Modeling and Simulation of Glucose-Insulin Metabolism

Esben Friis-Jensen (s042244)

Kongens Lyngby 2007
IMM-Bachelor-2007-

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

IMM-Bachelor: ISSN

Preface

This B.sc thesis was carried out at Informatics and Mathematical Modelling department, The Technical University of Denmark, under supervision of John Bagterp Jørgensen.

Summary

Diabetes is a widespread disease in the western world today. Many researchers are working on methods for diagnosing and treating diabetes. A tool used for this is mathematical models of the blood glucose and insulin kinetics.

In this thesis one of the models, Bergman's minimal model is described trough derivation and simulations. It is a model consisting of a glucose and an insulin kinetics part. The part, describing glucose kinetics has the problem that it overestimates glucose effectiveness S_G and underestimates insulin sensitivity S_I, which is interpretation parameters of a test called the IVGTT (Intravenous Glucose Tolerance Test).

Modifications and additions which could be done in order to describe the glucose and insulin kinetics more thoroughly is described. Based on Bergman's minimal model, two coupled models are proposed. A coupling between the two basic parts of Bergman minimal model and a coupling between the two modified parts of Bergman's minimal model. The basic coupling is called the original model. It can be used to describe the IVGTT for a healthy and a glucose resistant subject. Through calculation and simulation it is shown that the original model has a equilibrium problem, when a parameter p_5 is less than the basal concentration G_b.

The modified coupling, which is able to describe the glucose-insulin system for a type 1 diabetic on treatment is tested for reactions to insulin injections and change in basal insulin production. A PID controller,controlling insulin delivery is implemented, and it is shown how it can be used with the modified model, in order to test it for meal disturbance.

The final conclusion of the thesis is that both coupled models have problems, but they can be used to approximately simulate the blood glucose-insulin system, if you are aware where the problems occur.

Resumé

Diabetes er en udbredt sygdom i den vestlige verden i dag. Mange forskere arbejder derfor på at lave metoder til diagnose og behandling. Et værktøj brugt til dette er matematiske modeller, som beskriver blod glukose-insulin systemet.

I denne tesis bliver en af disse modeller, Bergman's minimale model, beskrevet via udledning og simulering. Modellen består af en glukose og en insulin kinetik beskrivelse. Den del, som beskriver glukose kinetiken har det problem at den overestimerer glukose effektivitet S_G og insulin følsomhed S_I, som er fortolkningsparametre af en test kaldet IVGTT (Intravenøs glukose tolerance test).

For at kunne beskrive glukose og insulin kinetikken bedre, bliver der foreslået udvidelser og modificeringer af modellen. Baseret på Bergmans minimale model, foreslås to sammenkoblinger. Den ene er en kobling mellem de basale dele af Bergmans minimale model og den anden er en kobling mellem den de udvidede dele af Bergmans minimale model. Den basale kobling bliver kaldt den originale model. Den kan bruges til at beskrive IVGTT'en for et rask og et glukose resistant subjekt. Gennem beregning og simulering viser det sig, at den originale model har et ligevægts problem når en bestemt parameter p_5 er mindre end den basale koncentration G_b.

Den udvidede kobling, som kan bruges til at beskrive type 1 diabetikere under behandling, bliver testet for reaktioner på insulin indsprøjtninger og ændringer i den basale insulin produktion. En 'PID controller', som kontrollerer insulin produktionen, bliver implementeret. For at vise hvordan den kan bruges med den udvidede model, med hensyn til test af forstyrrelse forsaget af måltider.

Den endelige konklusion på denne tesis er, at begge sammenkoblinger har prob-

lemer, men hvis man tager hensyn til problemerne,kan modellerne blive brugt til, at approksimativt simulere blod glukose-insulin systemet.

Acknowledgements

I would like to thank my supervisor John Bagterp Jørgensen, for guidance during the creation of this thesis and my family and girlfriend for support during the process.

Contents

Chapter 1

Introduction

1.1 Motivation

One of the biggest diseases in the western world today is diabetes. Many millions suffer from the disease and the number is growing. The grow is mostly due to the lifestyle in western world, with lots of unhealthy food. Because this is a large problem many researchers try to find methods for diagnosing and treating the disease. One of the approaches is to design a mathematical model describing the glucose-insulin system. Diabetes is a malfunction in exactly this system. These mathematical models can be used to diagnose, but also to create simulators to test different treatment types. One of the mathematical models describes the glucose-insulin system with a few number of parameters. This mathematical model is called Bergman's minimal model and was introduced in the eighties [13]. It is a model in two separate parts one describing the glucose kinetics and one describing the insulin kinetics. It is this model that will be described and analyzed in this thesis.

1.2 Problem Statement

In this thesis Bergman's minimal model's two parts will be analyzed for problems and possibilities. Modifications and additions will also be done. Then two coupled models will be proposed based on Bergman's minimal model. The original models and the coupled models will be implemented in simulators in order to show the functionalities and problems. All this is to find out whether or not the two coupled models could be used as trustworthy simulators for the blood glucose-insulin system. Such a simulator could be used for different purposes e.g testing of model predictive controllers [10]

1.3 Thesis Structure

Chapter 1-Introduction An introduction to the problems, which motivates this thesis.

Chapter 2-Bergman's Minimal Model A description of Bergman's Minimal Model and a proposing of two coupled models.

Chapter 3-U(t)-How it can be used A description of the possibilities with one of the coupled models. The one describing a type 1 diabetic. In this chapter a PID controller controlling the exogenous insulin infusion is described.

Chapter 4-Implementation Description of the implementation of the models in Matlab

Chapter 5-Simulations and Discussion Simulations showing the possibilities and problems of the different models. And a discussion about the use of the proposed coupled models.

Chapter 6-Conclusion Conclusion of the thesis

1.4 The Blood Glucose-Insulin System

The glucose-insulin system is an example of a closed-loop physiological system. A healthy person, normally has a blood glucose concentration at about $70 - 110 \frac{mg}{dL}$. The glucose-insulin system helps us to keep this steady state. In figure 1.1 a simple description of the system is shown. Most of the time a healthy person is in the green area, having normal blood glucose concentration.

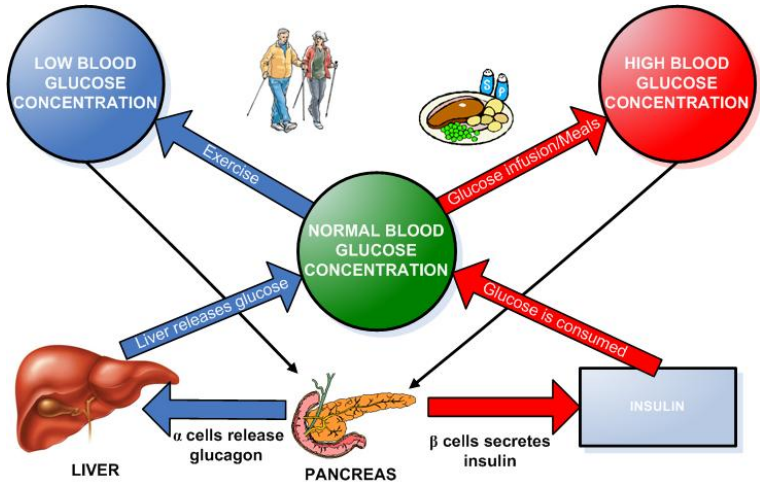

Figure 1.1: The blood glucose-insulin system

If the person then ingest additional glucose to the system e.g via a meal, the person moves to the red area, with a higher blood glucose concentration. When this happens a signal is sent to the pancreas, which β cells react by secreting the hormone insulin. This insulin increase the uptake of glucose by the cells, liver etc. and brings the person back in the green area. If the blood glucose concentration goes below the normal level, the person is in the blue area. This could happen as a response to exercise, which increase the glucose uptake. When the person is in the blue area with low blood glucose concentration a signal is also send to the pancreas. The pancreas α cells react by releasing glucagon. This glucagon affects the liver cells to release glucose in to the blood until the person is back in the green area again [2]. This is a very simple description of a more complicated system. But it is this simplistic way of explaining the metabolism, which will be presented in a mathematical model in this thesis.

1.5 Diabetes

Diabetes is a large problem today. According to the Diabetes Atlas 2003, 194 million people suffer from the disease. Diabetes is not a single disease, but actually many. The connection between the deceases is that they are caused by a disfunction in the blood glucose-insulin system. If not treated, diabetes can lead to heart diseases, blindness and other malfunctions. The two most frequently seen diabetes types are diabetes type 1 and diabetes type 2.

1.5.1 Diabetes Type 1

When you suffer from type 1, the β cells are destroyed by an auto-immune reaction in the body. This results in a very low insulin production (down to 10% of normal). At this production level the insulin cannot decrease the blood glucose level fast enough, when the person eats. The blood glucose rises even more because another missing function of insulin, namely the function to stop the production of glucagon, when the blood glucose level is high. All this results in a very high blood glucose level, if not treated. If the level goes beyond $180\frac{mg}{dL}$, some of the glucose is released with the urine. The symptoms of diabetes type 1 are tiredness, hunger and loss of weight. Today you treat this type of diabetes by injecting insulin into the body, by exercising and keeping a healthy diet. A person suffering from diabetes type 1, is dependent of getting insulin injected because nothing is secreted. Otherwise the person will die, because the body cannot handle the high glucose level. [5]

1.5.2 Diabetes Type 2

Diabetes type 2 is the most common type of diabetes. When you have this type of diabetes the pancreas is able to produce some insulin, and in some cases it can produce insulin as for a healthy person. The problem is that the insulin is not able to affect the cells, of the body to increase their uptake of glucose. Thus people suffering from type 2 diabetes are insulin resistant. Over time the number of β cells start to decrease, and then the type 2 diabetics should be treated with insulin injections like a type 1 diabetic. Type 2 almost also have the same symptoms as type 1 diabetes. [5]

These descriptions of the Diabetes types are simple, but actually it is complicated, to describe these diabetes types in a model as you will see in the simulation chapter.

1.5.3 Hyperglycemia

A person has hyperglycemia, when the blood glucose level is above $270\frac{mg}{dL}$. This can arise e.g. when a diabetic eats a large meal or has a low level of insulin in the blood. Hyperglycemia is extremely dangerous if not treated.

1.5.4 Hypoglycemia

A person has hypoglycemia when the blood glucose level is below $60\frac{mg}{dL}$. This can happen ex. after too much exercise, a too large insulin dosage, small amount of carbohydrates in the food or if the diabetic skips meals. Hypoglycemia can result in loosing of the conscience. Avoiding hypoglycemia is an important issue when you are using insulin as treatment.

1.5.5 Problems with Treatment

Today, as mentioned, you treat the insulin dependent patients, through exercise, a healthy diet and most important by injection of insulin. These injections are made with different devices, syringes, pumps etc. The problem with the devices today, is that they all have to be controlled by the patient. And the injections is not always done when actually needed. A perfect solution to the problem would be to create a treatment type, where the patient, doesn't have to think about having diabetes. This could be a self-regulated pump acting like an artificial pancreas.

1.6 Testing

Diabetes and other diseases caused by malfunctions in the glucose-insulin system, is one of the reasons that many mathematical models have been made over time to describe this dynamical system [7]. These mathematical models are based on and used to interpret tests. The models and tests, can help to improve the situation for many people suffering from diabetes.

1.6.1 The OGTT

One of the tests used is the Oral glucose tolerance test (OGTT) [15]. In this test the subject fast for an 8 hour period [1] after which the blood glucose and insulin concentrations are measured. Then the subject ingest glucose in a liquid solution orally. After this ingestion you take new measurements for a three hour period. The amount of glucose in the liquid is typically 75 g. From [15] the following interpretations of the test results is derived:

OGTT with a 75 g glucose drink (2 hours after ingestion)

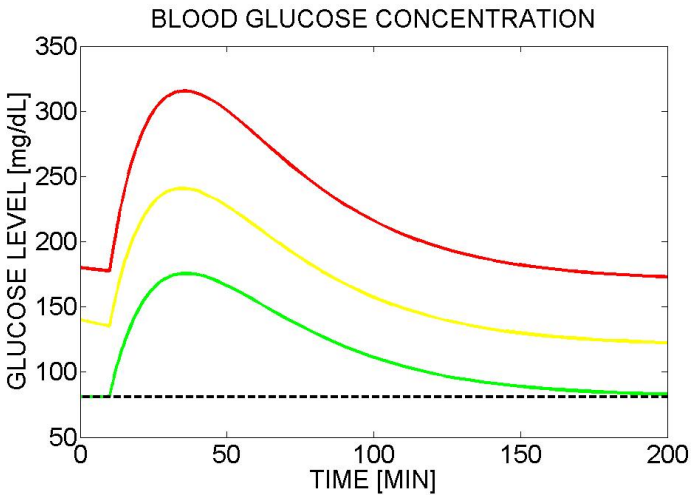

Figure 1.2: The pattern of an OGTT. Green graph: Normal glucose tolerance. Yellow graph: Pre-Diabetes. Red graph: Diabetes

Less than 140 mg/dL	Normal glucose tolerance
From 140 to 200 mg/dL	Pre-diabetes
more than 200 mg/dL	Diabetes

a graphical representation of these limits can be seen in figure 1.2

1.6.2 The IVGTT

Another test is the Intravenous Glucose Tolerance Test (IVGTT). Together with a mathematical model, this test can be used to estimate insulin sensitivity, S_I, glucose effectiveness, S_G, and the pancreatic responsiveness parameters ϕ_1 and ϕ_2 in a subject [11]. One of the mathematical models used to interpret the IVGTT is Bergman's minimal model introduced in the next chapter. Here you will also obtain information about the 4 parameters and how they are estimated.

The IVGTT test procedure begins with a injection of a glucose bolus intravenously, containing 0.30 g glucose pr. kg. bodyweight. Then you take blood samples frequently for a 3 hour period. These blood samples are analyzed and glucose and insulin levels are measured. A typical IVGTT for a normal subject,

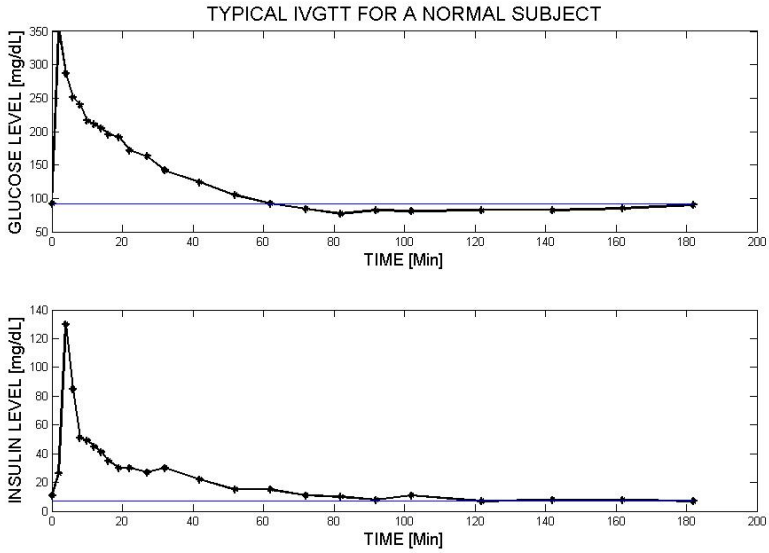

Figure 1.3: An IVGTT for a Normal subject

from studies by Bergman et al. [11] is shown in figure 1.3. As you can see the glucose level decays slowly to a minimum level below the basal value and then slowly reaches the basal value. The insulin peaks just after the injection, and then decays to a level above the baseline and then peaks a little again. finally it decays to the basal value. This is just a typical pattern and the glucose and insulin level may not behave exactly like this.

1.6.3 Fasting Blood Glucose

A third test, and a much easier test, is the fasting blood glucose. Here the subject/patient has to fast for a period of 8-10 hours, then a measurement of the glucose is made. [15]. The test results can be interpreted as:

From 70 to 99 mg/dL	Normal glucose tolerance
From 100 to 125 mg/dL	Pre-diabetes
more than 126 mg/dL	Diabetes

Chapter 2

Bergman's Minimal Model

2.1 Introduction to the Model

You can design very complicated models, with many parameters, to describe the glucose-insulin metabolism. But in many cases a simple model would be sufficient to make a good analysis. A simple method with few parameters, was introduced in the eighties by Richard N. Bergman and is called Bergman's minimal model [13] [7]. The model has been modified and examined many times. In this chapter the evolution of the minimal model will be described, and two models based on Bergman's minimal model will be introduced.

2.2 The Model

Bergman's minimal model is a one compartment model, meaning that the body is described as a compartment/tank with a basal concentration of glucose and insulin. The minimal model actually contains two minimal models. One describing glucose kinetics, how blood glucose concentration reacts to blood insulin concentration and one describing the insulin kinetics, how blood insulin concentration reacts to blood glucose concentration. The two models respectively take insulin and glucose data as an input. The two models have mostly been used to interpret the kinetics during the IVGTT test, and in their original form they cannot be used to much else [7], but with small additions or modifications they can also be used to describe meals and exogenous insulin infusion [10]. In this section a description of the two kinetics are done and finally two couplings are proposed, which could be used as simulators of the entire blood glucose-insulin system.

