization by the body including the central nervous system and muscles. The pancreas responds to the glucose level by producing insulin which is transported from the bloodstream to the "remote" compartment which is interstitial tissue where it can increase the glucose uptake by the muscles or fat tissue or reduce the glucose production by the liver. After a meal or an injection of glucose, the glucose level rises which causes a response by the pancreas. The pancreas produces insulin which is transferred into the interstitial tissue where it acts to return the glucose to the basal level. This creates a biological feedback loop. An analogous process control block diagram for the Minimal Model can be seen in Figure 4.

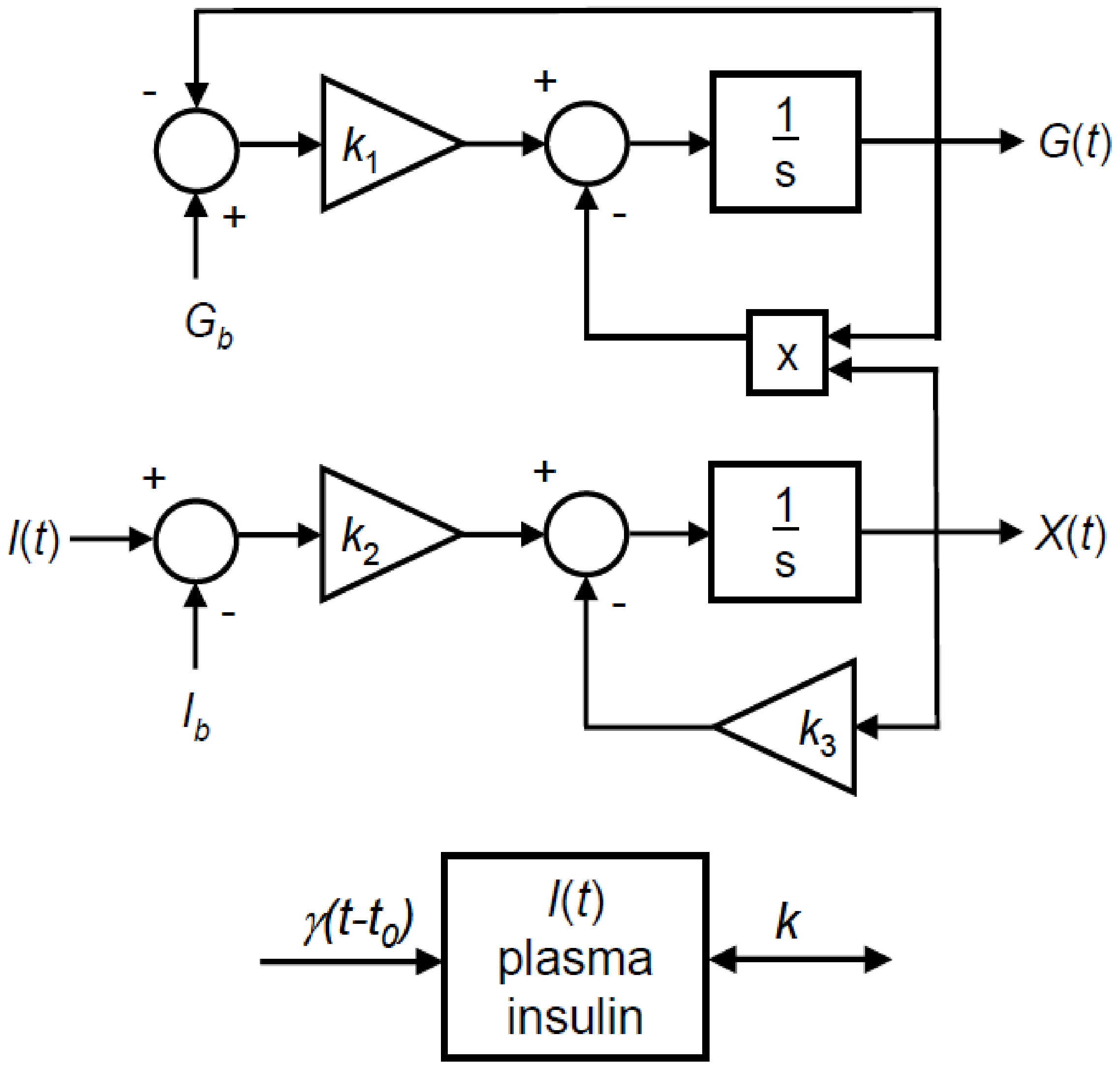

Figure 4: Control system analog for glucose and insulin kinetics [13]

G(t) and I(t) are the glucose and insulin concentrations as functions of time respectively. G_b and I_b are the basal or steady state glucose and insulin concentrations respectively. X(t) is not an actual physiological measurement quantity but the effect of active insulin or what is sometimes referred to as the effective insulin activity in min^{-1}. The insulin activity quantity accounts for the insulin having to travel from the bloodstream into the interstitial tissue to respond to the glucose level. The pancreas in the glucose and insulin system basically functions as a process controller. Glucose enters the plasma compartment at a rate proportional to the difference between the actual and basal concentrations. Glucose exits the plasma compartment at a rate proportional to the activity of insulin in the interstitial tissue. Insulin enters the plasma compartment at a rate proportional to the insulin responsivity by the pancreas multiplied by the time and exits at a rate proportional to the amount of insulin in the plasma compartment.

The equations for the Minimal Model are derived from unsteady state material balances in the body. The general unsteady state material balance is as follows:

$$In - Out + Generation - Consumption = Accumulation \tag{2.1}$$

The unsteady state material balances for glucose and insulin are taken from the reference and shown in the following equations below [11]:

$$k_1(G_b - G) - XG = \frac{dG}{dt} \tag{2.2}$$

Where k_1 is the glucose effectiveness in min^{-1} which may also be written as S_G, G_b is the basal glucose concentration in $\frac{mg}{dL}$, G is the glucose concentration in $\frac{mg}{dL}$, and X is the effect of active insulin also known as insulin activity in interstitial tissue in min^{-1}.

$$k_3\left(\frac{k_2}{k_3}(I - I_b) - X\right) = \frac{dX}{dt}$$

(2.3)

Where k_2 is the weighted external insulin input in min^{-1}, k_3 is the insulin clearance in min^{-1}, I is the insulin concentration in $\frac{microU}{mL}$, I_b is the basal insulin concentration in $\frac{microU}{mL}$, X is the insulin activity in min^{-1}, and the ratio of k_2 to k_3 is the insulin sensitivity (S_I). The insulin sensitivity reflects how effective the insulin is at returning to the basal concentration. Diabetics would have lower insulin sensitivities since they have impaired glucose tolerance. Equation 2.3 describes the dynamics of the transport of insulin from the blood to interstitial fluid. The final equation in the model describes the change in insulin concentration in the blood over time.

$$\gamma(G - G_b)t - k(I - I_b) = \frac{dI}{dt}$$

(2.4)

Where γ is the insulin responsivity by the pancreas in min^{-2}, G is the glucose concentration in $\frac{mg}{dL}$, G_b is the basal glucose concentration in $\frac{mg}{dL}$, t is the time in min, k is the insulin decay rate, I is the insulin concentration in $\frac{microU}{mL}$, and I_b is the basal insulin concentration in $\frac{microU}{mL}$. The material balances result in 3 coupled differential equations (2.2, 2.3, and 2.4) which form the Minimal Model.

Since the original Minimal Model has been proposed, there have been several proposed modifications to the model by other researchers. An example of this is a modification of the model to include the effect of physical exercise on insulin sensitivity and glucose effectiveness [14]. This is done by creating a more complex system that result in more than 3 coupled differential equations. Other modifications that have been proposed include the addition of parameters to account for genetic risk factors and obesity [15].

Matlab Implementation

The glucose insulin system was generated by simulating the original Bergman Minimal Model in MATLAB. The 3 coupled differential equations were solved using the MATLAB differential equation solver function ODE15s. This is a differential equation solver for stiff functions which was chosen after the standard differential equation solver ODE45 function failed to solve the system of equations in a timely manner. The system was simulated for 180 minutes with glucose measurements taken at every minute.

CHAPTER III

STATE ESTIMATION

3.1 Kalman Filter

There are several types of state estimators that have been introduced over the years. One popular type is the Kalman filter. The Kalman filter is a recursive data processing algorithm that combines all available measurement data, plus prior knowledge about the system and measurement devices to produce an estimate of the desired variables. The Kalman filter can be applied when the state equations are linear and there is a Gaussian distribution for the probability density function of the state for all of the time steps.

The following is a general linear system that the Kalman filter could be applied to which includes equations for the state as well as the measurement.

$$x_k = Ax_{k-1} + w_k \tag{3.1}$$

Where x_k is the state of the system at times step *k*, *A* is a constant, x_{k-1} is the state at time step *k-1*, and w_k is the process noise.

$$y_k = Cx_{k-1} + v_k \tag{3.2}$$

Where y_k is the measurement at times step k, *C* is a constant, x_{k-1} is the state at time step *k-1*, and v_k is the measurement noise.

3.2 Extended Kalman Filter

For non-linear systems, other state estimators are required. The extended Kalman filter attempts to apply the Kalman filter to nonlinear systems by linearizing the system and measurement equations for each time step. This is done through the use of a Taylor expansion for each time step. This results in a Jacobian matrix of partial derivatives.

$$F = \frac{\partial f}{\partial x} = \begin{bmatrix} 1 + (-S_G - X) * dt & -G * dt & 0 \\ 0 & 1 - k_3 * dt & S_I * k_3 * dt \\ k * \gamma * dt & 0 & 1 - k_1 * dt \end{bmatrix} \tag{3.3}$$

$$H = \frac{\partial h}{\partial x} = [1 \quad 0 \quad 0] \tag{3.4}$$

F and *H* are the Jacobian matrices of the state and measurement functions respectively. The Kalman gain (K_k), covariance (P_k) and estimate ($\hat{x}_k$) are updated as shown in equations 3.5, 3.6, and 3.7.

$$K_k = P_k H_k^T (H_k P_k H_k^T + R)^{-1} \tag{3.5}$$

$$P_k = P_{k-1} - K_k H_k P_{k-1} \tag{3.6}$$

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

(3.7)

Where $\hat{x}_k$ is the estimate state of the system at times step k, R is the system noise covariance matrix, $\hat{x}_{k-1}$ is the state at time step *k-1*, K_k is the Kalman gain at time step k, P_k is the system covariance matrix at time step k, y_k is the measurement at times step k, H is the Jacobian matrix of the measurement function. The reason matrix transposes and inverses are used in equation 3.5 is due to the behavior of Gaussian systems as well in order to obtain a feasible solution with the proper dimensions.

A drawback of the extended Kalman filter is that it requires the system and measurement functions to be differentiable and knowledge of what the derivatives of those function are. Some functions that are nonlinear may also not be linearized well by only a first order Taylor expansion.

3.3 Unscented Kalman Filter

The unscented Kalman filter takes another approach to apply the Kalman filter to nonlinear systems. The unscented Kalman filter involves a method of statistical linearization by deterministically sampling 2n+1 “sigma points” where n is the number of states that represent the system. The points that are sampled represent the current mean and the standard deviation from both sides of the mean. The points are fed through the system equation. As a result, the distribution of the sigma points is no longer symmetric. Sigma points are re-generated symmetrically by calculating the new weighted mean and covariance and choosing new points accordingly. A graphical representation of the sigma points can be seen in figure 4 from the reference [16].

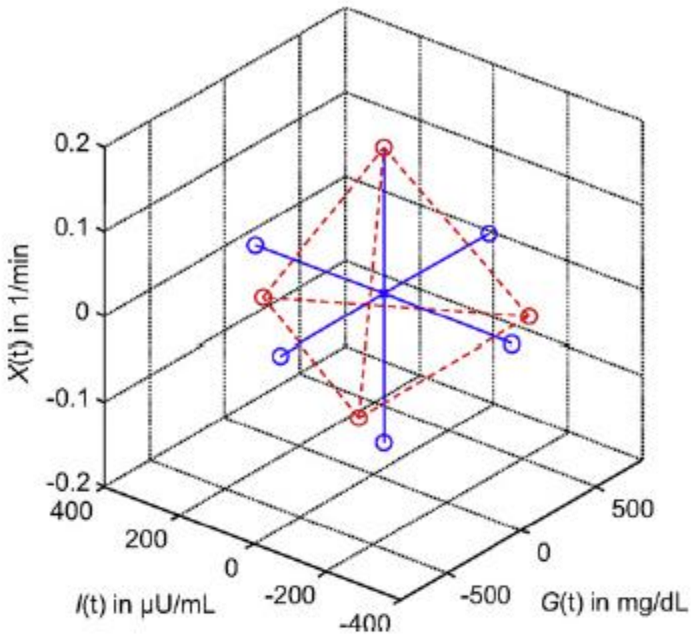

Figure 4: Symmetric sigma points for the Minimal Model system *[16]*

The dots connected by the solid line represent the initially symmetric sigma points while the dots connected by the dashed lines show a possible distribution that can result after the sigma points are passed through the nonlinear system. The sigma points are regenerated symmetrically using the following equations.

$$x_0 = \bar{x} \tag{3.8}$$

Where x_0 is the central point which is taken from the weighted mean of the sigma points ($\bar{x}$). The remaining sigma points are generated based off of the standard deviation from the mean. In the case of model used in this thesis, there are 3 sigma points on the positive side of the mean and 3 sigma points on the negative side of the mean. This is shown in equations 3.9 and 3.10.

$$x_i = \bar{x} + \left(\sqrt{(n+\lambda)P_{xx}}\right)_i$$

(3.9)

$$x_i = \bar{x} - \left(\sqrt{(n+\lambda)P_{xx}}\right)_i$$

(3.10)

The weighted mean for the estimate of the state of the system is calculated as follows:

$$\bar{x}_k = \sum_{i=0}^{2n+1} W_i x_{k,i}$$

(3.11)

The weights for the mean and covariance for the central sigma points are assigned using the following equations:

$$W_a^{(0)} = \frac{\lambda}{n+\lambda}$$

(3.12)

$$W_c^{(0)} = \frac{\lambda}{n+\lambda} + 1 - \alpha^2 + \beta$$

(3.13)

Where $W_a^{(0)}$ is the weight of the central sigma point for the mean and $W_c^{(0)}$ is the weight of the central sigma point for covariance. The weights for the mean and covariance for the rest of the sigma points are assigned using the following equations:

$$W_a^i = W_c^i = \frac{1}{2(n+\lambda)}$$

(3.14)

$$\lambda = \alpha^2(n + \kappa)$$

(3.15)

Where n is the number of sigma points, λ is a function of the tuning parameters α, β, and κ which have typical values of α=0.5, β=2, and κ=3-n [17]. The values for these parameters can be adjusted to "tune" the filter to the system with an unbiased estimate as long as the sum of the resulting weights is equal to 1.

The following equations are used to update the estimate of the state and the covariance:

$$P_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}[x_i - \bar{x}]\, W_i[h(x_i) - h(\bar{x})]^T$$

(3.16)

$$P_{yy} = \frac{1}{N-1}\sum_{i=1}^{N}[h(x_i) - h(\bar{x})]\, W_i[h(x_i) - h(\bar{x})]^T + R$$

(3.17)

$$K = P_{xy}P_{yy}^{-1}$$

(3.18)

$$\hat{x}_k = \hat{x}_{k-1} + K(y_k - h(\hat{x}_k))$$

(3.19)

An advantage of the unscented Kalman filter is that the system and measurement equations do not need to be linearized at each time step through Taylor expansions. This means there is no longer the need to calculate Jacobians as is the case in the extended Kalman filter.

3.4 Ensemble Kalman Filter

The ensemble Kalman filter is another way to utilize the Kalman filter for nonlinear systems. Instead of representing the mean and covariance of the state through deterministically chosen sigma points like the unscented Kalman filter, the ensemble Kalman filter utilizes a more stochastic process. The ensemble Kalman filter involves sampling a large number of random points to obtain the mean and covariance. In this thesis, 100 points which each had a value for the glucose concentration, insulin concentration, and effect of active insulin were used for the ensemble Kalman filter. The initial distribution of the points is assumed to be Gaussian. After the initial time step, the points are fed through the nonlinear state equation and their distribution becomes non-Gaussian. In this filter, the sample mean and sample covariance are used instead of some type of weighted mean and covariance.

$$P_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}[x_i - \bar{x}]\,[h(x_i) - h(\bar{x})]^T \tag{3.20}$$

$$P_{yy} = \frac{1}{N-1}\sum_{i=1}^{N}[h(x_i) - h(\bar{x})]\,[h(x_i) - h(\bar{x})]^T + R \tag{3.21}$$

$$K = P_{xy}P_{yy}^{-1} \tag{3.22}$$

$$\hat{x}_k = \hat{x}_{k-1} + K(y_k - h(\hat{x}_k))$$

(3.23)

$$P_{xx} = \frac{1}{N-1}\sum_{i=1}^{N}[x_i - \bar{x}]\,[x_i - \bar{x}]^T$$

(3.24)

The ensemble Kalman filter does not require the Gaussian assumption that is made for the propagation of sigma points in the unscented Kalman filter. A shortcoming of the ensemble Kalman filter is that all of the particles are assigned equal weights regardless of what their distribution [18].

3.5 Particle Filter

The last state estimator discussed in this thesis is the Monte Carlo based particle filter which was first proposed by Gordon et al [19]. In the particle filter, the assumptions of a Gaussian distribution and linear system are no longer required. Since there is no readily available formula for the probability density function of non-linear and non-Gaussian processes, the use of Monte Carlo simulation is required. Monte Carlo simulation involves drawing a large number of random samples to estimate integrals or areas. In the case of the particle filter, it is used to estimate the area of the probability density function.

