

 **DTU Compute**  
Department of Applied Mathematics and Computer Science

# Optimal Diabetes Treatment

## Meal detection

Emma Victoria Lind (s191159)  
Mona Saleem (s204226)  
Mariana de Sá Madsen (s201205)

Kongens Lyngby 2022



**DTU Compute**  
**Department of Applied Mathematics and Computer Science**  
**Technical University of Denmark**

Matematiktorvet  
Building 303B  
2800 Kongens Lyngby, Denmark  
Phone +45 4525 3031  
[compute@compute.dtu.dk](mailto:compute@compute.dtu.dk)  
[www.compute.dtu.dk](http://www.compute.dtu.dk)

# Abstract

---

## Background

Doctors' consultations with their patients are very brief and many diabetes patients do not treat themselves correctly. They either forget to give the bolus insulin or do not give enough. Therefore, to assist doctors the GRID algorithm has been optimized by tuning the  $G_{min}$  values.

## Methods

The GRID algorithm was tested on 57 virtual patients and simulated by a closed loop system over 30 days. In total, 90 meals were expected to be detected with size between 50-150 g CHO. The ranges for tuning the  $G_{min}$  values were:

$$G_{min,1} \in \{120, 125, 130, 135, 140\}$$

$$G'_{min,2} \in \{0.6, 1.1, 1.6, 2.1, 2.6\}$$

$$G'_{min,3} \in \{0.5, 1.0, 1.5, 2.0, 2.5\}.$$

An evaluation function finding the number of true positive and false positive meals was used to evaluate the performance of the GRID algorithm with the respective combination of  $G_{min}$  values. The optimal  $G_{min}$  values meet the criteria of maximum 1 false positive per day and detecting a minimum of 70% of the 90 correct meals. Lastly, the tuned GRID algorithm was tested on clinical data and evaluated.

## Results

The optimal  $G_{min}$  values were found to be  $G_{min,1} = 120$ ,  $G'_{min,2} = 0.6$  and  $G'_{min,3} = 0.5$ . The GRID algorithm used on the clinical data resulted in detecting 548 meals throughout approximately 5 month. Further, it detected 21 meals in 3 days that was chosen to analyse. The  $G_{min,1}$  value at some meals is seen to be too high mainly when the blood glucose concentration is too low when a meal is ingested. Furthermore, the  $G'_{min,2}$  and  $G'_{min,3}$  are seen to possibly be too low at some points. This is mostly during night time that these low values could be an issue due to a natural increase in the blood glucose concentration such that it is not reasonable to assume a meal should be detected.

## Conclusion

The main limitation of the developed GRID algorithm is the 2-hour counter. This results in several meals not being detected because a meal has been detected within two hours before hand.



# Preface

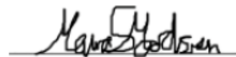
---

This 01666 Project work - Bachelor of Mathematics and Technology was prepared at the department of Applied Mathematics and Computer Science at the Technical University of Denmark.

Kongens Lyngby, June 20, 2022



Mona Saleem



Emma Victoria Lind (s191159)  
Mona Saleem (s204226)  
Mariana de Sá Madsen (s201205)



# Contents

---

<b>Abstract</b>	<b>i</b>
<b>Preface</b>	<b>iii</b>
<b>Contents</b>	<b>v</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Methods and background</b>	<b>3</b>
2.1 Control systems . . . . .	3
2.2 MVP model . . . . .	6
2.3 Numerical methods . . . . .	7
2.4 GRID algorithm . . . . .	10
<b>3 Simulation and evaluation</b>	<b>13</b>
3.1 Description of the clinical data . . . . .	13
3.2 Simulation scenarios . . . . .	13
3.3 Evaluation of the GRID algorithm . . . . .	22
<b>4 Results</b>	<b>25</b>
4.1 The optimal $G_{min}$ values . . . . .	25
4.2 Detection of meals on clinical data . . . . .	27
<b>5 Conclusion</b>	<b>35</b>
<b>A An Appendix</b>	<b>37</b>
<b>Bibliography</b>	<b>51</b>





# CHAPTER 1

## Introduction

---

As more people worldwide are being diagnosed with diabetes and with an approximation that over 750 million people will suffer from this disease in year 2045 [2] it is essential to find better solutions on how to treat this in people's everyday life. For people with diabetes their natural pancreas does not produce the correct amount or any insulin so people will have to inject insulin to keep their blood glucose concentration at a desirable level.

Artificial pancreas will help diabetes patients [3]. The point of an artificial pancreas is to reduce time in hypoglycemia where the blood glucose concentration is too low and also reduce the time in hyperglycemia where it is too high. Both hypo- and hyperglycemia can result in long term diseases, why it is desirable to improve quality of life for the patients.

Diabetes treatment has several challenges one of the most difficult ones being the estimation of the right meal size [3]. Therefore, being able to detect if a meal has been consumed recently together with the right amount of insulin can improve the treatment of diabetes by further assisting doctors in their brief consultations with their patients. This report will examine an implementation and tuning of the GRID algorithm in attempt to detect meals.

### *Problem area:*

This project will attempt to see whether it is possible to improve the GRID algorithm for detecting if a patient with type 1 diabetes has ingested a meal recently and if they forgot to take the corresponding amount of insulin with the meal. In this report we mainly shed light upon if tuning on solely the  $G_{min}$  values in the detection part of the algorithm is sufficient for an improvement of the GRID-algorithm or if other parameters are necessary as well.

### *Summary:*

In chapter 2 the methods used in this report is described. Chapter 3 includes a detailed description of how we have made our simulations and evaluations leading to the presentation of our results in Chapter 4. The results consists of a presentation of the optimal tuned parameters found in Section 4.1 and the analysis of the algorithm used on clinical data in Section 4.2. Furthermore, in this chapter the results are discussed. Lastly, in Chapter 5 the conclusion of the report is presented.



## CHAPTER 2

# Methods and background

---

## 2.1 Control systems

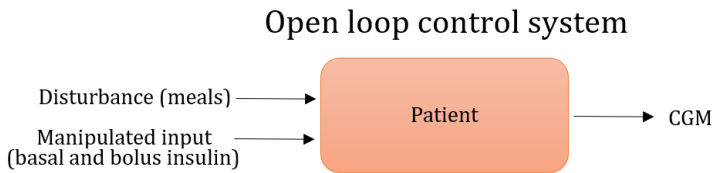
The pancreas works as the biological system inside the body controlling the insulin dose and the blood glucose concentration. However, this control system does not occur naturally for diabetes patients which means the patients will have to control the amount of insulin manually or rather use an artificial pancreas algorithm system [4]. This consists of a continuous glucose monitor denoted GCM sensor, a control algorithm and an insulin pump dosing the insulin [15]. There are two different types of insulin: the basal insulin which is a long acting insulin working through out the day and the bolus insulin which is a fast acting insulin injected at meals.

The artificial pancreas can be based on two types of control systems: open loop and closed loop. For open loop systems the input does not depend on the output whereas in the closed loop the input will depend on the output by evaluating the output continuously. That is, either the basal insulin is estimated beforehand and then not regularized subsequently or the basal insulin dose is automatically being controlled by the blood glucose concentration and vice versa. Feedback systems is what evaluates the output and controls the input in the closed loop but this will be elaborated later on.

For the clinical data the artificial pancreas is based on a closed loop system where the measurements of the blood glucose concentrations will be known. Later, we will be simulating several scenarios of virtual patients which is necessary for testing the validity of the GRID algorithm. Therefore, the virtual patients we will be simulating will also be controlled using this system. However, we will first try to simulate an open loop scenario since it is more simple than the closed loop.

### 2.1.1 Open loop

The main idea of the open loop system for the artificial pancreas is to control the basal insulin dose independently on the blood glucose concentration. That is the input does not depend on the output. This system is very simple and is illustrated in the figure below.

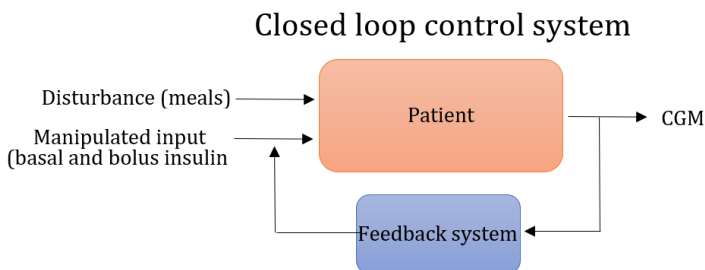


**Figure 2.1:** Open loop illustration

When we will be simulating this scenario it means that the basal insulin dose have been selected for every day beforehand. So the basal insulin dose will not be regularized continuously depending on the meals. We set the constant basal insulin rate at steady state which is  $25 \frac{\text{mU}}{\text{min}}$ . The open loop systems are good to start with since it is a simpler system that uses less computations given that it does not need to incorporate feedback system information into the control system [5]. However, using the open loop will not be enough later on when trying to approximate the artificial pancreas of the clinical data since the patients have been controlled by a closed loop with a feedback system.

### 2.1.2 Closed loop

The closed loop system improves the artificial pancreas by making the basal insulin dose and the blood measurements compatible. The figure below illustrates how the system works:



**Figure 2.2:** Closed loop illustration

The central principle is to use a feed back system to evaluate the output for every control measurement which then regularizes the input. Hence the feedback system is by definition what controls the right amount of basal insulin based on the current blood measurement.

### 2.1.3 Feedback systems

Feedback systems are commonly used in both technological and natural systems. The idea behind feedback systems are repeatedly correcting the difference in actual and desired results [6]. The feedback system will ensure that the basal insulin can be regularized based on the current blood glucose concentration. With this follows several properties such as being able to correct the difference in measurements and desired measurement. This property can be achieved by very simple feedback laws [7]. To begin with we illustrate a simple feedback

system described below:

$$u = \begin{cases} u_{max} & \text{if } e > 0 \\ u_{min} & \text{if } e < 0 \end{cases},$$

where  $e$  is the difference between the desired output and the actual output, and  $u_i, i \in \{max, min\}$  is the actuation command [7].

The advantages of this system is that it is very simple and does not use any parameters. However, the system tends to overreact whenever there is a small change in error since it will impact the entire output [8]. This results in the variables oscillating which can be a problem if the oscillations are not small enough [7]. Therefore, by extending the system the oscillation can be avoided.

### 2.1.3.1 PID control

The extended system we will be using is called the proportional–integral–derivative controller (the PID controller) which as the name indicates consists of three terms. The complete formula of the PID controller can be written as follows:

$$u(t) = k_p e(t) + k_i \int_0^t e(\tau) d\tau + k_d \frac{de(t)}{dt}. \quad (2.1)$$

The proportional term,  $k_p e$ , controls the oscillation of the variables. This is given by the control law as follows [8] :

$$u = \begin{cases} u_{max} & \text{if } e > 0 \\ k_p e & \text{if } e_{min} < e < e_{max} \\ u_{min} & \text{if } e < 0 \end{cases}. \quad (2.2)$$

When the error is in the interval given in the middle of equation (2.2) the behaviour of the controller is linear. This means that the control is proportional to the error with factor  $k_p$ , that is the controller gain.

However, the proportional control will often result in the output variables deviating from the desired variables which means that sometimes it will be necessary to achieve an error different from 0. [8] This leads to the integral term that ensures that the control is proportional to the integral of the error given as follows:

$$u(t) = k_i \int_0^t e(\tau) d\tau,$$

where  $k_i$  is the integral gain. [8]

Lastly, for providing the controller with the ability of predicting, the prediction of the error is used given by the following [8]:

$$e(t + k_d) \approx e(t) + k_d \frac{de(t)}{dt},$$

where  $k_d$  is the next time for the error.

Combining all these terms results in the fully mathematical formula for the PID control (2.1). Being able to utilize the PID controller for the purpose of controlling the basal insulin based on the glucose measurements, a closed loop simulation is required as described earlier.

### 2.1.4 Implementation of the PID controller as a function

The PID controller is implemented as a function for later simulating a closed loop scenario. The inputs will be the current blood glucose measurements ( $y_k$ ), the selected bolus insulin combined with the basal insulin as a vector  $\mathbf{U}$  ( $U$ ). The basal insulin will be zero, since it is to be updating continuously by the controller whereas the bolus insulin is already selected for each meal size. Furthermore, follows the vector of control parameters ( $\text{ctrlPar}$ ). This consists of 8 given parameters controlling the computation of the PID controller. Lastly, the vector of control states ( $\text{ctrlState}$ ) that consists of the previous blood glucose concentration and the current integral term. This is updated for every control step since it is also an output.

The implementation of this function can be seen in appendix A.

## 2.2 MVP model

The physiological processes inside the body is a network of dynamic changes where each compartment heavily depends and interacts with each other. For a person with diabetes there are three main components to always consider; the blood glucose concentration, the insulin concentration and the ingested carbohydrates from a given meal. These three factors are what will be used when modelling the dynamics.

The dynamics inside the body can be modelled for a simulated and virtual patient as a certain system of first order differential equations known as the Medtronic Virtual Patient model, also denoted MVP model. The MVP model is as follows [14]

$$\mathbf{f}(\mathbf{x}, \mathbf{u}, d, \mathbf{pf}) = \mathbf{x}'(t) = \begin{bmatrix} D_1'(t) \\ D_2'(t) \\ I_{sc}'(t) \\ I_p'(t) \\ I_{eff}'(t) \\ G'(t) \\ G_{sc}'(t) \end{bmatrix} = \begin{bmatrix} d(t) - \frac{D_1(t)}{\tau_m} \\ \frac{D_1(t) - D_2(t)}{\tau_m} \\ \frac{u(t)}{\tau_1 D_I} - \frac{I_{sc}(t)}{\tau_1} \\ \frac{I_{sc}(t) - I_p(t)}{\tau_2} \\ -p_2 I_{eff}(t) + p_2 S_I I_p(t) \\ -(GEZI + I_{eff}(t))G(t) + EGP_0 + \frac{1000 \cdot D_2(t)}{V_G \tau_m} \\ \frac{G(t) - G_{sc}(t)}{\tau_{sc}} \end{bmatrix}.$$

Where there the ingested meal is divided into two subsystems,  $D_1$  and  $D_2$ , and for the blood glucose and insulin concentration there are two measures for each; one in the blood plasma and the other beneath the skin denoted as subcutaneous. All these gives respectively  $G$ ,  $G_{sc}$ ,  $I_p$ ,  $I_{sc}$ . Lastly there is also the insulin effect given as  $I_{eff}$ . These measures change over time meaning they are time dependent. Furthermore, they depend on 10 individual parameters,  $\mathbf{pf}$ . These parameters can also be seen in the right-hand side of the MVP model, where each parameter represents an influence relevant to the mentioned variables,  $D(t)$ ,  $G(t)$  and  $I(t)$ , which differs for each person. This could for instance be the glucose effectiveness ( $GEZI$ ) or the endogenous glucose production ( $EGP_0$ ).