2.2.1 The Glucose Minimal Model

The original glucose minimal model describes how the glucose level behaves according to measured insulin data during an IVGTT. The model is a one compartment model divided into two parts. The first part is the main part describing the glucose clearance and uptake. The second part describes the delay in the active insulin I_2 which is a remote interactor which level affects the uptake of glucose by the tissues and the uptake and production by the liver. These two parts are described mathematically by two differential equations namely [7]:

$$\frac{dG(t)}{dt} = -(p_1 + X(t))G(t) + p_1 G_b \qquad G(0) = G_0 \qquad (1)$$

$$\frac{dX(t)}{dt} = -p_2 X(t) + p_3(I(t) - I_b) \qquad X(0) = 0 \qquad (2)$$

The best way to describe the meaning of these equations is to show how they are derived. A description of the parameters and the terms of the equations is then easier understood. The derivation is based on the description of the model by Steil et al. [8] and the rule of mass balances:

$$accumulated = in - out + generated - consumed$$

Such a derivation will be done in the next subsection. In the derivation the following parameters will be used:

Parameter	Unit	Description
t	[min]	Time
G(t)	[mg/dL]	Blood glucose Concentration
G_b	[mg/dL]	Steady state blood glucose concentration (baseline)
I_2(t)	[mU/L]	Active insulin concentration
X(t)	[1/min]	The effect of Active insulin.
I(t)	[mU/L]	Blood insulin concentration
I_b	[mU/L]	Steady state blood insulin concentration (baseline)
V_G	[dL]	Volume of the glucose compartment
V_{I_2}	[L]	Volume of the remote pool
Q_{G1}	[dL/min]	flow
Q_{G2}	[dL/min]	flow
Q_{I_21}	[L/min]	flow
Q_{I_22}	[L/min]	flow
w_1	$[dl^2/(min \cdot mU]$	effect factor
w_2	$[dl^2/(min \cdot mU]$	effect factor

2.2.1.1 Deriving the Model

The model is represented as a compartment/tank with a volume V_G. See figure 2.1. The glucose flows in and out of this compartment at a steady rate, resulting in a basal concentration G_b. However this steady state can be changed when ex. a bolus of glucose is injected. By using the rule of mass balances it is possible to describe what happens in this compartment mathematically. The accumulated part for the glucose compartment is the difference between the initial and final mass:

$$accumulated = V_G \cdot G(t_0 + \Delta t) - V_G \cdot G(t_0)$$

The income of glucose by the bolus is given by the initial condition G(0). As you can see on figure 2.1 there are two types of outgoing mass, namely uptake by the

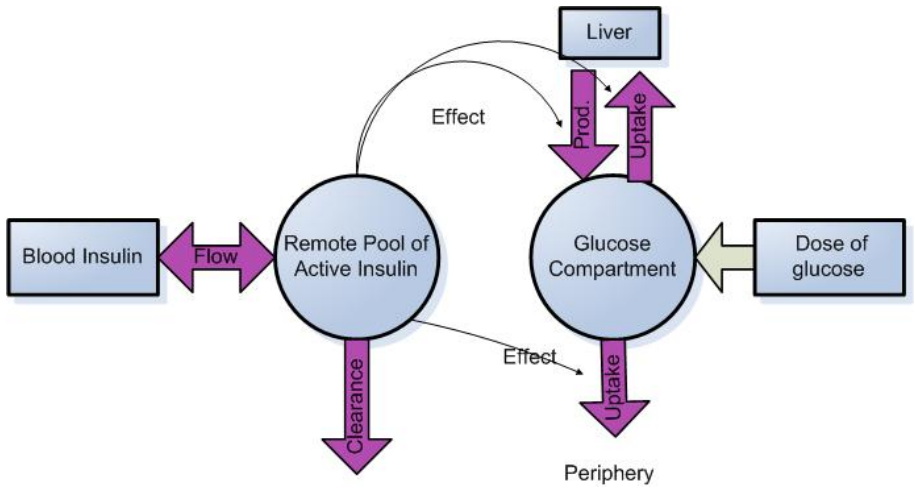

Figure 2.1: Graphical representation of the minimal glucose model

liver and uptake by the peripheral. There is one type of in-going mass (besides from the bolus) namely the production of glucose by the liver. This results in a 'in' and 'out' part determined by the threshold G_b basal glucose concentration. This basal concentration is according to Steil et al. [8], given by the difference between glucose and insulin independent production $prod_{gluinsind}$ and uptake $upt_{gluinsind}$. When the blood glucose level is above this basal concentration, the glucose disappears by uptake by the liver (upt_l) and the peripheral tissues (upt_p). But when the glucose level is below the basal concentration the liver produces glucose until the basal level is reached. This balance between the production and uptake by the liver, is referred to by Steil et al. [8] as NHGB (Net Hepatic Glucose Balance). Both the NHGB and the uptake by the peripheral tissues, can be enhanced by insulin. It is however not the blood insulin concentration that gives this effect directly, but the so-called active insulin placed in a remote pool. The NHGB and the uptake by the peripheral are given by:

$$upt_p = (Q_{G1} \cdot G(t) \cdot \Delta t + G(t) \cdot k \cdot w_1 \cdot I_2(t) \cdot \Delta t) + upt_{gluinsind}$$

$$NHGB = prod_{gluinsind} - (Q_{G2} \cdot G(t) \cdot \Delta t + G(t) \cdot k \cdot w_2 \cdot I_2(t) \cdot \Delta t)$$

k is just a constant changing L to dl, so this is set to 1. All this can be inserted in to the rule of mass balances

$$accumulated = in - out + generated - consumed \Leftrightarrow$$

$$accumulated = NHGB - upt_p \Leftrightarrow$$

$$V_G \cdot G(t_0 + \Delta t) - V_G \cdot G(t_0) = prod_{gluinsind} - ((Q_{G2} \cdot G(t) \cdot \Delta t$$

$$+ w_1 \cdot I_2(t) \cdot G(t) \cdot \Delta t) + (Q_{G1} \cdot G(t) \cdot \Delta t + w_2 \cdot I_2(t) \cdot G(t) \cdot \Delta t))$$

$$- upt_{gluinsind}$$

Steil et. al. [8] argues that the insulin/glucose independent terms are given by

$$prod_{gluinsind} - upt_{gluinsind} = Q_{G1} \cdot G_b \cdot \Delta t + Q_{G2} \cdot G_b \cdot \Delta t$$

This is what gives the threshold G_b. According to Cobelli et. al [12] this causes some problems, described later. By inserting this term in the rule of mass balance, you get the following

$$accumulated = in - out + generated - consumed \Leftrightarrow$$

$$accumulated = NHGB - upt_p \Leftrightarrow$$

$$V_G \cdot G(t_0 + \Delta t) - V_G \cdot G(t_0) = (Q_{G1} \cdot G_b \cdot \Delta t + Q_{G2} \cdot G_b \cdot \Delta t)$$

$$+ (Q_{G2} \cdot G(t) \cdot \Delta t + w_2 \cdot I_2(t) \cdot G(t) \cdot \Delta t)$$

$$+ (Q_{G1} \cdot G(t) \cdot \Delta t + w_1 \cdot I_2(t) \cdot G(t) \cdot \Delta t)$$

by dividing by Δt and V_G the following term is derived:

$$\frac{G(t_0+\Delta t)}{\Delta t} = \frac{Q_{G1}}{V_G}G_b + \frac{Q_{G2}}{V_G}G_b - (\frac{Q_{G1}}{V_G}G(t) + \frac{w_1}{V_G}G(t)I_2(t) + \frac{Q_{G2}}{V_G}G(t)$$

$$+ \frac{w_2}{V_G}G(t)I_2(t))$$

By setting $\frac{Q_{G1}}{V_G} = k_1, \frac{w_1}{V_G} = k_4$, $\frac{Q_{G2}}{V_G} = k_5$, $\frac{w_2}{V_G} = k_6$ and going to the limit $\Delta t \to 0$ it gives the following differential equation:

$$\frac{dG(t)}{dt} = k_1G_b + k_5G_b - (k_1G(t) + k_4I_2(t)G(t)) - (k_5G(t) + k_6I_2(t)G(t)$$

Now a differential equation is derived, but the active insulin I_2 is delayed during the transport across the capillaries, and the above differential equation does not take this into account. So a mathematical expression must be derived for this delay. The active insulin is in a remote pool. There is an outflow and a inflow which can be applied to the rule of mass balance. The accumulated part is given by:

$$accumulated = V_{I_2} \cdot I_2(t_0+\Delta t) - V_{I_2} \cdot I_2(t_0)$$

the 'in' and 'out' flow must again be considered together. When blood insulin concentration I(t) is above its basal value I_b, insulin flows into the remote pool. If the blood insulin concentration goes below its basal value insulin flows out of the remote pool, this is referred to as $balance_{ins}$. Another clearance $a_{clearance}$ from the remote pool is the one that is proportional to the level of active insulin $I_2(t)$. When the active insulin level in the remote pool rises this clearance rate also rises. all this can be formulated in the rule of mass balances as:

$$accumulated = in + out + generated - consumed \Leftrightarrow$$

$$accumulated = balance_{ins} - a_{clearance} \Leftrightarrow$$

$$V_{I_2} \cdot I_2(t_0 + \Delta t) - V_{I_2} \cdot I_2(t_0) = (Q_{I_2 1} \cdot (I(t) - I_b)\Delta t)$$

$$- (Q_{I_2 2} \cdot I_2(t)\Delta t)$$

by dividing by Δt and V_{I_2} and then going to the limit $\Delta t \to 0$ you get the following differential equation:

$$\frac{dI_2(t)}{dt} = -\frac{Q_{I_2 2}}{V_{I_2}} I_2 + \frac{Q_{I_2 1}}{V_{I_2}} (I(t) - I_b)$$

by setting $\frac{Q_{I_2 1}}{V_{I_2}} = k_2$ and $\frac{Q_{I_2 2}}{V_{I_2}} = k_3$ you get:

$$\frac{dI_2(t)}{dt} = -k_3 I_2 + k_2 (I(t) - I_b)$$

This describes the delay in the change of $I_2(t)$ but instead of using this in the mathematical expression, X(t), is introduced, which describes the effect of $I_2(t)$. This is created by setting $X(t) = (k_4 + k_6) I_2(t) \Leftrightarrow I_2(t) = \frac{X(t)}{k_4 + k_6}$. By inserting these in the two derived differential equations you get:

$$\frac{dG(t)}{dt} = -(k_1 + k_5 + X)G(t) + (k_1 + k_5)G_b$$

$$\frac{dX(t)}{dt} = -k_3 X(t) + k_2 (k_4 + k_6)(I(t) - I_b)$$

This model is described graphically in figure 2.2. The parameters are given in the following table:

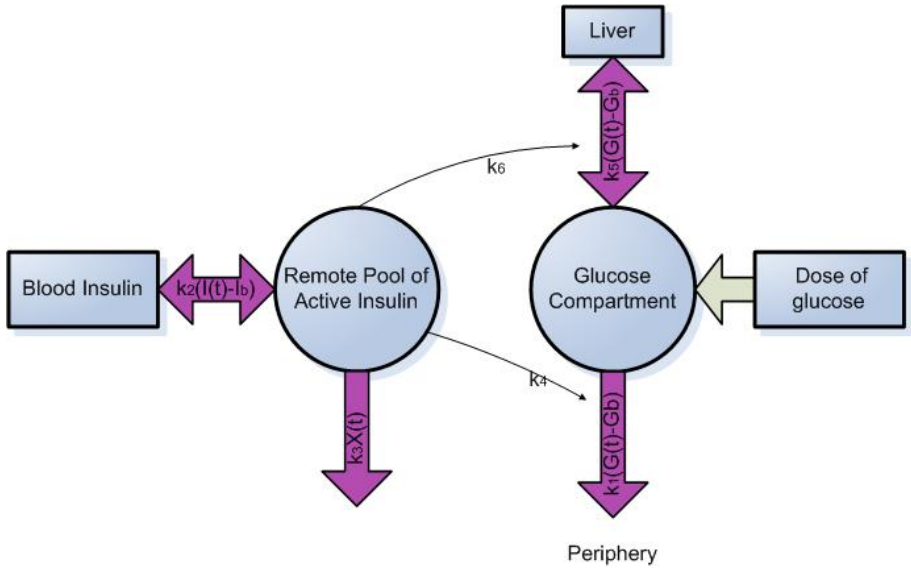

Figure 2.2: The minimal model describing glucose kinetics

Parameter	Unit	Description
k_1	[1/min]	Glucose ability to increase uptake by the peripheral
k_2	[1/min]	Insulin transport rate to remote pool
k_3	[1/min]	Rate of clearance of active insulin
k_4	[L/(min · mU)]	Active insulin effect on uptake by the peripheral
k_5	[1/min]	Glucose ability to change NHGB
k_6	[L/(min · mU)]	Active insulin effect on NHGB

However the model still does not look like (1) and (2) introduced in the beginning of this section. But by setting $p_1 = k_1 + k_5$, $p_2 = k_3$ and $p_3 = k_2(k_4 + k_6)$ (1) and (2) are achieved.

2.2.1.2 Glucose Effectiveness and Insulin Sensitivity

The minimal glucose model has mostly been used to interpret the IVGTT (see chapter 1). When you interpret you measure insulin levels during the test and use them as input in the glucose minimal model. Then via this model and a parameter estimation e.g. weighted non-linear least squares [11], two important

parameters: Glucose effectiveness and insulin sensitivity can be derived. The glucose effectiveness is defined by Gaetano et al. [7] as the insulin independent uptake rate and Cobelli et al. describes it as glucose ability to promote its own disposal. In the glucose minimal model the glucose effectiveness S_G is given by:

$$S_G = p_1$$

because p_1 is the sum of the two uptake rates k_1 and k_5, which are independent of X(t) but dependent on G(t). Insulin sensitivity is defined by Cobelli et al. [12] as the ability of insulin to enhance glucose effectiveness. To find an expression for insulin sensitivity in the glucose minimal model you must keep the effect X(t) on a steady state [13]. This results in:

$$\frac{dX(t)}{dt} = -p_2 X(t) + p_3(I(t) - I_b) = 0 \Leftrightarrow$$

$$X(t) = \frac{p_3}{p_2}(I(t) - I_b)$$

and by inserting this into (1) you get:

$$\frac{dG(t)}{dt} = -(p_1 + \frac{p_3}{p_2}(I(t) - I_b))G(t) + p_1 G_b$$

From this you can see that the ability of insulin to enhance the glucose effectiveness p_1 is given by $S_I = \frac{p_3}{p_2}$. Normal values for these parameters, when using the glucose minimal model to interpret are approximately in the interval $[4 \cdot 10^{-4}, 8 \cdot 10^{-4}]$ $\frac{L}{min \cdot mU}$ for S_I [11] and $[0.01, 0.03]$ $\frac{1}{min}$ for glucose effectiveness.

2.2.1.3 Additions to the Model

To increase the functionalities of the glucose minimal model, thus it could be used to simulate more than an IVGTT, some additions could be done. One of the additions is a function describing what a meal would do to the glucose level. This is done by adding a meal disturbance term D(t) to (1), so it looks like the following:

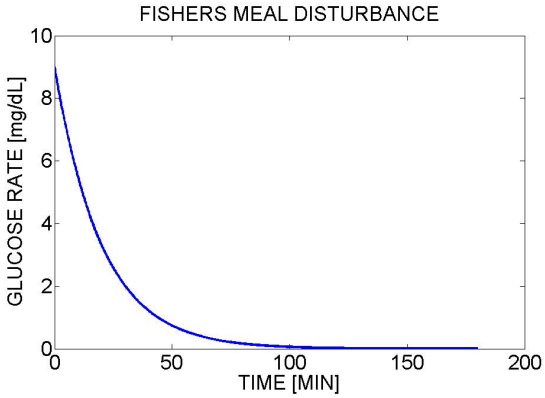

Figure 2.3: D(t) in Fishers form with drate = 0.05 and B=9

$$\frac{dG(t)}{dt} = -(p_1 + X(t))G(t) + p_1 G_b + D(t)$$

D(t) is the rate of mg glucose pr. dL entering the blood. This process of meal absorbtion needs a description. A simple description of this was suggested by Fisher [6] and looks like this:

$$D(t) = B \cdot exp(-drate \cdot t)$$

He suggested that the meal absorbtion description should be a function which rapidly increases after the meal, and then decays to 0 in 2-3 hours. If you use the values $B = 9$ and $drate = 0.05$ it gives the graph in figure 2.3.

Modeling with t is however not suitable, so instead of using the actual function a differentiation of it is done, and used instead. This results in the following differential equation

$$\frac{dD(t)}{dt} = -drate \cdot D(t)$$

To use this differential equation in a modelbased simulator, it must be used as a time-event. Time-events will be explained in the implementation chapter, but

basically during a simulation the simulation is stopped at a certain time, and the equation is given an initial value corresponding to B in the function. It is important to remember that this is also distributed into the compartment with volume V_G this means that in order to attain the rate in $\frac{mg}{min}$ you must multiply the function with this volume.

Another addition which could be done is a description of the glucose level in the subcutaneous layer. When you measure blood glucose concentration you often get the measurements from the subcutaneous layer. To make good comparisons to measurements, the function $G_{sc}(t)$ is introduced, it describes the glucose concentration in the subcutaneous layer. A differential equation describing the behavior of this function is introduced here:

$$\frac{dG_{sc}(t)}{dt} = \frac{G(t) - G_{sc}(t)}{5} - R_{utln} \qquad G_{sc}(0) = G(0) - 5 \cdot R_{utln}$$

This equation models a 5 min. first-order lag between the blood glucose concentration and the subcutaneous glucose concentration. The R_{utln}, Rate of utilization, is the difference between the two concentrations in the steady state. [10]. One of the major problems concerning creating a artificial pancreas, is the delay between these two concentrations.

2.2.1.4 Problems with the Model

Recent studies by Cobelli et al [12] [4] [3] has shown that when using the minimal glucose model to interpret an IVGTT, it overestimates S_G and underestimates S_I, when an insulin response is present. Steil et. al [8] gives a graphical presentation of this problem, and this will also be done in the simulation chapter of this thesis. Cobelli et al. [12] argues that the problem is due to the minimal model not being able to distinguish between the glucose effect on its own disposal (glucose effectiveness) and the total plasma clearance rate which decreases as the G(t) increases. They present a solution to the problem by introducing an extra non-accessible glucose compartment to the minimal model. Thus they can distinguish between glucose effectiveness and the plasma clearance rate. This model is not analyzed in this thesis, however it has proven to give a more accurate S_G and S_I. A result derived in [3] with the 2-compartment minimal model of glucose is that S_G is approximately 60% lower and S_I is approximately 35% higher than the values attained during an IVGTT using the one-compartment minimal model of glucose.