The conditioned probability density for the particle filter is based off of Bayes rule and is as follows:

$$posterior = \frac{likelihood \times prior}{evidence} = \frac{p(y_1|x_1)p(x_1|y_0)}{p(y_1)}$$

(3.25)

Where $p(y_1|x_1)$ is the probability of y_1 conditioned on x_1, and $p(x_1|y_0)$ is the probability of x_1 conditioned on y_0, and $p(y_1)$ is a normalizing constant.

The particle filter estimates the state of the system by using a weighted estimate. In order to determine the weights of the particles, the following equation was used.

$$Weight = \frac{1}{\sqrt{2\pi R}} exp\left(-\frac{(y-h(x))^2}{2R}\right)$$

(3.26)

Where R is the sensor noise variance, x is the state, y is the measurement, and h(x) is the measurement function of x. In this equation, the transition density was used as the importance density. This was done out of computational ease however there are other techniques for calculating the importance density.

The state was estimated by taking a weighted average of the particles at each time step in the simulation as shown in equation 3.27 below:

$$X_{estimate} = \sum \frac{x \times Weight}{\sum Weight}$$

(3.27)

Where $X_{estimate}$ is the estimated value of the state, x is the vector of particles created for the simulation, and the weight is the vector of weights for each particle which was normalized. This was performed for each time step to estimate the state of the process.

A resampling step was added to the particle filter algorithm. One reason resampling is performed is to prevent a condition known as degeneracy. This is where after a large number of time steps, only one particle has significant weight [20]. This means that computational effort is wasted on particles with negligible weight while only one particle determines the estimate of the state. In the resampling step which occurs at every time step, particles with low weights were eliminated while particles with high weights were duplicated. There are several types of resampling methods that vary in computational efficiency which include multinomial resampling, stratified resampling, systematic resampling, and residual resampling [21]. In this case, residual resampling was chosen. More information about the algorithm can be found from the reference [21].

CHAPTER IV

SIMULATION STUDY

The application of a process controller to function as an artificial pancreas will have to respond to the changes to the glucose insulin system during meal time. The Minimal Model that was described in chapter 2 was used as the simulation of the true state of glucose and insulin dynamics using the ODE15s function to solve the differential equations. The parameters used in the differential equations are taken from the reference and summarized in table II below [11].

S_G (min^{-1})	0.03082
S_I min^{-1}(microU/mL)$^{-1}$	5.07E-04
k_1 (min^{-1})	0.3
k_3 (min^{-1})	0.02093
Gamma min^{-2}(microU/mL)(mg/dL$^{)-1}$	0.003349
G_b (mg/dL)	89.5
I_b (microU/dL)	7.3

Table II: Minimal model parameters

Glucose measurements were also generated by adding random noise to the true glucose concentration. Figures 5, 6, and 7 show the system under normal circumstances which was used for the good filter initialization and poor filter initialization case studies.

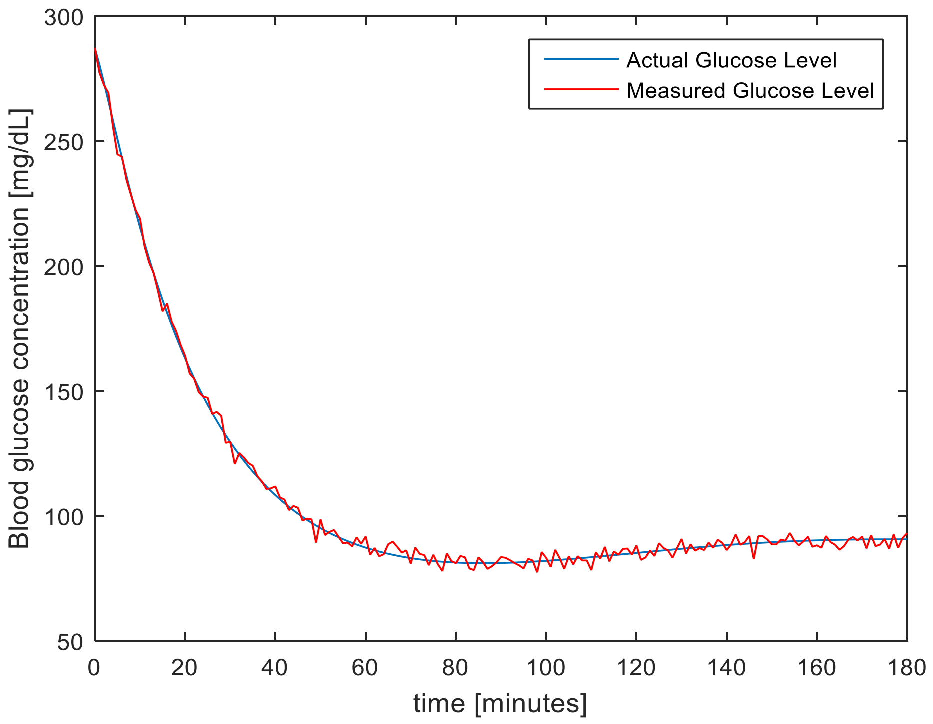

Figure 5: Glucose concentration vs. time

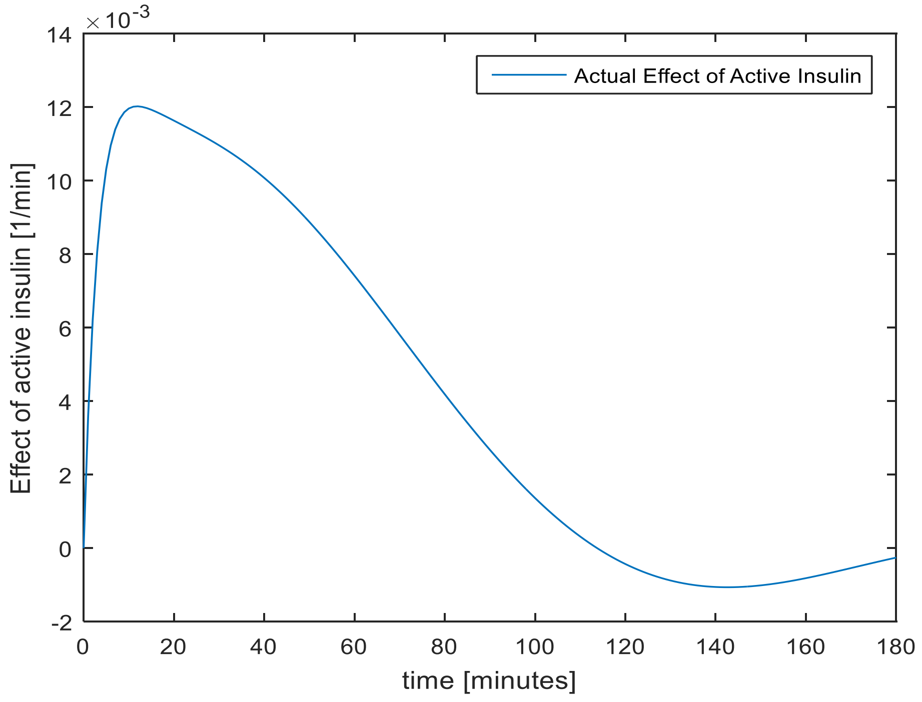

Figure 6: Effect of active insulin vs. time

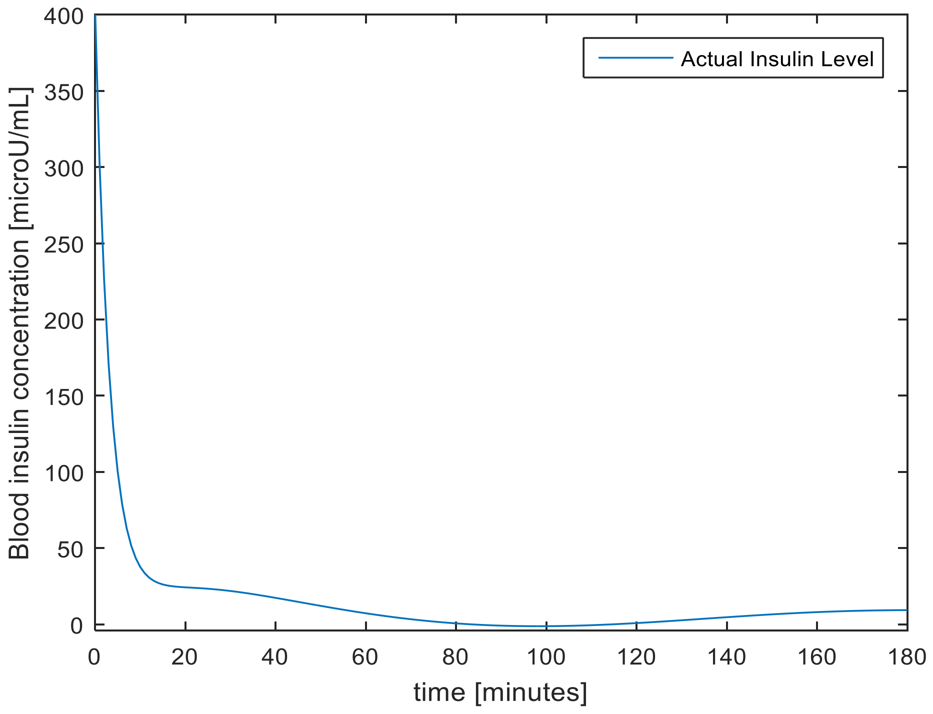

Figure 7: Insulin concentration vs. time

In order to evaluate the performance of the filters, the root mean squared error (RMSE) was computed as shown in the following equation.

$$RMSE = \sqrt{\frac{1}{k}\sum_{i=0}^{k}(X_t - X)^2} \tag{4.1}$$

Where k is the number of time steps, X_t is the true value of the state, and X is the state estimate. The comparison of the RMSE of the filters is given as an empirical observation and is not meant as an ultimate comparison as which filter is the absolute best. This is because of the selection of tuning parameters, number of samples generated, and selection of importance density will vary among other programmers and impact the results of the same type of filter. The computational times were calculated for illustrative purposes as no attempt was made to optimize the coded algorithm of the filters for speed. Another reason to calculate the computational times was to see if the filters could be implemented in real time even if they were not optimized for speed. The figures presented in each case study are for realizations of filter performance that had a RMSE close to the average and are presented for illustrative purposes.

4.1 Good Filter Initialization

The following results are for the case of good filter initialization where the initial estimate and state covariance are as follows:

$$x_0 = [289\ 1 \times 10^{-7}\ 406]$$

$$P_0 = \begin{bmatrix} 2^2 & 0 & 0 \\ 0 & 1 \times 10^{-3^2} & 0 \\ 0 & 0 & 2^2 \end{bmatrix} \tag{4.2}$$

The estimates from all 4 filters can be seen in figures 8, 9, and 10.

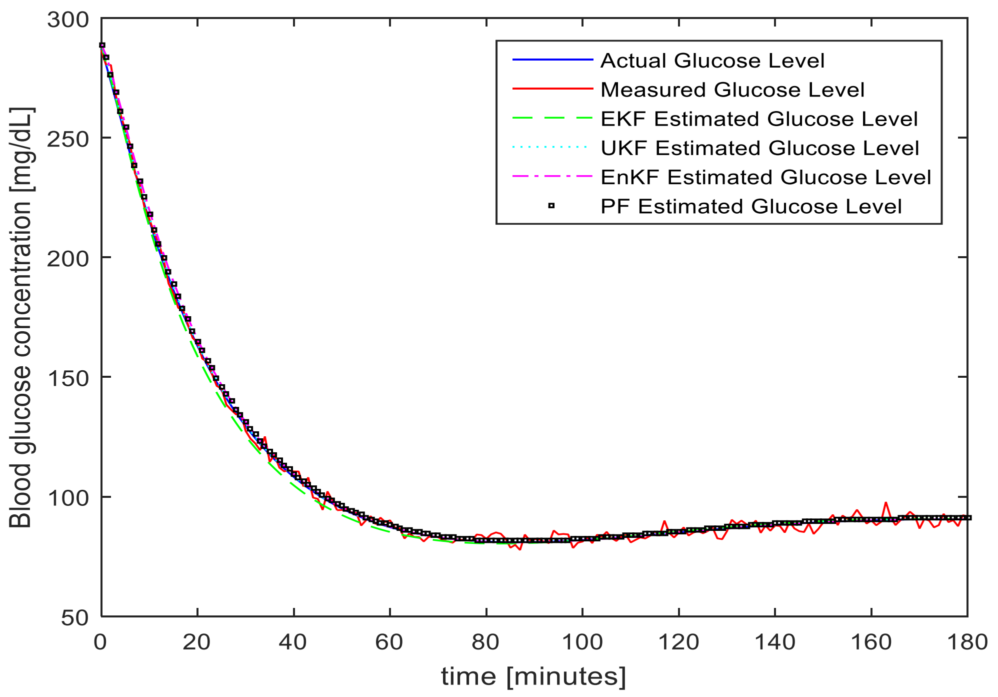

Figure 8: Good filter initialization, high confidence, glucose concentration vs. time

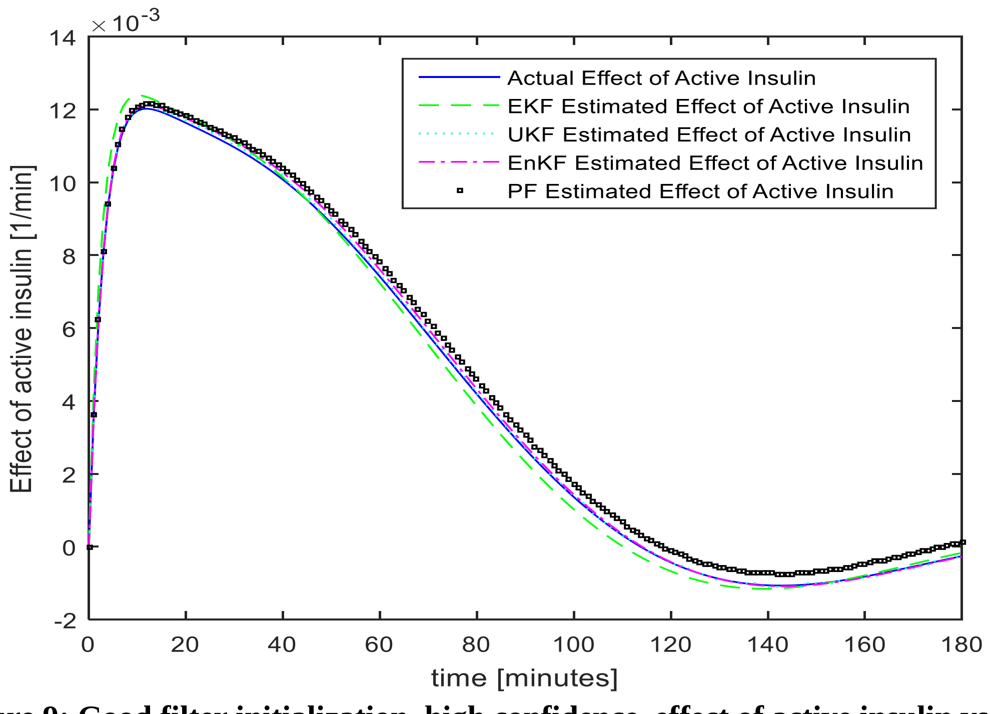

Figure 9: Good filter initialization, high confidence, effect of active insulin vs. time

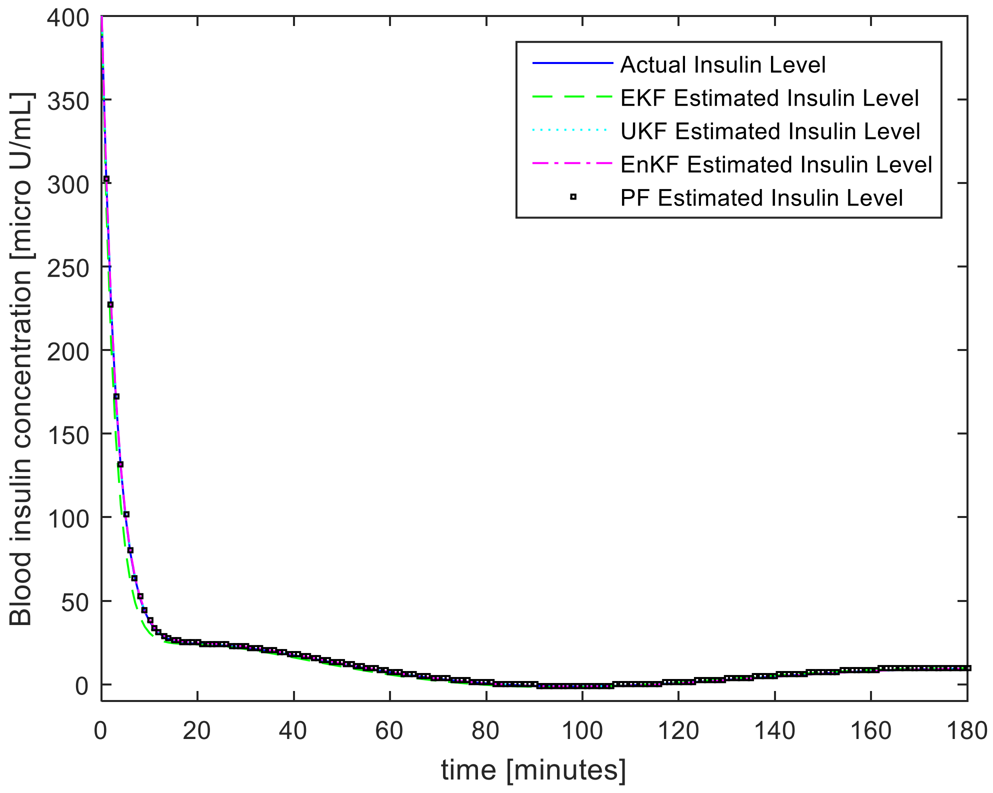

Figure 10: Good filter initialization, high confidence, insulin concentration vs. time

The results of the filters are summarized in table III below. It was observed that the unscented Kalman filter had the lowest RMSE for all of the states.