Therefore, the system can be described as a function  $\mathbf{f}(\mathbf{x}, \mathbf{u}, d, \mathbf{p}\mathbf{f})$  taking 4 inputs with different dimensions. The first is the state vector  $\mathbf{x}(t)$  given as the following:

$$\mathbf{x}(t) = \begin{bmatrix} D_1(t) \\ D_2(t) \\ I_{sc}(t) \\ I_p(t) \\ I_{eff}(t) \\ G(t) \\ G_{sc}(t) \end{bmatrix} .$$

The second input  $\mathbf{u}(t)$  is a 2-dimensional manipulated variable representing the basal and bolus insulin flow rate, the third  $d(t)$  is the ingestion rate of the meal and the fourth  $\mathbf{p}\mathbf{f}$  are the 10 individual parameters. The state vector  $\mathbf{x}(t)$  is the unknown independent variable that is to be solved from which we obtain the blood glucose concentration for the simulated patient. However, solving the model simply gives an output for one timepoint  $t$ . Therefore, in order to be able to estimate the desired basal and bolus insulin, it is necessary to solve the differential equation system for a longer frame of time. For this we have tackled the problem with a numerical approach which will be described in the following Section 2.3.

## 2.3 Numerical methods

The dynamics inside the body can as mentioned be approximated by using the MVP model, which means we can compute the change of the state vector, the disturbances given as meals and the insulin rate over time. This results in not being able to achieve the exact analytical function but instead using control steps of intervals of 5 minutes to approximate the exact glucose concentration and insulin rate at given time points.

For this purpose, we need to use numerical approaches which require an initial value and the derivative at initial state. Hence, we use the MVP model and guess an initial value that is most likely to approximate reality if we assume we begin at a steady state. At steady state we assume that the patient had just woken up: implying no food, no insulin and a glucose concentration at  $108 \frac{\text{mg}}{\text{dl}}$ . Our initial value guess will therefore satisfy the below equations.

$$\mathbf{f}(\mathbf{x}(0), \mathbf{u}, d, \mathbf{p}\mathbf{f}) = \mathbf{0}, \quad (2.3)$$

$$G - Gs = 0, \quad (2.4)$$

$$\text{bolus insulin} = 0, \quad (2.5)$$

where  $Gs$  is the blood glucose concentration at steady state  $108 \frac{\text{mg}}{\text{dl}}$  and  $G$  is the measured subcutaneous blood glucose concentration [9]. So equation (2.4) implies exactly that at the initial guess at steady state the measured blood concentration is  $108 \frac{\text{mg}}{\text{dl}}$ .

So based on the MVP model and this initial guess we will be able to compute the

derivative at  $t = 0$ , which we use to solve for the next state vector  $\mathbf{x}$  by using numerical approaches. Continuing this way we will be able to simulate several patients at any given time interval.

The last to mention is that we can not use the same numerical methods independently on the system since the system can either be stochastic or deterministic. A deterministic MVP model system is a simpler system where we with certainty compute the the same output every time. However, this is not a very good approximation of reality since there will always be some uncertainties of the processes inside the body and in the blood measurements hence we also need to expand to a stochastic system of differential equations. In the following Sections 2.3.1 and 2.3.2 we will elaborate which method to use when and how.

### 2.3.1 Explicit Euler

When the system of differential equations is deterministic the Explicit Euler method is used to simulate our patients. This method approximates solutions to the MVP model with the initial value problem [10].

The method uses the differentiated function value given by  $\mathbf{x}'(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}, d, \mathbf{pf})$  in a point  $\mathbf{x}_k$  to solve the function value in the point  $\mathbf{x}_{k+1}$ . When  $k = 0$  we have the initial function value  $\mathbf{x}_0$  given by the MVP model at steady state.

The method uses a fixed step size that is defined by the interval that the differential equation is solved over [10]. When simulating in this report we use control intervals of 5 minutes where we use the number of iterations  $N = 10$ . This means that for solving the differential equation in a specific interval  $[a, b]$  with a specific amount of iterations  $N$  the step size is given by  $h = \frac{[b-a]}{N}$ . Therefore, in this case the time intervals and the number of evaluations in each interval,  $N$ , does not change at any point, so the step size is fixed to  $h = \frac{5}{10} = 0.5$  and will not be updated in each iteration.

To specify, it means in every iteration the Explicit Euler method evaluates the next solution by computing the  $k$ 'th derivative, multiplying with the fixed step size  $h = 0.5$  and then adds the values of the  $k$ 'th  $\mathbf{x}$  vector. This gives the formula of the method as follows:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + h \cdot \mathbf{f}(\mathbf{x}_k, \mathbf{u}, d, \mathbf{pf}),$$

where  $k = 0, 1, \dots, N - 1$ . Overall this means that Explicit Euler method approximates the true function value for every step when solving the differential equation by the above equation.

### 2.3.2 Stochastic system of differential equations

As already mentioned, it is necessary to expand the system to a stochastic system of differential equations for ensuring an appropriate approximation of reality. This is due to the processes of blood glucose concentration inside the body undergoing some degree of randomness but also uncertainties in measurements of the blood glucose concentration. With this follows probabilities of the randomness and uncertainties which is most commonly assumed to be normally distributed. For the uncertainties of the blood measurements we add a zero-mean normal distributed measurement noise variable to the solved subcutaneous glucose concentration. Hence, we add it to the  $\mathbf{x}(6)$  in the MVP model. However, to add the process noise happening inside the body we will need a more complex way described in the following subsection.



### 2.3.2.1 Brownian motion

There are several ways to add the stochastic term of normally distribution to the system of differential equations. One common type of noise is the Brownian motion. It is one of the most important stochastic processes both in theory and in application [11]. Brownian motion is a random variable  $W(t)$  that depends on the time given over a discrete interval  $t \in \{0, 1, \dots, T\}$ . For the variable to be Brownian motion, the following three conditions must be fulfilled [12]:

- 1.  $W(0) = 0$  with probability 1.
- 2. For two different time points  $t$  and  $s$  in the time interval  $T$  such that  $0 \leq s \leq t \leq T$ , the increment  $W(t) - W(s)$  needs also to be a random variable that is normally distributed with mean zero and variance  $t - s$ . This is also given as  $W(t) - W(s) \sim \sqrt{t - s}N(0, 1)$ .
- 3. For  $0 \leq s < t < u < v \leq T$  the increments  $W(t) - W(s)$  and  $W(v) - W(u)$  are independent.

We compute the Brownian motion  $W(t)$  for each control step  $t_k$  meaning every 5 minutes. Given that it is discrete we need to compute a specific number of  $W(t)$  in the interval of 5 minutes which in our case is set to 10. This means that for every 5 minutes we compute the random variable as the vector  $W(t) = [W(t_1), W(t_2), \dots, W(t_{10})]$ . For the simulation of the Brownian motion we will use the code from the article *An Algorithmic Introduction to Numerical Simulation of Stochastic Differential Equations* [12].

### 2.3.2.2 Euler Maruyama

When expanding to the system of stochastic differential equations with the Brownian motion we get the following system of differential equations [12]:

$$d\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}, d, \mathbf{p}\mathbf{f})dt + \mathbf{g}(t)d\mathbf{W}(t). \quad (2.6)$$

Therefore, we are no longer able to solve the system using the Explicit Euler Method and instead we introduce the Euler Maruyama method that is applicable for stochastic systems of differential equations.

The idea behind the method stems from the Explicit Euler that also uses a fixed step size and an initial value guess at steady state. Again we wish to solve the system of equations for every 5 minutes with  $N = 10$  iterations. This implies once more a fixed step size of  $\Delta t = 0.5$ . Note that these steps are the same time steps we have computed the Brownian motion at so it matches. The method also uses the derivative from the MVP model, but then for every step we add the Brownian motion noise to the blood glucose concentration. After the 10 iterations the method has approximated the true value function which is done at every 5 minutes. This means that for the  $k$ 'th iteration the solution will be as follows [12]:

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, d_{k-1}, \mathbf{p}\mathbf{f}) \Delta t + \mathbf{g}(t) (W(t_k) - W(t_{k-1})), \quad k = 1, 2, \dots, N, \quad (2.7)$$

where  $\mathbf{g}(t) (W(t_k) - W(t_{k-1}))$  is the diffusion term and  $\mathbf{g}(t)$  is a vector consisting of zeros except for  $\mathbf{g}(6)$  that controls the intensity of noise meaning the greater it is the solution will be more noisy. This means that the noise is added only to the solution of the glucose concentration.

## 2.4 GRID algorithm

Finally, we introduce the GRID algorithm. This algorithm has been designed to detect meals hence it is called The Glucose Rate Increase Detector also denoted GRID. The main idea is to detect meals based on the slope of the blood glucose concentration for a patient[3]. Therefore, GRID will be used to detect continuous increase in the blood glucose concentration as a result of a meal [3]. This will be done by estimating the rate of change (ROC) of the blood glucose concentration. The algorithm can be divided in 3 main components: the CGM-filtering process, estimation of the ROC of glucose and lastly the detection part. These will be further described in the following.

### STEP 1.1) Noise-spike filter [3]

The first step is as mentioned to filter the CGM-data by using two types of filters.

The first filter is the noise-spike filter where the glucose measurement  $G_m(k)$  is filtered depending on which of the following three constraints apply to it as seen below:

$$G_{F,NS}(k) = \begin{cases} G_m(k) & \text{if } |G_m(k) - G_{F,NS}(k-1)| \leq \Delta G \\ G_{F,NS}(k-1) - \Delta G & \text{if } (G_{F,NS}(k-1) - G_m(k)) > \Delta G, \\ G_{F,NS}(k-1) + \Delta G & \text{if } (G_m(k) - G_{F,NS}(k-1)) > \Delta G \end{cases} \quad (2.8)$$

where  $G_m(k)$  is the blood glucose concentration for the sampling instant  $k$ ,  $G_{F,NS}(k)$  is the  $k$ th filtered blood glucose measurement from the noise-spike filter and  $G_{F,NS}(k-1)$  is the previous filtered blood glucose measurement. Additionally, there is  $\Delta G$  which is the maximum allowable ROC. This value is set to  $3 \frac{\text{mg}}{\text{dL}}$  for a 1-minute sampling, and given that our analysis will be based on 5-minute sampling it shall be mentioned that  $\Delta G$  is set to 15. What this filter does is therefore to add or subtract the maximum ROC depending on how big the change from the blood glucose measurement to the filter measurement is.

### STEP 1.2) Low pass filter [3]

The noise-spiked filtered glucose is further filtered through a low pass filter to lessen possible high frequency fluctuations. The noise-spike filter is given as below:

$$G_F(k) = \frac{\Delta t}{\tau_F + \Delta t} G_{F,NS}(k) + \left(1 - \frac{\Delta t}{\tau_F + \Delta t}\right) G_F(k-1), \quad (2.9)$$

where  $G_F(k)$  is the  $k$ th low pass filtered blood glucose measurement,  $G_F(k-1)$  is the previously low pass filtered blood glucose measurement and  $G_{F,NS}(k)$  is the noise-spike filtered measurement. Furthermore, the  $\Delta t$  is the sampling period from the current time point to the previous time point and  $\tau_F$  is a filter time constant which is used to smoothing the data.

### STEP 2) Estimation of ROC of glucose [3]

After the filter processes comes the estimation of ROC of glucose. The ROC of glucose is given as the derivative  $G'_F(k)$  calculated using the 3-point Lagrangian interpolation polyno-

mial evaluated at the most recent point [3]. The estimation is given below:

$$G'_F(k) \cong \frac{t(k) - t(k-1)}{(t(k-2) - t(k-1))(t(k-2) - t(k))} G_F(k-2) + \frac{t(k) - t(k-2)}{(t(k-1) - t(k-2))(t(k-1) - t(k))} G_F(k-1) + \frac{2t(k) - t(k-2) - t(k-1)}{(t(k) - t(k-1))(t(k) - t(k-2))} G_F(k) \quad (2.10)$$

We use the filtered measurements from the low pass filter at 3 different sampling times: the current time  $k$ , the previous  $k-1$  and  $k-2$ . This means for each current ROC of glucose, the 3 filtered blood glucose measurements are needed from 3 different sampling times.

#### STEP 3) Meal detection [3]

For the last part of detecting a meal, 3 new parameters  $G_{min,1}$ ,  $G'_{min,2}$  and  $G'_{min,3}$  are introduced which set the constraints for whether a meal is to be detected or not.  $G_{min,1}$  is the minimum glucose measurement for when a meal should be detected.  $G'_{min,2}$  and  $G'_{min,3}$  are selected as the minimum ROC we allow for respectively the current and previous ROC and the two previous and current ROC. The combination of these parameters are the ones we will try to tune. We will refer to these as the  $G_{min}$  values.

The detection is binary and is given as either 1 if a meal is detected or 0 if not. The equation is given as below:

$$GRID^+ = \begin{cases} 1 & \text{if } G_F(k) > G_{min,1} \wedge ((G'_F(k-2:k) > G'_{min,3}) \vee (G'_F(k-1:k) > G'_{min,2})) \\ 0 & \text{otherwise} \end{cases} \quad (2.11)$$

For a meal to be detected it is therefore a prerequisite for the filtered blood glucose measurement to be larger than the measurement of  $G_{min,1}$  and either the previous and current ROC to be larger than  $G'_{min,2}$  or the current and the two previous ROC to be larger than  $G'_{min,3}$ .

### 2.4.1 The $\tau_F$ value in the GRID algorithm

As mentioned one of the parameters in the GRID algorithm is the value of  $\tau_F$ . This value is the filter time constant and is a part of the low pass filter in step 1.2 of the algorithm. The constant  $\tau_F$  is the value that smooths the curve of the blood glucose concentration and will therefore have an effect on the process noise of the data. This means the process noise will fluctuate less when the  $\tau_F$  value is higher.

Tuning the  $\tau_F$  value could be done for a more optimal meal detection by the GRID algorithm, but later when optimizing the GRID algorithm by tuning the  $G_{min}$  values we have used the conclusions and the values from the article *Design of the Glucose Rate Increase Detector: A Meal Detection Module for the Health Monitoring System* [3] to determine the ranges for the  $G_{min}$  values. Therefore, we have chosen not to tune on this parameter and instead fix the  $\tau_F$  value to  $\tau_F = 6$  which is the optimal value found in the report.