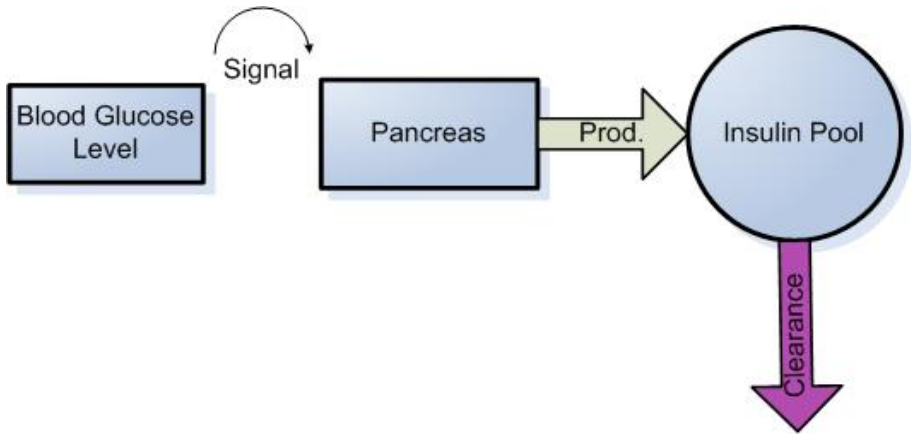

Figure 2.4: Graphical representation of the insulin minimal model

2.2.2 The Insulin Minimal Model

Now the model describing glucose kinetics as a product of insulin data input has been described. But a description of the insulin kinetics is missing. Bergman et. al [13] presented the following minimal model of insulin kinetics, represented here by the differential equation:

$$\frac{dI(t)}{dt} = p_6[G(t) - p_5]^+ t - p_4[I(t) - I_b] \qquad I(0) = I_0 \qquad (3)$$

Like the glucose model, this insulin model is used to interpret the IVGTT. The graphical representation of the model can be seen in figure 2.4, and like with the glucose minimal model, a derivation, based on the rule of mass balances,of the model is used to describe it. The derivation is based on assumptions by Gaetano et al. [7] and Bergman et al. [13]. The parameters used are:

Parameter	Unit	Description
I(t)	[mU/L]	Blood insulin concentration
I_b	[mU/L]	Basal blood insulin concentration
G(t)	[mg/dL]	Blood glucose concentration
p_5	[mg/dL]	Threshold for blood glucose concentration
V_I	[L]	Volume of insulin distribution pool
Q_{I1}	[L/min]	flow
Q_{I2}	$[\frac{mUdL}{mgmin}]$	flow

The accumulated part is the difference between the initial and the final blood insulin mass:

$$accumulated = V_I \cdot I(t_0 + \Delta t) - V_I \cdot I(t_0)$$

In a non type 1 diabetic subject, which this model can be used to describe, the pancreas is the source of insulin. In a healthy person a small amount of insulin is always created and cleared [5]. This helps to keep the basal concentration I_b. The glucose independent production and the clearance of insulin is proportional to the blood insulin concentration. If the insulin level is above basal concentration the clearance increases, if the insulin level is below basal concentration the basal production increases. When the glucose level gets high the pancreas reacts by releasing more insulin at a certain rate. To explain this mathematically you have to derive a mathematical function describing the reaction of the pancreas. This function is derived by Bergman et al. and adjusted by Gaetano et al. [13][7] to become $Pancreas(t) = [G(t) - p_5]^+ \cdot t$, in which $[G(t) - p_5]^+$ is a term which has the value $G(t) - p_5$ when positive and 0 when negative. So p_5 is the limit deciding when the pancreas should produce more insulin and when to stop. And the difference between $G(t) - p_5$ determines how much it should produce. The downside about this function is that it is very attached to the IVGTT. As you can see on figure 1.3 the insulin during an IVGTT respond in two peaks. The first peak is not described by this pancreas function but should be given as the initial value of the insulin concentration $I(0)$. The pancreas function describes the second peak. The multiplying by t is described by Gaetano et al. [7] as caused by the pancreas response being proportional not only to the hyperglycemia attained but also to the time elapsed from the glucose stimulus. By inserting the basal production/clearance term and the pancreas function as the 'in-out' part in the rule of mass balances you get:

$$accumulated = in - out + generated - consumed \Leftrightarrow$$

$$accumulated = Pancreas(t) + (Prod_{basal} - clearance) \Leftrightarrow$$

$$V_I \cdot I(t_0 + \Delta t) - V_I \cdot I(t_0) = (Q_{I2} \cdot [G(t) - p_5]^+ \cdot t \cdot \Delta t) - (Q_{I1} \cdot (I(t) - I_b)\Delta t)$$

Then by dividing by V_I and Δt and going to the limit $\Delta t \to 0$ the following differential equation is derived:

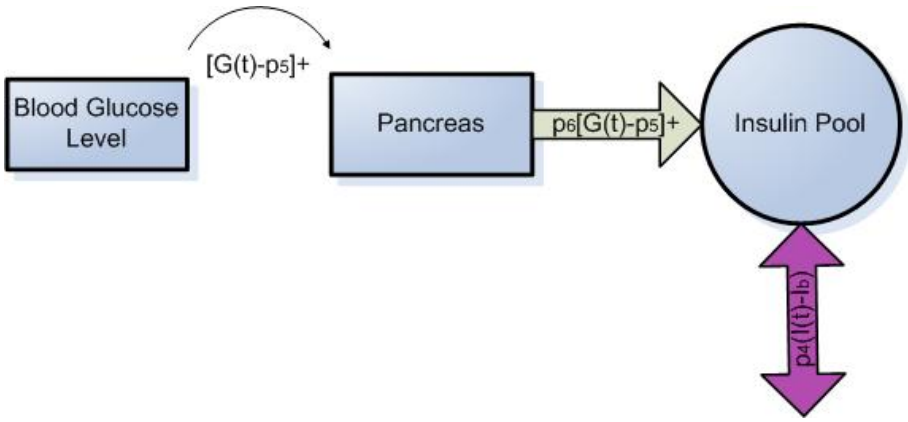

Figure 2.5: The Insulin Minimal Model

$$\frac{dI(t)}{dt} = \frac{Q_{I2}}{V_I}[G(t) - p_5]^+ \cdot t - \frac{Q_{I1}}{V_I}(I(t) - I_b)$$

Setting $p_6 = \frac{Q_{I2}}{V_I}$ and $p_4 = \frac{Q_{I1}}{V_I}$ you have the equation (3). This is described graphically in figure 2.5.

2.2.2.1 Pancreatic Response

The interpretation you can derive by using this original insulin minimal model together with an IVGTT, are the pancreatic response parameters ϕ_1 and ϕ_2. They describe the sensitivity of the pancreas at the first peak and at the second peak respectively, and are given by [11]:

$$\phi_1 = \frac{I_{max} - I_b}{p_4 \cdot (G_0 - G_b)} \qquad \phi_2 = p_6 \cdot 10^4$$

Normal values for these according to Bergman and Pacini derived in healthy subjects using an IVGTT is in the interval $2 - 4\frac{mUdLmin}{Lmg}$ for ϕ_1 and $20 - 35\frac{mUdl}{mgminL}$ for ϕ_2. In studies by Bergman et al. [14] they found an expression for insulin tolerance in the minimal model namely $\phi_2 \cdot S_I$. If this was lower

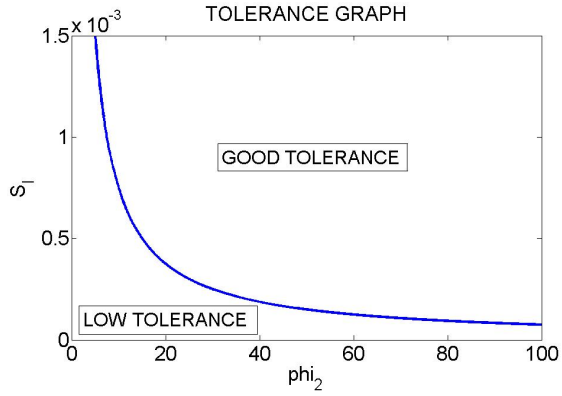

Figure 2.6:

than $75 \cdot 10^{-4}$ the person was low tolerant (see figure 2.6). Also the glucose effectiveness was lower, and this could be due to the problems with the glucose minimal model described earlier.

2.2.2.2 A Modification

The original insulin minimal model is very attached to the IVGTT, which makes it good for interpreting this, but bad for other purposes. One of these purposes could be to describe the insulin kinetics for a type 1 diabetic with no endogenous insulin production. This could be done by exchanging the incoming part/the pancreas with a function U(t) describing exogenous or endogenous insulin infusion [10], [6]. Then the differential equation would look like the following:

$$\frac{dI(t)}{dt} = -p_4 I(t) + \frac{U(t)}{V_I}$$

With this modification it is possible to describe the kinetics for a type 1 diabetic on different treatment types. e.g. a pump.

2.2.3 Coupling the Minimal Models

Now the two minimal models describing respectively glucose kinetics and insulin kinetics have been presented and modified. These are normally used indepen-

dently with glucose data and insulin data respectively to estimate the parameters, and interpret an IVGTT. A coupling of the two parts would result in a model describing the whole Blood glucose-insulin system. Two different couplings are introduced here. A coupling between the original minimal models, and a coupling between the modified minimal models.

2.2.3.1 The Original Model

The first coupling proposed is a coupling between the original minimal models, without additions and/or modifications. From now on this coupled model, will be called the original model. The model is represented by the following differential equations:

$$\frac{dG(t)}{dt} = -(p_1 + X(t))G(t) + p_1 G_b \qquad G(0) = G_0 \qquad (1)$$

$$\frac{dX(t)}{dt} = -p_2 X(t) + p_3(I(t) - I_b) \qquad X(0) = X_0 \qquad (2)$$

$$\frac{dI(t)}{dt} = p_6[G(t) - p_5]^+ t - p_4[I - I_b] \qquad I(0) = I_0 \qquad (3)$$

With the parameters:

Parameter	Unit	Description
G(t)	[mg/dL]	Blood glucose concentration
X(t)	[1/min]	The effect of active insulin
I(t)	[mU/L]	Blood insulin concentration
G_b	[mg/dL]	Basal blood glucose concentration
I_b	[mU/L]	Basal blood insulin concentration
p_1	[1/min]	Glucose clearance rate independent of insulin
p_2	[1/min]	Rate of clearance of active insulin (decrease of uptake)
p_3	[L/($min^2 mU$)]	Increase in uptake ability caused by insulin.
p_4	[1/min]	decay rate of blood insulin.
p_5	[mg/dL]	The target glucose level
p_6	$[\frac{mUdL}{Lmgmin}]$	Rate of pancreatic release after glucose bolus

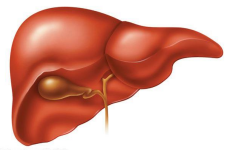

Figure 2.7: The liver, one of the main actors in the glucose-insulin system

This model is suited for simulating different IVGTT tests, for healthy and for glucose resistant(type 2 diabetics) subjects. Studies by Gaetano et al. have shown that a coupling between these original minimal models, does not allow a steady state for $p_5 < G_b$ [7]. This is easy to prove, because in a steady state $\frac{dG(t)}{dt} = \frac{dX(t)}{dt} = \frac{dI(t)}{dt} = 0$. This is only possible when $G(t) = G_b$,$X(t) = 0$ and $I(t) = I_b$. But for $p_5 < G_b$ the term $p_6[G_b - p_5]+ > 0$ This means that a steady state(equilibrium) can not be attained. Besides this problem, this coupling also adopts the problem with the glucose minimal model.

2.2.3.2 The Modified Model

The second coupling proposed is a coupling between the minimal models with modifications and additions. This coupled model will be referred to as the modified model. This model contains the following differential equations:

$$\frac{dG(t)}{dt} = -(p_1 + X(t))G(t) + p_1 G_b + D(t) \qquad G(0) = G_0$$

$$\frac{dX(t)}{dt} = -p_2 X(t) + p_3(I(t) - I_b) \qquad X(0) = X_0$$

$$\frac{dI(t)}{dt} = -p_4 I(t) + \frac{U(t)}{V_I} \qquad I(0) = I_0$$

$$\frac{dD(t)}{dt} = -drate \cdot D(t) \qquad D(0) = D_0$$

$$\frac{dG_{sc}(t)}{dt} = \frac{G(t) - G_{sc}(t)}{5} - R_{utln} \qquad G_{sc}(0) = G_0 - 5 \cdot Rut_l n$$

With the parameters given in the following table:

Parameter	Unit	Description
G(t)	[mg/dL]	Blood glucose concentration
X(t)	[1/min]	The effect of active insulin
I(t)	[mU/L]	Blood insulin concentration
D(t)	[mg/dL/min]	Meal disturbance function
Gsc(t)	[mg/dL]	Subcutaneous glucose concentration
U(t)	[mu/min]	exogenous insulin
G_b	[mg/dL]	Basal blood glucose concentration
I_b	[mU/L]	Basal blood insulin concentration
V_I	[L]	Volume of insulin distribution pool
p_1	[1/min]	Glucose clearance rate independent of insulin
p_2	[1/min]	Rate of clearance of active insulin (decrease of uptake)
p_3	[L/(min^{2}mU)]	Increase in uptake ability caused by insulin.
p_4	[1/min]	decay rate of blood insulin.
drate	[1/min]	decay rate of the meal disturbance

This model could be used to simulate the glucose-insulin system for a type 1 diabetic on treatment. The model is not attached to a single type of test. thus it has more possibilities concerning simulations of meal disturbance and insulin injections. It can be used to test model predictive controllers [10]. And this could make it a tool in the search of an artificial pancreas. This model also adopts the problem with the glucose minimal model.

2.3 Open-loop,Closed-loop and Semiclosed-loop models

In a mathematical model like the coupled model, you can distinguish between open-loop, closed-loop and a semiclosed-loop models. In the open-loop model there is no connection between the glucose and the insulin compartment. An open-loop model could describe a diabetic injecting a predetermined amount of insulin at certain times, where the injections not are based on the glucose level. The modified model in its pure form is an open loop model. In a closed-loop model there is a a full loop connection. The original model is an example of a closed loop model. At last there is a semi-closed model. In a semiclosed model the loop is not constant. An example of this could be if the U-term in

the modified minimal model was decided by a measurement of $G(t)$ at a certain time, and then was given a value for a certain period, until the next measurement was made. A Semiclosed-model using the modified model is described by M.E Fisher [6].

Chapter 3

$U(t)$-How it can be used

3.1 The U(t) function

One of the possibilities with the modified model, is to analyze/test different infusion ideas. You can do this through the exogenous insulin term U(t). In this chapter some of the possibilities with this U-term is described.

3.2 The Open Loop Model

The easiest way to use the infusion term U(t), is to use it to describe insulin injections, or to give it a constant value. In this way you can analyze how a day could look like for a Type 1 diabetic, using injections as treatment. You could also analyze how the blood glucose and insulin levels reacts to the injections.

3.3 The Closed Loop Model

Many researchers are working on how to create an insulin pump, which works as an artificial pancreas. Such a pump would have to react to the glucose level. You could call the pump a controller, because it controls the glucose level by manipulating the insulin dosage. By using the U-term in the modified minimal model, it is possible to implement such a controller and test it. This would turn the open-loop coupled model into a closed-loop model. In this section a controller will be described. This is just to show how a controller could be used together with the modified model. Thus this is not an attempt to create a perfect controller.

3.3.1 Introduction to the PID controller

In many processes, especially industrial, controllers are used to keep some kind of a steady state. This could be a tank filled with water, where you want to keep the water at a certain level. One of the frequently used controllers are the PID controller [9]:

$$u(t) = K_c[e(t) + \frac{1}{T_i}\int^t e(\tau)d\tau + T_d\frac{de(t)}{dt}] = P + I + D$$

This controller looks at the error $e = u_c - y$, where u_c is the setpoint, and y is the measured value. The controller consists of three control elements. P,the proportional part, I, the integral part, and D, the derivative part.

3.3.1.1 P

The proportional part of the controller $K_c e(t)$ is the one that increases or decreases u proportional to the error e. K_c is a constant known as proportional gain. proportion of the error e, u is changed. One way to estimate this constant is to use:

$$K_c = \frac{u_{max} - u_{min}}{p_B}$$

where u_{min} and u_{max} is the limitations of the output u, and p_B is the proportional band.

3.3.1.2 I

$K_c \frac{1}{T_i} \int^t e(t)d\tau$, the integral part of the controller, is also called the reset term. This part helps to obtain the steady state value automatically. The constant T_i is called the reset time.

3.3.1.3 D

The last term $K_c T_d \frac{de(t)}{dt}$ is the derivative part. This part you could also call the predictor. This part of the controller predicts what is happening next, and controls the output according to this. The constant T_d is derivative time.

3.3.2 Using the Controller

If you want to use the controller with a certain problem, you have to implement it in Matlab. One way to do this is by finding the transfer-function of the PID controller, to get the time response:

$$L(s) = K_c(1 + \frac{1}{T_i s} + T_d s)$$

However the derivative should not be implemented. instead of implementing this you approximate the $T_d s$ part with $T_d s \approx \frac{T_d s}{1+\frac{T_d s}{N}}$. This gives the following transfer-function:

$$L(s) = K_c(1 + \frac{1}{T_i s} + \frac{T_d s}{1 + \frac{T_d s}{N}}) \qquad \Leftrightarrow$$
$$L(s) = K_c \frac{T_i T_d (1 + \frac{1}{N})s^2 + (T_i + \frac{T_d}{N})s + 1}{T_i(\frac{T_d}{N}s^2 + T_d s)}$$

From this transfer function you can do a realization(going to the state-space domain) and obtain A,B,C and D in the system:

$$\frac{dx(t)}{dt} = Ax(t) + Be(t)$$

$$y = Cx(t) + De(t)$$

Which is easy to implement in e.g. Matlab using the Matlab function ssdata (see implementation chapter).

3.3.3 Controlling U(t)

The function of the controller could be to keep the blood glucose concentration at a steady state. In [10] a range of 60 mg/dL - 180 mg/dL is suggested. Thus a setpoint in that range could be chosen. Thus is the error term e(t) given by $e(t) = G(t) - setpoint$. When this error is zero U(t) should have a basal value, describing the normal flow of insulin into the I compartment in a healthy person. From the original model, you can see that this normal value is $U(t) = V_I p_4 \cdot$Normal insulin basal level. This normal basal level, is the amount, which makes the glucose stay at the basal concentration, when the error term is zero. This would be I_b for a healthy person. In this way you get a closed-loop model, which can be used in an attempt to describe the glucose-insulin metabolism in a diabetic with an artificial pancreas. The two new equations implemented in the model are:

$$U(t) = |V_I p_4 I_{b_{normal}} + Cx(t) + D(G(t) - setpoint)|$$

$$\frac{dx(t)}{dt} = Ax + B(G(t) - setpoint)$$

where $x(t) = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$, $A = \begin{pmatrix} A_1 & A_2 \\ A3 & A4 \end{pmatrix}$, $B = \begin{pmatrix} B_1 \\ B_2 \end{pmatrix}$, $C = \begin{pmatrix} C_1 & C_2 \end{pmatrix}$ and D is a scalar. Then the model is ready. But you still need to fit the parameters K_c,T_i and T_d in order to control U(t), almost as a real pancreas would do. This process is called tuning and this will be examined in the next section.

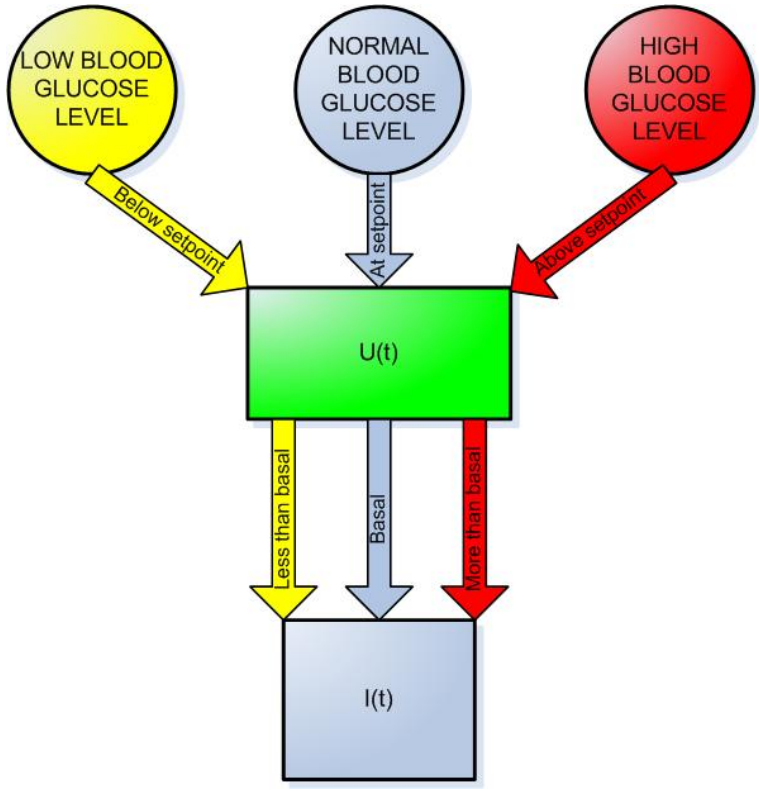

Figure 3.1: A graphical description of how the PID controller could work

3.3.4 Tuning the PID Controller, to the Model

The aim of the controller,is to act like an artificial pancreas, in order to do this you need to tune the parameters K_c, T_i and T_d. There are several methods you can use to tune the PID controller. Empirical tuning, where you adjust the parameters according to the output until you reach a good result, and tuning via a mathematical model. Before you tune the model, you should decide how it should work. What is the maximum and minimum glucose level allowed, how fast should the controller bring down the glucose level. And this brings another question namely what is the maximum amount of insulin possible to infuse. Questions like these must be answered before a tuning.