	EKF	UKF	EnKF	PF
CPU time (s)	0.0189	6.0395	111.275	101.9108
RMSE: Glucose (mg/dL)	1.0496	0.5453	1.8381	1.1954
RMSE: Insulin (micro U/mL)	12.2368	0.3919	0.4794	0.6703
RMSE: Effect of Active Insulin (1/min)	0.0024	5.86E-05	1.24E-04	3.27E-04

Table III: Good filter initialization, high confidence filter performance

A 2nd case was studied where the initial estimate remains the same as before but there is low confidence in the estimate. This is reflected in changing the value for the initial covariance matrix which was as follows:

$$P_0 = \begin{bmatrix} 40 & 0 & 0 \\ 0 & 1 \times 10^{-8} & 0 \\ 0 & 0 & 40 \end{bmatrix} \tag{4.3}$$

The new covariance is 10 times the covariance for high confidence. The estimates from all 4 filters for this case can be seen in figures 11, 12, and 13 below.

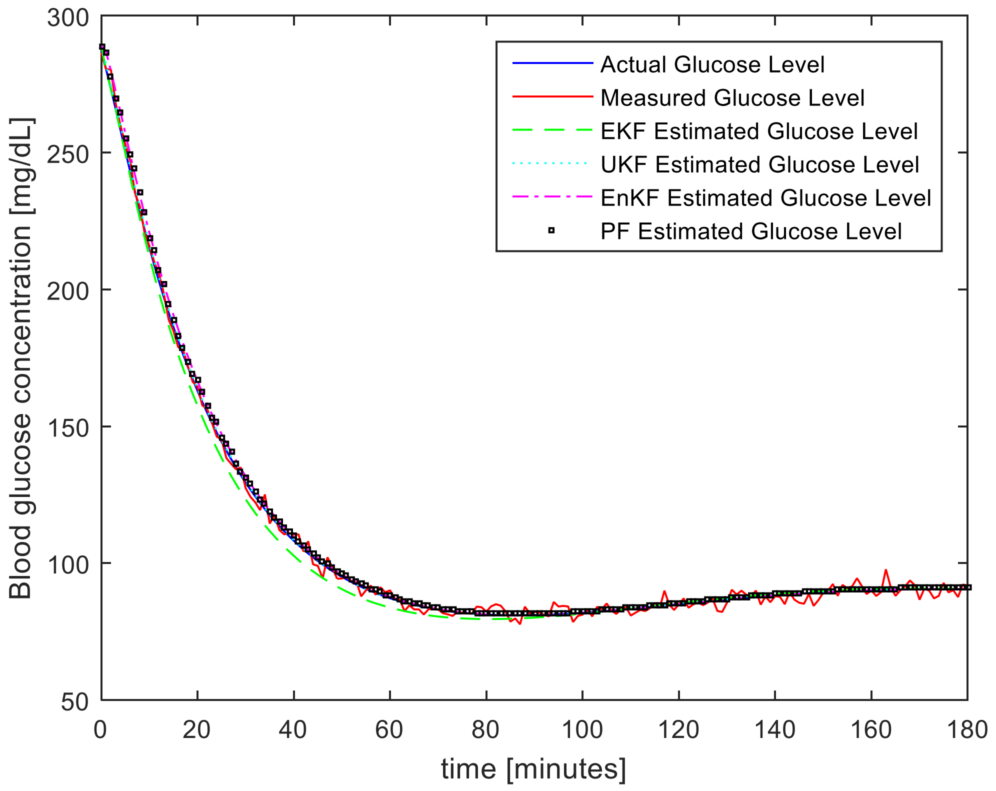

Figure 11: Good filter initialization, low confidence, glucose concentration vs. time

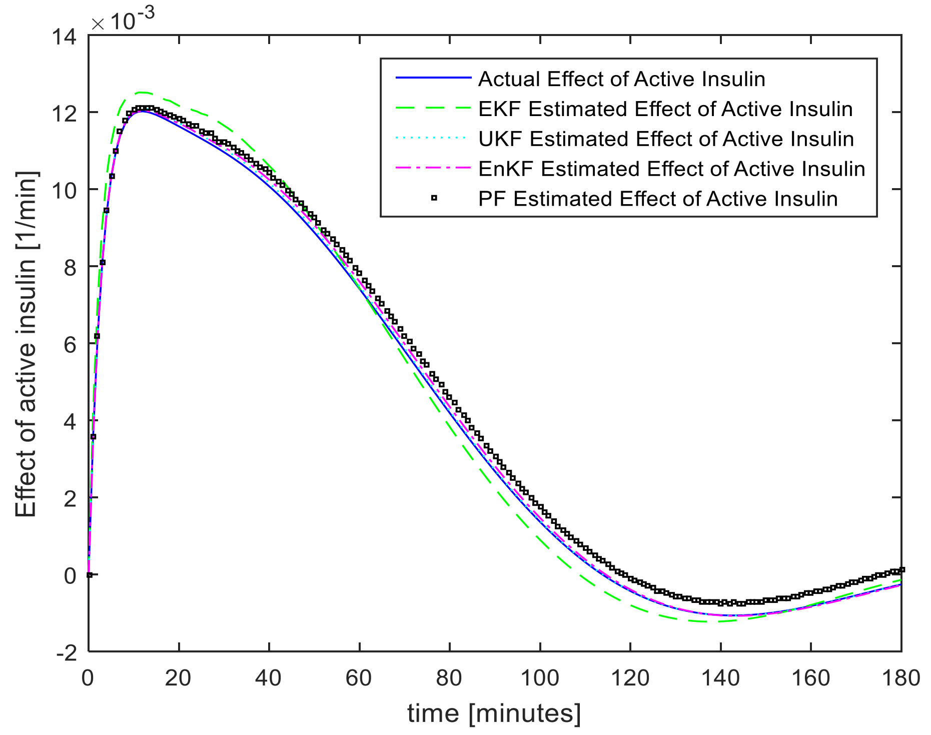

Figure 12: Good filter initialization, low confidence, effect of active insulin vs. time

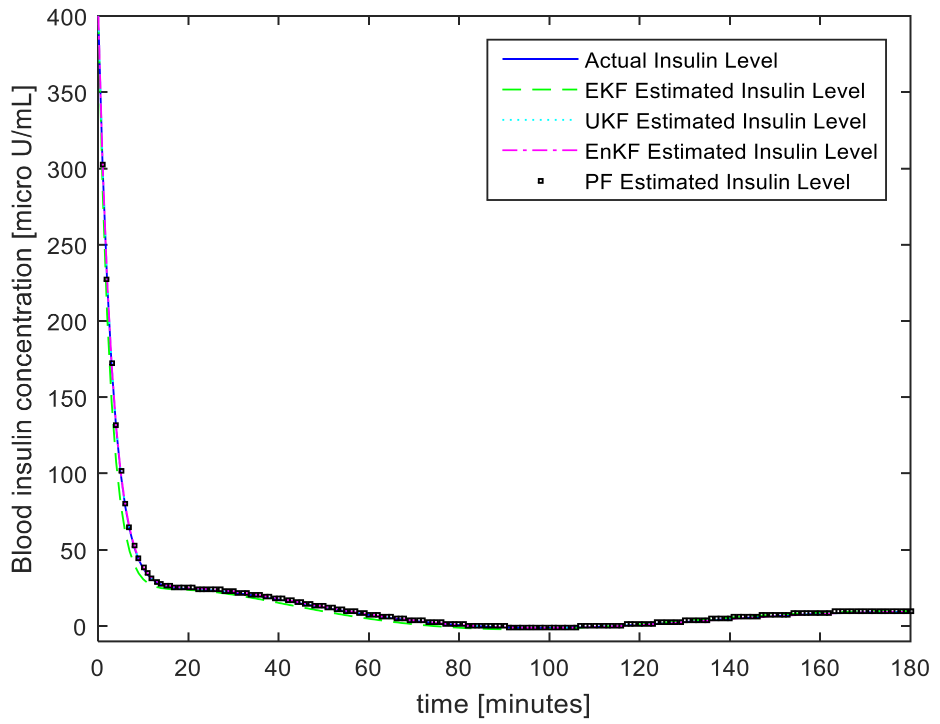

Figure 13: Good filter initialization, low confidence, insulin concentration vs. time

The results of the filters are summarized in table IV below. The RMSE increased slightly compared to the previous case of good filter initialization and high confidence. The unscented Kalman filter still had the lowest RMSE.

	EKF	UKF	EnKF	PF
CPU time (s)	0.0194	6.058	117.736	102.92
RMSE: Glucose (mg/dL)	1.0613	0.5258	2.0562	1.822
RMSE: Insulin (micro U/mL)	12.2351	0.3847	0.4852	0.6931
RMSE: Effect of Active Insulin (1/min)	0.0024	5.51E-05	1.15E-04	3.27E-04

Table IV: Good filter initialization, low confidence

4.2 Poor Filter Initialization

The following results are for the case of poor filter initialization where the initial estimate and state covariance are as follows:

$$x_0 = [385\ 1.33 \times 10^{-7}\ 541]$$

$$P_0 = \begin{bmatrix} 2^2 & 0 & 0 \\ 0 & 1 \times 10^{-3^2} & 0 \\ 0 & 0 & 2^2 \end{bmatrix} \tag{4.4}$$

The initial estimate is 33% higher than the estimate for good filter initialization. The initial covariance matrix was unchanged which placed high confidence in this poor initial estimate. The estimates from all 4 filters can be seen in figures 14, 15, and 16 below.

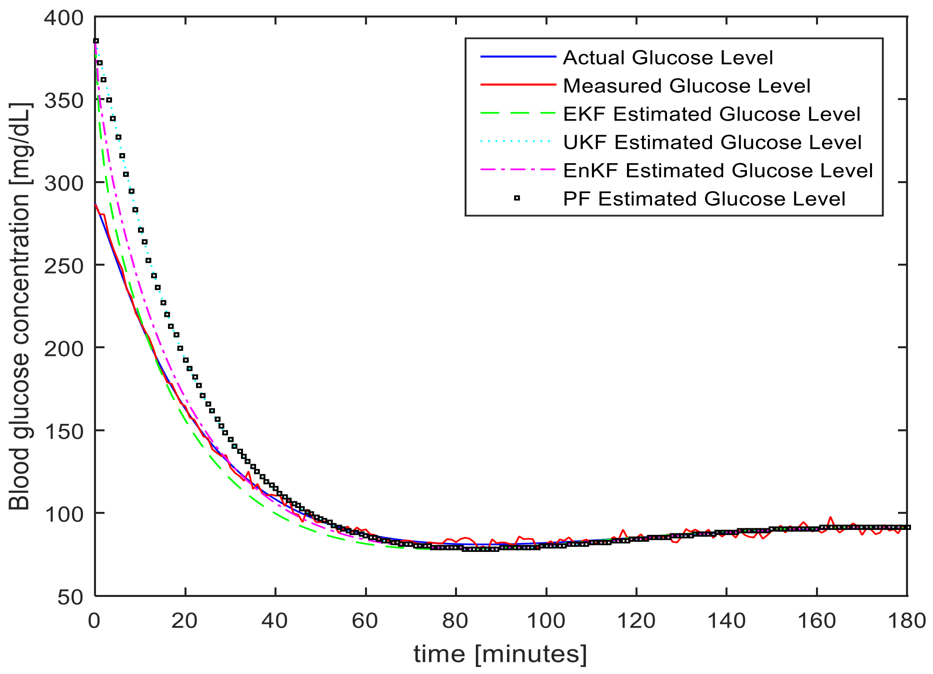

Figure 14: Poor filter initialization, high confidence, glucose concentration vs. time

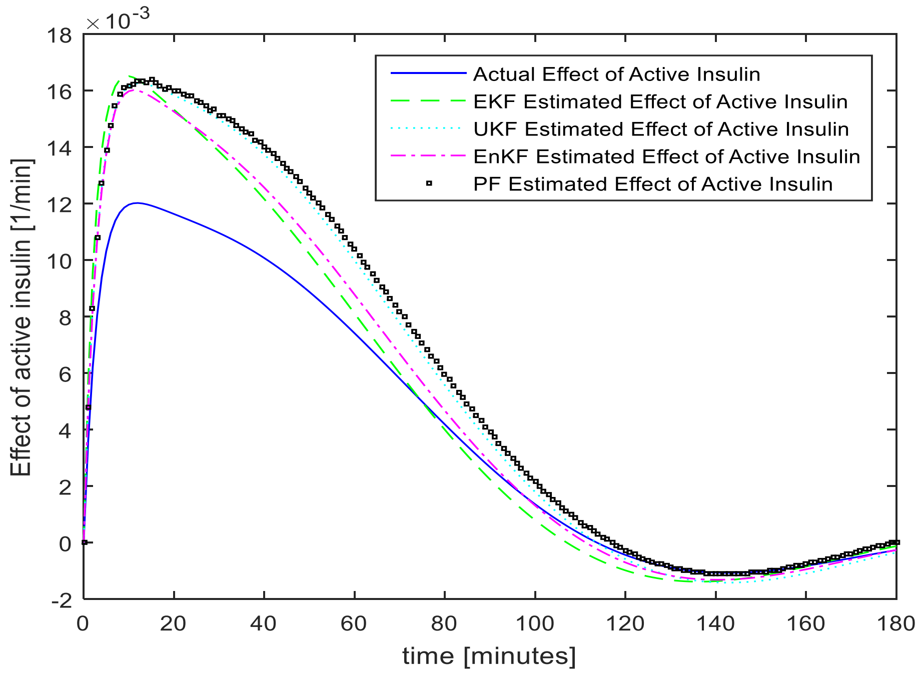

Figure 15: Poor filter initialization, high confidence, effect of active insulin vs. time

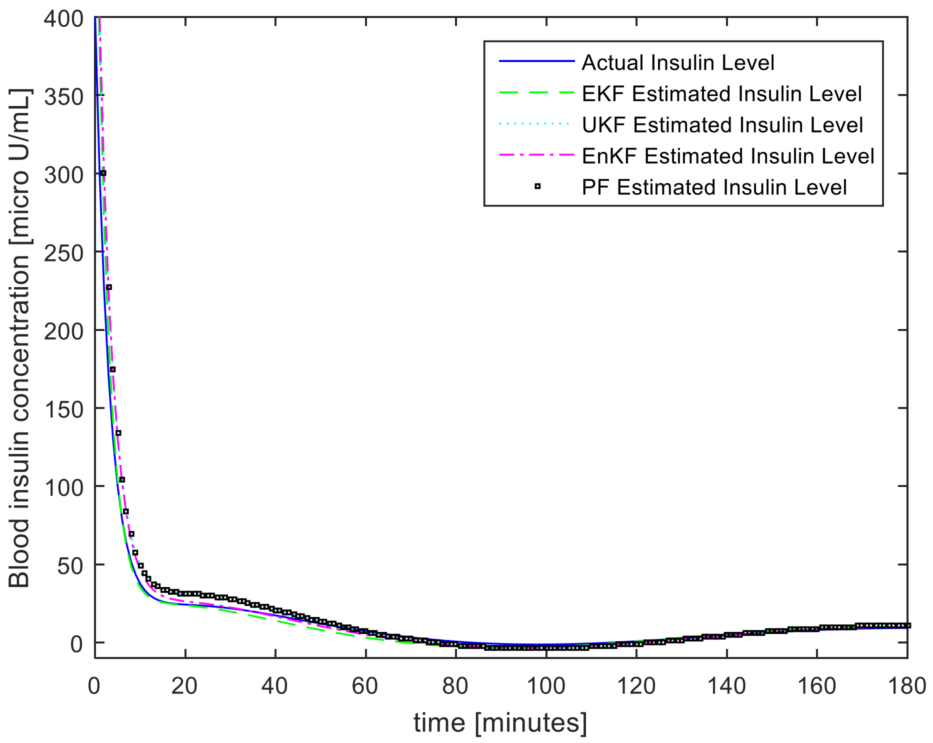

Figure 16: Poor filter initialization, high confidence, insulin concentration vs. time

The results of the filters are summarized in table V below. The extended Kalman filter had the lowest RMSE. In general, the filters had a higher RMSE than the case of good filter initialization which is probably since they took longer to converge to the true states.

	EKF	UKF	EnKF	PF
CPU time (s)	0.0189	6.1083	108.3806	101.927
RMSE: Glucose (mg/dL)	6.238	22.5345	12.5763	22.2365
RMSE: Insulin (micro U/mL)	2.2904	15.838	15.4067	15.7399
RMSE: Effect of Active Insulin (1/min)	3.71E-04	2.30E-03	1.80E-03	2.40E-03

Table V: Poor filter initialization, high confidence

The case of poor filter initialization was then repeated but with low confidence in the initial estimate. This means that the estimator will place less trust in the initial poor estimate. The new initial estimate and covariance are shown below.

$$x_0 = [385\ 1.33 \times 10^{-7}\ 541]$$

$$P_0 = \begin{bmatrix} 40 & 0 & 0 \\ 0 & 1 \times 10^{-8} & 0 \\ 0 & 0 & 40 \end{bmatrix} \tag{4.5}$$

The results from the 4 filters are shown in figures 17, 18, and 19 below.