### 2.4.2 Implementation of the GRID algorithm as a function

The GRID algorithm will be implemented in Matlab as a function which outputs a vector (zero\_one) of ones or zeros depending on a meal has been detected or not. Furthermore,

it outputs two vectors to update the previous measurements for equation 2.9, 2.8 and 2.10 such that they will be the previous measurements for the next control step. The first vector is a two dimensional vector consisting of  $G_{F,NS}(k-1)$  and  $G_F(k-1)$  used in equation 2.8 and 2.9 (filt\_prev). The other vector is a two dimensional vector consisting of the updated previous measurements of  $G_F(k-2)$  and  $G_F(k-1)$  used in equation 2.10 (Gfm\_vec). The last output is a flag counting how long time ago a meal has been detected (flag).

The GRID algorithm function takes in 9 inputs: the maximum allowable ROC (delta\_G), a three dimensional vector consisting of the two previous blood glucose measurements and the current glucose measurement (G\_vec), the  $\tau_F$  measurement (tau), the time interval between the current control step and the previous (tspan), the vector of previous filtered measurements that is also an output as already explained (filt\_prev) just as the input with the previous derivatives (Gfm\_vec), the  $G_{min}$  values in a vector in which we are meaning to tune (G\_min), a three dimensional vector consisting of the two previous time point and the current time points (t\_vec) and lastly a flag counting how long time ago a meal has been detected that is as mentioned also an output.

The GRID algorithm function depends on three other functions we implement called spikefilt\_func, lowfilt\_func and estimage lagrange. These respectively represents equation 2.8, 2.9 and 2.10. The GRID algorithm starts by calling the three functions such that it follows the 3 steps described earlier; STEP 1.1, STEP 1.2 and STEP 2. Afterwards, it computes STEP 3 of the algorithm; the meal detection part and outputs the zero\_one vector. Lastly, since we do control steps of 5 minutes we need to ensure that the same meal is not being detected more than once. The reason why this could happen is that only after 5 minutes the ROC change and the blood glucose measurement could still meet the criteria from 2.11 meaning that the same meal could end up being detected twice. We ensure that this does not happen by setting a counter given by the flag input/output so it updates continuously. If a meal has been detected the flag is set equal to  $\frac{120}{tspan}$  meaning that in two hours the function should not detect a meal again. This means that when the flag is input in the function in the next control step, a meal will not be detected again, instead, the flag will be subtracted by 1. Continuing this way we ensure that the function will be counting down in 2 hours after a meal has been detected instead of detected the same meal again. The implementation of this function can be seen in appendix A.

## CHAPTER 3

# Simulation and evaluation

---

### 3.1 Description of the clinical data

We have been provided with clinical data of real patients by Steno Diabetes Center Copenhagen. The clinical data we have chosen to analyze is a patient who is screened from 2nd of January 2021 till 1st of June 2021 which sums up to approximately 5 months of measurements. The data set consists of different measurements, however, we only regard the relevant measurements being the glucose measurements given by a CGM, the basal insulin levels, the patient's own announcement of bolus insulin and the ingested amount of carbs. Despite 5 months of measurements, we have chosen to only regard and analyze a time period of 3 days for practicality, visualisation and a more in depth analysis. The chosen time frame for analyzing the data is from the first initial measurement taken on 2nd of January at 00:02 am until 4th of January 23:59.

### 3.2 Simulation scenarios

The foundation of this report is as already mentioned the simulations of the virtual patients. It is based on these simulations that we will be able to analyse, tune on the  $G_{min}$  values and draw conclusion on the best combinations of the  $G_{min}$  values for the GRID algorithm. Therefore, we strive for the simulated data to look as much as possible as the clinical data, hence, we will simulate several scenarios. First we begin with one patient based on the open loop system over a month, then we expand for the closed loop system and lastly we simulate 100 patients over a month.

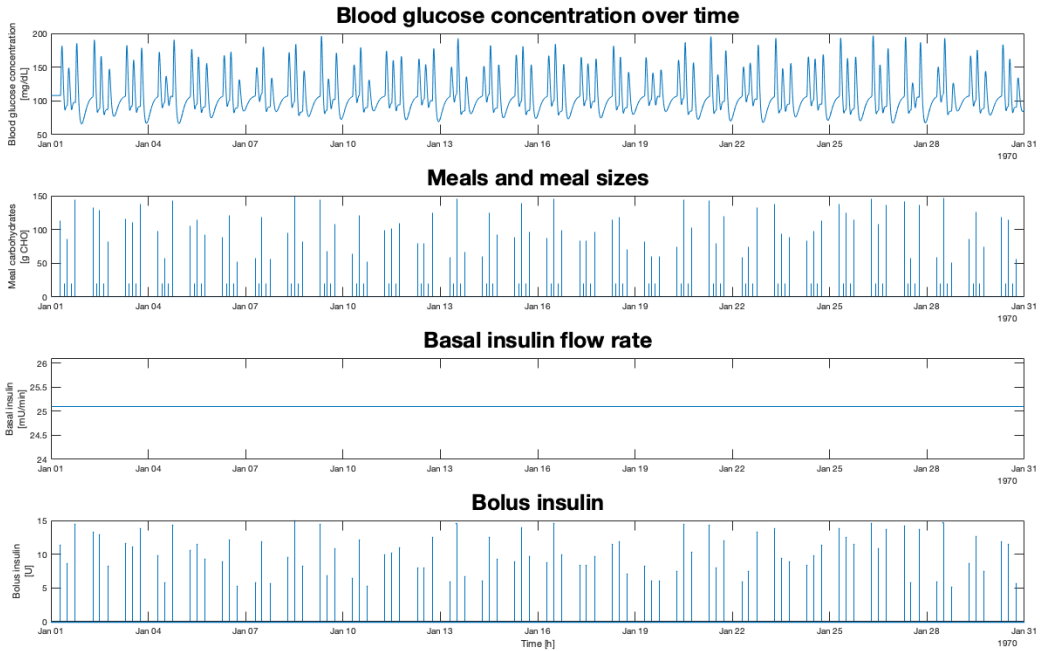
#### 3.2.1 1 patient: open loop

We start by simulating a response of 3 meals and 2 snack meals a day over 30 days for one patient meaning 90 meals and 60 snacks in total. We evaluate our patient at every 5 minutes which means that for 30 days we have 8,640 control steps. The three meals will be ingested at 7, 12 and 18 hours and the size will differ in the range of [50,150] g CHO. This is computed by using the 'randi' function in Matlab. The snack meals will have size 20 g CHO and will be ingested at 10 and 15 hours. For each meal the bolus insulin will be computed using the insulin to carbohydrate ratio denoted ICR [13]. This means that for 10 gram of carbohydrate we give 1 units of bolus insulin. For the snack meals we will not give bolus insulin - why is further elaborated in Section 3.2. This procedure will be the same for every simulation scenario.

The open loop simulation function has 8 inputs: an initial value guess from the steady state ( $\mathbf{x}_0$ ), the vector of control steps (tspan), the beforehand computed bolus insulin dose as an array and the basal insulin at steady state as an array which together constitute a matrix ( $\mathbf{U}$ ), the disturbance vector of different meal sizes at the given time points ( $\mathbf{D}$ ), the vector of individual parameters ( $\mathbf{p}$ ), the MVP model (simModel), the Explicit Euler function (simMethod) and the respective step size (NK). In this first scenario we use the Explicit Euler methods because we start out with a simulation based on the deterministic differential equations.

The implementation of the open loop function can be seen in appendix A.

The open loop simulation function solves the MVP model for every control step using the Explicit Euler method that uses the initial guess to get the first solution with the fixed step size  $h = 0.5$  as earlier explained. The function then stores all the solutions in a matrix  $\mathbf{x}$  and stores the time points in a vector  $\mathbf{t}$  which dimensions respectively will be  $7 \times 8640$  and  $1 \times 8640$ . From these solutions we extract the blood glucose concentration values using a CGM sensor function. The CGM function is really simple given that it only extracts the blood glucose concentration,  $\mathbf{x}(6)$ , from the state vector  $\mathbf{x}$ . Hereby, we can visualize the glucose concentrations, the meal sizes and the respective basal insulin rates and bolus insulin.

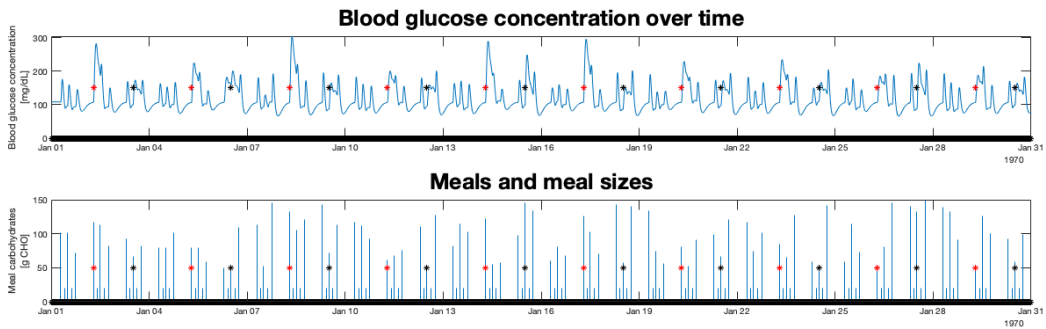


**Figure 3.1:** 150 meals/snacks response over 30 days on 1 patient computed by an open loop simulation

On Figure 3.1 the explained open loop simulation is being visualized. We see how the blood glucose concentration increases differently at every ingested meal depending on the size of the respective meals and further how the bolus insulin aids in dampen the glucose

fluctuations in getting too high. Furthermore, the basal insulin in the open loop is constant based on the steady state at  $25 \frac{\text{mU}}{\text{min}}$ .

However, since we seek to assist the doctors with detecting meals missing bolus insulin, we expect for the real patients and the clinical data to be not have given bolus insulin for some meals or not have given the right\*\* amount of bolus insulin for some meals. Therefore, we expand the simulation such that for some meals no bolus insulin has been given and for some meals only 20% of the bolus insulin have been given. We choose the meals missing bolus insulin to be at every 3th day at 7 hours starting on day 2 and the meals with incorrect amount of bolus insulin to be at every 3th day at 12 hours starting on day 3. This simulation is visualized below.



**Figure 3.2:** 150 meals/snacks response over 30 days on 1 patient with incorrect bolus for selected meals computed by an open loop system

On Figure 3.2 we see the red dots representing missed bolus insulin and the black dots representing the meals with the incorrect amount of bolus insulin. In the top subplot we see how the blood glucose concentration clearly fluctuates more when the bolus insulin has not been given correctly. So, looking at the red dots in the subplot, the blood glucose concentration always tend to fluctuate more.

The two simulations until now gives a good idea of what is going on and how the bolus insulin affects the blood glucose concentration at the respective given meal sizes.

### 3.2.2 1 patient: closed loop

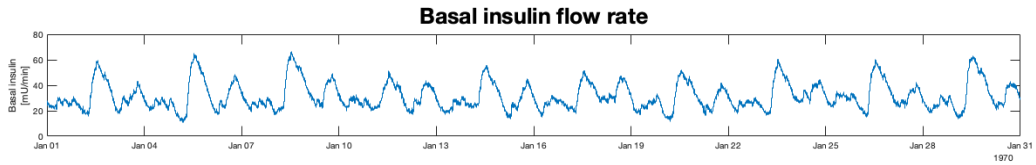
Next, we will expand to a closed loop simulation since the clinical data has been controlled by a closed loop system. At the closed loop system the basal insulin will no longer be constant but will instead be regularized continuously as mentioned earlier on. The simulation will proceed as at the open loop simulation with 3 meals and 2 snack meals for each day over 30 days and the bolus insulin will be incorrect at the same meals as before.

The closed loop simulation function has 12 inputs: an initial value guess from the steady state ( $x_0$ ), the vector of control steps (tspan), the disturbance vector of different meal sizes at the given time points (D), the vector of individual parameters (p), the bolus and basal insulin matrix (U), the MVP model (simModel), the Explicit Euler function (simMethod), the PID controller (ctrlAlgorithm), the CGM sensor function (observationMethod), the control parameters for the PID control function (ctrlPar), the control state for the initial integral term and the initial blood glucose measurement at  $108 \frac{\text{mg}}{\text{dL}}$  (ctrlState0)

and lastly the respective step size (NK).

The closed loop simulation functions solves the MVP model for every control step using the Explicit Euler method that uses the initial guess. However, before every control step the function uses the PID control function to determine the right amount of basal insulin to use when solving at the current control step. This will then automatically affect the next solution of the differential equation system given that the blood glucose concentration is being regularized by the bolus insulin as well. This means, that the PID controller only regularizes the basal insulin but bolus insulin is still determined for each meal beforehand. After every control step the function stores the solution in a matrix  $\mathbf{x}$  and stores the time points in a vector  $\mathbf{t}$  which dimensions respectively will be  $7 \times 8640$  and  $1 \times 8640$ . Then the function extracts the blood glucose concentration using the CGM sensor function. The implementation of the closed loop function can be seen in appendix A.

The main difference from the open loop is as mentioned that basal insulin is being regularized continuously which is visualized below.



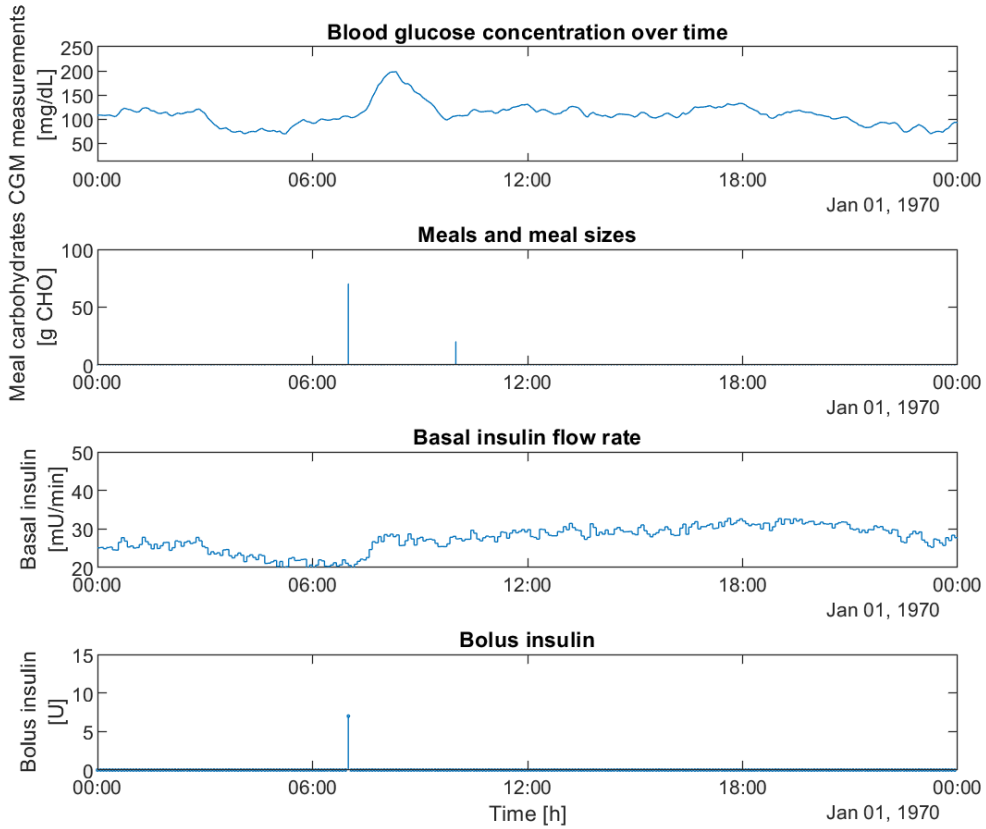
**Figure 3.3:** 150 meals/snacks response over 30 days on 1 patient computed by a closed loop system

On Figure 3.3 the basal insulin flow rate is being visualized. Instead of being constant it is clearly changing continuously depending on the meals sizes, however the bolus insulin is still necessary because the basal insulin works slower. The meals and bolus insulin is given at the same control steps as in the open loop simulation. This figure gives a good understanding of the affect of the closed loop system and how it improves the artificial pancreas since it is now possible to automatically control the basal insulin.