3.3.5 Using Other Controllers

Here a PID controller was chosen, to create a closed-loop model. But this is just to show how it could be done. A much better controller could be made, which could predict and control the insulin injections much better. This was just to show one of the possibilities with the model.

3.4 The Semi-Closed Loop Model

In a closed-loop model, the $U(t)$ reacts instantly every time something happens to the blood glucose concentration. In a semi-closed model, a reading is made of the blood glucose concentration at pre-decided times. This could be every 3rd hour. Then the size of U(t) is determined by the size of this reading. Examples of such controllers are described by Fisher [6].

Chapter 4

Implementation in Matlab

4.1 Introduction

In this section a short description of the different implementations in Matlab are given. In the first sections the general issues concerning implementing ODE-simulators are described. Then each of the simulators used in this thesis are given a short description. One of the simulators, namely the one based on the modified model, is given a graphical user interface. This simulator can be used to simulate many of the problems and possibilities with the minimal model, especially concerning meals and insulin infusion.

4.2 Choice of Solver

The solver used in the simulators is the ODE15s. Which has the following call:

```
[T,Y] = ode15s(odefun,tspan,y0,options)
```

odefun is the function containing the system,tspan is the timespan you want to integrate over, y0 are the initial values, and in options you can define differ-

ent settings for the solver. The method uses backward differentiation formulas (Gears method). It is an implicit multistep method. The reason why this solver is chosen instead of another Matlab solver, for instance ODE45, which could be more efficient, is because of its stability and that it still contain all the same functions as ODE45 does.

4.3 Discrete events

In the minimal model two types of discrete events are present, which must be taken into account when creating a simulator. Namely state-events and time-events.

4.3.1 State-Events

State-events, are changes in state of some parameter during the time period chosen. In the insulin minimal model the term $[G(t) - p_5]^+$ is a state-event. When $G(t)$ is larger than p_5 the term has the positive value given by the term inside the brackets. This is when insulin is secreted. But when $G(t)$ goes below p_5 the value is 0, because no insulin should be secreted. This is a change of state. This change of state must be done at the correct time when solving the system. A naive approach to handle this state-event would be to include the following if-statement(in pseudo) in the system to be solved:

```
IF G(t) > p5
   (G(t)-p5)+ = G(t)-p5
ELSE
   (G(t)-p5) = 0
END
```

But this could result in a change at a inexact time, because the timestep would have to be completed before an if-statement could be read. Instead of using this naive approach ODE15s has a function, where you can deal with state-events like this. Basically you create an event function containing the term $G(t) - p_5$ and then by inserting this function in the options and calling ODE15s with the following:

```
[T,Y,TE,YE,IE] = ode15s(odefun,tspan,y0,options)
```

you can detect zero crossings in the event function. Thus when you implement this in the simulators, you make two odefun. One containing the model with the term $G(t) - p5$ and one where the term is zero. Then you start solving and every time a zero crossing is detected you stop the solver and restart it with the other odefun, at the event time given in TE. ODE15s finds the exact time where the event happens. This gives a more exact picture, than in the naive approach.

4.3.2 Time-Events

In the modified model two time-events are present. The U-term can be changed at a certain time and the meal disturbance function can be given a initial value to decay from. Time events are easier to handle than state-events, because the user decides the time where the change should happen. So when dealing with time-events, you basically stop the solver at the time where the value should be changed, change the value and then restart the solver.

4.4 GLUSIM

GLUSIM is an implementation of the glucose minimal model (eq. 1-2). It can be used to simulate an IVGTT based on insulin measurement data. In this model, there is a time-event, for each new insulin data value. But instead of stopping the solver for each new datapoint, which wouldn't give a smooth solution, an interpolation of the points are made using the Matlab function interp1. Then the system is solved with ODE15s. The call of the function is

```
[GE,SI,RES,T] = glusim(parametertype,data)
```

parametertype defines which parametergroup to use. The parametergroups are found in the file parameters1.m. data is the input data, typically measurement data. The simulator gives you the solutions RES to the times T. And in addition the glucose effectiveness GE and the insulin sensitivity SI. The function containing the model in this simulator is bergmanpart1. All of the files related to GLUSIM can be found in the appendix.

4.5 INSSIM

INSSIM is the implementation of the insulin minimal model (eq 3). It can also be used to simulate an IVGTT. Here the glucose measurement data is used as input and a interpolation is done like with GLUSIM. The insulin model contains the state-event mentioned in a previous section, and is dealt with in the way described there. The call of the function is:

```
[pan2,RES,T] = inssim(parametertype,data)
```

again parametertype is the choice of parametergroup. The parametergroups can be found in the file parameters2.m. data is again the data you use as input. Besides the solution and time the simulator gives you the second pancreatic response parameter. The functions containing the model are bergmanpart21 and bergmanpart22. The event function used is implemented in bergmanpart2event. All the files are in the appendix.

4.6 BERSIMU

To have the ability to simulate a IVGTT for the entire blood glucose-insulin system a simulator based on the original model is implemented and it is called BERSIMU. The state-event is handled like in INSSIM. The functions used are bermod1,bermod2 and bermodevent. The call of the function is:

```
[PAN2,GE,SI,RES,T] = bersimu(parametertype,tspan)
```

Like GLUSIM and INSSIM it uses a parameter file (parameters) and gives the solution + additional parameters. The files can be seen in the appendix

4.7 ESIM

All of the previous simulators GLUSIM, INSIM and BERSIMU, are all very simple simulators, with very limited possibilities in concern to to meal disturbance and simulations of diabetics. In this section an implementation of the simulator ESIM, based on the modified model is described. This simulator has

more possibilities than the previous mentioned simulators, and it will be given a graphical user interface to make it easier to use.

4.7.1 Simulator

The simulation program used by ESIM is called MODBERSIM, and it has the following call

```
[SG,SI,RES,T] = modbersim(infuse,control,tspan,initval
,p,b,a,tmeals,mealsam,tin,inam)
```

In which the input are

infuse the size of the basal insulin delivery

Control decides whether or not to use the PID controller, if control = 0 the controller is not used, if not, the controller is used

tspan timespan

initval initial values

p vector containing the 4 p-parameters

b vector containing the 3 values G_b,I_b and V_I

a vector containing the setpoints for the pid controller

tmeals vector containing time of meals in min.

mealsam vector containing initial rate of meal absorbtion

tin vector containing time of insulin injections

inam vector containing amount of insulin injected

Besides this input, the program uses the file parametersesim, where the parameter for the PID controller can be found. The output is the results, times, S_G and S_I given by the definition of the glucose minimal model. The model the simulator uses is implemented in the function MODBERMOD.

4.7.2 PID controller

if *control* $\neq 0$ the PID controller is used by the program MODBERSIM. The A,B,C and D described in chapter 3 in the section about the PID controller, is found by using the following code:

```
Kc = 0.2; Ti = 500; Td = 120;

N = 10;
num = conv(Kc,[Ti*Td*(1+1/N) (Ti+Td/N) 1]);
den = conv([Ti 0],[Td/N 1]);

sys = tf(num,den);
[Apid,Bpid,Cpid,Dpid]=ssdata(sys);
```

First the tuning parameters are defined. Then the transfer function is created (tf). Then finally you go to the state-space domain by using the function ssdata. MODBERSIM and the files related parametersesim and MODBERMOD can be found in the appendix.

4.7.3 GUI

To make the simulator MODBERSIM user friendly a GUI (Graphical user interface) has been created, and is called ESIM. When creating a GUI in Matlab, you first create the components: textboxes, frames etc. and then you give them callbacks, meaning that you attach a function to the component, and when something happens with the component the attached function is called. This function then contains the reaction to the change. Another important thing when creating GUI's is to make error messages, that makes it impossible for the user, to make errors. These error boxes are also found in the attached callback function. The components are created with the function uicontrol and here you can also define the callback function. The program ESIM is in the appendix together with screenshots and a short description of the program.

Chapter 5

Simulations and Discussion

5.1 Introduction

In the previous chapters Bergman's minimal model, has been described and modified. In this chapter different kind of simulations will be done, in order to show the problems and possibilities described. Initially simulations with GLUSIM, which is an implementation of the glucose minimal model will be done. The issue regarding underestimation and overestimation of S_I and S_G will be analyzed. Next the insulin minimal model implemented in INSSIM will be used to describe the basics of this model. n. Then the two coupled models, the original and the modified model, will be looked upon and used for simulation. Finally a short discussion,based on all the simulations, about the use of these coupled models will be done.

5.2 Simulations with the Glucose Minimal Model

The glucose minimal model (see chapter 2) has been implemented into the Matlab program GLUSIM. In this section some of the possibilities and problems with this model will be shown.

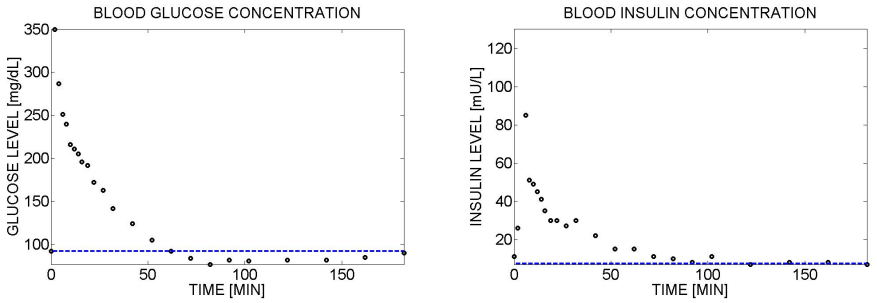

Figure 5.1: Graphical representation of the measured data used to estimate parameters of an IVGTT [11]

5.2.1 The IVGTT

The glucose minimal model is used to interpret the glucose kinetics of an IVGTT. These interpretations are based on parameter estimation, using measured blood insulin concentrations during the test as input to the model. Bergman et al. [11] have made the program MINMOD, which uses a weighted non-linear least squares method to estimate the parameters. They use the measured data shown in figure 5.1. These data are for a normal subject.

The basal values measured are $G_b = 92\frac{mg}{dL}$ and $I_b = 7.3\frac{mU}{L}$ (shown on figure 5.1 with blue dotted lines). By inserting the insulin data as input, and using MINMOD they derive the following parameters:

$$p_1 = 0.03082 \quad p_2 = 0.02093 \quad p_3 = 1.062 \cdot 10^{-5} \quad G(0) = 287\frac{mg}{dL}$$

This gives a glucose effectiveness $S_G = 0.03082$ and a insulin sensitivity $S_I = \frac{p_3}{p_2} = 5.07 \cdot 10^{-4}$ which are both inside the normal range [11]. The parameters are inserted into GLUSIM, and the insulin data are given as input. The graphs derived by doing this can be seen in figure 5.2.

As you can see the GLUSIM simulation follows the measured data nicely like it should. Now a decrease and increase of the parameter p_1 which is also the glucose effectiveness S_G, will be tried in order to show the influence of this parameter according to the minimal model. The rest of the parameters are kept at the previous given values. In figure 5.3 you can see the results. When S_G is halved the decay of the glucose level becomes slower. When S_G is doubled the level decays faster. This parameter does not change anything according to

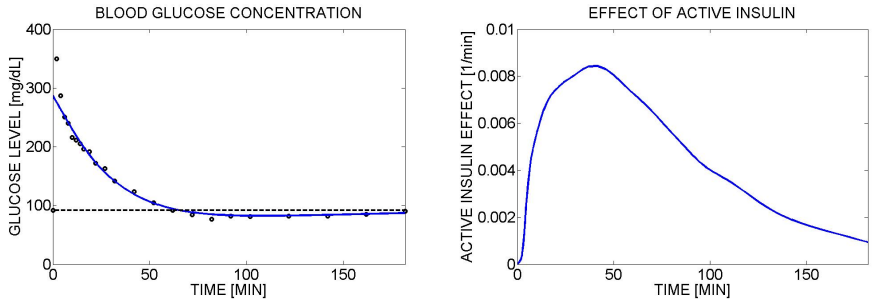

Figure 5.2: Simulation of MINMOD data with GLUSIM. The black dots on graph 1 represents measured glucose values, and the dotted line is G_b

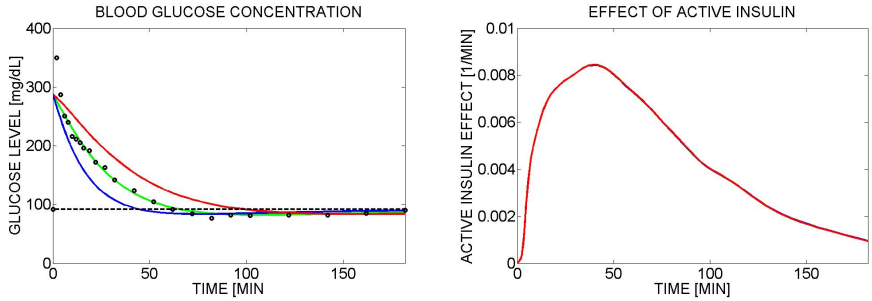

Figure 5.3: Green graph: $S_G = 0.03$. Blue Graph: $S_G = 0.06$. Red graph: $S_G = 0.015$. Black circles: Measured data (MINMOD). Dotted line: G_b

active insulin effect as you can see on the second graph.

Now the same test is done with insulin sensitivity S_I. The results are shown in figure 5.4. When S_I is doubled by doubling p_3 the effect of active insulin is also approximately doubled. And when S_I is halved due to a halved p_3, the effect is approximately halved. When you do the same by halving and doubling p_2 it also changes the effect but not as much as a change in p_3. Basically this shows how a high insulin sensitivity increases the effect of active insulin to decay the glucose level and how a low sensitivity decreases the effect of active insulin. But it also shows that there is a difference in having a high S_I due to a high p_3 in contrary to a high S_I due to a low p_2.

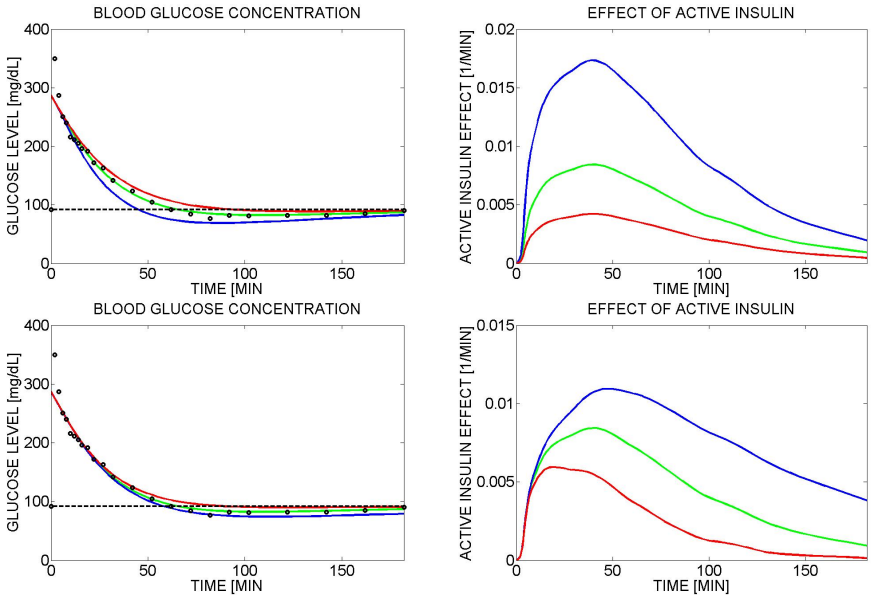

Figure 5.4: In the top graphs $p_3 = 1.062 \cdot 10^{-5}$ and in the bottom graphs: $p_2 = 0.02093$. Green graphs: $S_I = 5 \cdot 10^{-4}$, Blue graphs: $S_I = 10 \cdot 10^{-4}$,Red graphs: $S_I = 2.5 \cdot 10^{-4}$

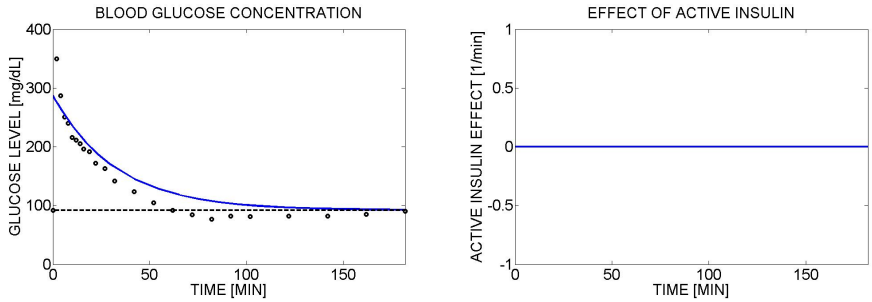

Figure 5.5: The IVGTT with no insulin response simulated with GLUSIM

5.2.2 The Problem with the Glucose Minimal Model

In the first chapter, the problem with the glucose minimal model was described. This problem turned up when parameters were estimated with an IVGTT with insulin response. In such case the glucose effectiveness S_G was overestimated and the insulin sensitivity was underestimated. This can be shown graphically by using the same parameters, used in the previous section to simulate an IVGTT. But instead of using the measurement data as input the insulin is constantly set to the basal value $I_b = 7.3\frac{mg}{dL}$. Describing that the insulin has no effect. The results of doing this are visualized in figure 5.5.

As you can see the glucose still decays very fast, and according to Steil et al. [8] this decay is too fast. Cobelli et al. [3] uses their 2-compartment model, which is not analyzed in this thesis, to estimate that S_G should be about 60% lower and S_I should be 35% higher. This is used in a simulation with GLUSIM, by setting $p_1 = p_1 - p_1 \cdot 60\%$ and $p_3 = p_3 \cdot 135\%$. These parameters are simulated with and without an insulin response, and the result can be seen in figure 5.6. When using these parameter-estimation the curve do not quite fit the glucose curve when insulin is responding, but the parameters are also just estimated. Without the insulin response and the new parameters, the glucose level decay much slower than before, but not slow enough according to the results given by Steil et al. in which the baseline not even is reached after 240 min. All this shows that the glucose minimal model has a lack in giving a S_G and S_I which fit in all situations. This makes it difficult to use it in a simulator, because its difficult to describe a subject using this model.

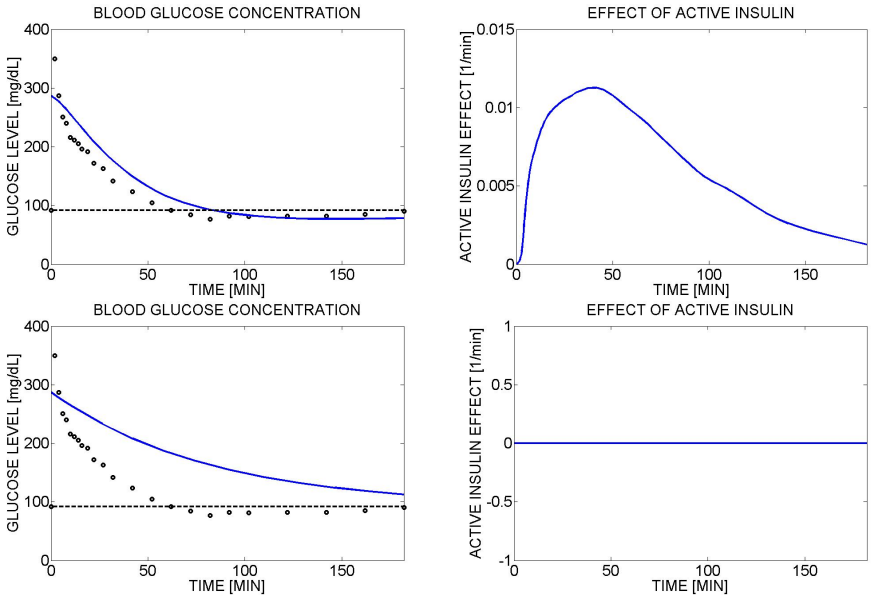

Figure 5.6: The IVGTT with and without insulin response simulated with GLUSIM using the parameter estimations of S_G and S_I from [3]

5.3 Simulations with the Insulin Minimal Model

In this section simulations are done with INSSIM, an implementation of the insulin minimal model.