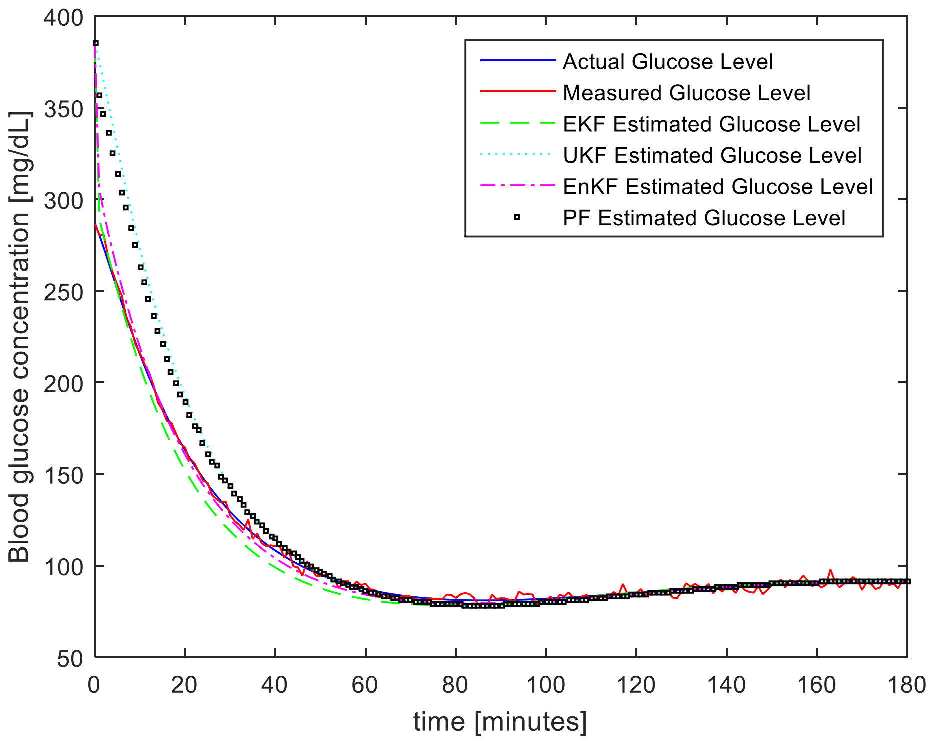

Figure 17: Poor filter initialization, low confidence, glucose concentration vs. time

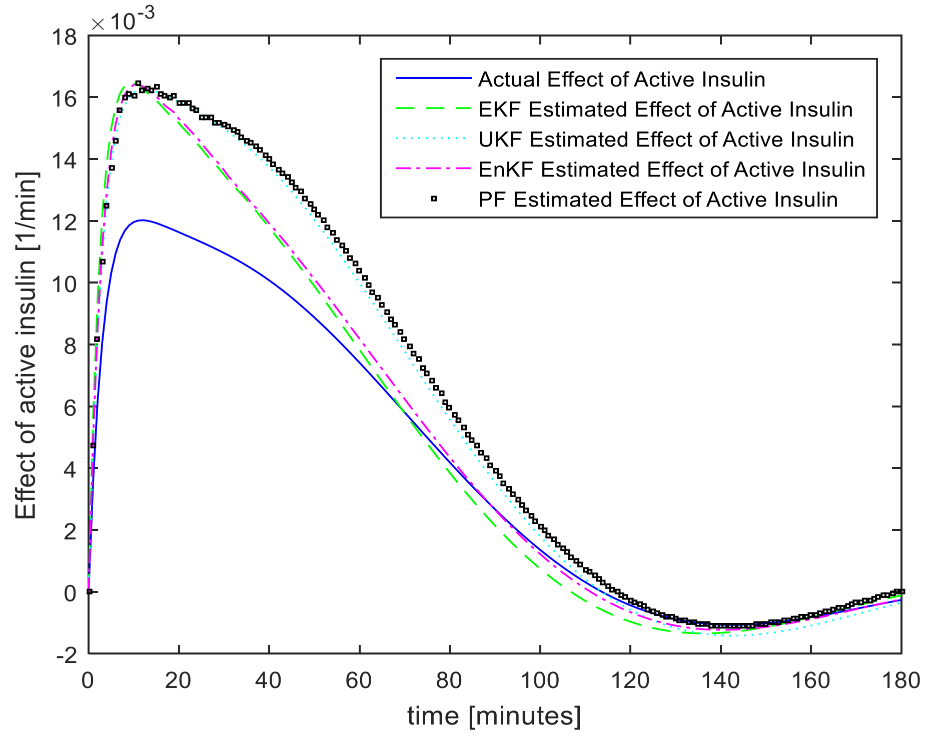

Figure 18: Poor filter initialization, low confidence, effect of active insulin vs. time

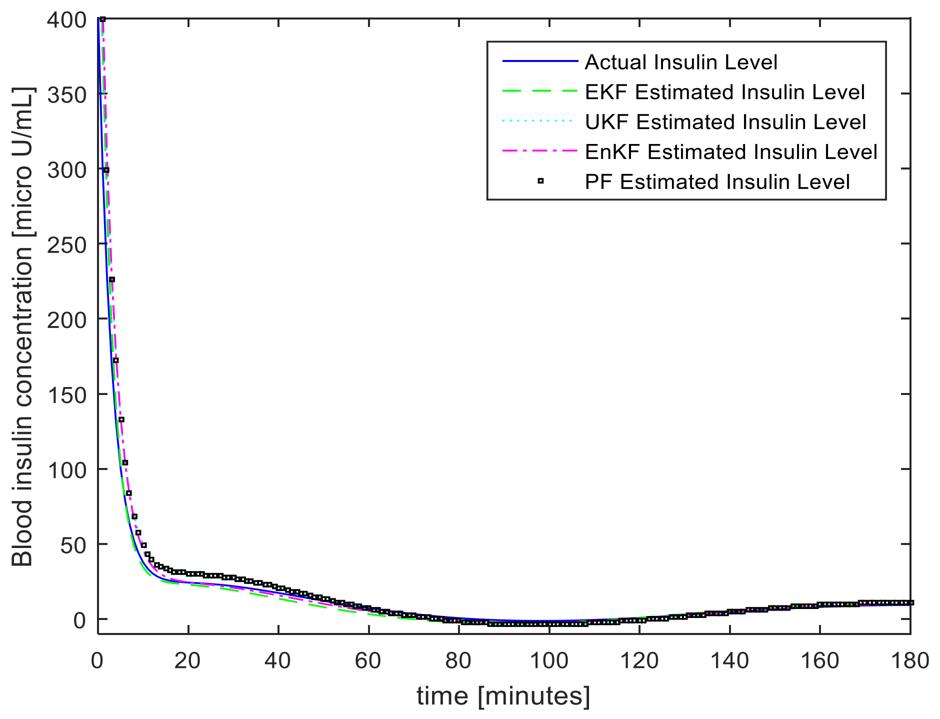

Figure 19: Poor filter initialization, low confidence, insulin concentration vs. time

The results of the filters are summarized in table VI below. The extended Kalman filter had the lowest RMSE. In general, the RMSE was lower than the case of poor filter initialization and high confidence.

	EKF	UKF	EnKF	PF
CPU time (s)	0.0161	5.7939	114.4744	90.8206
RMSE: Glucose (mg/dL)	1.0682	22.5478	8.1905	19.7673
RMSE: Insulin (micro U/mL)	2.3299	15.8395	15.3371	15.7338
RMSE: Effect of Active Insulin (1/min)	2.72E-04	2.30E-03	1.50E-03	2.40E-03

Table VI: Poor filter initialization, low confidence

4.3 Plant-Model Mismatch

The next case study was that of plant-model mismatch where the system equation used by the estimator does not accurately describe the true system. The first case of this is where the insulin sensitivity parameter for the estimator is 33.2% higher than the actual sensitivity. This was chosen since it reflects the upper confidence interval for insulin sensitivity reported from the reference [11]. This is a likely that can arise from the results of a glucose tolerance test. The initial state estimate and covariance were the same from equation 4.2. The results from the 4 filters can be seen in figures 20, 21, and 22 below.

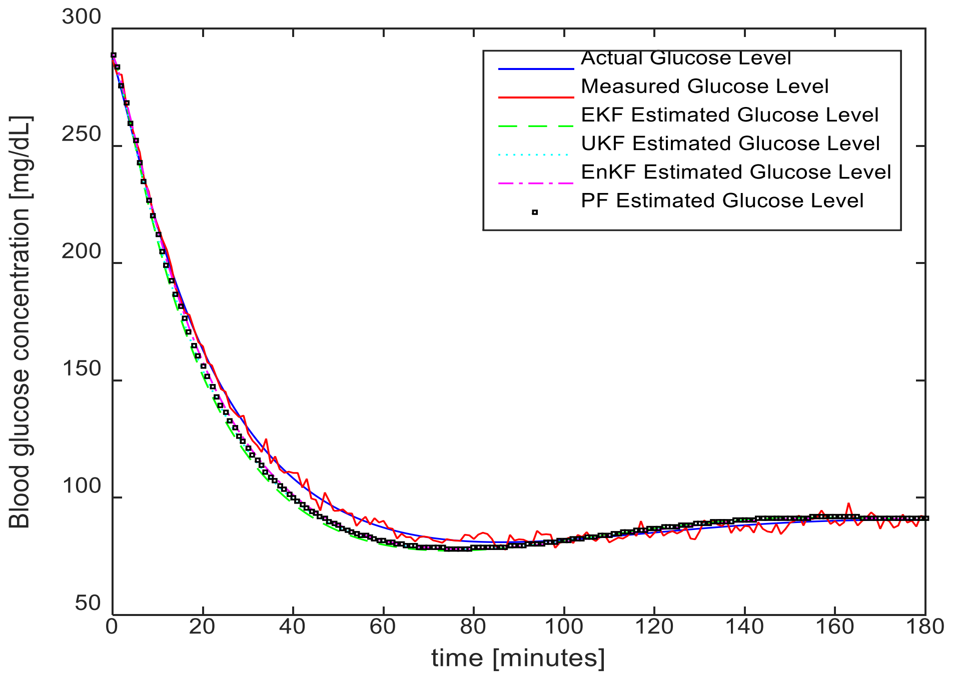

Figure 20: Plant-model mismatch: insulin sensitivity, glucose concentration vs. time

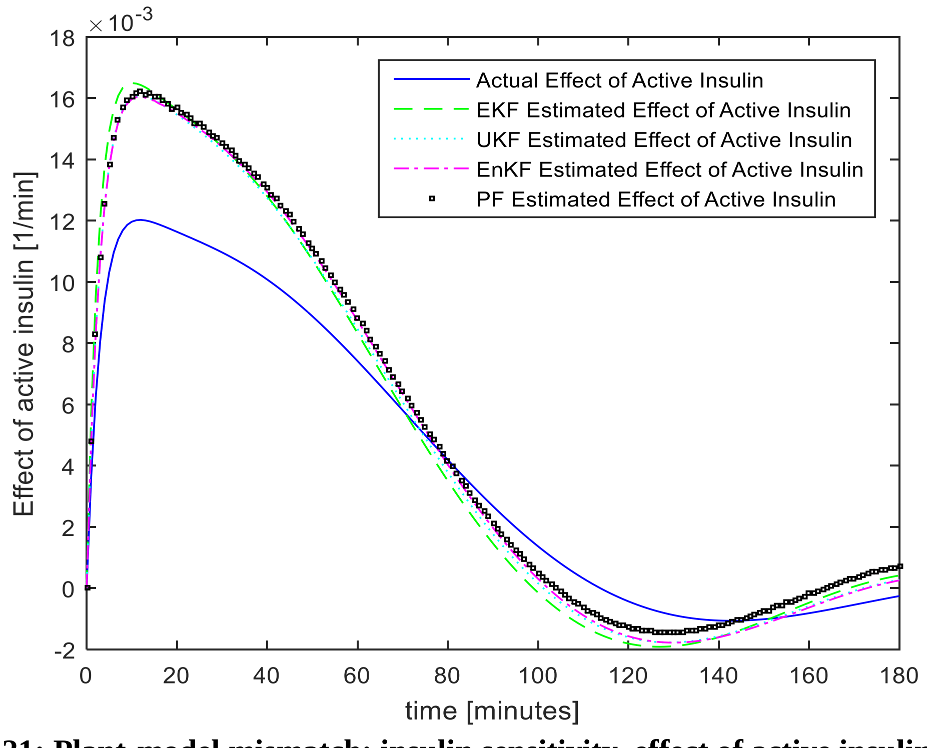

Figure 21: Plant-model mismatch: insulin sensitivity, effect of active insulin vs. time

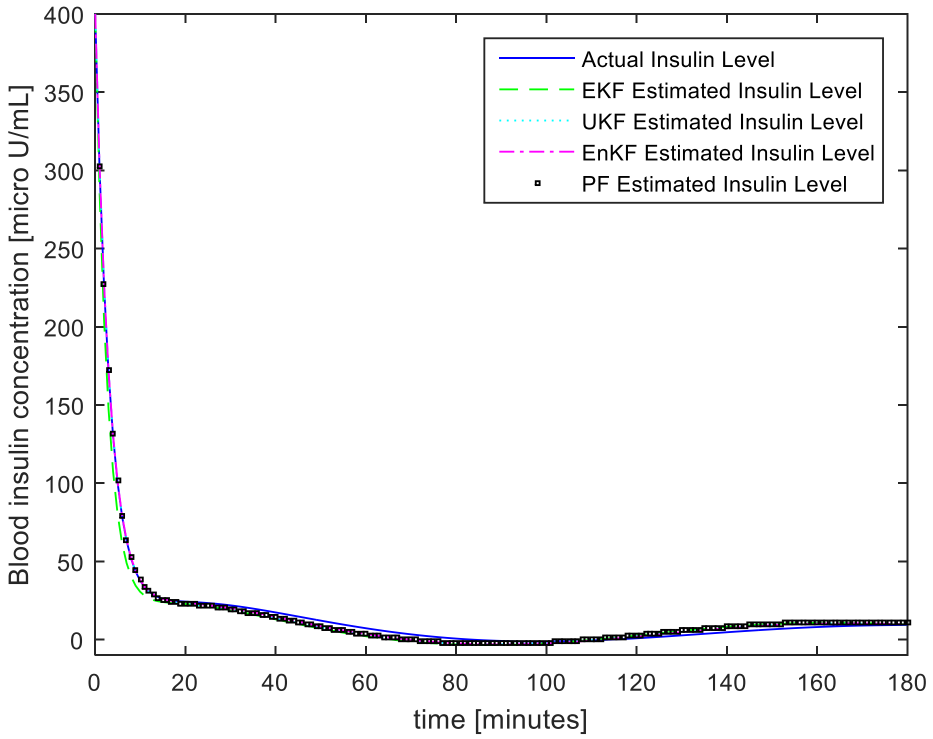

Figure 22: Plant-model mismatch: insulin sensitivity, insulin concentration vs. time

The results of the filters are summarized in table VII below. The ensemble Kalman filter had the lowest RMSE.

	EKF	UKF	EnKF	PF
CPU time (s)	0.0227	5.9297	127.2572	102.7042
RMSE: Glucose (mg/dL)	7.372	4.8735	3.7011	4.2731
RMSE: Insulin (micro U/mL)	12.7049	2.6171	2.2833	2.6021
RMSE: Effect of Active Insulin (1/min)	3.50E-03	1.90E-03	1.90E-03	2.00E-03

Table VII: Plant-model mismatch, insulin sensitivity

The case of plant-model mismatch was then studied for the case where the true system was that of an elderly patient while the parameter values from the estimator were still based off of the values from the reference [11]. A summary of the parameters for the elderly patient are shown in the table below.

S_G (min^{-1})	0.01572
S_I min^{-1}(microU/mL)$^{-1}$	3.10E-04
k_1 (min^{-1})	0.3606
k_3 (min^{-1})	0.01301
Gamma min^{-2}(microU/mL)(mg/dL$)^{-1}$	0.001785
G_b (mg/dL)	105.5
I_b (microU/dL)	7.3

Table VIII: Elderly patient minimal model parameters

The goal of this case study was to see how using completely different model parameters effects the performance of the filters. The results from the 4 filters can be seen in figures 23, 24, and 25 below.