### 3.2.3 The ingested snack meals in the simulation

When ingesting a snack meal we can see on our simulations that it has little influence on the blood glucose concentration. We recall that the ingested snacks in our simulation is of size 20 g CHO. This means that the ROC in the blood glucose concentration is very small. We have visualized this in the figure below:





**Figure 3.4:** Closed loop simulation of one meal and one snack meal

On Figure 3.4 we see that the blood glucose concentration does not increase drastically after a snack meal has been ingested. When using the closed loop simulation the PID-controller changes the basal insulin for every 5 minutes based on the blood glucose concentration. So, for snack meals the basal insulin will therefore even out the increase hence we will not give any bolus insulin.

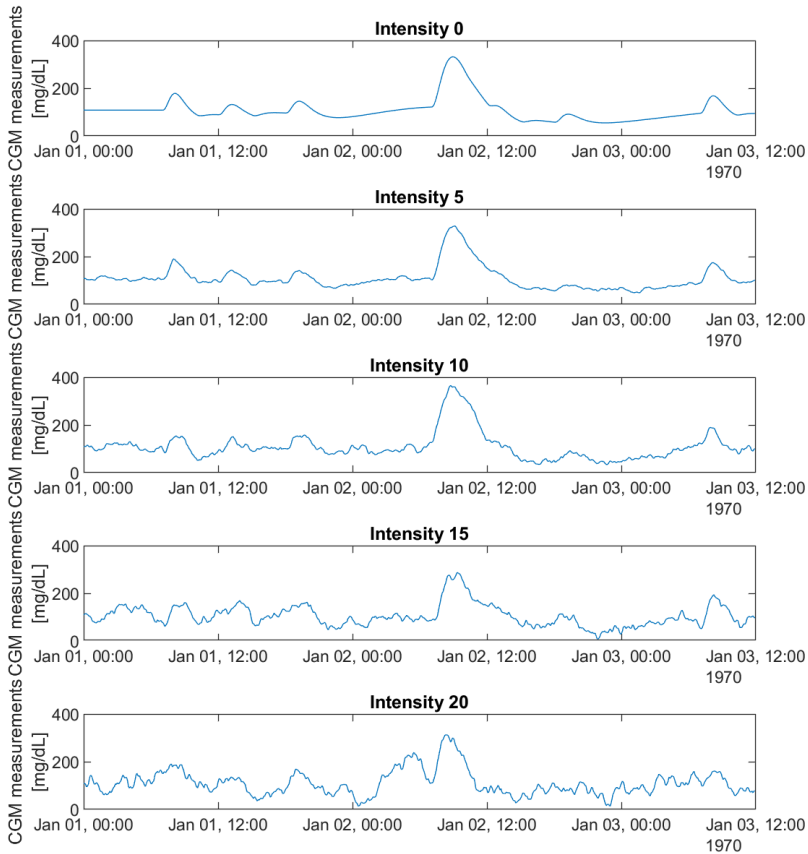
Another reason for the little influence of snacks might be due to the *GEZI* parameter given in the **pf** vector in the MVP model which corresponds to the body's natural uptake of glucose. The body could be using more glucose than the carbohydrates ingested such that the concentration curve decreases to the steady state at  $108 \frac{\text{mg}}{\text{dL}}$  until more and larger carbohydrates are ingested again. Therefore despite the fact, that the blood glucose concentration might be above the  $G_{\min,1}$  value, the ROC of glucose for a snack meal do not satisfy the constraint given in the GRID algorithm and again the snacks are not detected.

All this will result in the GRID algorithm not detecting snacks as meals and therefore we have not taken the snack meals into consideration when evaluation the GRID function. A question to be asked is then: why do our simulations include snack meals? This is done to match the clinical data as much as possible. It would not be realistic to simulate a patient only eating 3 large meals a day.

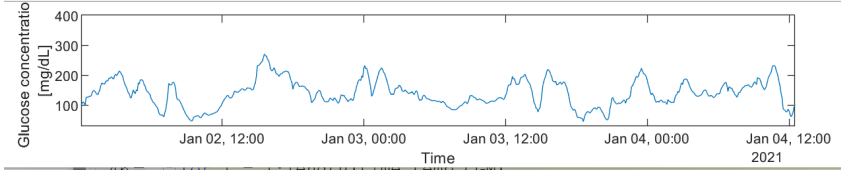
### 3.2.4 Closed loop: noise intensity parameter

As this report is trying to approximate real patients as much as possible it is as already mentioned necessary to construct simulation scenarios that considers measurement noise and process noise by making the differential equations stochastic. Then, instead of solving the system using Explicit Euler method use the Euler Maruyama method.

The intensity of process noise is controlled by the  $\mathbf{g}(t)$  vector that is multiplied to the Brownian motion and added to the system of differential equations. We have determined the most optimal value of intensity by comparing different curves of the blood glucose concentration with different intensity values to the glucose concentration of the clinical data. The comparison will be done with a simulation over 3 days such that it matches approximately 3 days in the clinical data. We have examined 5 different intensity values in the set  $\mathbf{g} \in \{0, 5, 10, 15, 20\}$ . This is visualized in Figure 3.5.



**Figure 3.5:** Blood glucose concentration with different intensity values over 3 days



**Figure 3.6:** Blood glucose concentration of the clinical data over approximately 3 days

The clinical data is visualized in Figure 3.6 over approximately the same period of time to compare which of the intensities give the most similar process noise compared to the clinical data. From the plots it is clear to see that both intensity value 0 and 5 is too low. With these intensities there is not enough process noise on the data because the curves look too smooth compared to the clinical data. With intensity 20 there is too much process noise. The blood glucose concentration decrease almost down to 0, which would be unrealistic for a real patient. Furthermore, the curve oscillates too much. In conclusion, we disregard intensity 0, 5 and 20. Looking at the curves with intensity 10 and 15 they look very similar but again we see huge decreasing with intensity 15 meaning the blood glucose concentration is at some measurements lower than  $10 \frac{\text{mg}}{\text{dL}}$ . This is way too low from the clinical data and the reason why we have chosen intensity 10 to proceed with in this report.

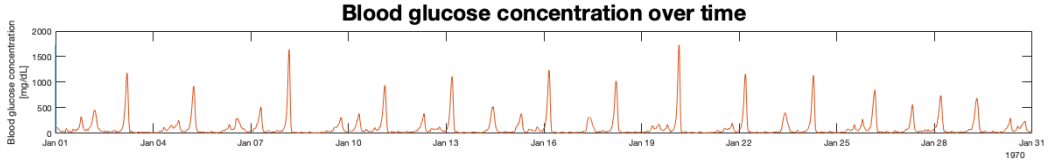
So with these in place, we continue our simulations based on the closed loop simulations with both process noise and the measurement noise described in Chapter 2.

### 3.2.5 100 patients simulation - closed loop

So far, we have tried to approximate the clinical data by simulating only one patient, however, to draw further conclusions later, we need to simulate several patients to be able to optimize the GRID algorithm such that it fits more patients. People e.g. react very differently on insulin and meals sizes depending on the **pf** vector, so therefore, by simulating several patients, we hopefully fit more patients overall when optimizing the GRID algorithm.

To begin with, we simulate 100 patients. For this purpose we need to simulate 100 individual vectors of parameter values **pf** which is done by implementing a function we denote the `pmatrx` function. This is based on the article *Identification of Intraday Metabolic Profiles during Closed-Loop Glucose Control in Individuals with Type 1 Diabetes* [14] which has given the parameter values for 10 patients. From these patients we use the maximum and the minimum value from each parameter value and use the `randn` in Matlab to normally distribute 100 values in between the maximum and minimum value. We notice that the units of the  $S_I$  in the article is  $\frac{\text{ml}}{\mu\text{U}}$  but in our computations the unit is  $\frac{\text{ml}/\mu\text{U}}{\text{min}}$ . By testing, we conclude that the article must have meant to write  $\frac{\text{ml}/\mu\text{U}}{\text{min}}$  so therefore we continue without recalculating the units for this parameter.

We are now able to simulate the 100 patients using the `pmatrx` function. However, after the simulation we see that some patients tend to get very high blood glucose concentrations as seen in Figure 3.7 below.



**Figure 3.7:** Patients with unrealistic blood glucose concentraions

On Figure 3.7 we see a patient which blood glucose concentration gets unrealistic high up to more than  $1,500 \frac{\text{mg}}{\text{dL}}$ . This is where the parameters from the `pmatrix` function might have an influence on how the individual patient e.g. takes up the insulin and reacts on different meal sizes. Since these patients do not impact realistically to the final results we decide to add a restriction on what blood glucose concentrations we tolerate. We allow for the minimum value to be  $20 \frac{\text{mg}}{\text{dL}}$  and the maximum value to be  $550 \frac{\text{mg}}{\text{dL}}$ . This results in disregarding 43 patients, meaning we only use the simulations from 57 patients.

The fact that we out of 100 patients disregard 43 of them could mean that the implementation of the `pmatrix` function should have been reconsidered. Instead of using the maximum and the minimum value from the article we could have used the mean of the 10 patients parameter values and then e.g. accepted a standard deviation of 1 and then computing the 100 parameter vectors in between that range.

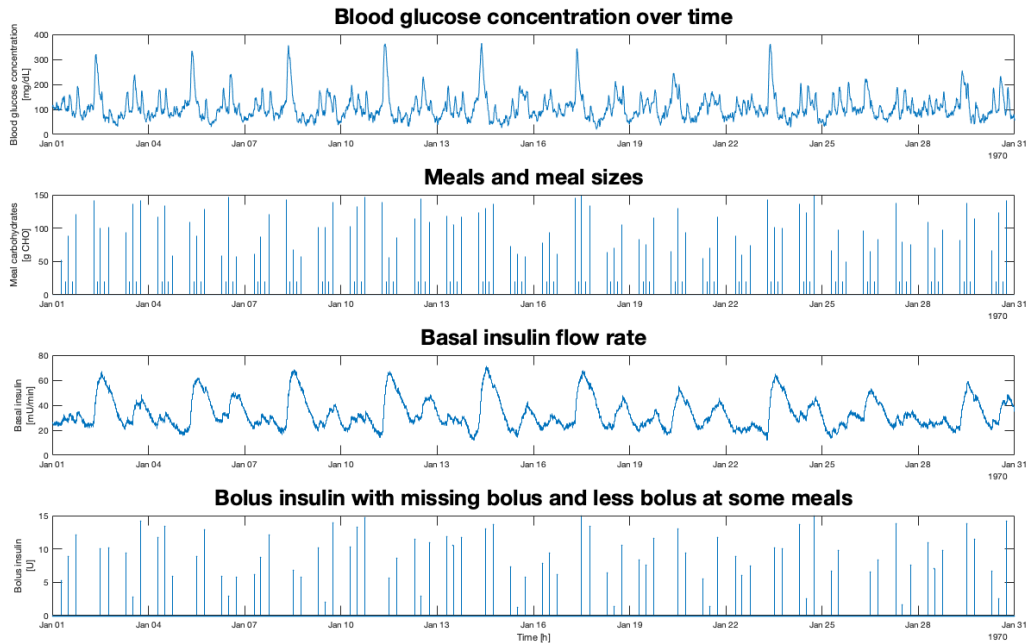
More over, the number of patients are also not that high, therefore, by simulating e.g. 1,000 patients we could still have ended up with a lot of simulated patients after disregarded the ones not meeting the restrictions.

Furthermore, when we use `randn` in Matlab we do not consider the combination of the parameter values, we just computed them randomly independently. So maybe, by tuning the parameters dependently on each other, we could make sure that we did not simulate unrealistic patients.

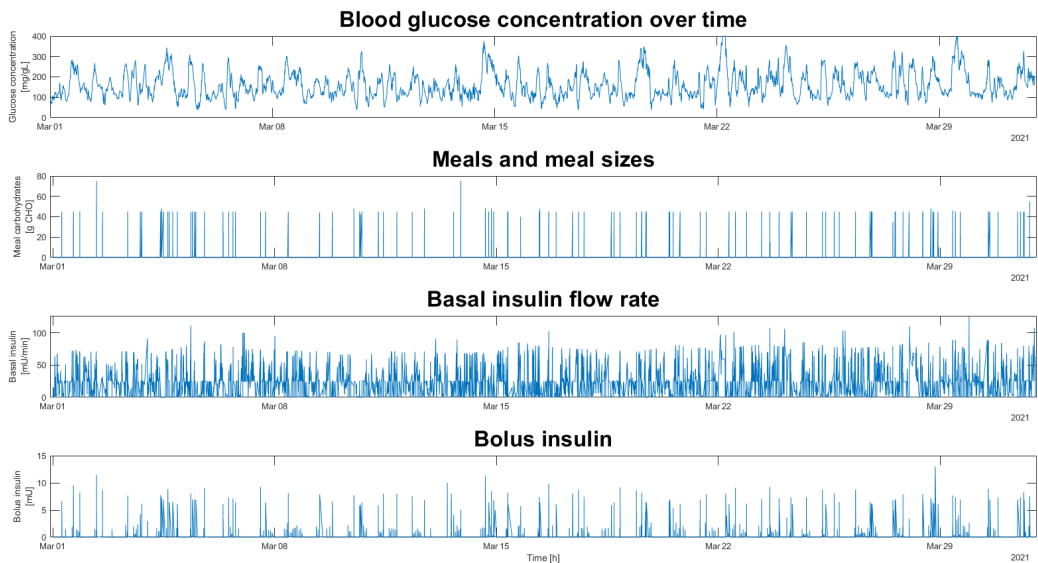
However, we decide to continue our simulation with the 57 patients so these considerations are only important to have in mind when concluding on the optimal GRID algorithm later.