5.3.1 The IVGTT

Like with the glucose minimal model, Bergman et al. [11] have also used their MINMOD program to estimate parameters for the insulin minimal model, by using the glucose measurements shown in figure 5.1. They derive the following parameters:

$$p_4 = 0.3 \qquad p_5 = 89.5 \qquad p_6 = 0.3349 \cdot 10^{-2} \qquad I(0) = 403.4\frac{mU}{L}$$

Before the first pancreatic response ϕ_1 is calculated the computed values at time

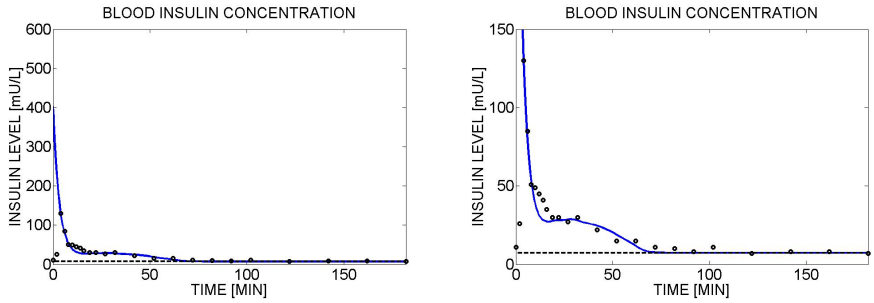

Figure 5.7: Simulation with MINMOD data with INSSIM. the black dots are measured data, the dotted line is I_b.

0 and time 2 are neglected, because this are the times (see measured data) where the insulin level rises to the first peak, and in the insulin minimal model the rise to the first peak cannot be computed. So instead of having $I_{max} = 403.4$ they have $I_{max} = 132.5$ [11]. and G(0), in the formula is then actually the measured value 4 min. after the injection. Then the pancreatic responses can be calculated:

$$\phi_1 = \frac{I_{max} - I_b}{p_4(G(0) - G_b)} = \frac{132.5 - 7.3}{0.3(287 - 92)} = 2.14$$

$$\phi_2 = p_6 \cdot 10^4 = 33.49$$

By using the measured glucose data and the derived parameters as input you get the graph on figure 5.7. The INSSIM data follows the measured data. Computed times until the 4th min. should be neglected as earlier described.

5.4 Simulations with The Original Model

The coupling between the two original minimal models is implemented in the Matlab program BERSIMU. This coupling gives a full picture of the blood glucose-insulin system, and need no measured data as input. In this section the possibilities, ex. simulation of a type 2 diabetic and a graphical presentation of the problems that occurs when $p_5 < G_b$ will be done.

5.4.1 Comparison

When using the original coupled model, it would be possible to fit the model to an IVGTT, in a single step. But would this give the same parameters as with the separate minimal models In the lack of a estimation program, a comparison, between the graphs simulated with BERSIMU and GLUSIM AND INSSIM, is done to check for differences. The parameters used for the models are the ones derived with MINMOD, introduced in the previous sections:

$$p_1 = 0.03082 \quad p_2 = 0.02093 \quad p_3 = 1.062 \cdot 10^{-5} \quad G(0) = 287\frac{mg}{dL}$$
$$p_4 = 0.3 \qquad p_5 = 89.5 \qquad p_6 = 0.3349 \cdot 10^{-2} \qquad I(0) = 403.4\frac{mU}{L}$$

The results are shown in figure 5.8. The comparison shows small differences between the original model and the separate model simulation. This shows that a parameter estimation using the original model, would result in a different set of parameters, and a new index for the interpretation parameters S_I, S_G, ϕ_1 and ϕ_2.

5.4.2 The Problem with The Original Model

As explained in the section about the original model, the coupling has a problem, namely that no equilibrium can be obtained when $p_5 < G_b$. In figure 5.9 this problem is illustrated by a comparison between two simulations with BERSIMU. One where $p_5 = 89.5$ and one where $p_5 = 94$. The parameters from the previous section is used so $G_b = 92\frac{mg}{dL}$. The graphs show how the equilibrium is found when $(G(t) = G_b, X(t) = 0, I(t) = I_b$ for the simulation with $p_5 = 94$. But for the simulation with $p_5 = 89.5$, no equilibrium is found. instead the vibrations for the blood insulin concentration seems to grow.

5.4.3 Simulating a Type 2 Diabetic

One of the possibilities given by the original model is to simulate an IVGTT for a insulin-independent type 2 diabetic. In studies by Bergman et al. [14] they found that a low S_G and a $\phi_2 S_I$ below $75 \cdot 10^{-4}$ was in common for low tolerant (glucose resistant) subjects. The study showing that a low glucose effectiveness is present in a low tolerant subject, can be due to the problem

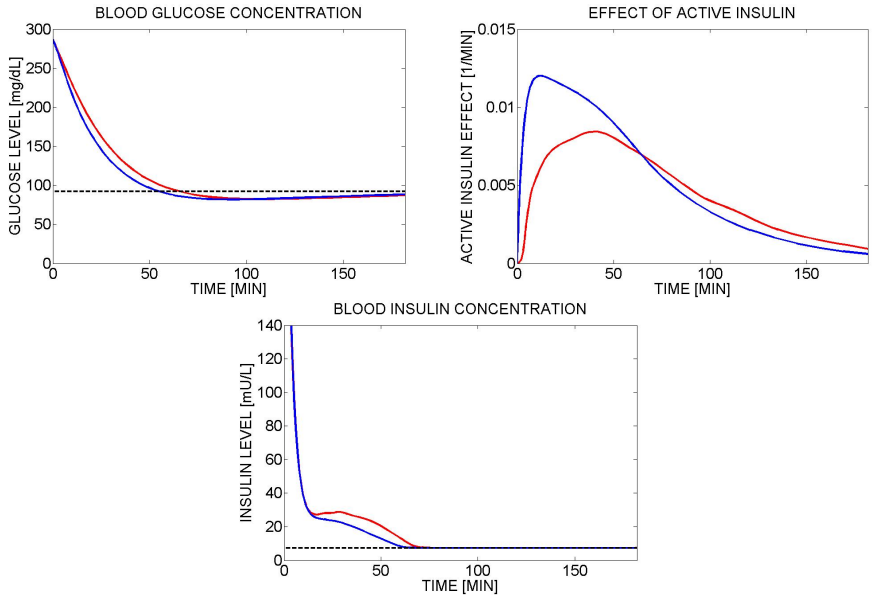

Figure 5.8: Comparison between data simulated with the separate minimal models and the coupled model. Red graphs are simulated with either GLUSIM or INSSIM and blue graphs are simulated with BERSIMU

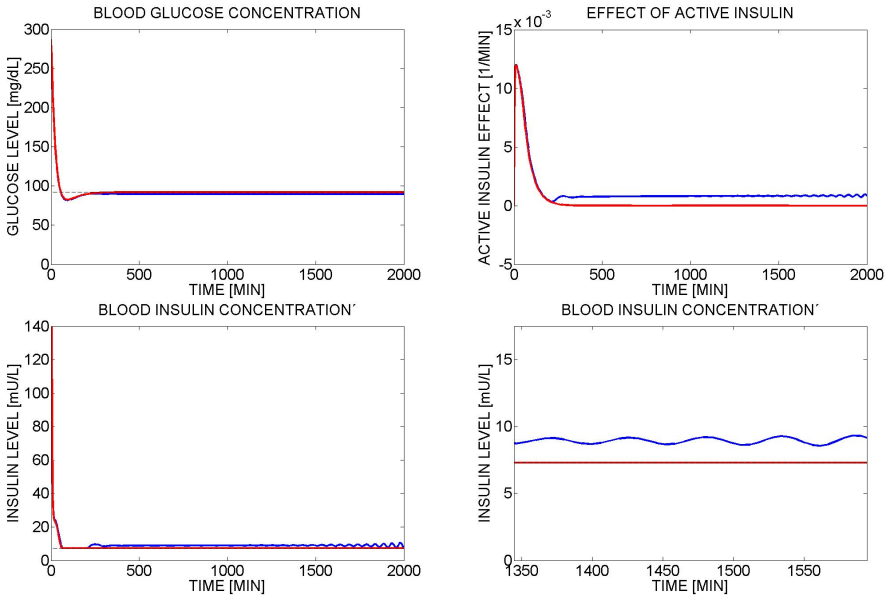

Figure 5.9: Comparison between simulations done with $p_5 < G_b$ (blue graphs) and $p_5 > G_b$ (red graphs), baselines: G_b and I_b (black dotted lines)

with the glucose minimal model, in which glucose effectiveness is overestimated and insulin sensitivity is underestimated. In figure 5.10 the IVGTT for subjects with low S_G , low $\phi_2 S_I$ and with both low S_G and $\phi_2 S_I$ respectively are shown. These are compared to a good tolerant subject described with the previously used parameters, but with $p_5 = 94$, due to the equilibrium problem with the coupling. The simulation results show how the subjects with low S_G and low $\phi_2 S_I$ respectively has a slower decay. And when both of these terms are low the decay is very slow. These simulations shows how the pattern of a type 2 diabetic (glucose resistant) could look like.

5.5 Simulations with The Modified Model

In this section simulations with the modified model are done. The modified model contains many possibilities in regard to meal disturbance and insulin infusion. First an attempt to simulate the pattern of a type 1 diabetic during fasting is attempted, then the effect of meals and insulin injections are shown. Finally the PID controller is tuned and tested, in order to show the possibility of using a controller with the model.

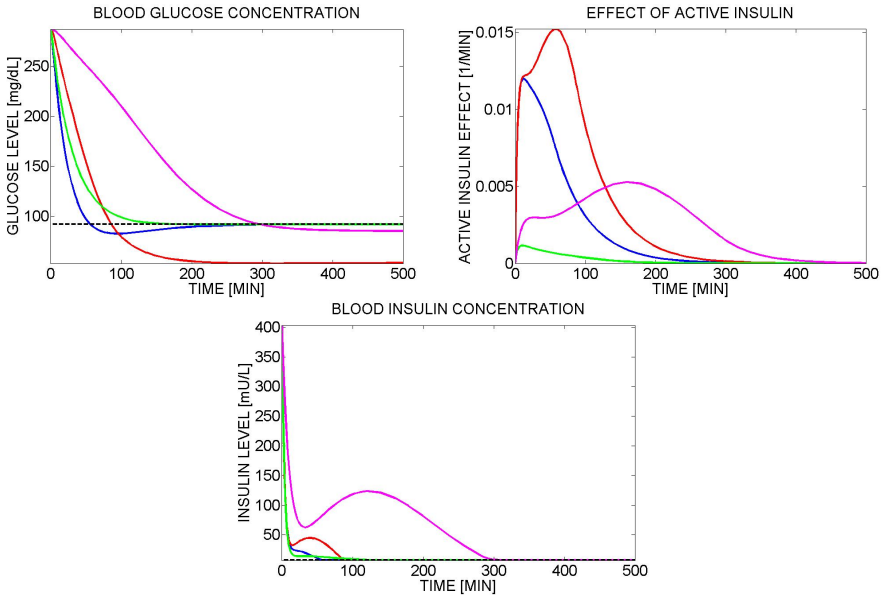

Figure 5.10: Blue graphs: Normal subject $S_G = 0.0308$ and $\phi_2 S_I = 169.93$, Red graphs: Low $S_G = 0.0001$ and normal $\phi_2 S_I = 169.93$, Green graphs: Normal $S_G = 0.0308$ and low $\phi_2 S_I = 5.07$, Magenta graphs: low $S_G = 0.0001$ and low $\phi_2 S_I = 5.07$

5.5.1 A Type 1 Diabetic

In this section an attempt to simulate the pattern of a type 1 diabetic during fasting is done. In the section about the glucose minimal model it was shown that if the parameters were estimated based on a IVGTT with insulin response, they could not be used to show how the subject would react if no insulin response was present. Here an attempt to simulate a diabetic during fasting is tried. An untreated type 1 diabetic has a very high blood glucose concentration, due to the fact that none or almost none insulin is produced. In these simulations the function U(t) is given the constant value 0. G_b is set to an arbitrary high value $G_b = 200\frac{mg}{dL}$. The basal insulin concentration I_b for a type 1 diabetic is set to $I_b = 0\frac{mU}{L}$. By using these values and using the following parameters for a type 1 diabetic derived in a study by Lynch et al. [10]:

$$p_1 = 0.028735\frac{1}{min}, \quad p_2 = 0.028344\frac{1}{min} \quad p_3 = 5.035 \cdot 10^{-5}\frac{L}{min^2 mU}$$

you get the results shown in figure 5.11. everything stays at the steady state. One could argue that it is wrong to set $I_b = 0$, because then the effect of any ínsulin coming in, would be the same as for a healthy person having a basal insulin concentration at e.g. $I_b = 15\frac{mU}{L}$. Having $I_b = 15\frac{mU}{L}$ and zero insulin production would give the effect of insulin a negative effect, which would increase the glucose value. This does not seem likely. Whether or not insulin has the same effect independently of the basal concentration would require a better physiological understanding of the system.

5.5.2 Meal disturbance

One of the possibilities with the modified model, is to see how the model/system reacts to a meal disturbance. By using the parameters from the previous section a meal disturbance simulation is done. In figure 5.12 the result of three meals of different sizes are given, using Fisher's meal disturbance function and values of 3, 5 and 12. no insulin response is given. These graphs show that the glucose level rises and then decays inside a 2-hour period. This fast clearance of glucose is most likely due to the parameters estimated by Lynch et al. has been affected by the problems with the glucose minimal model, an overestimation of S_G and a underestimation of S_I. So p_1 should have a lower value, which would slow down the decay. But this was just to show that the model reacts to the meal disturbance.

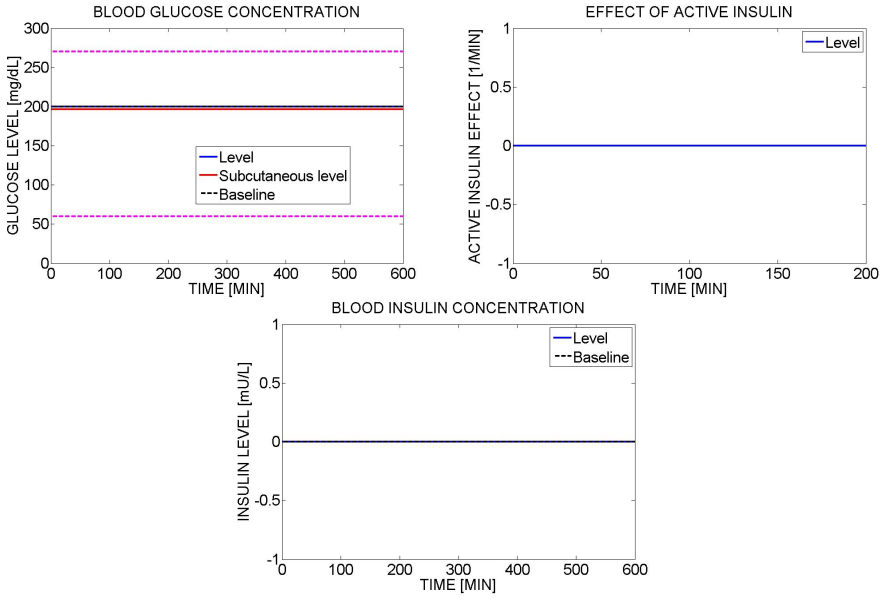

Figure 5.11: Attempt to simulate a fasting period for a type 1 diabetic.

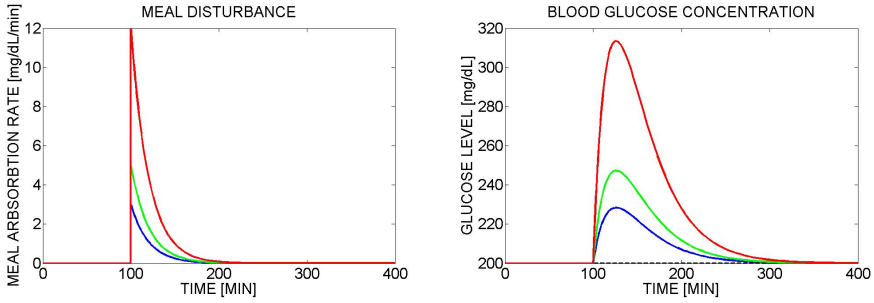

Figure 5.12: Meal disturbance graphs at zero insulin. Black dotted lines are baseline G_b

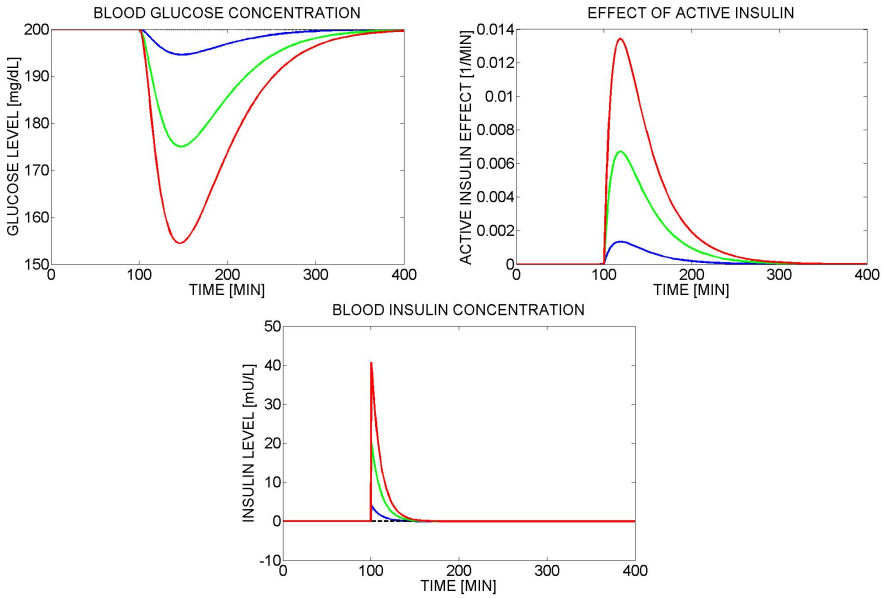

Figure 5.13: Insulin Injections given at time 100. Black dotted lines are baselines G_b and I_b.