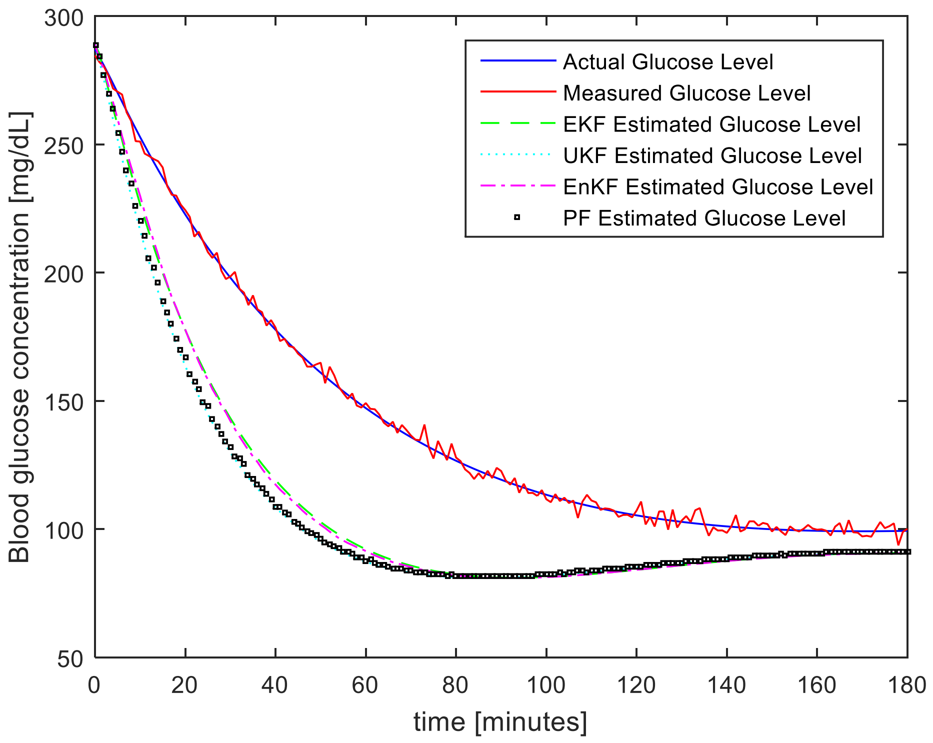

Figure 23: Plant-model mismatch: elderly patient, glucose concentration vs. time

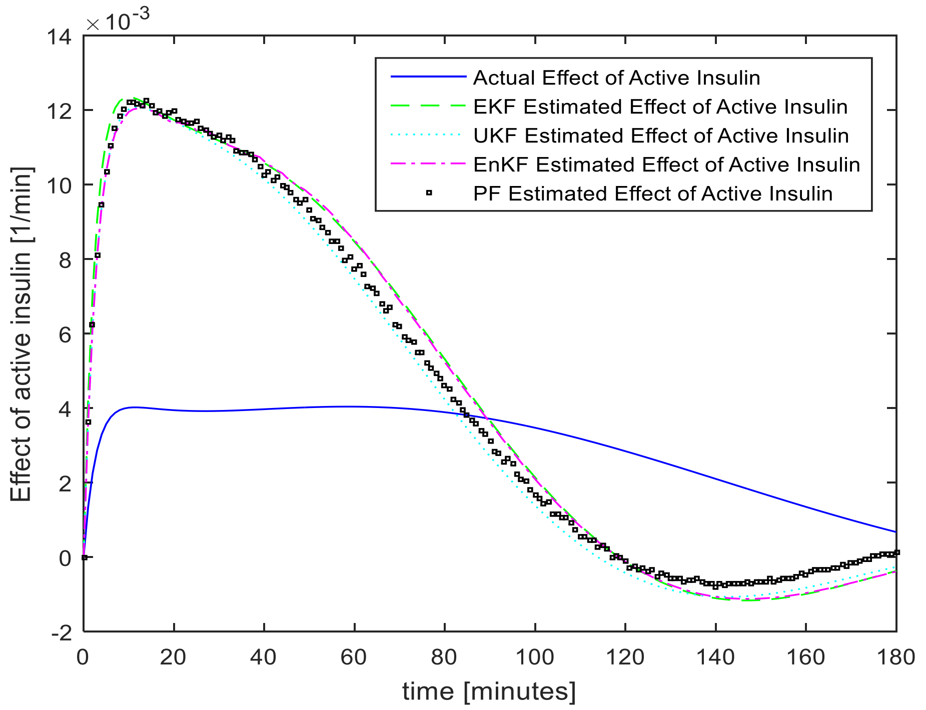

Figure 24: Plant-model mismatch: elderly patient, effect of active insulin vs. time

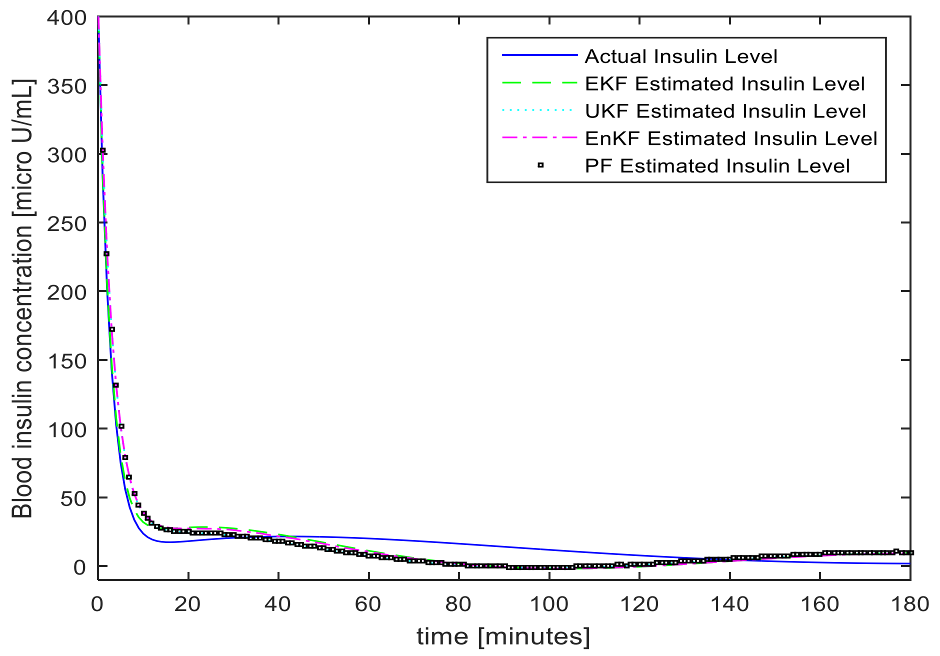

Figure 25: Plant-model mismatch: elderly patient, insulin concentration vs. time

The results of the filters are summarized in table IX below. The extended Kalman filter had the lowest RMSE but none of the filters were actually able to converge to the true state for glucose concentration, insulin concentration, or effect of active insulin.

	EKF	UKF	EnKF	PF
CPU time (s)	0.0206	5.7411	128.5475	101.5256
RMSE: Glucose (mg/dL)	34.2906	41.706	37.6606	40.7346
RMSE: Insulin (micro U/mL)	9.2112	10.6548	10.3877	10.4799
RMSE: Effect of Active Insulin (1/min)	4.20E-03	4.20E-03	4.40E-03	4.20E-03

Table IX: Plant-model mismatch, elderly patient

4.4 Increased Measurement Noise

The accuracy of glucose monitors varies across manufacturers due to the design and sampling method that is utilized for the monitor. The current FDA standard for glucose monitors available on the consumer market requires the monitor to be within +/- 15% of a reference measurement throughout the range of the monitor [22]. As a result, the filters were compared using the initial estimate and covariance from the case of good filter initialization and high confidence using (equation 4.2) but with an increased measurement noise of 10 mg/dL and 15 mg/dL. In the case of a measurement noise of 10 mg/dL, the results from the 4 filters can be seen in figures 26, 27, and 28 below.

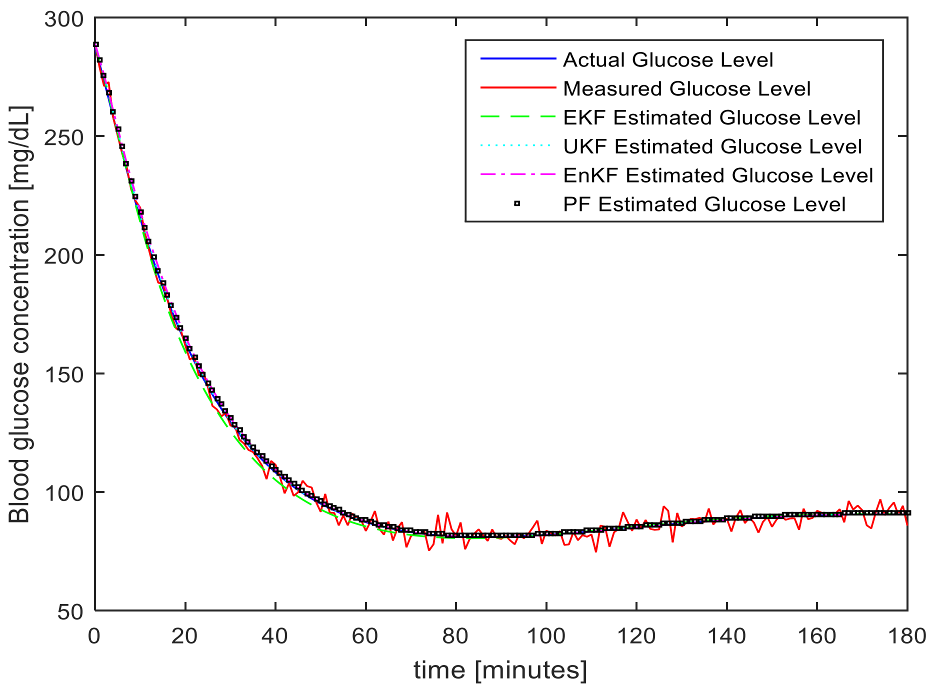

Figure 26: Increased measurement noise: 10 mg/dL, glucose concentration vs. time

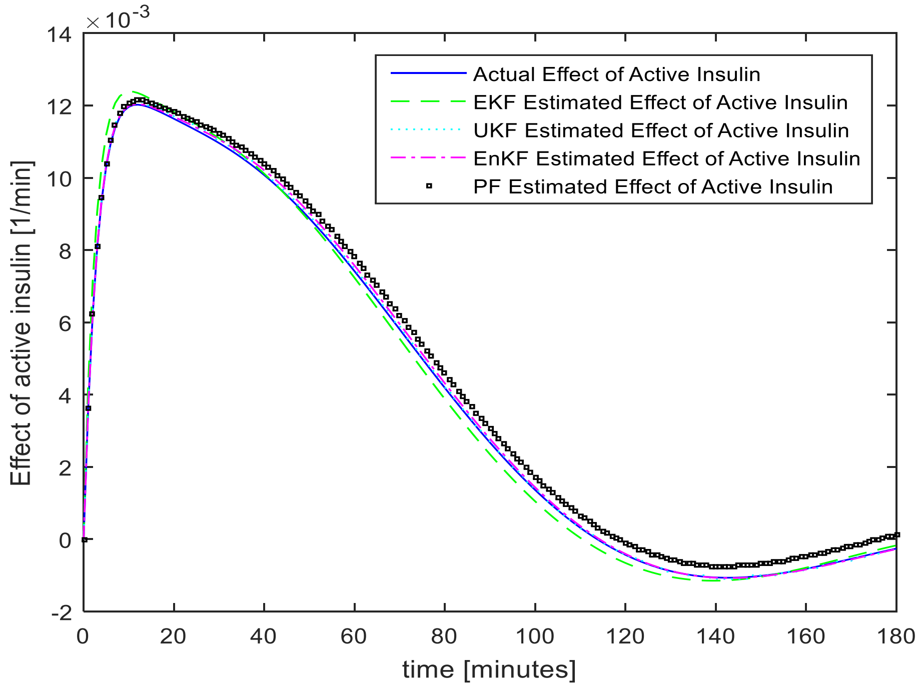

Figure 27: Increased measurement noise: 10 mg/dL, effect of active insulin vs. time

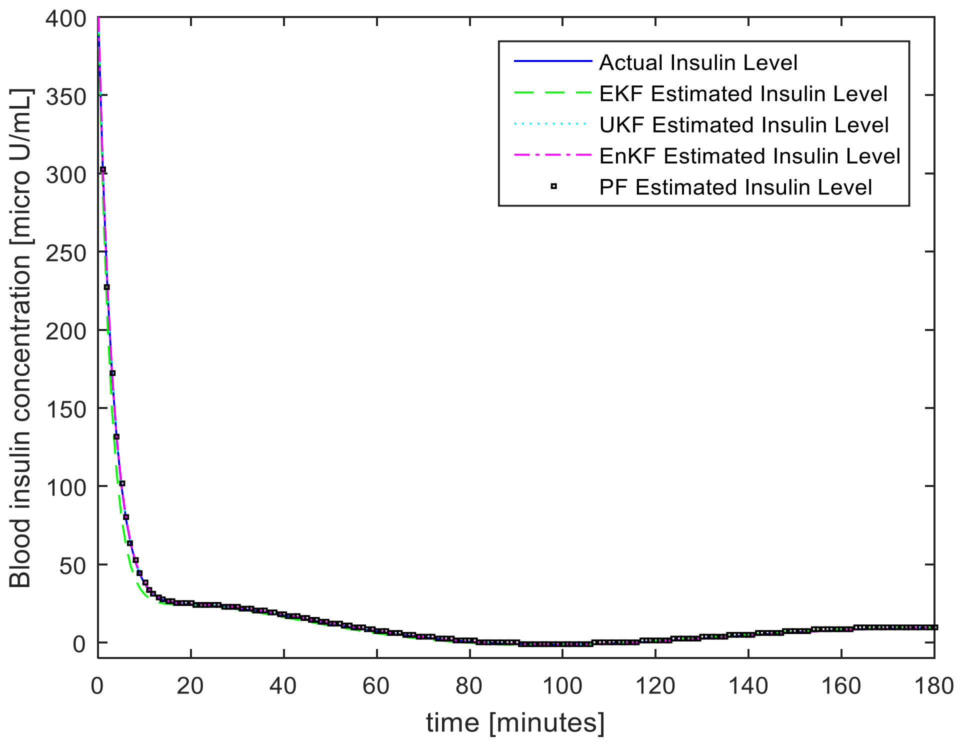

Figure 28: Increased measurement noise: 10 mg/dL, insulin concentration vs. time

The results of the filters are summarized in table X below. The unscented Kalman filter had the lowest RMSE. In general, there was a negligible difference in the RMSE between the cases of measurement noise of 10 mg/dL and 5 mg/dL from section 4.1.

	EKF	UKF	EnKF	PF
CPU time (s)	0.0191	6.0214	121.6147	101.1347
RMSE: Glucose (mg/dL)	3.1281	0.5285	1.3434	1.1003
RMSE: Insulin (micro U/mL)	12.1693	0.3903	0.3912	0.6676
RMSE: Effect of Active Insulin (1/min)	1.30E-03	5.62E-05	9.89E-05	3.27E-04

Table X: Measurement noise 10 mg/dL

The case study was then conducted using the same initial estimate and covariance from equation 4.2. In the case of a measurement noise of 15 mg/dL, the results from the 4 filters can be seen in figures 29, 30, and 31 below.

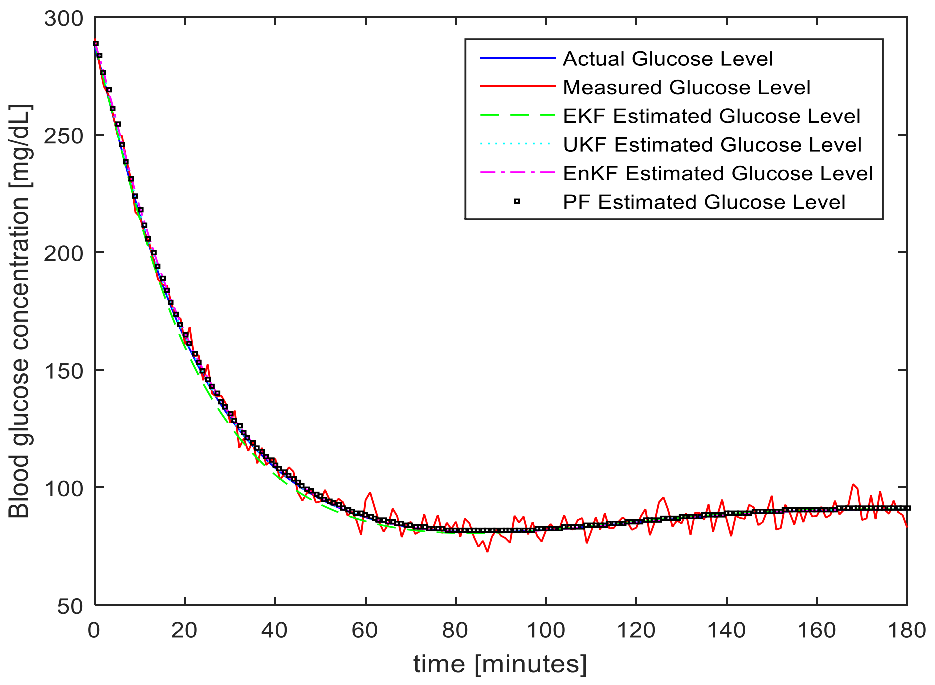

Figure 29: Increased measurement noise: 15 mg/dL, glucose concentration vs. time

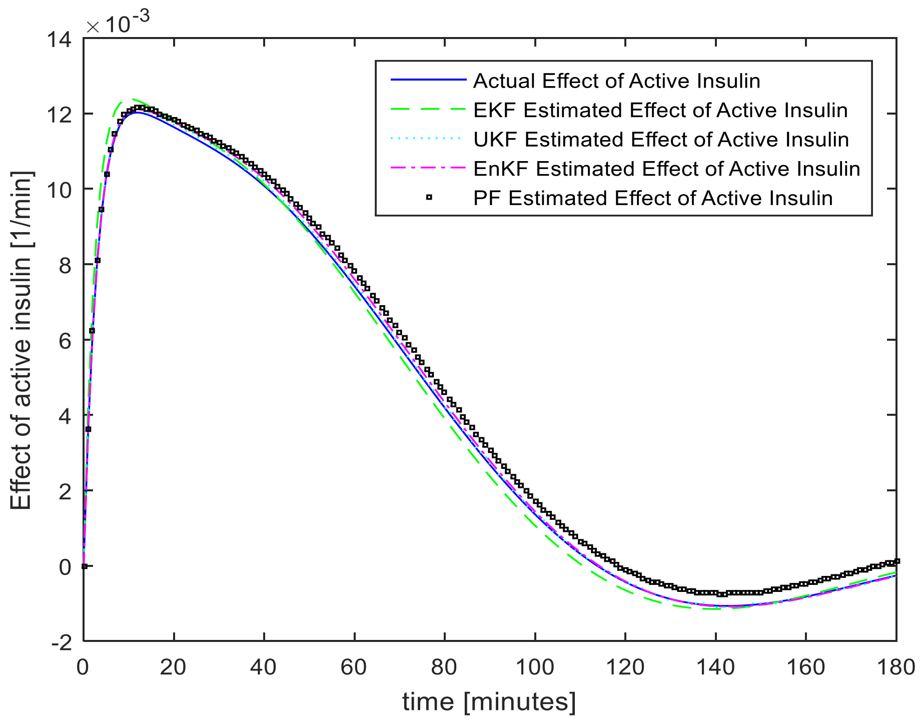

Figure 30: Increased measurement noise: 15 mg/dL, effect of active insulin vs. time

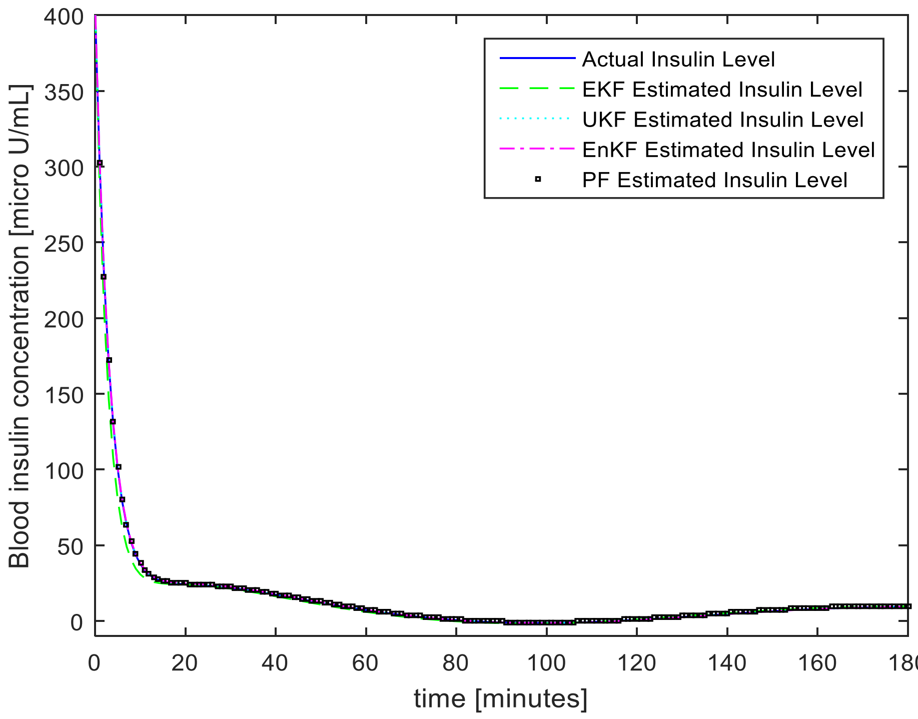

Figure 31: Increased measurement noise: 15 mg/dL, insulin concentration vs. time

The results of the filters are summarized in table XI below. The unscented Kalman filter had the lowest RMSE. In general, there was a negligible difference in the RMSE between the cases of measurement noise of 15 mg/dL and 10 mg/dL.