### 3.2.6 Final simulation compared to clinical data

We have now developed a simulation to approximate the clinical data as much as possible by using a closed loop system, changing the bolus insulin to be incorrect for some meals and choosing the intensity value of  $\mathbf{g}(t) = [0, 0, 0, 0, 0, 10, 0]$ . The final simulation for one patient will now be compared to the clinical data and is visualized in the figures below.



**Figure 3.8:** Final simulation using closed loop and intensity 10 over a month



**Figure 3.9:** Clinical data visualized over a month

The intention with both figures is to get a brief overview of how similar the simulated data and the clinical data are. Therefore, both are visualized over approximately one month.

The first thing we notice is that the patient in the clinical data must have used an other feedback system to regularize the basal insulin. The curve oscillates much more than on the

simulated data which may indicate that the control systems are not identical.

Looking at the blood glucose concentration, we see that the simulated patient's concentration is more steady in the range  $[50-200] \frac{\text{mg}}{\text{dL}}$  unless the bolus insulin is missing and the blood glucose concentrations increases to almost  $400 \frac{\text{mg}}{\text{dL}}$ . The blood glucose of the real patient is in general much higher, this could indicate that the patient more often skips or gives the incorrect amount of bolus insulin.

Lastly, the meal sizes for the clinical patients are general a bit smaller than the simulated patient. However, the patient seems to eat within smaller time intervals, which could indicate that the patient eats smaller meals and therefore more meals or that the patient maybe forgot to notify everything at the same time so notifies two meals right after each other. Furthermore, given that the patient eats more times with smaller intervals than our simulated patient can also be why the blood glucose concentration in general is a bit higher overall. Nevertheless, realistic meal sizes should be greater than  $50 \frac{\text{mg}}{\text{dL}}$  hence we keep to the meal sizes we have simulated until now.

We have tried to approximate the real data as much as possible but yet there could be some improvements on the simulation. The simulation are not completely similar since several differences can be seen. Therefore, we could have given smaller meal sizes with smaller time intervals or we could have used another type of feedback system. However, this is a good starting point and we will keep to the simulation done on Figure 3.8 but then expanding as mentioned to 100 patients.

We are now ready to use the GRID algorithm on the simulated data and the clinical data. However, we need to decide how to evaluate the true positives and false positives given by the GRID algorithm. This will be elaborated in Section 3.3.

### 3.3 Evaluation of the GRID algorithm

The GRID-algorithm is the main component in the task of meal detection. The main goal is to be able to detect all meals over a time period for a patient to assist doctors in their very brief consultations. Therefore, we want to be able to evaluate the performance of the GRID algorithm. As mentioned, the parameters that will be tuned are the  $G_{min}$  values. For this purpose, a function has been made that computes the number of true positives and false positives. We notice that a true positive is when the GRID algorithm detects a meal and a meal has been ingested, whereas a false positive is when the algorithm detects a meal but no meal has been ingested.

#### 3.3.1 Description of the evaluation function

We implement a function with the purpose of computing the true positives and false positives. Before the simulation we create a vector **D** that is a vector with zeros unless a meal has been given so the entry is the randomly selected ingestion rate of the meal. By the GRID algorithm function, we achieve a binary vector called **D\_detected** with ones for detected meals and zeros for undetected meals. These two vectors are input in the evaluation function and it is based on these we will be able to find the true positives (TP) and false positives (FP).

We start by changing the data type of the **D** vector for binary. Here we make sure to not consider the snacks meals, meaning the meals less than 50 g CHO. This means that at every entry in the vector **D** larger than 50 g CHO we change to 1 and else the entry

will be changed to 0. Based on these two vectors it is clear that some meals are not being detected up to 1.5 hours after the true meal has been ingested. With this in mind, we have to consider a delay of up to 90 minutes when finding the FP and TP which means that a stride has to be input in the function as well. We accepted this delay since the counter in the GRID function assures that a new meal will not be detected within two hours. Therefore, the stride will not affect the amount of false positives and true positives to be found with the function.

After the function has changed the data type for binary, the function finds the indices for the ones in the **D** vector and the **D\_detected** vector. Then by using for-loops the function are able to loop over all the indices in each vector that have ones. The true positives are counted if there is a 1 in the vector **D** and at the same time a 1 in the vector **D\_detected** within the range of the stride after eating. The false positives are counted if there is a 1 in the vector **D\_detected** but no 1 in the vector **D** within the range of the stride before the 1 in the **D\_detected** vector.

This function will make it possible for us to try to find the most optimal  $G_{min}$  values which will be elaborated in the following section.

A further look at the implementation of this function can be seen in appendix A.

### 3.3.2 How to evaluate the $G_{min}$ values

After having run the evaluation function we achieve a number of true positive and false positive meals from the GRID algorithm with some respective  $G_{min}$  values. Given that the goal is to get as many true positive meals as possible and as few false positive meals as possible, we need to find a way to evaluate how many false positives and how few true positives we allow for the optimal GRID algorithm to have.

In total we want 90 meals over 30 days for one patient to be detected. In addition, we compute the percentage of true positives out of the 90 true meals by  $\frac{TP}{90}$ . Furthermore, we want to decide how many false positive we accept per day, so we compute  $\frac{FP}{30}$ , since we simulate over 30 days. Based on these two calculations we can set a threshold for each calculation at respectively  $\frac{FP}{30} < t_{FP}$  and  $\frac{TP}{90} \geq t_{TP}$ . This means that we maximum allow for the GRID algorithm to get  $t_{FP}$  false positive per day and minimum allow for the GRID algorithm to detect  $t_{TP}$  out of the 90 meals. In conclusion, we evaluate the performance of the GRID algorithm with the respective  $G_{min}$  values that is to be tuned based on how many true positives and false negatives the GRID algorithm gives.





# CHAPTER 4

## Results

---

### 4.1 The optimal $G_{min}$ values

Earlier we specified how we would evaluate the GRID algorithm by computing the false positives and the true positives. This leads to the optimization of the GRID in this report; tuning the  $G_{min}$  values. We have chosen to tune the  $G_{min}$  values by combining different values for each parameter in the ranges:

$$G_{min,1} \in \{120, 125, 130, 135, 140\}$$

$$G'_{min,2} \in \{0.6, 1.1, 1.6, 2.1, 2.6\}$$

$$G'_{min,3} \in \{0.5, 1.0, 1.5, 2.0, 2.5\}.$$

The ranges are chosen based on the optimal values given from the article *Design of the Glucose Rate Increase Detector: A Meal Detection Module for the Health Monitoring System* [3], where they were found to be  $G_{min,1} = 130$ ,  $G'_{min,2} = 1.6$  and  $G'_{min,3} = 1.5$ . Hereby, we choose the ranges to be around these values with the same margin size on each side of the values with a step size of 5 for  $G_{min,1}$  and 0.5 for both  $G'_{min,2}$  and  $G'_{min,3}$ . Combing all the  $G_{min}$  values gives 125 combinations.

By computing all the combinations of these values, we can run the simulation for 100 patients and evaluate the detection by the GRID algorithm with each  $G_{min}$  value combination using the evaluation function elaborated earlier. However, this means that we have computed the true positives and false positives for each patient and each combination but to be able to draw conclusions we take the mean over all patients for each  $G_{min}$  combination. Therefore, we get one mean number of true positives and one mean number of false positives for each combinations. We recall that the true positives are the meals where a meal has been detected and also ingested, and a false positive is a meal that has been detected but not ingested.

Next, we evaluate every combination of the  $G_{min}$  values with the GRID algorithm and for this we set the thresholds  $t_{TP} = 0.7$  and  $t_{FP} = 1$ . This means that we accept for maximum 1 false positive to be detected per day and at least 70% of the 90 true meals to be detected.

By detecting the meals of the 100 simulated patients using the GRID algorithm with the 125 combinations of  $G_{min}$ , we end up with 18  $G_{min}$  combinations that all meet our criteria of the two thresholds. These are shown in Table 4.1.

$G_{min}$ combination	Percentage true positive	False positive pr day
[120, 0.6, 0.5]	0.76062	0.89415
[120, 1.1, 0.5]	0.75263	0.83977
[120, 1.6, 0.5]	0.75283	0.83918
[120, 2.1, 0.5]	0.75283	0.83918
[120, 2.6, 0.5]	0.75283	0.83918
[125, 0.6, 0.5]	0.73528	0.72632
[120, 1.1, 0.5]	0.72807	0.68187
[125, 1.6, 0.5]	0.72827	0.68129
[125, 2.1, 0.5]	0.72827	0.68129
[125, 2.6, 0.5]	0.72827	0.68129
[120, 0.6, 1.0]	0.7501	0.77135
[120, 0.6, 1.0]	0.72242	0.62573
[120, 0.6, 1.5]	0.7501	0.77135
[125, 0.6, 1.5]	0.72242	0.62573
[120, 0.6, 2.0]	0.7501	0.77135
[125, 0.6, 2.0]	0.72242	0.62573
[120, 0.6, 2.5]	0.7501	0.77135
[125, 0.6, 2.5]	0.72242	0.62573

**Table 4.1:** Table with the results from the evaluation of combinations of  $G_{min}$  values

Because all the  $G_{min}$  combinations above have less than one false positive pr day, we will focus on the highest possible percentage of true positives detected. From the result in Table 4.1 we see that one combination of the  $G_{min}$  values has a percentage of true positive at 0.76062 meaning that 76% of true meals ingested are detected by the GRID algorithm with this  $G_{min}$  combination. This is the highest percentage which means the conclusion of the optimal  $G_{min}$  values is the ones where  $G_{min,1} = 120$ ,  $G'_{min,2} = 0.6$  and  $G'_{min,3} = 0.5$ . These are the values we will use in Section 4.2 to detect meals with the GRID algorithm on the clinical data.

#### 4.1.1 Discussion of the optimal $G_{min}$ values

It is important to take into consideration that our results are influenced by choices made earlier in the report.

First, the parameter  $\tau_F$  in the GRID algorithm is not being tuned as mentioned in Section 2.4.1. The  $\tau_F$  value controls the smoothing of the glucose concentration curves as already mentioned. If the fluctuations that might be from the process noise is not smoothed, meaning that the  $\tau_F$  value is small, then there can be steep slopes meaning high ROC of glucose from one measurement to another even though no meal has been ingested. Therefore, the lower the  $\tau_F$  value is the more fluctuation the blood glucose concentration will have which could lead to more false positive meals. Also vice versa, the higher  $\tau_F$  is the fewer meals would be detected, so with a very smooth blood glucose concentration there will be detected less meals overall which could influence the number of true meals detected which we are not interested in as well. Therefore, to improve the GRID algorithm further we could have tuned on the  $\tau_F$  value as well since it might have a high impact on the result.

Furthermore, we decided to determine the ranges of the  $G_{min}$  values based on the report *Design of the Glucose Rate Increase Detector: A Meal Detection Module for the Health*

*Monitoring System* [3], such that we could fix  $\tau_F$ , but if the  $\tau_F$  value was also to be tuned, we could have chosen some very different ranges for all the parameters.

In addition, the ranges we have used could have been larger. The minimum values for  $G'_{min,2}$  and  $G'_{min,3}$  are not necessarily that low and since these influence on the slope of the ROC we allow, it might have lead to higher percentage of true positives if the values were even lower. On the other hand, if the values were too low, we would also accept the slopes of the blood glucose concentration curves to be smaller which could lead to more false positives as well.

Another choice we made was to simulate only 100 patients. As already mentioned we were forced to only use 57 patients of them since the simulation ended up with 43 patients being too unrealistic. Therefore, the GRID algorithm with the  $G_{min}$  combination found may not fit as many randomly chosen patients as intended given that the simulation is only based on 57 patients.

Lastly, our evaluation is based on how many false positives we tolerate per day and how few true positives we tolerate out of the 90 true ingested meals. We could also have chosen to look at this differently e.g. taken the percentage of false positives out of the number of the detected meals in total. However, we thought it was easier to understand and draw conclusion when evaluating in this way.

Furthermore, the threshold we have chosen for the number of allowed false positives per day,  $t_{FP}$ , might be limiting the number of true positives we achieve. Therefore, if we would have made a further examination of optimizing the GRID algorithm we could also have tuned the threshold values. This could have lead to a deeper examination where we could have used more statics and e.g. error rates of different threshold values.

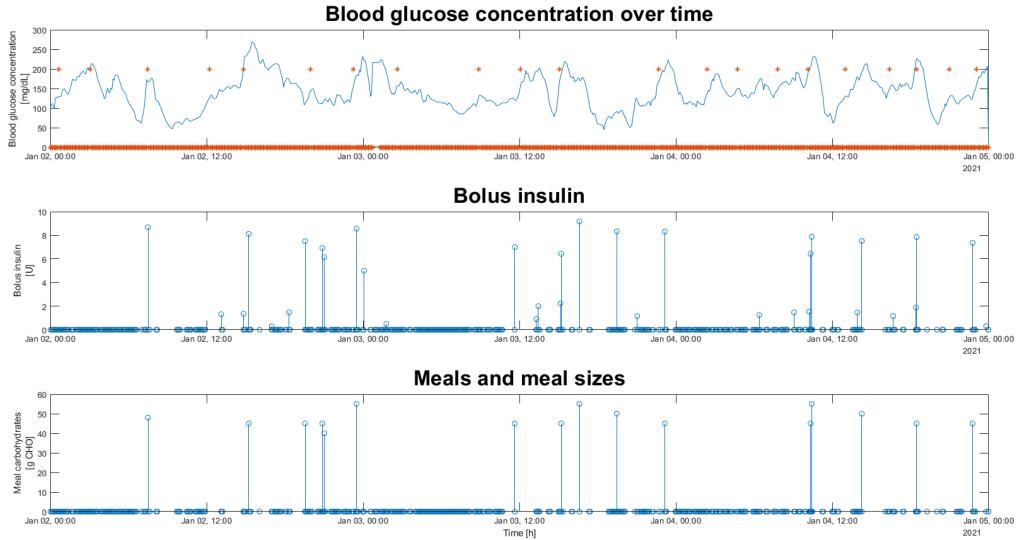
## 4.2 Detection of meals on clinical data

The several simulations and considerations of tuning the GRID algorithm finally allows for testing it on the clinical data. With the found optimal  $G_{min}$  values to be  $G_{min,1} = 120$ ,  $G'_{min,2} = 0.6$  and  $G'_{min,3} = 0.5$ , we proceed with the GRID algorithm with these values. We have chosen to test the GRID algorithm on the whole data set, however only visualizing and analyzing a time period of 3 days.