5.5.2.1 Insulin Injections

Another possibility with the modified model, is to use the U(t) function and see how the model reacts to an insulin shot, and or a change in the basal delivery. If it is assumed that the injections is injected in a time period of 1 minut, a simulation can be done of an insulin injection. The parameters used are the same as before. In figure 5.13 graphs for 3 injections of sizes $100mU$,$500mU$ and $1000mU$ is given. This shows how the model describe an insulin injection effect on the blood glucose concentration. This is however a very simplistic attempt to simulate an injection, because there are different types of insulin, fast-acting and short acting. By changing p_4 the clearance rate of insulin, you can change how long time the insulin should act (see figure 5.14). but a model describing some delays could be a good idea to add to the model.

Another way to use the U(t) function is to give it a constant value, so it describes a constant delivery of a certain size. For the type 1 diabetic used in the previous simulations the delivery is 0. In figure 5.15 the basal delivery is changed from 0 to 20 $\frac{mU}{min}$. This results in a new steady state for blood glucose concentration at $G(t) = 100\frac{mg}{dL}$. So with a change in basal delivery it is possible to change the basal concentration of glucose. This basal delivery could be given by a pump.

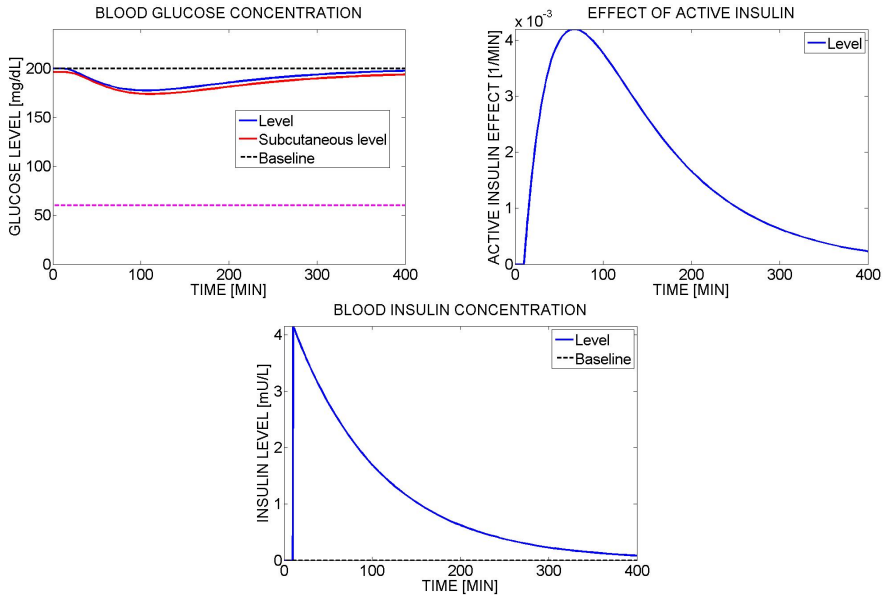

Figure 5.14: Insulin Injection at time 10 of $100mU$ with a low $p_4 = 0.01$. Black dotted lines are baselines G_b and I_b.

Such a controlled pump is described in the next section by the PID controller, introduced in chapter 3.

5.5.3 The Artificial Pancreas

In chapter 3, a PID controller, was introduced. This PID controller could be used to create an artificial pancreas. In this section a tuning and a testing of this PID controller is described. Some of the issues regarding the creation of a controller will be discussed. The PID controller, is not the best choice of controller and this is not the purpose. The purpose is to show how a controller can be used together with the minimal model.

5.5.3.1 Tuning the PID Controller

If the PID controller, has to work as an artificial pancreas it has to be tuned. Here an empirical tuning is done based on the pattern of an IVGTT and the reactions to a meal disturbance. The parameters used are:

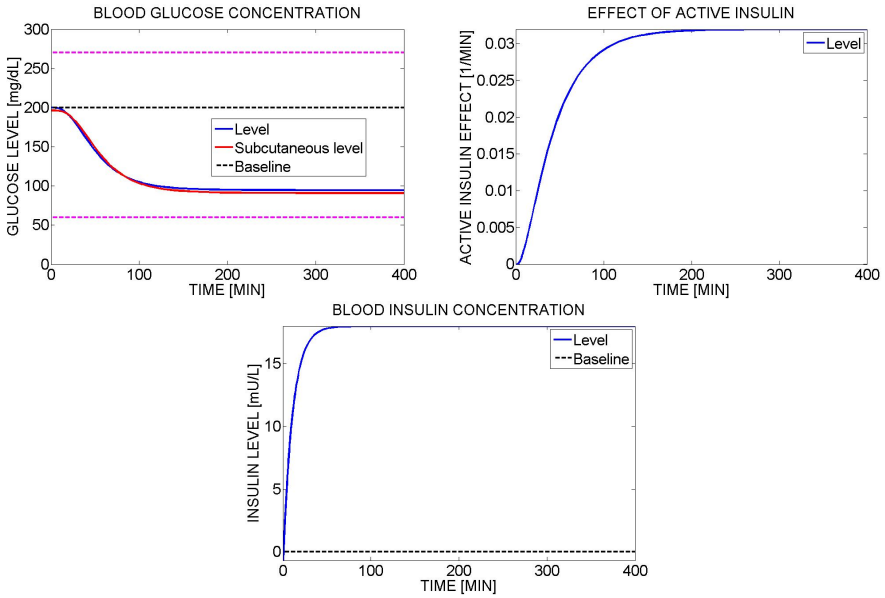

Figure 5.15: Basal delivery changed to U(t) = 20. Black dotted lines are baselines G_b and I_b

$$G_b = 81\frac{mg}{dL}, \quad I_b = 15\frac{mU}{L}, \quad V_I = 12L, \quad p_4 = 0.3\frac{1}{min}$$
$$p_1 = 0.028735\frac{1}{min}, \quad p_2 = 0.028344\frac{1}{min} \quad p_3 = 5.035 \cdot 10^{-5}\frac{L}{min^2 mU}$$

These are the parameters describing a type 1 diabetic [10]. One thing to notice is that $I_b = 15\frac{mU}{L}$. This means that is assumed that the controller has been there for sometime and has created a basal concentration of $I_b = 15\frac{mU}{L}$. In order to keep this delivery a basal rate of $p_4 I_b V_I$ is infused as mentioned in chapter 3. This means that the controller controls the additional delivery, but can also subtract from the basal. By looking at the IVGTT derived by Bergman et al. [11], which is a typical pattern of IVGTT, and comparing it to the output of MODBERSIM an empirical tuning is done and the following parameters are derived:

$$K_c = 0.2, \quad T_i = 500, \quad T_d = 120 \quad N = 10$$

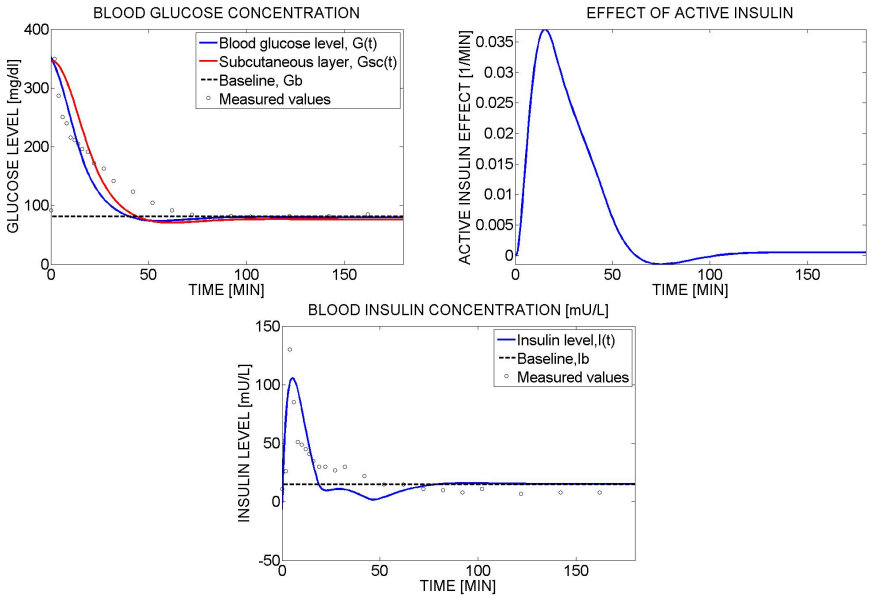

Figure 5.16: The comparison between the IVGTT from MINMOD [11] and the tuned IVGTT

As you see the integral part of the controller $\frac{1}{T_i}$ is very small, and can almost be neglected, this makes the PID controller a PD controller. The graphs of the simulated IVGTT, with the "Artificial Pancreas" can be seen in figure 5.16.

As it can be seen this controller is not fitted perfectly. However if you want to create a good artificial pancreas, this is not important. The main target should be to bring down the glucose level fast, without infusing so much insulin, that it results in hypoglycemia, where the glucose level becomes too low. The two major issues when you want to create an artificial pancreas, is that it has to react fast, and has to be able to determine infusion based on subcutaneous measurements.

5.5.3.2 Testing the Controller

Further testing of the PID-controller based artificial pancreas is done, in order to see if it works correctly. Testing of controllers, like this, which have great influence on peoples life are very important.

The minimal model could be used to test these controllers, but this would be

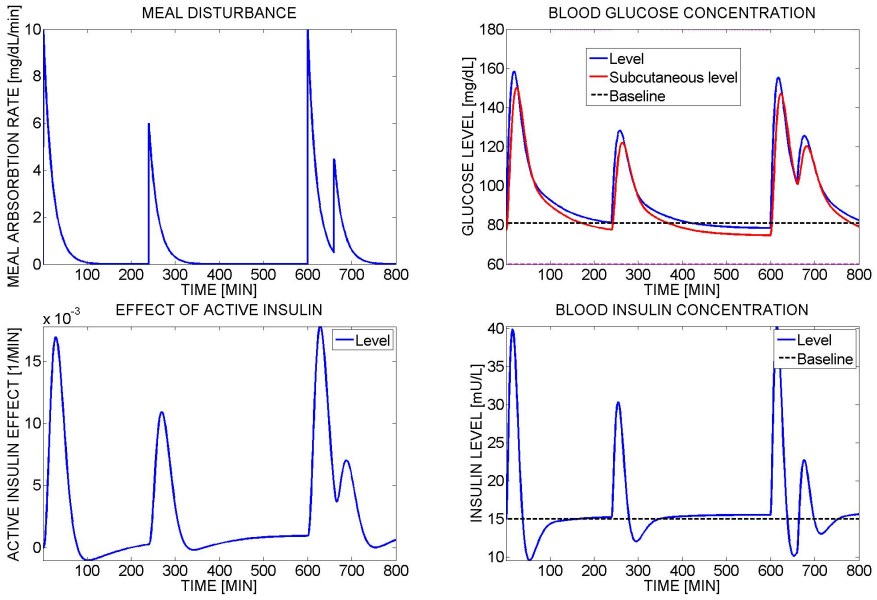

Figure 5.17: Testing the tuned PID controller. Meal test, with breakfast,lunch,dinner and snack. The first graph show the meal rates. initial values for the meal rates are chosen to be between 5-10 $\frac{mg}{dL}$

a very simple test. A more detailed model, is therefore necessary. In this case the controller is tested by simulating an OGTT and a whole day with different meals. For the OGTT the result should show a healthy subject (Blood glucose concentration less than $140\frac{mg}{dL}$ after 2 hours). For the meal test a set of demands are derived from the study by Lynch et al. [10], The blood glucose level should return to the steady state within 3 hours, $0 < U(t) < 100\frac{mu}{min}$ and $60\frac{mg}{dL} < G(t) < 180\frac{mg}{dL}$. Both the OGTT and the meals are simulated with Fishers meal disturbance function. figure 5.17 and 5.18 show the results of the test. The results show that the controller keeps the glucose inside the given boundaries in both scenarios.

5.5.3.3 The Problems with the Controller

Even though the tests gave good results. The controller still has some problems. The empirical tuning is very rough, and further testing would probably show some problems, e.g. hypoglycemia, with the PID controller. Another problem is that the controller is tuned to a certain set of parameters, so it would only work for this single subject.Thus this controller is not perfect, but just an example of

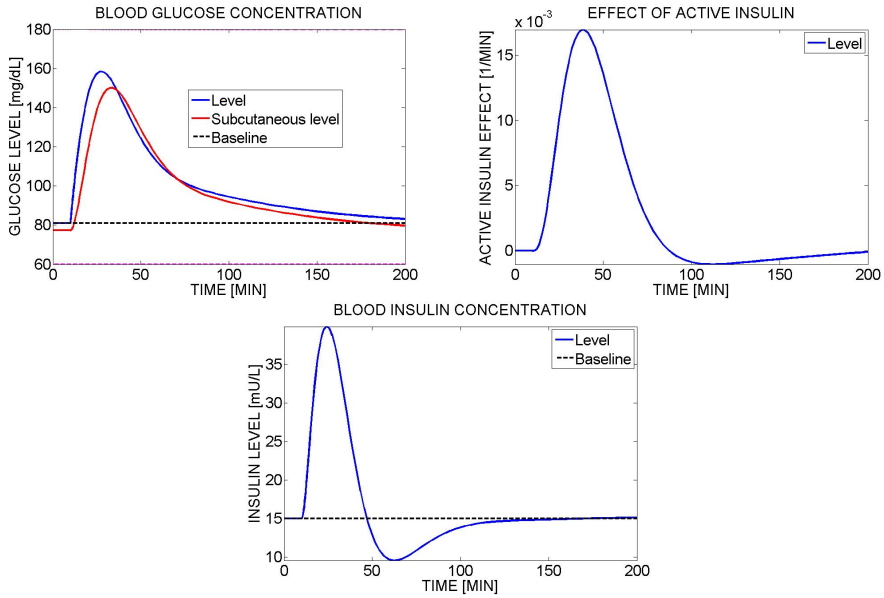

Figure 5.18: Testing the tuned PID controller. OGTT test, at time 10 the initial rate of fishers meal disturbance function is set to 10 $\frac{mg}{dL}$

how a controller can be used together with the modified model.

5.6 Discussion about the coupled models

The two coupled models which both are based on the glucose-and insulin minimal models, has now been analyzed according to possibilities and problems. In this section a summary is given, in order to discuss whether or not the coupled models are usable.

5.6.1 The Original Model

The original model, based on the two basic minimal models, has some good functionalities, shown in the simulations. First of all it could be used to interpret an IVGTT in a single step for the entire blood glucose system. Secondly it can be used to simulate different IVGTT's for both healthy and glucose resistant subjects. Finally due to the low number of parameters it is very easy to use.

The model however have some problems. If the parameter $p_5 < G_b$ no equilibrium can be found. Another problem is that the model adopts the problem of the glucose minimal model: overestimation of S_G and underestimation of S_I. This means that a inaccurate picture is given on the influence of glucose and insulin respectively.

The original model, can be used inside a limited area, namely to simulate and interpret an IVGTT. In this limited area the equilibrium problem plays a minor role so the model could be used to give a approximate picture of an IVGTT. However this picture would be inaccurate due to the problem with the glucose minimal model.

5.6.2 The Modified Model

The modified model has a many possibilities. It can be used to show the reactions of a type 1 diabetic to a meal disturbance, an insulin shot and a change in basal delivery of insulin. Due to the few number of parameters it is easy to use, compared to other models (Sorenson etc. [8]).

The most interesting function of the modified model is the possibility to attach a controller, which controls the glucose level by manipulating the insulin delivery. This is interesting, because one of the great issues in the diabetes-world today is the search for a controller acting like an artificial pancreas. With the modified model with a controller attached it is possible to test how such a controller would react to meal disturbance. Even though it would only give an approximate picture, it would be a good starting point, if you want to know whether your controller works or not.

As described the modified model has some good functionalities. But like the original model it also adopts the problem of the glucose minimal model. This makes it difficult to describe a subject using the minimal model, because the model does not have the ability to slow down the total plasma clearance rate due to the rise in glucose, when no insulin response is present.

The modified model can be used to give a rough picture of how the blood glucose-insulin system is for a type 1 diabetic with exogenous insulin delivery, and how this subject reacts to a given meal disturbance. But the model does not give a full picture of the system in all of its aspects, so when using it to e.g. test a controller, you have to be cautious and aware of the problems, which can occur.

5.6.3 Could the models be used

The original model and the modified model, are both very simple models, which does not give a full picture of the blood glucose-insulin system. But both of the models are able to give an approximate picture of the system.The original model can be used to simulate IVGTT's. But due to the problems of the adopted glucose minimal model, it is not a good model to use to interpret these tests.

The modified model has good functionalities for testing of controllers. But it only gives a very general picture, so you could not use the modified model to certify that a controller is working 100% correct.

Basically it is almost impossible to make a model, which describes the blood glucose-insulin system 100% correct, but mathematical models are a good tool to give an approximate picture. Both of the models are able to give this approximate picture, with a few number of parameters. Both of the models could be used for simulations as long as you are aware, that this is very simple models, giving a rough picture of the blood glucose-insulin system, and that this picture is not always correct due to the problems of the models.

Chapter 6

Conclusion

In this thesis the construction of a mathematical model describing the whole blood glucose-insulin system was tried. Two models was derived. They were both based upon the two minimal models of Bergman's minimal model, which is primarily used to interpret an IVGTT.

These two minimal models carried some problems. One of the problems was that the glucose minimal model overestimates S_G and S_I when fitted to data with a insulin response. This makes it difficult to simulate different scenarios with this model. And maybe this model is to minimal to describe the aspects when no insulin response is present.

One of the coupled models the original model is very attached to the IVGTT and is able of simulating IVGTT for both healthy and glucose resistant subjects. It did however have the problem that no equilibrium could be obtained for $p_5 < Gb$, and that makes it mathematically incorrect. It also adopted the problem of the glucose minimal model, which makes it a poor model for interpretation of the IVGTT.

The other coupled model, the modified model was not attached to a single test. It contained the possibility to simulate the reaction to a meal disturbance and the reaction to a insulin injection or a change in basal insulin delivery. A PID controller was also implemented to show how the model could be used to test

controllers. Such a test however would only be very rough due to the simplicity of the minimal model.

The modified model, like the original model, adopted the problem of the glucose minimal model, and among other things this made it difficult to simulate a type 1 diabetic without treatment.

Thus the coupled models were not perfect, but but if used with caution to the problems, they could be used to give approximations to how the blood glucose-insulin system would react in a certain situation and one of the positive sides of the coupled models is that they are very easy to use.