	EKF	UKF	EnKF	PF
CPU time (s)	0.0183	6.0064	122.9872	98.0083
RMSE: Glucose (mg/dL)	3.2576	0.498	1.2318	1.0833
RMSE: Insulin (micro U/mL)	12.2939	0.2752	0.3786	0.6673
RMSE: Effect of Active Insulin (1/min)	1.30E-03	4.41E-05	9.41E-05	3.27E-04

Table XI: Measurement noise 15 mg/dL

4.5 Multiple Glucose Ingestions

Since eating schedules may be irregular at times, this case study compared the performance of the filters if a 2nd ingestion of approximately half of the original amount of glucose occurs 60 minutes after the original ingestion. This was done in the simulation by defining the glucose concentration at 60 minutes as 140 mg/dL while leaving the insulin concentration and effect of active insulin unchanged. This allows for a more realistic response by the system after the sudden introduction of glucose. This could represent the patient eating a snack an hour after a regular meal. The performance of the filters was first compared for the scenario where the filters did not know about the second glucose ingestion. The initial state estimate and covariance were the same as equation 4.2. The results from the 4 filters can be seen in figures 32, 33, and 34.

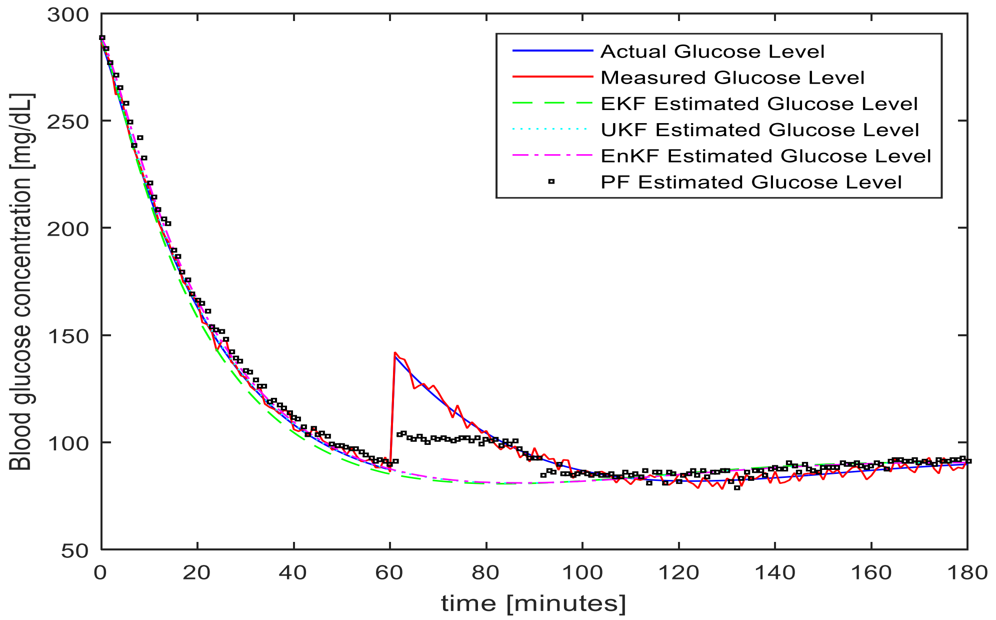

Figure 32: Multiple glucose ingestions, filter does not know, glucose concentration vs. time

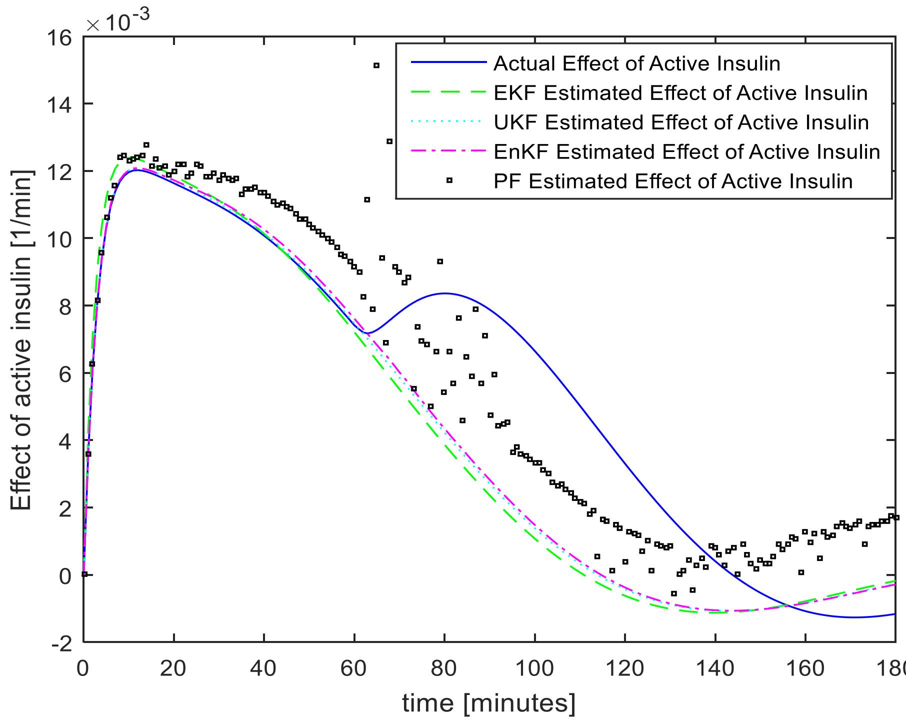

Figure 33: Multiple glucose ingestions, filter does not know, effect of active insulin vs. time

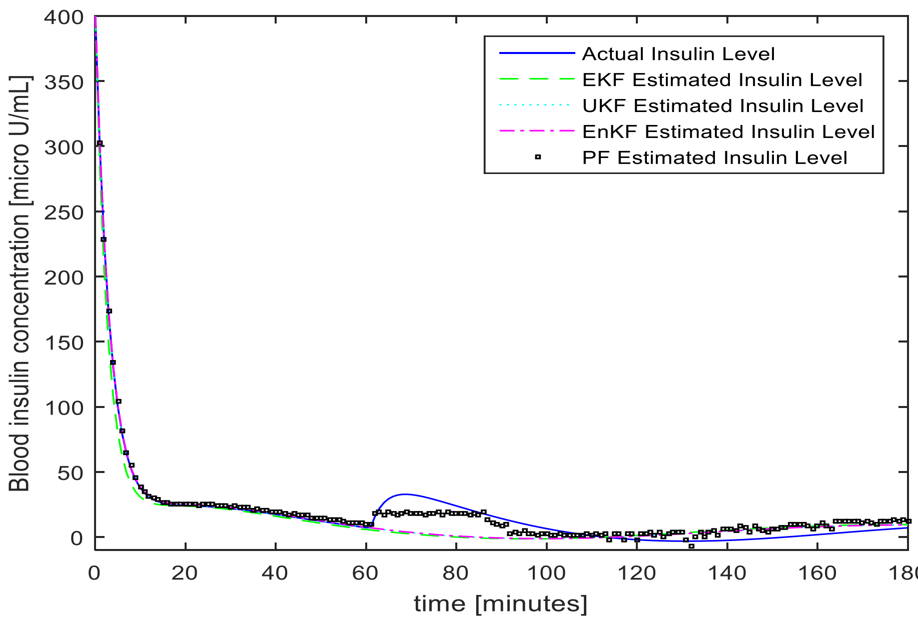

Figure 34: Multiple glucose ingestions, filter does not know, insulin concentration vs. time

The results of the filters are summarized in table XII below. The particle filter had the lowest RMSE. As seen in the figures above, the particle filter was the only filter that was able to respond to the 2nd ingestion of glucose.

	EKF	UKF	EnKF	PF
CPU time (s)	0.020313	6.0273	103.683	103.9756
RMSE: Glucose (mg/dL)	14.01532	13.6922	13.7116	8.3992
RMSE: Insulin (micro U/mL)	16.20965	10.4184	10.3612	5.7244
RMSE: Effect of Active Insulin (1/min)	3.02E-03	2.60E-03	2.60E-03	2.10E-03

Table XII: Multiple glucose ingestions, filter does not know

The scenario of multiple glucose ingestions was then studied where the filters do know a second glucose ingestion has occurred, this could be done in practice by having the patient specify to the process controller that they have eaten a meal. The importance of this is due to the equation for the change in insulin concentration being an explicit function of time since the glucose ingestion or injection. The results of this case study can be seen in figures 35, 36, and 37 below.

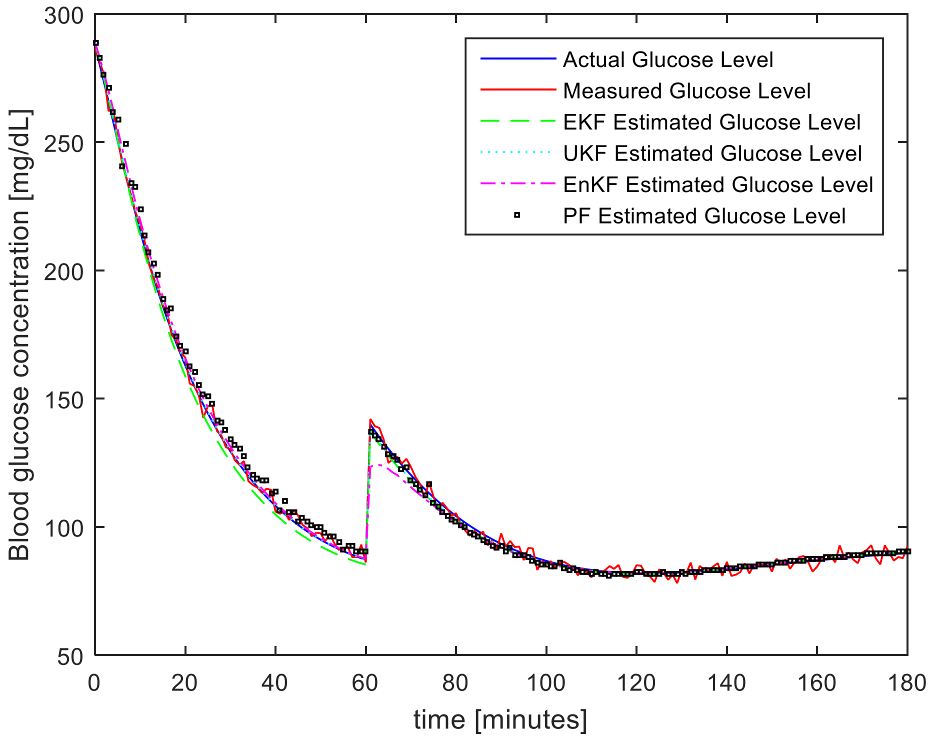

Figure 35: Multiple glucose ingestions, filter knows, glucose concentration vs. time

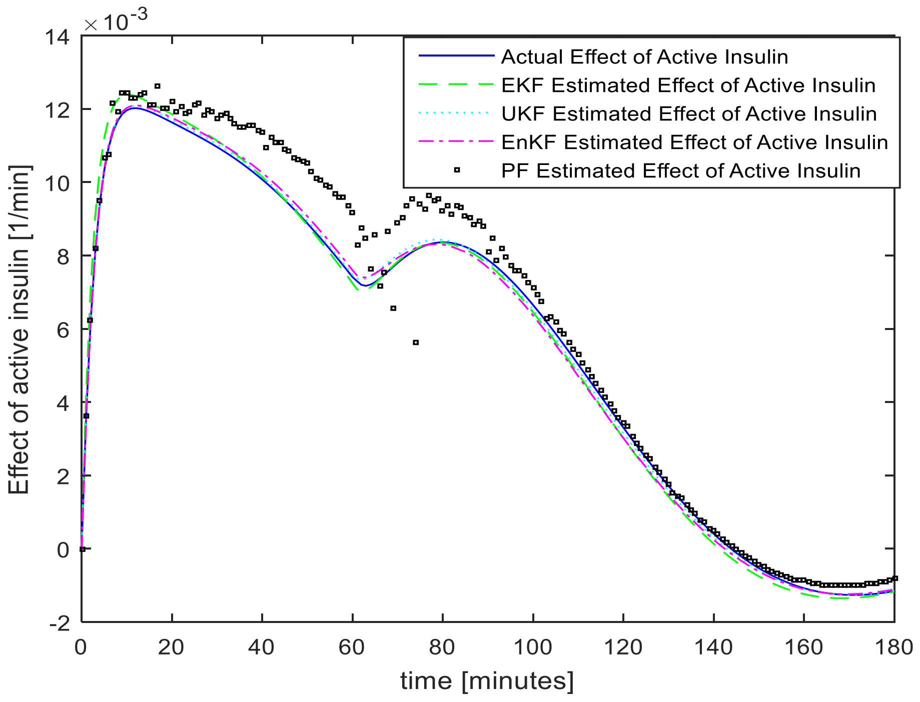

Figure 36: Multiple glucose ingestions, filter knows, effect of active insulin vs. time

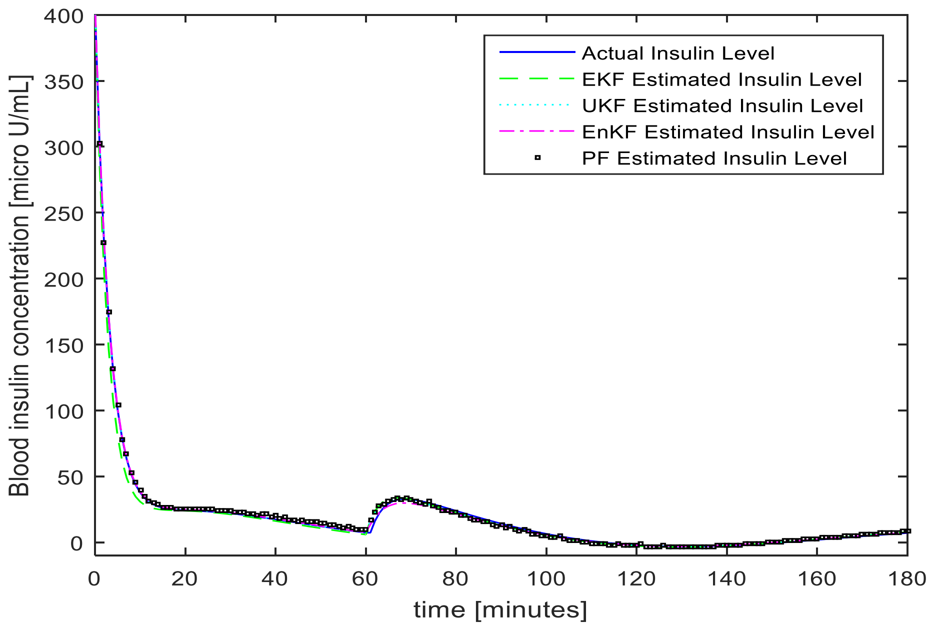

Figure 37: Multiple glucose ingestions, filter knows, insulin concentration vs. time

The results of the filters are summarized in table XIII below. The unscented Kalman filter had the lowest RMSE. In this scenario, all 4 of the filters were able to effectively respond to the 2nd glucose ingestion and eventually converge to the true state.

	EKF	UKF	EnKF	PF
CPU time (s)	0.020781	6.595625	118.0676563	99.1173
RMSE: Glucose (mg/dL)	4.964501	0.9659688	2.597856583	3.1916
RMSE: Insulin (micro U/mL)	12.26026	1.1124967	1.142380911	1.9218
RMSE: Effect of Active Insulin (1/min)	1.29E-03	1.39E-04	1.65E-04	7.98E-04

Table XIII: Multiple glucose ingestions, filter knows

CHAPTER V

CONCLUSION AND FUTURE WORK

Robust control of glucose and insulin levels is essential for diabetics to live healthy normal lives. The current technology that is used to control the glucose concentration does not take into account the dynamics of the glucose and insulin system but rather keeps the concentrations within a range that does not result in serious medical emergencies. The current treatment of diabetes is to inject insulin with syringes or insulin pumps in order to compensate for the impaired glucose tolerance of the patient. The dosages are increased or decreased by measuring the glucose concentration. The design of future insulin pumps could be coupled with continuous glucose monitors in order to create a process controller that would function as an artificial pancreas.