Testing the GRID algorithm on the whole data set results in a total meal detection of 764 meals throughout the approximately 5 months. Interestingly, the patient has only announced 548 meals which already might indicate that the GRID algorithm is detecting some possibly unannounced meals.

Furthermore, there has been given 1,332 times of bolus insulin which could be due to forgetting to announce a meal or perhaps not giving the optimal amount of bolus insulin during the initial meal leading to a second dose of bolus insulin.

However for further assessment we continue by considering the earlier described 3 days time period. In this time period the patient has announced 16 meals but the total amount of detected meals in this time frame is 21. These are indicated with red dots shown in Figure 4.1 with the associated blood glucose concentration curve, the bolus insulin and the announced meals.

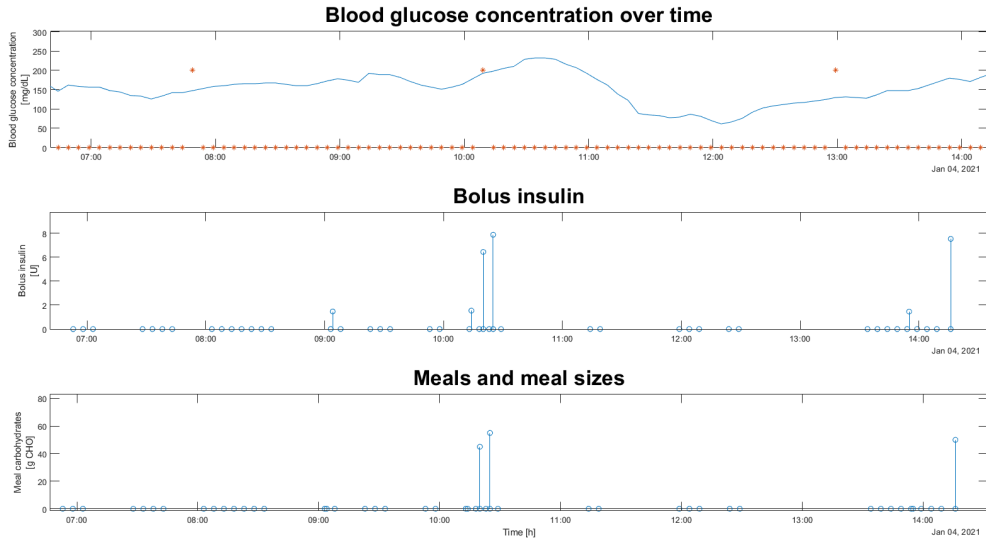


**Figure 4.1:** Test of GRID algorithm on clinical patient over 3 days with associated blood glucose concentration curve, meals, announced bolus insulin and detected meals by GRID algorithm with optimal  $G_{min}$  values

It is seen on Figure 4.1 that the GRID algorithm seems to correctly detect 11 out of the 16 meals that are announced by the patient. This can for instance be seen on the 2nd of January around 7 AM where bolus is also given to the meal.

In continuation of this, it is seen on the figure that the patient sometimes ingests two meals and gives bolus twice within two hours time but the GRID algorithm only detects one meal. For this we are going to analyze two cases where the patient has announced two meals within a 2 hour range. In the first case, the meals are announced within a few minutes and the next case the meals have been announced with more than 30 minutes in between.

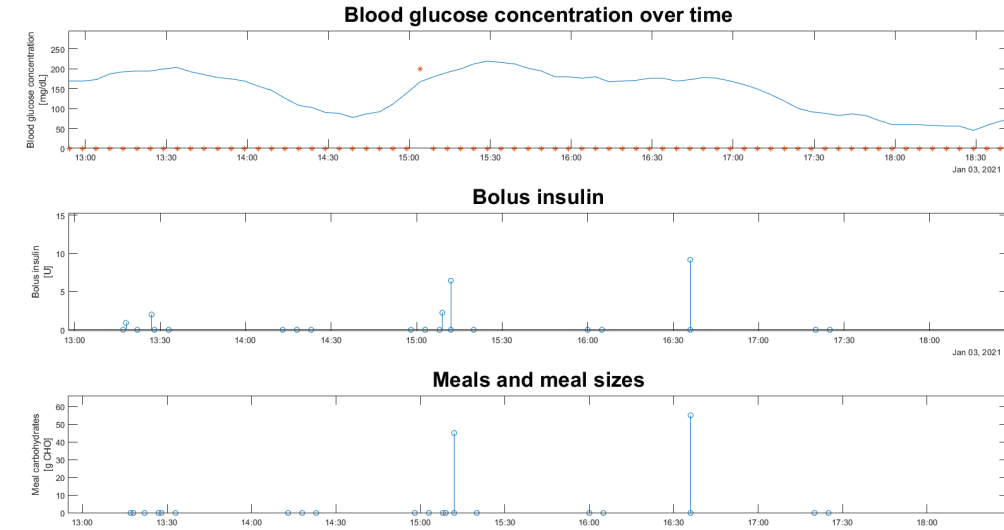
First looking at 4th of January at 10:20 and at 10:25, the patient ingests two meals within a few minutes but simply one meal is detected at 10.09. A reason for announcing two meals and two times of bolus insulin within a few minutes could be that the patient forgot to announce the whole meal and the whole bolus insulin dose to begin with. It shall also be noticed that the meal is detected before the meal is ingested which might be because the patient announces the meal after already ingesting it. A better visualization of the ingested meals and bolus insulin given within few minutes can be seen in Figure 4.2.



**Figure 4.2:** Blood glucose concentration curve when ingesting two announced meals within few minutes

The figure shows how the slope for the blood glucose concentration is continuously increasing without halts leading to a reasonable assumption that it can be considered as one meal. This scenario can be seen again on January 2nd at 20.53 and 21.02 meaning this occurs twice in total. Based on the assumption of these meals in practise being one meal, the 16 announced meals can be reduced to 14 meals meaning that the GRID algorithm only missed 3 meals.

However, when the time between the two meals is more than just a few minutes the same conclusion cannot be made. For instance on January 3rd which is visualized on Figure 4.3, there is a detection at 15.09 where the patient ingests a meal at 15.12 and again at 16.36.

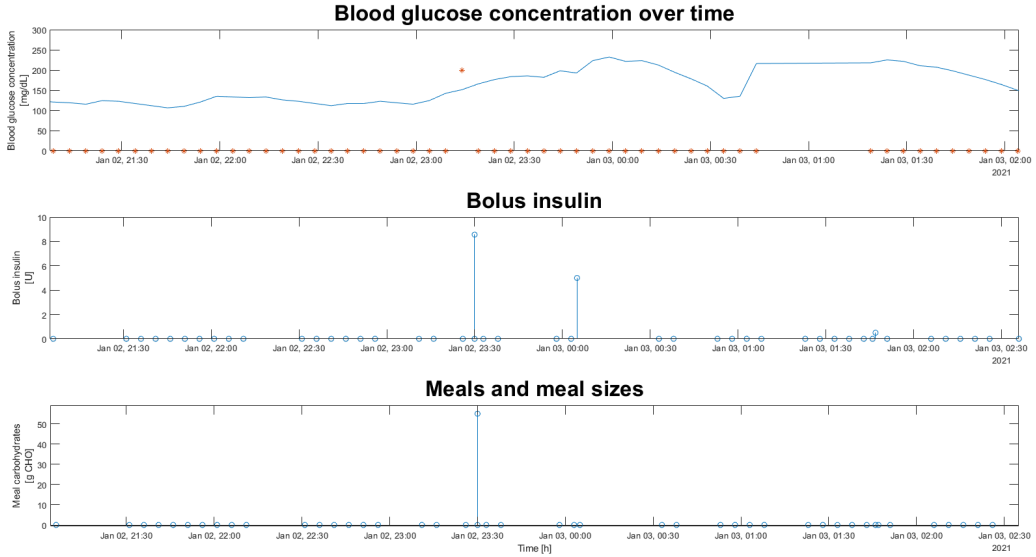


**Figure 4.3:** Blood glucose concentration curve when patient has ingested two meals and given bolus twice within 2 hours

In contrary to the example before, the blood glucose concentration decreases before the second announced meal is ingested. Therefore, it is not reasonable to consider these two meals as one meal. Nevertheless, only one meal is being detected. The reason for this is either due to the 2-hour counter or that the blood glucose concentration might still not meet the criteria of the  $G_{min}$  values in the constraints in the GRID-algorithm. So even if the counter was set to detect more often the  $G_{min}$  values could still be the reason.

This scenario happens once again at January 2nd at 19.59 with meals ingested at 19.33 and 20.53. This means once again that due to either the 2-hour counter or the  $G_{min}$  values we experience that the GRID algorithm does not detect two meals that the patient has announced.

Back to Figure 4.1, we recall that the patient sometimes only announces one meal but gives bolus insulin twice. For instance, this happens on the 2nd of January which is visualized on Figure 4.4. The patient announces the meal at 23.30 but gives bolus insulin both at 23.30 and again at 00.05. In this time frame the GRID algorithm simply detects one meal at 23.14.



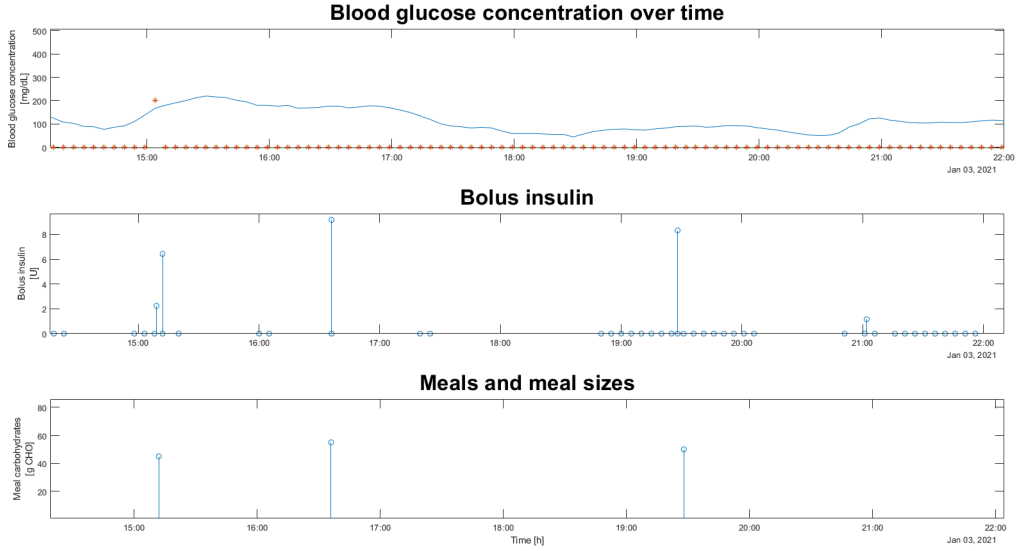
**Figure 4.4:** Blood glucose concentration curve when bolus insulin is given with 34 minutes in between

Although it was mentioned that the patient might sometimes give another dose of corrected bolus insulin in addition to the possibly not optimal first dose, we don't suspect that to be the case in this particular example. Therefore, despite only one announcement of a meal, we reckon that the patient has actually ingested a second meal when the second dose of bolus insulin is taken. If we assume that the patient has forgotten to notify the second meal when the bolus insulin is injected, the GRID algorithm should have detected the meal. A reason that the possible meal is not being detected in this situation is the same as before either due to the 2-hour counter or that the ROC does not meet the criteria of the  $G_{min}$  values.

Still looking at Figure 4.4, one more interesting point can be noticed. At 00.39 the blood glucose starts to increase drastically even though no meal or bolus insulin has been announced. Here we would have expected for the GRID algorithm to detect this possible meal given that it most likely does meet the  $G_{min}$  criteria. Therefore, we must assume that the GRID algorithm does not detect the meal because of the 2-hour counter.

This scenario happens several times during the 3 days always within two hours after a detected meal. Hence, the 2-hour counter definitely limits the performance of the GRID algorithm.

We see at Figure 4.1 on January 3rd that a meal has been announced at 19.28 but this meal goes by undetected even though the last meal is detected over 2 hours before at 15.04. This is further visualized in the figure below 4.5

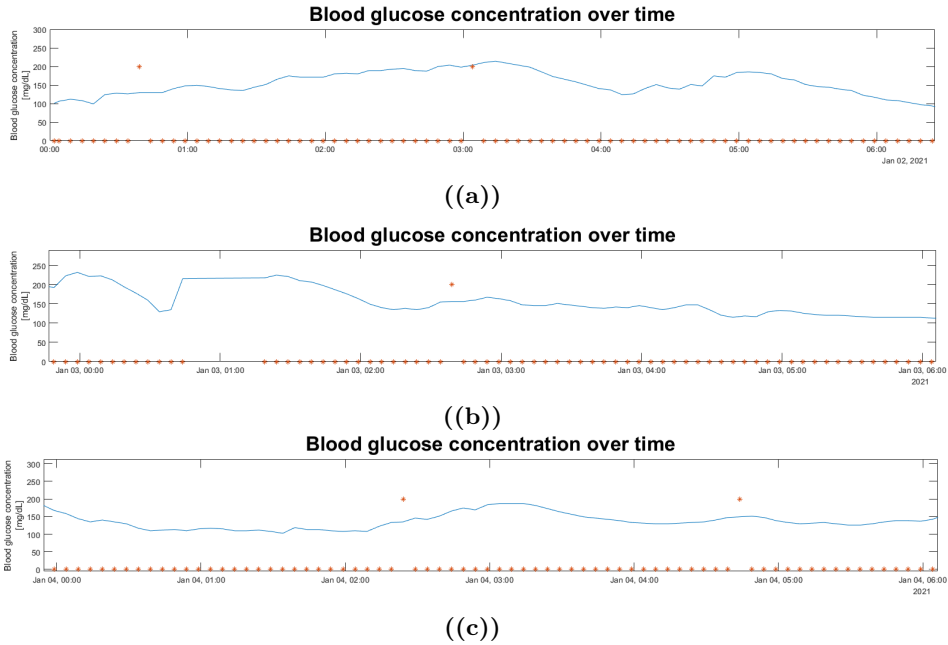


**Figure 4.5:** Blood glucose concentration curve, bolus insulin and meals when patient has ingested a meal and taken bolus insulin after two hours of the previous meal without a detection of the GRID-algorithm.