Appendix A

Matlab Programs

A.0.4 glusim.m

```
function [GE,SI,RES,T] = glusim(parametertype,data)

parameters1

p = [p1 p2 p3];
b = [Gb Ib];
startval = [G0 X0];
GE = p1;
SI = p3/p2;
tmin = Data(1,1);
tmax = Data(end,1);
tmin = 0;
tmax = 182;
tspan = [tmin tmax];

[T,RES] = ode15s(@bergmanpart1,tspan,startval,[],Data,p,b);


% Gbvec = Gb*ones(size(T));
% figure
% plot(T,RES(:,1),'-b','linewidth',3)
% hold on
% plot(T,Gbvec,'--black','linewidth',3)
% plot(Data(:,1),Data(:,2),'*black')
```

```
% V = axis; axis([0 182 V(3) V(4)]);
% title('BLOOD GLUCOSE LEVEL,SIMULATED WITH GLUSIM','fontsize',16)
% xlabel('TIME [MIN]','fontsize',14)
% ylabel('GLUCOSE LEVEL [mg/dL]','fontsize',14)
% legend('Glucose level','Baseline Gb','Measured data')
% figure
% plot(T,RES(:,2),'-b','linewidth',3)
% title('ACTIVE INSULIN EFFECT,SIMULATED WITH GLUSIM','fontsize',16)
% xlabel('TIME [MIN]','fontsize',14)
% ylabel('ACTIVE INSULIN EFFECT [1/MIN]','fontsize',14)
% V = axis; axis([0 182 V(3) V(4)]);
% figure
% plot(Data(:,1),Data(:,3),'oblack','linewidth',3)
% title('MEASURED INSULIN DATA','fontsize',16)
% xlabel('TIME [MIN]','fontsize',14)
% ylabel('BLOOD INSULIN LEVEL','fontsize',14)
```

A.0.5 bergmanpart1.m

```
%************************************************************************
% Minimal Model part 1 (glucose)
% Matlab Implementation by Esben Friis-Jensen, s042244
% DTU. 2007
%************************************************************************
% This function should be solved with a ODESOLVER, this could be one of the
% Matlab standard ODESOLVERS like ODE45 or ODE15s.
%************************************************************************
% Besides the basic time t, and the res vector it should have 2 more inputs:
%************************************************************************
% p is a vector of size (3,1), containing the 3 parameters p1, p2 and p3
%************************************************************************
% b is a vector of size (2,1) containing the basal values Gb and Ib
%************************************************************************
function [dres] = bergmanpart1(t,res,input,p,b)

dres = zeros(2,1); % a column vector with 7 elements

if size(input,1) > 1 || size(input,2) > 1
I = interp1(input(:,1),input(:,3),t);
else
I = input;
end

%************************************************************************
% The equations
%************************************************************************
dG = -p(1)*(res(1)-b(1)) - res(2)*res(1);
dX = -p(2)*res(2)+ p(3)*(I-b(2));

dres(1) = dG;
dres(2) = dX;
%************************************************************************
```

A.0.6 parameters1.m

```
%**************************************************************************
% Parameters for the Minimal Model part 1 : Glucose kinetics
% Implemented by Esben Friis−Jensen, DTU. To be used together with the
% program glusim
%**************************************************************************
% Parameters for the p−vector
%**************************************************************************
% p0 − Theoretical glucose level at time zero above baseline.
% p1 − Glucose effectiveness (Insulin independent).
% p2 − Rate of the spontaneous decrease of tissue glucose uptake ability.
% p3 − increase in uptake ability per unit of insulin conc. over baseline
% (insulin dependent).
%**************************************************************************
% Parameters for the b−vector
%**************************************************************************
% Gb − Baseline glycemia.
% Ib − Baseline insulemnia.
%**************************************************************************

Data = data;


%**************************************************************************
% Here the parameters are setup
%**************************************************************************
if parametertype == 1 % Parameter group 1
    %**********************************************************************
    % p−values
    %**********************************************************************
    p0 = 195; p1 = 0.03082; p2 = 0.02093; p3 = 1.062E−5;
    %**********************************************************************
    % b−values
    %**********************************************************************
    Gb = 92; Ib = 7.3;
    %**********************************************************************
    % Initial values
    %**********************************************************************
    G0 = Gb + p0; X0 = 0;
end

if parametertype == 2 % Parameter group 1
    %**********************************************************************
    % p−values
    %**********************************************************************
    p0 = 195; p1 = 0.03082; p2 = 0.01046; p3 = 1.062E−5;
    %**********************************************************************
    % b−values
    %**********************************************************************
    Gb = 92; Ib = 7.3;
    %**********************************************************************
    % Initial values
    %**********************************************************************
    G0 = Gb + p0; X0 = 0;
```

```
end

if parametertype == 3 % Parameter group 1
    %**************************************************************************
    % p−values
    %**************************************************************************
    p0 = 195; p1 = 0.03082; p2 = 0.04186; p3 = 1.062E−5;
    %**************************************************************************
    % b−values
    %**************************************************************************
    Gb = 92; Ib = 7.3;
    %**************************************************************************
    % Initial values
    %**************************************************************************
    G0 = Gb + p0; X0 = 0;
end
```

A.0.7 inssim.m

```
function [pan2,RES,T] = inssim(parametertype,data)

parameters2

p = [p4 p5 p6];
b = [Ib];
startval = [I0];
options = odeset('Events',@bergmanpart2event);
tspan = Data(:,1);

tstart = Data(1,1);
tmax = Data(end,1);
    begin = 1;
    while tstart < tmax

        state2 = (bergmanpart2event(tstart,startval,Data,p,b) == 0);


        if bergmanpart2event(tstart,startval,Data,p,b) > 0
            [T1,RES1,Te,Ye,Ie]=ode15s(@bergmanpart21,[tstart tmax],startval,options,Data,p,b);
        else
            [T1,RES1,Te,Ye,Ie]=ode15s(@bergmanpart22,[tstart tmax],startval,options,Data,p,b);
        end


        if state2 == 1
            startval = Ye(1,:);
            tstart = Te(1,1);
            T1 = [];
            RES1 = [];
        elseif ~isempty(Ie)
            startval = Ye(end,:);
```

```
            tstart = Te(1,1);
        else
            tstart = tmax;
        end

        Ye = [];
        Ie = [];
        Te = [];


        theend = begin + (size(T1,1)-1);
        RES(begin:theend,1) = RES1;
        T(begin:theend,1) = T1;
        begin = theend + 1;
    end


pan2 = p6*10^4;
```

A.0.8 bergmanpart21.m

```
%************************************************************************
% Minimal Model part 2 (Insulin). state 1
% Matlab Implementation by Esben Friis-Jensen, s042244
% DTU. 2007
%************************************************************************
% This function should be solved with a ODESOLVER, this could be one of the
% Matlab standard ODESOLVERS like ODE45 or ODE15s.
%************************************************************************
% Besides the basic time t, and the res vector it should have 2 more inputs:
%************************************************************************
% p is a vector of size (3,1), containing the 3 parameters p4, p5 and p6
%************************************************************************
% b is a vector of size (1,1) containing the basal value Ib
%************************************************************************
function [dres] = bergmanpart21(t,res,input,p,b)

global G

dres = zeros(1,1); % a column vector

G = interp1(input(:,1),input(:,2),t);


%************************************************************************
% The equations
%************************************************************************
dI = p(3)*(G-p(2))*t - p(1)*(res(1)-b(1));

dres(1) = dI;
%************************************************************************
```

A.0.9 bergmanpart22.m

```
%************************************************************************
% Minimal Model part 2 (Insulin). state 1
% Matlab Implementation by Esben Friis−Jensen, s042244
% DTU. 2007
%************************************************************************
% This function should be solved with a ODESOLVER, this could be one of the
% Matlab standard ODESOLVERS like ODE45 or ODE15s.
%************************************************************************
% Besides the basic time t, and the res vector it should have 2 more inputs:
%************************************************************************
% p is a vector of size (2,1), containing the 3 parameters p4, p5 and p6
%************************************************************************
% b is a vector of size (1,1) containing the basal value Ib
%************************************************************************
function [dres] = bergmanpart22(t,res,input,p,b)

global G

dres = zeros(1,1); % a column vector

G = interp1(input(:,1),input(:,2),t);

%************************************************************************
% The equations
%************************************************************************
dI = − p(1)∗(res(1)−b(1));

dres(1) = dI;
%************************************************************************
```

A.0.10 bergmanpart2event.m

```
function [rese,isterminal,direction]= bergmanpart2event(t,res,input,p,b)

G = interp1(input(:,1),input(:,2),t);


rese = G−p(2);
isterminal = 1;
direction = 0;
```

A.0.11 parameters2.m

```
%************************************************************************
% Parameters for the Minimal Model
```

```
% Implemented by Esben Friis−Jensen, DTU. To be used together with the
% program BERSIMU
%************************************************************************
% Parameters for the p−vector
%************************************************************************
% p0 − Theoretical glucose level at time zero above baseline.
% p1 − Glucose effectiveness (Insulin independent).
% p2 − Rate of the spontaneous decrease of tissue glucose uptake ability.
% p3 − increase in uptake ability per unit of insulin conc. over baseline
% (insulin dependent).
% p4 − first order decay rate contant for insulin (n).
% p5 − Target glucose level
% p6 − Rate of pancreatic release of insulin after glucose bolus pr. min.
% pr. mg/dL above target glycemia
%************************************************************************
% Parameters for the b−vector
%************************************************************************
% Gb − Baseline glycemia.
% Ib − Baseline insulemnia.V_I − Volume of insulin compartment in liters.
% Rutln − Baseline for the glucose level in the subcutaneuos layer.
% Gbsc − Rate of utilization
% drate − Decay rate for meals, also known as alpha.
%************************************************************************

Data = data;

%************************************************************************
% Here the parameters are setup
%************************************************************************
if parametertype == 1 % Parameter group 1
    %********************************************************************
    % p−values
    %********************************************************************
     p4 = 0.3; p5 = 89.5; p6 = 0.003349;
    %********************************************************************
    % b−values
    %********************************************************************
    Ib = 7.3;
    %********************************************************************
    % Initial values
    %********************************************************************
    I0 = Ib+396;
end
```

A.0.12 bersimu.m

```
%************************************************************************
% BERSIMU − A simulator for Bergmans original minimal model. (coupled)
% Implemented by Esben Friis−Jensen, s042244, DTU
%
% It is used together with the file parameters, and the functions bermod1
% and bermod2.
```

```
%*************************************************************************
% The call
%*************************************************************************
% [SG,SI] = bersimu(parametertype,tspan)
%*************************************************************************
% Where the input are:
%*************************************************************************
%*************************************************************************
% PARAMETERTYPE:
% An integer. This is the choice of parameters. The parametergroups can be
% found in the parameters file, where you can change the values as you
% like.
%*************************************************************************
% TSPAN:
% Should be a vector containing [t_min t_max]
%*************************************************************************


function [pan2,GE,SI,RES,T] = bersimu(parametertype,tspan)


%*************************************************************************
% In the file parameters, all the parameters are setup
%*************************************************************************
parameters;
%*************************************************************************
% the values from parameters are loaded into the vectors p and b
%*************************************************************************
p = [p1 p2 p3 p4 p5 p6];
b = [Gb Ib];
GE = p(1);
SI = p(3)/p(2);
pan2 = p(6)*10^4

%*************************************************************************
% Here the initial values and options are chosen
%*************************************************************************
if size(tspan,2) ~= 2
    display('----------------WARNING---------------------')
    display('NO SIMULATION EXECUTED DUE TO THE FOLLOWING')
    display('Your TSPAN vector should contain 2 elements')
    display('--------------------------------------------')
    return
end
t_min = tspan(1);
t_max = tspan(2);
startval = [G0 X0 I0]; % Initial values
options = odeset('Events',@bermodevent);


%*************************************************************************
% Basic simulation, without any disturbance
%*************************************************************************
tstart = t_min;
begin = 1;
```

```
while tstart < t_max

    state2 = (bermodevent(tstart,startval,p,b) == 0);



    if bermodevent(tstart,startval,p,b) > 0
        [T1,RES1,Te,Ye,Ie]=ode15s(@bermod1,[tstart t_max],startval,options,p,b);
    else
        [T1,RES1,Te,Ye,Ie]=ode15s(@bermod2,[tstart t_max],startval,options,p,b);
    end

    if state2 == 1
        startval = Ye(1,:);
        tstart = Te(1,1);
        T1 = [];
        RES1 = [];
    elseif ~isempty(Ie)
        startval = Ye(1,:);
        tstart = Te(1);
    else
        tstart = t_max;
    end
    Ye = [];
    Ie = [];
    Te = [];

    theend = begin + (size(T1,1)-1);
    RES(begin:theend,1:3) = RES1;
    T(begin:theend,1) = T1;
    begin = theend + 1;
end


%*************************************************************************

%*************************************************************************
% If you want the numbers in mmol/L instead of mg/dL for glucose
%*************************************************************************
% gi = gi./18;
% Gb = Gb./18;
% RES(:,1) = RES(:,1)./18;
%*************************************************************************


%*************************************************************************
```

A.0.13 bermod1.m

```
%*************************************************************************
% Original Minimal Model Matlab Implementation by Esben Friis-Jensen, s042244
% DTU. 2007
```

```
%*************************************************************************
% This function should be solved with a ODESOLVER, this could be one of the
% Matlab standard ODESOLVERS like ODE45 or ODE15s.
%*************************************************************************
% Besides the basic time t, and the res vector it should have 2 more inputs:
%*************************************************************************
% p is a vector of size (6,1), containing the 4 parameters p1, p2, p3 and
% p4,p5,p6
%*************************************************************************
% b is a vector of size (2,1) containing the 2 basal values Gb, Ib
%*************************************************************************

function [dres] = bermod1(t,res,p,b)


dres = zeros(3,1); % a column vector with 5 elements


%*************************************************************************
% The Minimal Model function 1
%
% res(1) = G
% res(2) = X
% res(3) = I
%*************************************************************************

%*************************************************************************
% The equations
%*************************************************************************
dG = -p(1)*(res(1)-b(1)) - res(2)*res(1);
dX = -p(2)*res(2)+ p(3)*(res(3)-b(2));
dI = p(6)*(res(1)-p(5))*t - p(4)*(res(3)-b(2));


dres(1) = dG;
dres(2) = dX;
dres(3) = dI;

%*************************************************************************
```

A.0.14 bermod2.m

```
%*************************************************************************
% Minimal Model Matlab Implementation by Esben Friis-Jensen, s042244
% DTU. 2007
%*************************************************************************
% This function should be solved with a ODESOLVER, this could be one of the
% Matlab standard ODESOLVERS like ODE45 or ODE15s.
%*************************************************************************
% Besides the basic time t, and the res vector it should have 2 more inputs:
%*************************************************************************
% p is a vector of size (6,1), containing the 6 parameters p1, p2, p3 and
```

```
% p4,p5,p6
%************************************************************************
% b is a vector of size (2,1) containing the 2 basal values Gb, Ib
%************************************************************************
function [dres] = bermod2(t,res,p,b)


dres = zeros(3,1); % a column vector with 5 elements


%************************************************************************
% The Minimal Model function 2
%
% res(1) = G
% res(2) = X
% res(3) = I
%************************************************************************

%************************************************************************
% The equations
%************************************************************************
dG = -p(1)*(res(1)-b(1)) - res(2)*res(1);
dX = -p(2)*res(2)+ p(3)*(res(3)-b(2));
dI = - p(4)*(res(3)-b(2));


dres(1) = dG;
dres(2) = dX;
dres(3) = dI;

%************************************************************************
```

A.0.15 bermodevent.m

```
function [rese,isterminal,direction]= bermodevent(t,res,p,b)

rese = res(1)-p(5);
isterminal = 1;
direction = 0;
```

A.0.16 parameters.m

```
%************************************************************************
% Parameters for the Minimal Model
% Implemented by Esben Friis-Jensen, DTU. To be used together with the
% program BERSIMU
%************************************************************************
```

```
% Parameters for the p−vector
%*************************************************************************
% p0 − Theoretical glucose level at time zero above baseline.
% p1 − Glucose effectiveness (Insulin independent).
% p2 − Rate of the spontaneous decrease of tissue glucose uptake ability.
% p3 − increase in uptake ability per unit of insulin conc. over baseline
% (insulin dependent).
% p4 − first order decay rate contant for insulin (n).
% p5 − Target glucose level
% p6 − Rate of pancreatic release of insulin after glucose bolus pr. min.
% pr. mg/dL above target glycemia
%*************************************************************************
% Parameters for the b−vector
%*************************************************************************
% Gb − Baseline glycemia.
% Ib − Baseline insulemnia.V_I − Volume of insulin compartment in liters.
%*************************************************************************


%*************************************************************************
% Here the parameters are setup
%*************************************************************************
if parametertype == 1 % Parameter group 1
    %*********************************************************************
    % p−values
    %*********************************************************************
    p0 = 195; p1 = 0.03082; p2 = 0.02093; p3 = 1.062E−5; p4 = 0.3; p5 = 94; p6 = 0.003349;
    %*********************************************************************
    % b−values
    %*********************************************************************
    Gb = 92; Ib = 7.3;
    %*********************************************************************
    % Initial values
    %*********************************************************************
    G0 = Gb + p0; X0 = 0; I0 = Ib + 396;

end

if parametertype == 2 % Parameter group 2
    %*********************************************************************
    % p−values
    %*********************************************************************
    p0 = 195; p1 = 0.0001; p2 = 0.02093; p3 = 1.062E−5; p4 = 0.3; p5 = 94; p6 = 0.003349;
    %*********************************************************************
    % b−values
    %*********************************************************************
    Gb = 92; Ib = 7.3;
    %*********************************************************************
    % Initial values
    %*********************************************************************
    G0 = Gb + p0; X0 = 0; I0 = Ib + 396;

end

if parametertype == 3 % Parameter group 3
```

```
    %**********************************************************************
    % p-values
    %**********************************************************************
    p0 = 195; p1 = 0.0308; p2 = 0.02093; p3 = 1.062E-6; p4 = 0.3; p5 = 94; p6 = 0.0010;
    %**********************************************************************
    % b-values
    %**********************************************************************
    Gb = 92; Ib = 7.3;
    %**********************************************************************
    % Initial values
    %**********************************************************************
    G0 = Gb + p0; X0 = 0; I0 = Ib + 396;

end

if parametertype == 4 % Parameter group 3
    %**********************************************************************
    % p-values
    %**********************************************************************
    p0 = 195; p1 = 0.0001; p2 = 0.02093; p3 = 1.062E-6; p4 = 0.1; p5 = 94; p6 = 0.0010;
    %**********************************************************************
    % b-values
    %**********************************************************************
    Gb = 92; Ib = 7.3;
    %**********************************************************************
    % Initial values
    %**********************************************************************
    G0 = Gb + p0; X0 = 0; I0 = Ib + 396;

end
```