The extended Kalman filter, unscented Kalman filter, ensemble Kalman filter, and particle filter were applied to estimate the glucose concentration, insulin concentration, and effect of active insulin in the human body. The Minimal Model created by Bergman was used as the system model. The performance of the filters was compared to the cases of good filter initialization, poor filter initialization, plant-model mismatch, and multiple glucose ingestions.

It was observed that it is feasible to estimate the glucose concentrations, insulin concentrations, and effect of active insulin based off of only glucose measurements. Although computational times varied between the filters and for each case study, the computational time was low enough that the filters would be able to be implemented in real time for a process controller. In the case of poor filter initialization, the filters eventually did converge to the true state. As seen in the plant-model mismatch case studies, the performance of the filters is sensitive to the estimation of the parameters during the glucose tolerance test. Since these parameters change as patients age, the parameters will have to be re-estimated periodically. The case study that involved increasing the measurement noise from the glucose measurement had a negligible effect on the performance of the filters. As evidenced by the performance of the filters in the case study involving multiple glucose ingestions, the actual occurrence of a meal may have to be specified to the process controller depending on the choice of filter as only the particle filter was able to respond to the ingestion without having access to a system equation describing it.

Future Work

The variants of the system model that was used for this study could be investigated for similar case studies to see if they yield better results. Increasing the complexity may result in an increased burden on the state estimators but may result in better filtering of the data. In the future, other Kalman based or Monte Carlo based filters could be applied to the problem. Once the state estimation problem of the glucose and insulin system is solved, the next stage of the research will be the actual design and implementation of a potential process controller.

REFERENCES

[1] P. S. Maybeck, Stochastic Models, Estimation, and Control, New York, NY: Academic Press, 1979.

[2] C. M. Crowe, "Data reconciliation - progress and challenges," *Journal of Process Control*, vol. 6, no. 2, pp. 89-98, 1996.

[3] S. Bai, J. Thibault and D. McCean, "Dynamic data reconciliation: Alternative to Kalman filter," *Journal of Process Control*, vol. 16, pp. 485-498, 2005.

[4] Center for Disease Control and Prevention, "Diabetes Latest Data and Statistics," 2016. [Online]. Available: http://www.cdc.gov/features/diabetesfactsheet/.

[5] U.S. National Library of Medicine, "Diabetes," 2016. [Online]. Available: https://www.nlm.nih.gov/medlineplus/diabetes.html.

[6] U. S. National Library of Medicine, "Blood sugar test," 2016. [Online]. Available: https://www.nlm.nih.gov/medlineplus/ency/article/003482.htm.

[7] Diabetes Knowledgebase, "The Why and How of Maintaining Normal Blood Glucose Level," 2016. [Online]. Available: http://diabeteskb.org/the-why-and-how-of-maintaining-normal-blood-glucose-level/.

[8] National Institute of Diabetes and Digestive and Kidney Diseases, "Continuous Glucose Monitoring," 2016. [Online]. Available: http://www.niddk.nih.gov/health-information/health-topics/Diabetes/continuous-glucose-monitoring/Pages/index.aspx.

[9] Medtronic, "MiniMed 530G System," 2016. [Online]. Available: http://www.medtronicdiabetes.com/products/minimed-530g-diabetes-system-with-enlite.

[10] A. J. Laguna, P. Rossetti, J. Ampudia-Blasco, J. Vehi and J. Bondia, "Experimental blood glucose interval identification of patients with type 1 diabetes," *Journall of Process Control*, vol. 24, pp. 171-181, 2014.

[11] G. Pacini and R. N. Bergman, "MINMOD: a computer program to calculate insulin sensitivity and pancreatic responsivity from the frequently sampled intravenous glucosetolerance test," *Computer Methods and Programs in Biomedicine,* vol. 23, pp. 113-122, 1986.

[12] R. N. Bergman, "Minimal Model: Perspective from 2005," *Hormone Research*, vol. 64, no. 3, pp. 8-15, 2005.

[13] N. Van Riel, "Minimal Models for Glucose and Insulin Kinetics," 5 February 2004. [Online]. Available: http://cbio.bmt.tue.nl/~nvriel/parameter_estimation/VanRiel 20Minimal 20Models 20for 20Glucose20 and 20Insulin.pdf.

[14] A. Kartono, "Modified minimal model for effect of physical exercise on insulin sensitivity and glucose effeciveness in type 2 diabetes and healthy human," *Theory in Biosciences*, vol. 132, pp. 195-206, 2013.

[15] D. Araujo-Vilar, C. A. Rega-Liste, D. A. Garcia-Estevez, F. Sarmiento-Escalona, V. Mosquera-Tallom and J. Cabezas-Cerrato, "Minimal model of glucose metabolism: Modified equaitons and its application in the study of insulin

sensitivity in obese subjects," *Diabetes Research and Clinical Practice,* vol. 39, pp. 129-141, 1998.

[16] C. Eberle and C. Ament, "The Unscented Kalman Filter estimates the plasma insulin from glucose measurement," *Biosystems,* vol. 103, pp. 67-72, 2011.

[17] S. Ungarala, "Computing arrival cost parameters in moving horizon estimationusing sampling based filters," *Journal of Process Control,* vol. 19, pp. 1576-1588, 2009.

[18] Z. Shen and Y. Tang, "A modified ensemble Kalman particle filter for non-Gaussian systems with nonlinear measurement functions," *Journal of Advances in Modeling Earth Systems,* vol. 7, pp. 50-66, 2014.

[19] N. J. Gordon, D. J. Salmond and A. F. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *IEEE Proc.-F Radar Signal Process,* vol. 140, no. 2, pp. 107-113, 1993.

[20] J. D. Hol and F. Gustafsson, "On Resampling Algorithms for Particle Filters," *IEEE Nonlinear Statistical Signal Processing Workshop,* pp. 79-82, 2006.

[21] S. Sarkka, *Particle Filtering - Sequential Importance Resampling and Rao - Blackwellized Particle Filtering,* 2012.

[22] G. Freckman, C. Schmid, A. Baumstark, S. Pleus, M. Link and C. Haug, "System Accuracy of 43 Blood Glucose Monitoring Systems for Self-Monitoring of Blood Glucose according to DIN EN ISO 15197," *Journal of Diabetes Science and Technology,* vol. 6, no. 5, pp. 1060-1075, 2012.

APPENDIX

Extended Kalman Filter

```
%EKF_MM
clc
clear
global k
%Initial state
x = [287;0;404];
n = length(x);
m = 1;
%Number of time steps
N = 180;
t = [1:N];
%Noise covariances
Q = 0.0001^2*eye(n);
R = sqrt(5)^2*eye(m);
%Handles for Models and Jacobian functions
f_func = @glucose_f1;
df_func = @glucose_df1_dx;
h_func = @glucose_h;
dh_func = @glucose_dh_dx;
fe_func = @glucose_f1;
dfe_func = @glucose_df1_dx;

for j=1:1
    EKF_cputime=cputime;
%Storage
X = zeros(n,N);
%Generate states
X(:,1) = feval(f_func,x)+(diag(Q).^0.5).*randn(size(x));
for k = 2:N
    X(:,k) = feval(f_func,X(:,k-1))+(diag(Q).^0.5).*randn(size(x));
end

Y = feval(h_func,X);
Y = Y+diag(R).^0.5.*randn(size(Y));
%Filter initialization
xf = [289;1e-7;406]; %good filter initialization
%xf = [385;1.333e-7;541]; %bad filter initialization
Pf = [4 0 0;0 1e-9 0;0 0 4]; %high confidence
%Pf = [40 0 0;0 1e-8 0;0 0 40]; %low confidence
%Pf = 0.5^2*eye(n);
%Storage;
Xf = zeros(size(X));
%Filter loop
```

```
for k=1:N
    F = feval(dfe_func,xf);  %will be different than system during
plant model mismatch dfe_func and fe_func
    xf = feval(fe_func,xf);
    Pf = F*Pf*F'+Q;
    H = feval(dh_func,xf);
    K = Pf*H'*inv(H*Pf*H'+R);
    xf = xf+K*(Y(:,k)-feval(h_func,xf));
    Pf = Pf-K*H*Pf;
    Xf(:,k) = xf;
    Kh(:,k)=K;
end
EKF_cputime=cputime-EKF_cputime
comptime(j,:)=EKF_cputime;
%Mean Squared Error
     Eg=sqrt(sum((1/k)*(X(1,1:k)-Xf(1,1:k)).^2));
     Ex=sqrt(sum((1/k)*(X(2,1:k)-Xf(2,1:k)).^2));
     Ei=sqrt(sum((1/k)*(X(3,1:k)-Xf(3,1:k)).^2));

     Error(j,:)=[Eg Ex Ei]
end
 Avg_Error=mean(Error)
 Avg_CPU=mean(comptime)

%MSE = sum(sum((X-Xf).^2))/(n*N);
figure(1)
plot(t,X(1,:),t,Y(1,:),t,Xf(1,:),'--g')
   xlabel('time [minutes]');
    ylabel('Blood glucose concentration [mg/dL]');
    legend('Actual Glucose Level','Measured Glucose Level','EKF
Estimated Glucose Level');
figure(2)
plot(t,X(2,:),t,Xf(2,:),'--g')
 xlabel('time [minutes]');
    ylabel('Effect of active insulin [1/min]');
    legend('Actual Effect of Active Insulin','EKF Estimated Effect of
Active Insulin');

figure(3)
plot(t,X(3,:),t,Xf(3,:),'--g')
axis([0,180,-10,400])
  xlabel('time [minutes]');
    ylabel('Blood insulin concentration [micro U/mL]');
    legend('Actual Insulin Level','EKF Estimated Insulin Level');
```

<u>Function file for EKF: System function</u>

```
%glucose_f1
function x_out = glucose_f1(x)
global t k
%parameters
SG = 0.03082; %[1/min] glucose effectiveness
SI = 5.07e-4; %[mL/uU*min] insulin sensitivity (affects frequency of
oscillation)
```

```
k1 = 0.3; %[1/min] decay rate of blood insulin
k3 = 0.02093; %[1/min]
Gamma = 0.003349; %[1/min^2]

Ib = 7.3; % [mU/L] basal blood insulin concentration
Gb = 89.5; % [mg/dL] basal blood glucose concentration
dt = 1;

x_out(1,:) = x(1)+dt*(SG*(Gb-x(1))-x(2)*x(1));
x_out(2,:) = x(2)+dt*(k3*(SI*(x(3)-Ib)-x(2)));
x_out(3,:) = x(3)+dt*((Gamma*k*(x(1)-Gb))-(k1*(x(3)-Ib)));
```

Function file for EKF: System Jacobian

```
%glucose_df_dx
function F = glucose_df1_dx(x)
global k t
%parameters



SG = 0.03082; %[1/min] glucose effectiveness
SI = 5.07e-4; %[mL/uU*min] insulin sensitivity (affects frequency of
oscillation)
k1 = 0.3; %[1/min] decay rate of blood insulin
k3 = 0.02093; %[1/min]
Gamma = 0.003349; %[1/min^2]

Ib = 7.3; % [mU/L] basal blood insulin concentration
Gb = 89.5; % [mg/dL] basal blood glucose concentration

dt = 1;
F=[1+dt*(-SG-x(2)),dt*-x(1),0;0,1-dt*k3,dt*k3*SI;dt*Gamma*k,0,1-dt*k1];
```

Function file for EKF: Measurement function

```
%glucose_h
function y = glucose_h(x)
y = x(1,:);
```

Function file for EKF: Measurement Jacobian

```
%glucose_dh_dx
function H = glucose_dh_dx(x)
H=[1 0 0];
```

Function file: Minimal Model

```
%minimod_v86



function dydt = minimod_v86(t,y)

global G I X Param
% dydt = zeros(size(y));
G = y(1); %[mg/dL] Glucose level
X = y(2); %[1/min] Effect of Active Insulin
I = y(3); %[uU/mL] Insulin level
%Parameters from Bergman 1986
SG = 0.03082;%0.03082; %[1/min] glucose effectiveness 2.6e-2
SI = 5.07e-4;%1.2e-4; %[mL/uU*min] insulin sensitivity (affects
frequency of oscillation)
k1 = 0.3;%0.3; %[1/min] decay rate of blood insulin 0.27
k3 = 0.02093;%0.02093; %[1/min] 0.025
Gamma = 0.003349;%0.003349; %[1/min^2]  0.0041

Ib = 7.3; % [microU/mL] basal blood insulin concentration
Gb = 89.5; % [mg/dL] basal blood glucose concentration

Param = [SG SI k1 k3 Gamma];

dGdt = SG*(Gb-G)-X*G;
dXdt = k3*(SI*(I-Ib)-X);
dIdt = (Gamma*(G-Gb)*t)-(k1*(I-Ib));
dydt = [dGdt; dXdt; dIdt];
end
```

Unscented Kalman Filter

```
%minimalmodel_v18_UKF.m
%original Bergman minimal model for blood glucose
clc
clear
```

```
global G I X  Gm  t
Q=0.0001^2;%*eye(3);
R=5; %measurement noise covaraiance
for t=1:180
s=1; %sampling interval (minutes)
%Solve ODE
tspan = [0:t];

y0 = [287; 0; 404];
[t,y] = ode15s('minimod_v86', tspan, y0);
G=y(:,1);
X=y(:,2);
I=y(:,3);
%disp([t,G,X,I]);
L=length(tspan);

%next time step initial conditions

y0=[G; X; I];

end


  xt=[G X I];

  Gm=G+sqrt(R)*randn(L,1);
  yt=[Gm];


figure(1)
plot(t,G,t,Gm,'-r');
xlabel('time [minutes]');
ylabel('Blood glucose concentration [mg/dL]');
legend('Actual Glucose Level','Measured Glucose Level');


figure(2)
plot(t,X);
xlabel('time [minutes]');
ylabel('Effect of active insulin [1/min]');
legend('Actual Effect of Active Insulin');

figure(3)
plot(t,I);
xlabel('time [minutes]');
ylabel('Blood insulin concentration [microU/mL]');
legend('Actual Insulin Level');
    axis([0,180,-4,400])

%apply Unscented Kalman Filter

%Unscented Kalman Filter

%generate weights for mean
```

```
n=3; %number of states
N=7; %number of sigma points
alpha=0.8; %usually 0.5
beta=2;  %usually 2
kappa=3-n; %usually 3-n
lambda=alpha^2*(n+kappa)-n;
Wa0=lambda/(n+lambda);
Wc0=lambda/(n+lambda)+1-alpha^2+beta;
Wai=1/(2*(n+lambda));
Wci=1/(2*(n+lambda));
Wa=[repmat(Wai,3,1); Wa0; repmat(Wai,3,1)];   %weighted average 7x1
matrix
Wc=[repmat(Wci,3,1); Wc0; repmat(Wci,3,1)];   %weighted covariance
7x1matrix
Wad=diag(Wa);
Wcd=diag(Wc);


%100 realizations
for m=1:100 %change to 100 when ready

    % xt=xt+randn(size(xt))*(sqrt(Q));
 UKF_cputime=cputime;
%initial condition at t=0 or k=0

%generate initial sigma points (need 7)
Gavg=289*ones(1,7);
Xavg=1e-7*ones(1,7);
Iavg=406*ones(1,7);
sigma_points=[Gavg; Xavg; Iavg];
P=[4 0 0; 0 1e-9 0; 0 0 4]; %initial covariance matrix  Pxx High
Confidence
%P=[40 0 0; 0 1e-8 0; 0 0 40]; %initial covariance matrix  Pxx Low
Confidence
%Psqrt=chol(P)+chol(P)'-diag(diag(chol(P)))
 Psqrt=sqrtm(P)
Pmat=[-Psqrt zeros(3,1) Psqrt]
sigma_points=sigma_points+Pmat;

  Xhat=zeros(length(t),3); %pre-allocate space for estimate
  Xhat(1,:)=sigma_points(:,4); %store initial state estimate
  x=zeros(3,7);

  for rt=1:length(t)-1

%Solve ODE for each set of sigma points
tspan = [rt-1:rt];

for j=1:7
ic(j,:)=[sigma_points(:,j)];
[te,y] = ode15s('minimod_v86', tspan, ic(j,:));
x(:,j)=[y(end,1); y(end,2); y(end,3)];

end
    disp(x) %3x7 matrix
```

```
%generate new sigma points based on new mean and covariance

    %compute weighted mean
    xhat=x*Wa; %3x7 x 7x1 = 3x1 matrix

    sigma_points=[xhat(1)*ones(1,7);
xhat(2)*ones(1,7);xhat(3)*ones(1,7)];

    %compute covariance

    diff_co=x-sigma_points; %3x7 matrix
    P=diff_co*Wcd*diff_co'; %updated covariance matrix Pxx

    %Psqrt=chol(P)+chol(P)'-diag(diag(chol(P)))
    Psqrt=(sqrtm(P))
    Pmat=[-Psqrt zeros(3,1) Psqrt]
    sigma_points=sigma_points+Pmat;
    x=sigma_points
   % pause
  %try to update P
  xhat=x*Wa; %3x7 x 7x1 = 3x1 matrix

    sigma_points=[xhat(1)*ones(1,7);
xhat(2)*ones(1,7);xhat(3)*ones(1,7)];

    %compute covariance

    diff_co=x-sigma_points; %3x7 matrix
 %generate "fake" or "predictive" measurements

  Per=[sqrt(R)^2 sqrt(R)^2 sqrt(R)^2];
  Per=diag(Per);
  %Persqrt=chol(Per)+chol(Per)'-diag(diag(chol(Per)))
  Persqrt=sqrtm(Per)
  Persqrt=diag(Persqrt)
  Pmater=[-Persqrt; 0; Persqrt]'
  y=x(1,:)+Pmater
  ynew=yt(rt+1)*ones(size(y))
  diff=ynew-y
%pause
    %Kalman Gain

    xhatv=x*Wa; %weighted average of state sigma points

    yhatv=y*Wa; %weighted average of "fake" measurements

    Pyy=diff*Wcd*diff'+R; %measurement variance            maybe delete
this R

    Pxy=diff_co*Wcd*diff'; %updated covariance matrix

    K= Pxy*inv(Pyy)
    sumk=K*diff

    x=x+sumk      %Apply UKF to state


    %compute weighted average for estimate
    xhat=x*Wa;

    %generate new sigma points for next time step

    sigma_points=[xhat(1)*ones(1,7);
xhat(2)*ones(1,7);xhat(3)*ones(1,7)];
     %compute covariance

    diff_co=x-sigma_points;
    P=diff_co*Wcd*diff_co'; %updated covariance matrix

   % Psqrt=chol(P)+chol(P)'-diag(diag(chol(P)))
    Psqrt=(sqrtm(P));
    Pmat=[-Psqrt zeros(3,1) Psqrt];
    %initial conditions for net time step
    sigma_points=sigma_points+Pmat


  %Storage
  Xhat(rt+1,1)=xhat(1);
  Xhat(rt+1,2)=xhat(2);
  Xhat(rt+1,3)=xhat(3);
  Khat(rt,:)=K;
  diffhat(rt,:)=diff;
  Pxyhat(rt,:)=Pxy;
  Pyyhat(rt,:)=Pyy;
  diff_cohat(rt,1)=diff_co(1);
  diff_cohat(rt,2)=diff_co(2);
  diff_cohat(rt,3)=diff_co(3);
  end
UKF_cputime=cputime-UKF_cputime
disp(Xhat)

  %add filter estimate to figures

    figure(4)
    plot(t,xt(:,1),'-b',t,Gm,'-r', t,Xhat(:,1),'--g');
    xlabel('time [minutes]');
    ylabel('Blood glucose concentration [mg/dL]');
    legend('Actual Glucose Level','Measured Glucose Level','UKF
Estimated Glucose Level');


    figure(5)
    plot(t,xt(:,2),'-b', t, Xhat(:,2), '--g');
    xlabel('time [minutes]');
    ylabel('Effect of active insulin [1/min]');
    legend('Actual Effect of Active Insulin','UKF Estimated Effect of
Active Insulin');

    figure(6)
    plot(t,xt(:,3),'-b', t, Xhat(:,3),'--g');
    axis([0,180,-5,400])
    xlabel('time [minutes]');
    ylabel('Blood insulin concentration [micro U/mL]');
    legend('Actual Insulin Level','UKF Estimated Insulin Level');

%Mean Squared Error
     Eg=sqrt(sum((1/rt)*(xt(1:rt+1,1)-Xhat(1:rt+1,1)).^2));
     Ex=sqrt(sum((1/rt)*(xt(1:rt+1,2)-Xhat(1:rt+1,2)).^2));
     Ei=sqrt(sum((1/rt)*(xt(1:rt+1,3)-Xhat(1:rt+1,3)).^2));
     Error(m,:)=[Eg Ex Ei]
     comptime(m,:)=UKF_cputime;
end

 Avg_Error=mean(Error)
 Avg_CPU=mean(comptime)
```