We notice on Figure 4.5, that when ingesting this meal the blood glucose concentration is lower than  $120 \frac{\text{mg}}{\text{dL}}$  which means that the criteria from  $G_{min,1}$  is not met. In this case, for the GRID algorithm to be able to detect the meal, the  $G_{min,1}$  value should be lower than  $93.6 \frac{\text{mg}}{\text{dL}}$ . However, when the blood glucose concentration is so low such that a patient is in need of ingesting a meal, it might in general be difficult for any GRID algorithm to detect these meals without also detecting too many false meals because the criteria of  $G_{min,1}$  would be too low.

Again back to Figure 4.1, we will look at the detected meals from the GRID algorithm where the patient has not announced neither given bolus insulin nor ingested meals. In total this happen 10 times, where some of the times are shown in the figure below.





**Figure 4.6:** (a) Night of the 2th January (b) Night of the 3th January (c) Night of the 4th January. No meal or bolus has been announced on either detection in all subfigure.

On all the scenarios seen in Figure 4.6, the blood glucose concentration is higher than  $G_{min,1} = 120 \frac{\text{mg}}{\text{dL}}$ . Most of the scenarios happen during the night time after midnight. For some diabetes patients it is natural to experience increasing blood glucose concentrations during their sleep known as the Dawn Phenomenon[16] which can be due to the release of hormones and a release of glycogen from the liver [17]. Therefore it is uncertain whether the patient is an example of this or if the GRID-algorithm is in fact detecting some meals that the patient has ingested without announcement. If they are not correct meals, the reason for the GRID algorithm to detect them anyway, could be because  $G'_{min,2}$  and  $G'_{min,3}$  are really low but should have been chosen higher. On the other hand, since the GRID algorithm has made this detection we can not reject that a meal has been ingested.

#### 4.2.1 Discussion of the GRID algorithm used on real data

Overall, the primary tendency is that the our GRID algorithm is limited due to the 2-hour counter. There are 3 times a meal has been announced but no meal has been detected and several times where a possible meal should have been detected but was not due to the 2-hour counter.

In the real world, the number of times a meal is ingested and when is sporadic compared to how we have simulated the data. In our simulations, we have assumed that no meals are ingested within two hours but this is not the case for the clinical patient. It will therefore be necessary to tune on the counter for a more optimal GRID algorithm.

Furthermore there are 10 scenarios where the GRID algorithm detects a meal that not always looks like a meal especially during the night time. This could indicate that the  $G_{min}$  values were chosen too low. However as already discussed, we would maybe not end up

with the GRID algorithm detecting all the meals if the values were higher. However, given that this happens mostly during the night this could therefore be a topic of conversation for the doctor and the patient since it could be because of a natural influence.

Still worth mentioning in the discussion of these results is that the optimal  $G_{min}$  values are chosen based on tuning with 57 simulated patients. As mentioned a higher number of simulated patients would be desirable and would lead to a GRID algorithm that might be used more widely on different clinical patients. However, we only test our GRID algorithm on one clinical patient and it is therefore difficult to conclude that it have effected the performance.

A consideration also discussed earlier was the influence of the value  $\tau_F$  and how too low values could result in the algorithm detecting too many false meals. However, we decided to fix this value even though we know it can influence the performance a lot. To follow up, we did not see any indication that this decision gave any wrong detections that could be due to the  $\tau_F$  value. So, the choice of  $\tau_F = 6$  seems to have been a good estimation.

# Conclusion

---

The GRID algorithm has a lot of parameters that could have been tuned and taken into consideration. However, in attempt to optimize the GRID algorithm for the purpose of assisting doctors with their brief consultations we have only focused on tuning the combinations of the  $G_{min}$  values.

The simulations are the foundation of the report, however, we do not entirely succeed in making the simulations exactly resemble the clinical data. This could have required that we had to use another feedback system or given smaller but more meals. However, we accept our simulations as a good foundation and starting point.

We make an evaluation function that is able to evaluate the performance of the GRID algorithm with respective  $G_{min}$  values based on the number of true positives and false positives. Hereby, we set two thresholds deciding in how many of each we will tolerate. We decide to choose the thresholds based on how many false positives we accept per day and how few true positives we accept to be detected out the total number of 90 true meals. In addition, this makes sense for us but another way to evaluate and examine the errors could have been done.

We end up with the optimal  $G_{min}$  combination to be  $G_{min} = [120, 0.6, 0.5]$  which we then use in the GRID algorithm when detecting meals on the clinical data. However, we only simulate 100 patients and have to disregard 43 of them. Therefore, we need to emphasize that these  $G_{min}$  values perhaps do not fit as many randomly clinical patients as we would desire.

We use the GRID algorithm on the whole data set that was given throughout approximately 5 month. The algorithm detected 764 out of 548 meals but the bolus insulin was given 1,332 times, which could indicate that the patient at some meals did not give the right amount to begin with and therefore had to give more.

During the 3 chosen days we analyze, the GRID algorithm detects 21 meals.

We saw that some of the meals detected might not have been meals, but rather a natural increase in the blood glucose concentration during night time. So, if this is the case the reason for the GRID algorithm to detect could be because of  $G'_{min,2}$  and  $G'_{min,3}$  value being too low.

Furthermore, we saw that the  $G_{min,1}$  was sat too high when a meal is ingested at a time when the blood glucose concentration is too low. However, setting this value lower could result in a lot of false meals being detected.

Mainly, we noticed that the GRID algorithm is limited due to the 2-hour counter. It becomes a problem several times, both when a meal is announced but is not being detected, and also when we see on the blood glucose concentration curve that the patient supposedly ingest a meal even though no meal or bolus is notified. This parameter should therefore definitely be tuned to optimize the GRID algorithm further.



# APPENDIX A

## An Appendix

---

The referred scripts can be seen on the following page in the order:

1. Implementation of the PID controller.
2. Implementation of the GRID algorithm.
3. Implementation of the open loop simulation.
4. Implementation of the closed loop simulation.
5. Implementation of the evaluation function.

To access all the functions and Matlab script look at the attached zip file denoted Hold\_11.

---

20-06-22 10:32 /Users/marianadesamadsen/D.../PIDControl2.m 1 of 2

---

```

function [uk,ctrlstate] = PIDControl2(yk, U, ctrlPar, ctrlState)
%
% PIDControl()
%
% DESCRIPTION:
% This function implements a discretized proportional-integral-derivative
% (PID) controller for controlling the insulin flow rate. This differs from
% PIDControl by additionally having the insulin vector as input
%
% INPUT:
% yk          - Current blood glucose concentration
% U           - Insulin vector of both basal and bolus insulin
%
% ctrlPar     - vector of the following:
%                * Ts          - Sampling time, 5 min
%                * Kp          - Proportional gain
%                * Ki          - Integrator gain
%                * Kd          - Derivative gain
%                * ybar        - The target glucose concentration, y=108
%                * ubar        - Nominal insulin flow rate
%                * Ti          - Tuned parameters
%                * Td          - Tuned parameters
%
% ctrlState   - vector of the following:
%                * Ik          - The integral term (Ik)
%                * ykm1        - Previous glucose concentration, yk-1
%
% OUTPUT:
% uk          - a vector of manipulated inputs
% ctrlstate   - the updated controller state
%
% PROJECT:
% Fagprojekt 2022
% A diabetes case study - Meal detection
%
% GENEREL:
% BSc          : Mathematics and technology
% University   : The Technical University of Denmark (DTU)
% Department   : Applied Mathematics and Computer Science
%
% AUTHORS:
% Emma Victoria Lind
% Mariana de Sá Madsen
% Mona Saleem
%
% CONTACT INFORMATION
% s201205@student.dtu.dk
% s191159@student.dtu.dk
% s204226@student.dtu.dk
%

% Unpack control parameters
Ts      = ctrlPar(1); %      Sampling time
Kp      = ctrlPar(2); %      Proportional gain
%Ki      = ctrlPar(3); %      Integrator gain (not used since we calculate it in the✓
function line 68)
%Kd      = ctrlPar(4); %      Derivative gain (not used since we calculate it in the✓

```

---

20-06-22 10:32 /Users/marianadesamadsen/D.../PIDControl2.m 2 of 2

---

```

function line 69)
ybar  = ctrlPar(5); %      Target blood glucose concentration
ubar  = ctrlPar(6); %      Nominal insulin flow rate
Ti    = ctrlPar(7); %      Tuned parameters
Td    = ctrlPar(8); %      Tuned parameters

% Unpack control state
Ik = ctrlState(1); %      Value of integral at previous time step
ykm1 = ctrlState(2); %    Previous observed glucose concentration

% Computing

ek = yk-ybar; % Setpoint error

Ki = Kp * Ts/Ti; % Helps controlling the steady state
Kd = Kp * Td/Ts; % The top

Pk = Kp * ek; % Proportional term. Controls how fast the error change

Ikp1 = Ik + Ki * ek; % Integral term. The area of the error

Dk = Kd * (yk-ykm1); % Derivative term. The top of the curve

uba = ubar + Pk + Ik + Dk; % Basal insulin flow rate
ubo = U; % Bolus insulin flow rate

% OUTPUT

% The controlled manipulated inputs at time step
uk = [uba,ubo];

% Controller state OUTPUT
ctrlstate = [Ikp1; yk];

end

```

---

20-06-22 10:40 /Users/marianadesamadsen/Doc.../GRID\_func.m 1 of 3

---

```
function [ Gfm_vec , filt_prev , flag, zero_one ] = GRID_func( ...
    delta_G , G_vec , tau, tspan , filt_prev , Gmin, Gfm_vec , t_vec, flag)

%
% GRID_func()
%
% DESCRIPTION:
% The function is a part of the GRID algortihm. This is part of the
% dectection logic, the last part of the algortihm.
% The function takes in the glucose measurements, calls the two filter
% functions and finds the derivatives. After this it is able to detect
% weather or not there a meal has been detected. Lastly, it counts down
% such that a meal will not be detected twice within two hours
%
% INPUT:
% delta_G          - The maximum ROC (rate of change)
%
% G_vec            - Vector consisting of the glucose value and the
%                  two previous glucose measurements.
%                  As follows: [Gm-2, Gm-1, Gm].
%
% tspan            - The interval step given as a number
%
% filt_prev        - Vector of previous filteret glucose measurements
%                  As follows: [G_{F,NS}(k-1), G_{F}(k-2)].
%                  For equation (1)&(3).
%
% Gmin             - Vector of minumum glucose measurements
%                  As follows: [G_{min,1},G_{min,2},G_{min,3}].
%                  For equation (4).
%
% Gfm_vec          - Vector of previous derivatives
%                  As follows: [G'_{F}(k-2),G'_{F}(k-2)].
%                  For equation (4).
%
% t_vec            - Vector of sampling time respectively for G
%
% flag             - For counting the time from last detected meal
%
% OUTPUT:
% Gfm_vec          - The stored new vector of the previous filtered
%                  glucose measurements.
%                  As follows: [G'_{F}(k-1),G'_{F}(k)].
%
% G_prev           - The stored new vector of previous glucose
%                  measurements.
%                  As follows: [G_{F,NS}(k-2),G_{F}(k-2)].
%
% zero_one         - 1 or 0 for detected meal.
%
% flag             - For counting the time from last detected meal
%
% PROJECT:
% Fagprojekt 2022
% A diabetes case study - Meal detection
%
% GENEREL:
% BSc              : Mathematics and technology
% University       : The Technical University of Denmark (DTU)
```



---

20-06-22 10:40 /Users/marianadesamadsen/Doc.../GRID\_func.m 2 of 3

---

```
% Department          : Applied Mathematics and Computer Science
%
% AUTHORS:
% Emma Victoria Lind
% Mariana de Sá Madsen
% Mona Saleem
%
% CONTACT INFORMATION
% s201205@student.dtu.dk
% s191159@student.dtu.dk
% s204226@student.dtu.dk
%
%

% Inisializing all values

Gfm_m2 = Gfm_vec(1);      % The second previous derivative used in euqation 4
Gfm_m1 = Gfm_vec(2);      % The previous derivative used in equation 4

Gfnsn2_prev = filt_prev(1); % The second previous noise-spike filtered value
                        % used in equation 1
Gfm2_prev   = filt_prev(2); % The second previous low filterd value used in
                        % equation 2

% Minimum values used in equation 4
Gmin1 = Gmin(1);
Gmin2 = Gmin(2);
Gmin3 = Gmin(3);

% The two previous measured glucose values and the one at control state
Gm2    = G_vec(1);      % The second previous glucose value
Gm1    = G_vec(2);      % The previous glucose value
G      = G_vec(3);      % The glucose value at control state

% COMPUTING

% The noise-spike filter at the 3 sampling times
Gfnsn2 = spikefilt_func(Gm2,Gfnsn2_prev,delta_G);
Gfnsn1 = spikefilt_func(Gm1,Gfnsn2,delta_G);
Gfns   = spikefilt_func(G,Gfnsn1,delta_G);

% The low filter at the 3 sampling times
Gfm2 = lowfilt_func(tau,tspan,Gfnsn2,Gfm2_prev);
Gfm1 = lowfilt_func(tau,tspan,Gfnsn1,Gfm2);
Gf    = lowfilt_func(tau,tspan,Gfns,Gfm1);

% Inisializing input for estimate_lagrange
Gf_vec = [ Gfm2 , Gfm1 , Gf ];

% Computing the first derivative using lagrange
Gfm     = estimate_lagrange(t_vec,Gf_vec); % Returns the derivative

% The detection part from equation 4
if (Gf > Gmin1) && ...
    ((Gfm > Gmin3) && (Gfm_m1 > Gmin3) && (Gfm_m2 > Gmin3) ...
    || (Gfm > Gmin2) && (Gfm_m2 > Gmin2) )
```

---

**20-06-22 10:40 /Users/marianadesamadsen/Doc.../GRID\_func.m 3 of 3**

---

```
    zero_one = 1; % A meal has been detected
else
    zero_one = 0; % No meal has been detected
end

% Output
filt_prev = [Gfns2,Gfm2]; % Outputting the updated filtered values
Gfm_vec = [Gfm_1,Gfm]; % Outputting the updated derivative values

% COUNTING PART
if flag > 0
    % flag larger than 0 means that a meal has been detected within 120 min
    % implying that a meal should not be detected again already. So zero_one
    % is set to 0. Therefore, flag is subtracted by -1, such that it will
    % count down so a meal can be detected again after 120 min.

    flag = flag-1;
    zero_one = 0;
elseif flag == 0 && zero_one == 1
    % flag equal to zero means a meal may be detected again
    % but only is if zero_one equals 1. When this happen flag start over
    % counting down 120 min.

    flag = 120/tspan;
end

end
```

---

20-06-22 10:33 /Users/marianadesam.../OpenLoopSimulation.m 1 of 2

---

```

function [T,X] = OpenLoopSimulation(x0, tspan, U, D, p, simModel, simMethod, NK)
% OpenLoopSimulation()
%
% DESCRIPTION:
% Function performs an open-loop simulation for given initial condition of
% the state vector, time, intervals, disturbance variables, parameters, and
% simulation model and methods. The open-loop simulation uses the MVPmodel
% and ExplicitEuler to compute both the subcutaneous glucose concentration,
% Gsc(t), the blood glucose concentration G(t) and the statevector x(t) for
% each time step.
%
%
% INPUT:
% x0          - initial state vector                      (dimension: 7)
% tspan       - time interval to integrate over          (dimension N+1)
% U           - bolus and basal insulin (manipulated input) (dimension nu x N)
% D           - meal rate (disturbance)                  (dimension nd x N)
% p           - parameter values                         (dimension np)
% simModel    - simulation model, MVPmodel               (function handle)
% simMethod   - simulation method, ExplicitEuler         (function handle)
% NK          - Number of timesteps in each time interval
%
% OUTPUT:
% T - The control state of time for each step            (dimension: N
N+1)
% X - The statevector x(t) for each time step stored in a matrix (dimension: nx
x N+1)
%
% PROJECT:
% Fagprojekt 2022
% A diabetes case study - Meal detection
%
% GENERAL:
% BSc          : Mathematics and technology
% University   : The Technical University of Denmark (DTU)
% Department   : Applied Mathematics and Computer Science
%
% AUTHORS:
% Emma Victoria Lind
% Mariana de Sá Madsen
% Mona Saleem
%
% CONTACT INFORMATION
% s201205@student.dtu.dk
% s191159@student.dtu.dk
% s204226@student.dtu.dk
%

% Number of control steps
N = numel(tspan) - 1;

% Number of states
nx = numel(x0);

% Number of time steps in each control interval
Nk=NK;

```

---

20-06-22 10:33 /Users/marianadesam.../OpenLoopSimulation.m 2 of 2

---

```
% Allocate memory
T = zeros(1, N+1);
X = zeros(nx, N+1);

% Initial condition in each control interval
xk = x0;

% Store solution
T(1) = tspan(1);
X(:,1) = x0;

% Loop for each time step. Computes Gsc(t), G(t), x(t) from k = 0 to N.
for k = 1:N
    % Times
    tk    = tspan(k);
    tkp1  = tspan(k+1);

    % Manipulated inputs and disturbance variables
    uk = U(:,k);
    dk = D(:,k);

    % Time interval
    tspank = linspace(tk, tkp1, Nk+1);

    % Solve initial value problem
    [Tk, Xk] = simMethod(simModel, tspank, xk, uk, dk, p);

    % Update initial condition
    xk = Xk(end, :)';

    % Store solution
    T(k+1) = Tk(end)';
    X(:, k+1) = Xk(end, :)';
end

end
```

---

20-06-22 10:33 /Users.../ClosedLoopSimulation\_withnoise2.m 1 of 3

---

```
function [T, X, Y, U, ctrlState] = ClosedLoopSimulation_withnoise2(tspan,x0,D,U,p, ...
    ctrlAlgorithm, simMethod, simModel, observationMethod, ctrlPar,ctrlState0,NK,
intensity)
%
% ClosedLoopSimulation()
%
% DESCRIPTION:
% Performs a closed-loop simulation of a model-based control algorithm for
% given time range, initial condition, disturbance variables, insulin
% levels, parameters, control algorithm, simulation model, observation
% model, control parameters, control state, control intervals and intensity
% level.
% Closed-loop is used when the input depends on the
% output; this function is part of the PID-controller.
%
% INPUT:
%   tspan                - boundaries of the control intervals          (dimension: N+1)
%   x0                   - initial state                                (dimension: nx)
%   D                    - disturbance variables for each control interval (dimension: nd x N)
%   U                    - Insulin levels of both bolus and basal        (dimension: nu x N)
%   p                    - parameters                                    (dimension: np)
%   simModel             - simulation model                             (function handle)
%   ctrlAlgorithm        - control algorithm                           (function handle)
%   ctrlPar              - controller parameters
%   ctrlState0           - initial controller state                    (dimension: nc)
%   simMethod            - simulation method                            (function handle)
%   NK                   - Number of steps in each control interval     (scalar)
%   intensity            - The intensity value used for Euler Maruyama   (scalar)
%
% OUTPUT:
%   T - boundaries of control intervals (=tspan)          (dimension: N+1)
%   X - the states in the simulation model                 (dimension: nx x N+1)
%   Y - the observed variables                             (dimension: ny x N+1)
%   U - the computed manipulated inputs                   (dimension: nu x N)
%   ctrlState - matrix of controller states               (dimension: nc x N+1)
%
% PROJECT:
% Fagprojekt 2022
% A diabetes case study - Meal detection
%
% GENEREL:
% BSc                : Mathematics and technology
% University         : The Technical University of Denmark (DTU)
% Department         : Applied Mathematics and Computer Science
%
% AUTHORS:
% Emma Victoria Lind
% Mariana de Sá Madsen
% Mona Saleem
```

---

20-06-22 10:33 /Users.../ClosedLoopSimulation\_withnoise2.m 2 of 3

---

```
%
% CONTACT INFORMATION
% s201205@student.dtu.dk
% s191159@student.dtu.dk
% s204226@student.dtu.dk
%

% Initial time
t0 = tspan(1);

% Observed variables of glucose at steady state
y0 = observationMethod(x0,p);

% manipulated inputs calculated
uDummy = ctrlAlgorithm(y0,U(1,1), ctrlPar, ctrlState0);

% Number of each variable
nx = numel(x0);           % states
ny = numel(y0);           % glucose concentration
nu = numel(uDummy);       % manipulated inputs
nc = numel(ctrlState0);   % glucose concentration and integral term

% Number of control intervals
N = numel(tspan)-1;

% Number of time steps in each control interval
Nk = NK;

% Initialising Output
T = zeros( 1, N+1);
X = zeros(nx, N+1);
Y = zeros(ny, N+1);
ctrlState = zeros(nc, N+1);

% Storing solution
T(1) = t0;
X(:,1) = x0;
Y(:,1) = y0;
ctrlState(:, 1) = ctrlState0;

% Copying initial condition to another name
tk = t0;
xk = x0;
yk = y0;

for k = 1:N

    %%% Initializing the loop

    % Time
    tkp1 = tspan(k+1);

    % Time interval
    tspank = linspace(tk, tkp1, Nk+1);

    % Controller state
    ctrlStatek = ctrlState(:, k);
```

---

**20-06-22 10:33 /Users.../ClosedLoopSimulation\_withnoise2.m 3 of 3**

---

```
% Disturbance variables
dk = D(:, k);

%% Start computing

% Compute manipulated inputs after the PID control
[uk, ctrlStatekp1] = ctrlAlgorithm(yk, U(2 ,k), ctrlPar, ctrlStatek);

% Solving the differential equation with euler maruyama
[Tk, Xk] = simMethod(simModel, tspank, xk, uk, dk, p,intensity);

%% Overwriting for the next loop

% States at the next time step
tkp1 = Tk(end);
xkp1 = Xk(end,:);

% Observed variables at the next time step
ykp1 = xkp1(7);

% Update initial condition
tk = tkp1;
xk = xkp1;
yk = ykp1;

%% Storing solution

% Store solution and updating conditions
T( k+1) = tkp1;
X(:, k+1) = xkp1;
Y(:, k+1) = ykp1;
U(:, k ) = uk;
ctrlState(:, k+1) = ctrlStatekp1;
```

end

---

20-06-22 10:32 /Users/marianadesam.../evaluationfunction.m 1 of 3

---

```
function [truepositive,falsepositive] = evaluationfunction(stride,D,D_detected,Ts,N)
%
% detectionrates()
%
% DESCRIPTION:
% This function computes the TP, FP, TN, FN based on a true vector with
% detected meals D, and a computed vector with detected meals D_detected.
% It uses a stride since the meal will be detected a time period after the
% meals was given.
% It uses the indices for the true meals and the detected meals to compare
% if in the stride a meals should be detected or not and vise versa.
%
% INPUT:
% stride          - The maximal time period it takes from the meal to be
% detected
% D               - The true vector of real meals.
% D_detected      - The estimated vecor of 0 or 1. 1 meaning meals is
% detected
% U               - Bolus insulin
%
% Ts              - The time between control steps
% N               - The number of observations
%
% OUTPUT:
% Two outputs being TP, FP
%
% PROJECT:
% Fagprojekt 2022
% A diabetes case study - Meal detection
%
% GENEREL:
% BSc              : Mathematics and technology
% University       : The Technical University of Denmark (DTU)
% Department       : Applied Mathematics and Computer Science
%
% AUTHORS:
% Emma Victoria Lind
% Mariana de Sá Madsen
% Mona Saleem
%
% CONTACT INFORMATION
% s201205@student.dtu.dk
% s191159@student.dtu.dk
% s204226@student.dtu.dk
%
% Initializing
falsenegative = 0;
falsepositive = 0;
truepositive  = 0;
truemeals     = zeros(1,N);
mealdetec     = zeros(1,N);

% Changing datatype of D to binary
for i = 1:N
    if D(1,i) >= 50/Ts % Not considering the snackmeals
        D(1,i) = 1;
    end
end
```



---

**20-06-22 10:32 /Users/marianadesam.../evaluationfunction.m 2 of 3**

---

```
    else
        D(1,i) = 0;
    end

end

% Finding the indices for the true meals
for i = 1 : N
    if D(1,i) == 1
        true meals(i) = i;
    end
end

% Finding the indices for the detected meals
for i = 1 : N
    if D_detected(i) == 1
        meal detec(i) = i;
    end
end

% Removing all the zeros so there is only the indices left
idx detec meals = nonzeros(meal detec)';
idx true meals = nonzeros(true meals)';

% Examine if there are no true meals where there are detected meals
for i = 1:length(idx detec meals)

    % The idx value when meal has been detected
    k = idx detec meals(i);

    if (k-stride) < 1
        j = k-1;

        if sum(D(1,k-j:k)) == 0
            false positive = false positive + 1;
        end

    elseif sum(D(1,k-stride:k)) == 0
        false positive = false positive + 1;
    end
end

% Examine if there are true meals where there are detected meals
for i = 1:length(idx detec meals)

    % The idx value when meal has been detected
    k = idx detec meals(i);

    if (k-stride) < 1
        j = k-1;

        if sum(D(1,k-j:k)) == 0
            false positive = false positive + 1;
        end

    elseif sum(D(1,k-stride:k)) == 1
        true positive = true positive + 1;
    end
end
```

---

20-06-22 10:32 /Users/marianadesam.../evaluationfunction.m 3 of 3

---

end  
end

end

# Bibliography

---

- [1] *Hvad er type 1-diabetes?*, last updated 23rd of March 2022, Videncenter for Diabetes, accessed 15th of June 2022.  
<https://videncenterfordiabetes.dk/viden-om-diabetes/type-1-diabetes/hvad-er-type-1-diabetes>
- [2] *Diabetes på verdensplan*, Diabetes foreningen, accessed 15th of June 2022.  
<https://diabetes.dk/forskning/viden-om-diabetes/diabetes-pa-verdensplan>
- [3] Harvey et al. 2014, "*Design of the Glucose Rate Increase Detector: A Meal Detection Module for the Health Monitoring System*", Journal of Diabetes Science and Technology 2014, Vol. 8(2), p. 308.
- [4] Meng, Q., Mäkinen, V. P., Luk, H., and Yang, X. 2013. "*Systems Biology Approaches and Applications in Obesity, Diabetes, and Cardiovascular Diseases*", Current cardiovascular risk reports, 7(1), 73–83.  
<https://doi.org/10.1007/s12170-012-0280-y>
- [5] Nazmul Siddique 2014, "*Intelligent Control: A Hybrid Approach Based on Fuzzy Logic, Neural Networks and Genetic Algorithms*", Springer, p. 40 l. 26.
- [6] Åström, J. K, and Murray, M. Richard 2012, "*Feedback Systems: An Introduction for Scientists and Engineers* ", Version v2.11b (28 September 2012), Princeton University Press, p 17.
- [7] Åström, J. K, and Murray, M. Richard 2012, "*Feedback Systems: An Introduction for Scientists and Engineers* ", Version v2.11b (28 September 2012), Princeton University Press, p 23.
- [8] Åström, J. K, and Murray, M. Richard 2012, "*Feedback Systems: An Introduction for Scientists and Engineers* ", Version v2.11b (28 September 2012), Princeton University Press, p 24.
- [9] Dickson, J. L., Hewett, J. N., Gunn, C. A., Lynn, A., Shaw, G. M., and Chase, J. G. (2013). *On the problem of patient-specific endogenous glucose production in neonates on stochastic targeted glycemic control*, Journal of diabetes science and technology, 7(4), 913–927.  
<https://doi.org/10.1177/1932296300700414>
- [10] Cheney, W. and Kincaid, D. 2012, "*Numerical Mathematics and Computing*", Seventh Edition, Cengage Learning pp. 125-128.
- [11] Szabados, T., 1994, "*An Elementary Introduction to the Wiener Process and Stochastic Integrals*", Technical University of Budapest, p. 2 l.1.

- [12] Higham, J. D., 2001, "*An Algorithmic Introduction to Numerical Simulation of Stochastic Differential Equations*" SIAM REVIEW Vol. 43, No. 3, pp. 525–546, p. 2.
- [13] Dinsmoor, S. R., last updated 13th of May 2020, "*Insulin-to-Carbohydrate Ratio: Definition and Overview*", accessed 12th of June 2022.  
<https://www.diabetesselfmanagement.com/diabetes-resources/definitions/insulin-to-carbohydrate-ratio/>
- [14] Kanderian et al. 2009, "*Identification of Intraday Metabolic Profiles during Closed-Loop Glucose Control in Individuals with Type 1 Diabetes*", Journal of Diabetes Science and Technology, Volume 3, Issue 5, p. 1051.
- [15] John Bagterp Jørgensen, Dimitri Boiroux, Zeinab Mahmoudi, "*An artificial pancreas based on simple control algorithms and physiological insight*", IFAC-PapersOnLine, Volume 52, Issue 1, 2019, Pages 1018-1023, ISSN 2405-8963, <https://doi.org/10.1016/j.ifacol.2019.06.196>.  
<https://www.sciencedirect.com/science/article/pii/S2405896319302848>
- [16] O'Neal TB, Luther EE. Last updated 2021 May 23, "*Dawn Phenomenon*", accessed 19th of June 2022  
<https://www.ncbi.nlm.nih.gov/books/NBK430893/>
- [17] Castro, R., M., last updated Nov. 13 2020, "*The dawn phenomenon: What can you do?*", accessed 19th of June 2022.  
<https://www.mayoclinic.org/diseases-conditions/diabetes/expert-answers/dawn-effect/faq-20057937>