Ensemble Kalman Filter

```
%minimalmodel_v14EnKF.m
%original Bergman minimal model for blood glucose
clc
clear
global G I X  Gm  t
R = 5; %measurement covariance
%Process Simulation

for t=1:180
s=1; %sampling interval (minutes)
%Solve ODE
tspan = [0:t];

y0 = [287; 0; 404];
[t,y] = ode15s('minimod_v86', tspan, y0);
G=y(:,1);
X=y(:,2);
I=y(:,3);
%disp([t,G,X,I]);
L=length(tspan);
```

```
%next time step initial conditions
y0=[G; X; I];

end

  Gm=G+sqrt(R)*randn(L,1); %glucose measurement

  xt=[G X I];
  yt=[Gm];


figure(1)
plot(t,G, t,Gm,'-r');
xlabel('time [minutes]');
ylabel('Blood glucose concentration [mg/dL]');
legend('Actual Glucose Level','Measured Glucose Level');


figure(2)
plot(t,X);
xlabel('time [minutes]');
ylabel('Effect of active insulin [1/min]');
legend('Actual Effect of Active Insulin');

figure(3)
plot(t,I);
xlabel('time [minutes]');
ylabel('Blood insulin concentration [mU/L]');
legend('Actual Insulin Level');


%apply Ensemble Kalman Filter

%Ensemble Kalman Filter
%use 100 realizations
for n=1:1  %change to 100 when ready
 EnKF_cputime=cputime;
%initial condition at t=0 or k=0
Nsamples=100;               %use 100 random possibilities of true state
at t=0
GEnKF=289+sqrt(4)*randn(Nsamples,1);
XEnKF=(1e-7+sqrt(1e-9)*randn(Nsamples,1));
IEnKF=406+sqrt(4)*randn(Nsamples,1);
x=[GEnKF XEnKF IEnKF];   %initial conditions: Nsamples X 3 Matrix

  Xhat=zeros(length(t),3); %pre-allocate space for estimation matrix
  Xhat(1,:)=[289 1e-7 406];
  %dynamics


  for rt=1:length(t)-1
         k=length(rt);
```

```
%Solve ODE

for j=1:length(x)  %pass each point through ode
      ic=x(j,:);

tspan = [rt-1:rt];

[te,y] = ode15s('minimod_v86', tspan, ic);
x(j,:)=[y(end,1),y(end,2),y(end,3)];
disp(x);
L=length(tspan);

%Storage

end

     y=x(:,1)+sqrt(R)*randn(length(x(:,1)),1);

      ynew=yt(rt)*ones(size(y));
      diff=ynew-y;


    %Kalman Gain

    xhatv=mean(x);

    yhatv=mean(y); %average of "fake" measurements


    Pyy=var(y)+R;
    Gcov=cov(x(:,1),y);
    Xcov=cov(x(:,2),y);
    Icov=cov(x(:,3),y);
    Pgy=Gcov(1,2);
    Pxy=Xcov(1,2);
    Piy=Icov(1,2);
    PXy=[Pgy; Pxy; Piy];

    K= PXy*(Pyy.^-1);
  %  K= Pxy*inv(Pyy)

    sumk=diff*K';

    x=x+sumk;      %Apply EnKF to state

    %compute average for estimate

    xhat=mean(x);



    %Storage
  Xhat(rt+1,1)=xhat(1);
  Xhat(rt+1,2)=xhat(2);
  Xhat(rt+1,3)=xhat(3);
  Khat(rt,:)=K;
  Pyyhat(rt,:)=Pyy;
  PXyhat(rt,:)=PXy;
  diffhat(rt,:)=diff;


  end
  EnKF_cputime=cputime-EnKF_cputime

    %add filter estimate to figures

  figure(4)
    plot(t,xt(:,1),'-b',t,Gm,'-r', t,Xhat(:,1),'--g');
    xlabel('time [minutes]');
    ylabel('Blood glucose concentration [mg/dL]');
    legend('Actual Glucose Level','Measured Glucose Level','EnKF
Estimated Glucose Level');


    figure(5)
    plot(t,xt(:,2),'-b', t, Xhat(:,2), '--g');
    xlabel('time [minutes]');
    ylabel('Effect of active insulin [1/min]');
    legend('Actual Effect of Active Insulin','EnKF Estimated Effect of
Active Insulin');

    figure(6)
    plot(t,xt(:,3),'-b', t, Xhat(:,3),'--g');
    axis([0,180,-5,400])
    xlabel('time [minutes]');
    ylabel('Blood insulin concentration [micro U/mL]');
    legend('Actual Insulin Level','EnKF Estimated Insulin Level');

%Mean Squared Error
     Eg=sqrt(sum((1/rt)*(xt(1:rt+1,1)-Xhat(1:rt+1,1)).^2));
     Ex=sqrt(sum((1/rt)*(xt(1:rt+1,2)-Xhat(1:rt+1,2)).^2));
     Ei=sqrt(sum((1/rt)*(xt(1:rt+1,3)-Xhat(1:rt+1,3)).^2));
    % format long
     Error(n,:)=[Eg Ex Ei]
     comptime(n,:)=EnKF_cputime;

end

 Avg_Error=mean(Error)
 Avg_CPU=mean(comptime)
```

Function file: Resample

```
function Y = resampleX_test(X,alpha,r)
%
%  By: Ron Abileah, Vista Research Inc
%
%  Original version:  November 10, 2005
%  Version 1.1        December 1, 2005
%                         - Corrected comments and mentioned similarity
to
%                         MATLAB function "resample."
%                         - Output Y is row (column) if input X is row
%                         (column)
%                         - Sets alpha to default value (1) if alpha is
not provided
%  Version 1.2        December 15, 2005
%                         - Corrected indexing bug discovered by Eike
%                         Rietsch.  This improved accuracy.
%  Version 1.3        October 23, 2006
%                         - Corrected bug in handling the value at
Nyquist,
%                         which caused a slight increase in
interpolation
%                         error.  The problem was pointed out by a user
who
%                         found that resampleX(x,1) did not return
exactly
%                         x.
%
%  Resamples X(n).  Y(n) = X(alpha*n), where alpha is a resample
interval.
%  For example, if X is data sampled at 1000 samples per second and you
%  would like to transform it to the equivalent of 1100 samples per
second
%  use alpha= 1000/1100 (.9091); for 800 sample per second
%  use alpha = 1000/800 (1.25).
%
```

```
%  ResampleX is similar to the MATLAB "resample" function (in the
Signal
%  Processcing Toolbox).  There are two differences:
%
%  (1) The MATLAB resample does some fancy schmancy interpolation of
the
%       original time series;  resampleX works on the Fourier trasnform
of
%       the time series.  The main benefit of FT processing is speed.
%  (2) MATLAB resample cputime depends on the value of alpha.  It runs
faster with
%       simple rational numbers.  The cputime of resampleX is
indpendent of
%       alpha.
%
%  The main reason for using resampleX instead of resample is speed.
In
%  test cases resampleX was generaly 5-20 times faster.    Use resample
if
%  your alpha values are simple rational numbers or numerical accuracy
is more
%  important.  Use resampleX for very general values of alpha or where
some
%  accuracy can be traded for speed.
%
%  ResampleX uses the fact that resampling a time series X by a factor
alpha is
%  equivalent to resampling the frequency samples of its transform by
m/alpha,
%  where m is a frequency index.
%
%  The calling sequence is one of the following
%
%              Y = resampleX(X,alpha)
%              Y = resampleX(X,alpha,r)
%              Y = resampleX
%
%  X can be real or complex. Output Y is the same length as X.
%  If alpha > 1 some Y's will be extrapolations beyond the end of X.
%  Extrapolated values are not realiable, so throw them away.  Keep
only
%  the first N/alpha values, where N is the original length of the
data.
%  Whne X is real, Y's may have small imaginary values due to
%  approximations.
%
%  Optional parameter r is an integer frequency interpolation factor.
The
%  function will work with r = 1, but r = 2,4, or more produces more
%  accurate results.  Use r=64 if you want very accurate results and
are
%  not too concerned about computing time.  The default value is r = 8.
%
%  Calling the function with no arguments produces a test signal and a
plot
%  of the original and resampled signal with alpha =0.95, r = 8.
%
```

```
%  -- Set r to its default value if not specified as input argument
if ~exist('r')
    r = 8; end

%  -- Produce test signal if there are no input arguments
if nargin == 0
    x1=0:pi/64:2*pi;
    X=exp(i*x1)+0.5*exp(i*3*x1)+0.25*exp(i*x1.^2);
    alpha = 0.95;
end

% The default value of alpha is 1
if nargin == 1
    alpha =1;
end

% -- zero pad X for frequency interpolation
Y=X;
N0=length(Y);  N=r*N0; N2 = N/2;
if r >1
    Y(N)=0;
end

%  -- Fourier transform the padded time series
Y = fft(Y);
n = 1:length(X);

%  -- Resample the Fourier transform
m = round((0:(N2-1))./alpha) - (0:(N2-1));
f = floor(alpha.*N2); m((f+1):end)=NaN;
m = [ m  0 -fliplr(m(2:end)) ] + (1:N) ;
m(find(isnan(m)))=N2+1;
Y=ifft(Y(m));
Y=Y(1:N0)/alpha;


if nargin>0
    return
end

%  -- Display results
figure
n=1:N0;
plot( n ,real(X),'-k' ,n ,imag(X),'--k',...
    n ,real(Y),'-r' ,n ,imag(Y),'--r')
xlabel('Sample number'); ylabel('Amplitude')
axis([1 N0 -2 2])
legend ('Original real', 'Original imaginary ', 'Resampled
real','Resampled imaginary')

end
```

Particle Filter

```
%minimalmodel_v1PF.m
%original Bergman minimal model for blood glucose
clc
clear
global G I X  Gm t
 R=5; %measurement noise variance (mg/dL)
%Process Simulation

for t=1:180
s=1; %sampling interval (minutes)
%Solve ODE
tspan = [0:t];

y0 = [287; 0; 404];
[t,y] = ode45('minimod_v86', tspan, y0);
G=y(:,1);
X=y(:,2);
I=y(:,3);
%disp([t,G,X,I]);
L=length(tspan);

%next time step initial conditions
y0=[G; X; I];

end

  Gm=G+sqrt(R)*randn(L,1);

  xt=[G X I];
  yt=[Gm];


figure(1)
plot(t,G, t,Gm,'-r');
xlabel('time [minutes]');
ylabel('Blood glucose concentration [mg/dL]');
legend('Actual Glucose Level','Measured Glucose Level');


figure(2)
plot(t,X);
xlabel('time [minutes]');
ylabel('Effect of active insulin [1/min]');
legend('Actual Effect of Active Insulin');

figure(3)
plot(t,I);
xlabel('time [minutes]');
ylabel('Blood insulin concentration [mU/L]');
legend('Actual Insulin Level');
```

```
%apply Gordon Salmond and Smith Particle Filter

%Particle Filter
%use 100 realizations
for n=1:1  %change to 100 when ready
    PF_cputime=cputime;
%initial condition at t=0 or k=0
Nsamples=100;               %use 100 random possibilities of true state
at t=0
GEnKF=289+sqrt(4)*randn(Nsamples,1);
XEnKF=(1e-7+sqrt(1e-9)*randn(Nsamples,1));
IEnKF=406+sqrt(4)*randn(Nsamples,1);
x=[GEnKF XEnKF IEnKF];   %initial conditions: Nsamples X 3 Matrix

  Xhat=zeros(length(t),3);
  Xhat(1,:)=[289 1e-7 406];
  %dynamics


  for rt=1:length(t)-1
        % k=length(rt);


%Solve ODE

for j=1:length(x)  %pass each point through ode
      ic=x(j,:);

tspan = [rt-1:rt];

[te,y] = ode15s('minimod_v86', tspan, ic);
Gen=y(end,1);
Xen=y(end,2);
Ien=y(end,3);
x(j,:)=[Gen,Xen,Ien];
disp([Gen,Xen,Ien]);
L=length(tspan);

%Storage

end

      %pause

  y=x(:,1)+sqrt(R)*randn(length(x(:,1)),1);

      ynew=yt(rt)*ones(size(y));
      diff=ynew-y;

    %compute weight of particles (using Gaussian PDF)

     w=(1/sqrt(2*pi*R))*exp((-1*(diff).^1.8)/(2*R)); %was 1.8
    %normalize weight
    w=w/sum(w);
    w=[w w w];

    %weighted average
    xhat=sum(x.*w);

    %Storage

  Xhat(rt+1,1)=xhat(1);
  Xhat(rt+1,2)=xhat(2);
  Xhat(rt+1,3)=xhat(3);


    %Resample
    x(:,1)=resampleX(x(:,1),0.9995); %was 0.35
    x(:,2)=resampleX(x(:,2),0.9995);
    x(:,3)=resampleX(x(:,3),0.9995);
  end
     PF_cputime=cputime-PF_cputime
    %add filter estimate to figures

  figure(4)
    plot(t,xt(:,1),'-b',t,Gm,'-r', t,Xhat(:,1),'--g');
    xlabel('time [minutes]');
    ylabel('Blood glucose concentration [mg/dL]');
    legend('Actual Glucose Level','Measured Glucose Level','PF
Estimated Glucose Level');


    figure(5)
    plot(t,xt(:,2),'-b', t, Xhat(:,2), '--g');
    xlabel('time [minutes]');
    ylabel('Effect of active insulin [1/min]');
    legend('Actual Effect of Active Insulin','PF Estimated Effect of
Active Insulin');

    figure(6)
    plot(t,xt(:,3),'-b', t, Xhat(:,3),'--g');
    axis([0,180,-5,400])
    xlabel('time [minutes]');
    ylabel('Blood insulin concentration [micro U/mL]');
    legend('Actual Insulin Level','PF Estimated Insulin Level');

%Mean Squared Error
     Eg=sqrt(sum((1/rt)*(xt(1:rt+1,1)-Xhat(1:rt+1,1)).^2));
     Ex=sqrt(sum((1/rt)*(xt(1:rt+1,2)-Xhat(1:rt+1,2)).^2));
     Ei=sqrt(sum((1/rt)*(xt(1:rt+1,3)-Xhat(1:rt+1,3)).^2));

     Error(n,:)=[Eg Ex Ei]
     comptime(n,:)=PF_cputime;

end
```

```
Avg_Error=mean(Error)
Avg_CPU=mean(comptime)
```