A.0.17 modbersim.m

```
%**************************************************************************
% MODBERSIM - A simulator for the Modified version of Bergmans minimal
% model. Implemented by Esben Friis-Jensen, s042244, DTU
%**************************************************************************
% The call
%**************************************************************************


function [SG,SI,RES,T] = modbersim(infuse,control,tspan,initval,p,b,a,...
tmeals,mealsam,tin,inam)

%**************************************************************************
% Defining parameters not defined in the GUI
%**************************************************************************
parametersesim
startval = initval;
SG = p(3)/p(2);
SI = p(1);
%**************************************************************************
```

```
% Creation of the table with time events
%*************************************************************************
ttimeevents = [tmeals';tin'];
mtimeevents = [mealsam';zeros(size(tin))'];
itimeevents = [zeros(size(tmeals))';inam'];
timeevents = [ttimeevents,mtimeevents,itimeevents];
timeevents = sortrows(timeevents,[1 3]); % sort the rows according to time.

% find rows where two timeelements are the same.
deletevec = [];
i = 1;
for k=1:size(ttimeevents,1)-1
    if timeevents(k,1) == timeevents(k+1,1)
        if timeevents(k,2) ~= 0 && timeevents(k+1,2) == 0
            timeevents(k+1,2) = timeevents(k,2);
            deletevec(i) = k;
            i = i+1;
        end
        if timeevents(k,3) ~= 0 && timeevents(k+1,3) == 0
            timeevents(k+1,3) = timeevents(k,3);
            deletevec(i) = k;
            i = i+1;
        end
        if timeevents(k,2) == 0 && timeevents(k,3) == 0 && timeevents(k+1,2) ~= 0
            deletevec(i) = k;
            i = i+1;
        end
        if timeevents(k,2) == 0 && timeevents(k,3) == 0 && timeevents(k+1,3) ~= 0
           deletevec(i) = k;
           i = i+1;
        end
        if timeevents(k,2) == 0 && timeevents(k,3) == 0 && timeevents(k+1,2) == 0 ...
           && timeevents(k+1,3) == 0
           deletevec(i) = k;
           i = i+1;
        end
    end
end


% delete the rows not needed anymore
for w = 1:size(deletevec,2)
timeevents(deletevec(w),:) = [];
deletevec = deletevec - 1;
end
initialrow = [tspan(1)-0.1 0 0];
timeevents = [initialrow;timeevents];
%*************************************************************************

%*************************************************************************
% THE SIMULATION
%*************************************************************************
infusion = infuse;
tstart = tspan(1);
size1 = 1;
```

```
size3 = size(startval,2);
options = odeset('RelTol',1e−4,'AbsTol',[1e−8 1e−8 1e−8 1e−8 1e−8 1e−8 1e−8]);
timeinj = 0.5;

        for i = 1:size(timeevents,1)−1

            [T1,RES1] = ode15s(@modbermod,[timeevents(i,1)+timeinj...
            timeevents(i+1,1)],...
            startval,options,p,b,a,infusion,control);

            startval = RES1(end,:);
            startval(1,5) = startval(1,5)+ timeevents(i+1,2);

            infusion = timeevents(i+1,3);

            [T2,RES2] = ode15s(@modbermod,[timeevents(i+1,1),...
            timeevents(i+1,1)+timeinj],...
            startval,options,p,b,a,infusion,control);

            infusion = infuse;


            startval = RES2(end,:);

            size2 = size1 + size(RES1,1)+size(RES2,1)−1;
            RES(size1:size2,1:size3) = [RES1;RES2];
            T(size1:size2,1) = [T1;T2];
            size1 = size2;
            A = i;
        end

        [T3,RES3] = ode15s(@modbermod,...
        [timeevents(A+1,1)+timeinj,tspan(2)],startval,options,p,b,a,...
    infusion,control);
        RES = [RES;RES3];
        T = [T;T3];

% figure
% plot(T,RES(:,1))
% figure
% plot(T,RES(:,5))

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% If you want the numbers in mmol/L instead of mg/dL for glucose
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% gi = gi./18;
% Gb = Gb./18;
% RES(:,1) = RES(:,1)./18;
% RES(:,4) = RES(:,4)./18;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Here the plots are chosen
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%************************************************************************
```

A.0.18 modbermod.m

```
%************************************************************************
% Modified Minimal Model Matlab Implementatin by Esben Friis−Jensen, s042244
% DTU. 2007
%************************************************************************
% This function should be solved with a ODESOLVER, this could be one of the
% Matlab standard ODESOLVERS like ODE45 or ODE15s.
%************************************************************************
% Besides the basic time t, and the res vector it should have 3 more inputs:
%************************************************************************
% p is a vector of size (4,1), containing the 4 parameters p1, p2, p3 and
% p4.
%************************************************************************
% b is a vector of size (6,1) containing the 5 basal values Gb, Ib, Gbsc,
% V_I,Rutln and alpha
%************************************************************************
% infuse is a parameter deciding the size of the infusion u, if infuse is
% negative the artificial pancreas based on a PID controller is used.
%************************************************************************
function [dres] = modbermod(t,res,p,b,a,infuse,control)

global Apid Bpid Cpid Dpid drate Rutln

dres = zeros(7,1); % a column vector with 7 elements


%************************************************************************
% The Modified Minimal Model
%
% res(1) = G
% res(2) = X
% res(3) = I
% res(4) = Gsc
% res(5) = D
% res(6) = x1
% res(7) = x2
%************************************************************************

% ************************************************************************
% Here it is decided to use a controller, or to infuse insulin as a certain
% amount.
%************************************************************************
if control == 0
    u = infuse;
else
   % u = b(3)*p(4)*a(2);
    u = abs(b(3)*p(4)*a(2)+ Cpid*[res(6);res(7)]+Dpid*(res(1)−a(1)));
end
%************************************************************************
```

```
% The equations
%**************************************************************************
dG = -p(1)*res(1)+p(1)*b(1) - res(2)*res(1)+ res(5);
dX = -p(2)*res(2)+ p(3)*(res(3)-b(2));
dI = -p(4)*res(3)+ u/b(3);
dGsc = (res(1)-res(4))/5 - Rutln;
dD = -drate*res(5);
dx = Apid*res(6:7,1) + Bpid*(res(1)-a(1));

dres(1) = dG;
dres(2) = dX;
dres(3) = dI;
dres(4) = dGsc;
dres(5) = dD;
dres(6:7,1) = dx;
%**************************************************************************
```

A.0.19 parametersesim.m

```
%**************************************************************************
% Parameters for the Modified Minimal Model
% Implemented by Esben Friis-Jensen, DTU. To be used together with the
% program MODBERSIM
%**************************************************************************

global Apid Bpid Cpid Dpid drate Rutln

Rutln = 0.74;
%**************************************************************************
% Used for meal simulation
%**************************************************************************
RES = [];
T = [];
drate = 0.05; % this is used for all simulations with meals.
%**************************************************************************
%**************************************************************************
% The PID Controller is loaded here
% *************************************************************************
Kc = 0.2; Ti = 500; Td = 120;

N = 10;
num = conv(Kc,[Ti*Td*(1+1/N) (Ti+Td/N) 1]);
den = conv([Ti 0],[Td/N 1]);

sys = tf(num,den);
[Apid,Bpid,Cpid,Dpid]=ssdata(sys);
```

A.0.20 esim2.m

```
function varargout = esim(varargin)
close all
clear all


%*************************************************************************
% Default values for the parameters
%*************************************************************************

p = zeros(1,4);
b = [92,15,12];
a = [92,15];
tmeals = [1,60,120,180];
mealsam = zeros(1,4);
tin = [1 60 120 180];
inam = zeros(1,4);
tspan = [0,300];
initval = [b(1) 0 b(2) b(1)-5*0.74 0 0 0];
tmaxcheck = max(tmeals(4),tin(4));
tmincheck = min(tmeals(1),tin(1));
newt = [];
decider = 0;
decider2 = 0;


%*************************************************************************
% Creating the window
%*************************************************************************
window = figure('name','ESIM','position',[0,0,1024,745],'menubar',...
'none','toolbar','none',...
    'color',[0,0,0]);
headlinepanel = uipanel('Position',[.04 .91 .42 .07],...
    'backgroundcolor',[0,0,0],'foregroundcolor',[1,1,1],'bordertype'...
,'none');
tspanpanel = uipanel('Title','Time Span','FontSize',12,...
    'Position',[.04 .84 .42 .07],'fontweight','bold',...
    'backgroundcolor',[0,0,0],'foregroundcolor',[1,1,1]);
initvalpanel = uipanel('Title','Initial Values','FontSize',12,...
    'Position',[.04 .765 .42 .07],'fontweight','bold',...
    'backgroundcolor',[0,0,0],'foregroundcolor',[1,1,1]);
parameterpanel = uipanel('Title','Subject','FontSize',12,...
    'Position',[.04 .60 .42 .16],'fontweight','bold',...
    'backgroundcolor',[0,0,0],'foregroundcolor',[1,1,1]);
unitpanel = uipanel('Title','Units','FontSize',12,...
    'Position',[.04 .535 .42 .06],'fontweight','bold',...
    'backgroundcolor',[0,0,0],'foregroundcolor',[1,1,1]);
mealpanel = uipanel('Title','Meals','FontSize',12,...
    'Position',[.04 .36 .42 .17],'fontweight','bold',...
    'backgroundcolor',[0,0,0],'foregroundcolor',[1,1,1]);
controllerpanel = uipanel('Title','Controller','FontSize',12,...
    'Position',[.04 .29 .42 .06],'fontweight','bold',...
    'backgroundcolor',[0,0,0],'foregroundcolor',[1,1,1]);
infusionpanel = uipanel('Title','Infusion','FontSize',12,...
    'Position',[.04 .16 .42 .12],'fontweight','bold',...
```

```
    'backgroundcolor',[0,0,0],'foregroundcolor',[1,1,1]);
simulatorpanel = uipanel('Position',[.04 .05 .2 .10],'fontweight','bold',...
    'backgroundcolor',[0,0,0],'foregroundcolor',[1,1,1],'bordertype','none');
datapanel = uipanel('title','Data','fontsize',12,'Position',...
[.25 .05 .21 .10],'fontweight','bold',...
    'backgroundcolor',[0,0,0],'foregroundcolor',[1,1,1]);
axespanel = uipanel('Position',[0.47 .05 .53 0.95],'fontweight','bold',...
    'backgroundcolor',[0,0,0],'foregroundcolor',[1,1,1],'bordertype','none');
setpointpanel = uipanel('Title','Setpoints','FontSize',12,...
    'Position',[.04 .20 .42 .07],'fontweight','bold',...
    'backgroundcolor',[0,0,0],'foregroundcolor',[1,1,1]);
set(setpointpanel,'visible','off')

%************************************************************************
% Defining the default colors
%************************************************************************
bcwindow = [0,0,0];
fcwindow = [1,1,1];
bctspanpanel = [0,0,0];
fctspanpanel = [1,1,1];
bcparameterpanel = [0,0,0];
fcparameterpanel = [1,1,1];
bcmealpanel = [0,0,0];
fcmealpanel = [1,1,1];
bccontrollerpanel = [0,0,0];
fccontrollerpanel = [1,1,1];
bcinfusionpanel = [0,0,0];
fcinfusionpanel = [1,1,1];
bccontrollerpanel = [0,0,0];
fccontrollerpanel = [1,1,1];
%************************************************************************


% bccontrollerpanel = [0,0.3,0.6];
% fccontrollerpanel = [1,1,1];

%************************************************************************
% Constructing the components
%************************************************************************
%AXES
%************************************************************************
ax1 = axes('Parent',axespanel,'units','pixels',...
    'Position',[55 480 450 150],'nextplot','replace');
ax2 = axes('Parent',axespanel,'units','pixels',...
    'Position',[55 260 450 150],'nextplot','replace');
ax3 = axes('Parent',axespanel,'units','pixels',...
    'position',[55 40 450 150],'nextplot','replace');
%************************************************************************
% STATIC TEXT
%************************************************************************
Headline = uicontrol('parent',headlinepanel,'Style','text','String','ESIM',...
'Position',[140,2,150,50],...
    'fontsize',34,'backgroundcolor',bcwindow,...,
    'foregroundcolor',fcwindow);
ax1header = uicontrol('parent',axespanel,'Style','text','String',...
```

```
'Blood glucose level','Position',[180,640,150,20],...
    'fontsize',12,'backgroundcolor',[0,0,0],...,
    'foregroundcolor',[1,1,1],'fontweight','bold');
ax2header = uicontrol('parent',axespanel,'Style','text','String'....,
'Active Insulin Effect','Position',[180,420,150,20],...
    'fontsize',12,'backgroundcolor',[0,0,0],...,
    'foregroundcolor',[1,1,1],'fontweight','bold');
ax3header = uicontrol('parent',axespanel,'Style','text','String',...
'Insulin Level','Position',[180,200,150,20],...
    'fontsize',12,'backgroundcolor',[0,0,0],...,
    'foregroundcolor',[1,1,1],'fontweight','bold');


%**************************************************************************
% STATIC TEXT FOR TSPAN
%**************************************************************************
tminimum = uicontrol('parent',tspanpanel,'Style','text','String',...
'Starttime [min]:','Position',[12,8,100,20],...
    'fontsize',10,'backgroundcolor',bctspanpanel,...,
    'foregroundcolor',fctspanpanel);
tmaximum = uicontrol('parent',tspanpanel,'Style','text','String',...
'End time [min]:','Position',[210,8,100,20],...
    'fontsize',10,'backgroundcolor',bctspanpanel,...,
    'foregroundcolor',fctspanpanel);
%**************************************************************************
% EDIT TEXT FOR TSPAN
%**************************************************************************
tminimumedit = uicontrol('parent',tspanpanel,'Style','edit','String',tspan(1),...
    'Position',[120,8,60,20],...
    'Callback',{@tminimum_Callback});
tmaximumedit = uicontrol('parent',tspanpanel,'Style','edit','String',tspan(2),...
    'Position',[320,8,60,20],...
    'Callback',{@tmaximum_Callback});

%**************************************************************************
% STATIC TEXT FOR INITVAL
%**************************************************************************
G0 = uicontrol('parent',initvalpanel,'Style','text','String','G0:',...
'Position',[20,8,20,20],...
    'fontsize',10,'backgroundcolor',bctspanpanel,...,
    'foregroundcolor',fctspanpanel);
I0 = uicontrol('parent',initvalpanel,'Style','text','String','I0:',...
'Position',[175,8,20,20],...
    'fontsize',10,'backgroundcolor',bctspanpanel,...,
    'foregroundcolor',fctspanpanel);

Gsc0 = uicontrol('parent',initvalpanel,'Style','text','String','Gsc0:',...
'Position',[300,8,40,20],...
    'fontsize',10,'backgroundcolor',bctspanpanel,...,
    'foregroundcolor',fctspanpanel);
%**************************************************************************
% EDIT TEXT FOR INITVAL
%**************************************************************************
G0edit = uicontrol('parent',initvalpanel,'Style','edit','String',initval(1),...
    'Position',[50,8,60,20],...
```

```
    'Callback',{@G0_Callback});
I0edit = uicontrol('parent',initvalpanel,'Style','edit','String',initval(3),...
    'Position',[210,8,60,20],...
    'Callback',{@I0_Callback});
Gsc0edit = uicontrol('parent',initvalpanel,'Style','edit','String',initval(4),...
    'Position',[350,8,60,20],...
    'Callback',{@Gsc0_Callback});

%************************************************************************
% POPUPMENU-LOADING OF PARAMETERS
%************************************************************************
popup = uicontrol('parent',parameterpanel,'Style','popupmenu',...
            'String',{'Default','Type 1 (use control)',...
            'Type 1 (No control)','Last example'},...
            'Position',[170,75,100,20],...
            'Callback',{@popup_Callback});

%************************************************************************
% STATIC TEXT FOR PARAMETERS (POPUP)
%************************************************************************
popuptext = uicontrol('parent',parameterpanel,'Style','text',...
'String','Predefined parameters:','Position',[15,75,150,20],...
    'fontsize',10,'backgroundcolor',bcparameterpanel,...,
    'foregroundcolor',fcparameterpanel);


%************************************************************************
% STATIC TEXT FOR PARAMETERS
%************************************************************************
p1 = uicontrol('parent',parameterpanel,'Style','text','String',...
'p1:','Position',[20,45,20,20],...
    'fontsize',10,'backgroundcolor',bcparameterpanel,...,
    'foregroundcolor',fcparameterpanel);
p2 = uicontrol('parent',parameterpanel,'Style','text','String',...
'p2:','Position',[120,45,20,20],...
    'fontsize',10,'backgroundcolor',bcparameterpanel,...,
    'foregroundcolor',fcparameterpanel);
p3 = uicontrol('parent',parameterpanel,'Style','text','String',...
'p3:','Position',[220,45,20,20],...
    'fontsize',10,'backgroundcolor',bcparameterpanel,...,
    'foregroundcolor',fcparameterpanel);
p4 = uicontrol('parent',parameterpanel,'Style','text','String',...
'p4:','Position',[320,45,20,20],...
    'fontsize',10,'backgroundcolor',bcparameterpanel,...
    'foregroundcolor',fcparameterpanel);
Gb = uicontrol('parent',parameterpanel,'Style','text','String',...
    'Gb:','Position',[20,10,20,20],...
    'fontsize',10,'backgroundcolor',bcparameterpanel,...,
    'foregroundcolor',fcparameterpanel);
Gbunit = uicontrol('parent',parameterpanel,'Style','text','String'...
'[mg/dL]','Position',[95,10,50,20],...
    'fontsize',10,'backgroundcolor',bcparameterpanel,...
    ,'foregroundcolor',fcparameterpanel);
Gbunit2 = uicontrol('parent',parameterpanel,'Style','text','String',...
'[mmol/L]','Position',[95,10,51,20],...
```

```
    'fontsize',10,'backgroundcolor',bcparameterpanel,...,
    'foregroundcolor',fcparameterpanel,'visible','off');
Ib = uicontrol('parent',parameterpanel,'Style','text','String','Ib:',...
'Position',[175,40,10,20],...
    'fontsize',10,'backgroundcolor',bcparameterpanel,...,
    'foregroundcolor',fcparameterpanel);
Ibunit = uicontrol('parent',parameterpanel,'Style','text','String',...
'[mU/L]','Position',[255,10,50,20],...
    'fontsize',10,'backgroundcolor',bcparameterpanel,...,
    'foregroundcolor',fcparameterpanel);
VI = uicontrol('parent',parameterpanel,'Style','text','String','VI:'...
,'Position',[320,10,20,20],...
    'fontsize',10,'backgroundcolor',bcparameterpanel,...,
    'foregroundcolor',fcparameterpanel);
VIunit = uicontrol('parent',parameterpanel,'Style','text','String',...
'[L]','Position',[395,10,20,20],...
    'fontsize',10,'backgroundcolor',bcparameterpanel,...,
    'foregroundcolor',fcparameterpanel);
%*************************************************************************
%EDIT TEXT BOXES USED FOR PARAMETERS
%*************************************************************************
p1edit = uicontrol('parent',parameterpanel,'Style','edit','String',p(1),...
    'Position',[50,45,60,20],...
    'Callback',{@p1_Callback});
p2edit = uicontrol('parent',parameterpanel,'Style','edit','String',p(2),...
    'Position',[150,45,60,20],...
    'Callback',{@p2_Callback});
p3edit = uicontrol('parent',parameterpanel,'Style','edit','String',p(3),...
    'Position',[250,45,60,20],...
    'Callback',{@p3_Callback});
p4edit = uicontrol('parent',parameterpanel,'Style','edit','String',p(4),...
    'Position',[350,45,60,20],...
    'Callback',{@p4_Callback});
Gbedit = uicontrol('parent',parameterpanel,'Style','edit','String',b(1),...
    'Position',[50,10,40,20],...
    'Callback',{@Gb_Callback});
Ibedit = uicontrol('parent',parameterpanel,'Style','edit','String',b(2),...
    'Position',[210,10,40,20],...
    'Callback',{@Ib_Callback});
VIedit = uicontrol('parent',parameterpanel,'Style','edit','String',b(3),...
    'Position',[350,10,40,20],...
    'Callback',{@VI_Callback});

%*************************************************************************
% CHECKBOX FOR UNITS
%*************************************************************************
checkboxunits = uicontrol('parent',unitpanel,'Style','checkbox','String',...
'Use mmol/L instead of mg/dL','Position',[25,6,300,20],...
    'fontsize',10,'backgroundcolor',bccontrollerpanel,...,
    'foregroundcolor',fccontrollerpanel,'Callback',...
{@checkboxunits_Callback});

%*************************************************************************
% STATIC TEXT FOR MEALS
%*************************************************************************
```

```
time = uicontrol('parent',mealpanel,'Style','text','String','Time [min]:',...
'Position',[20,70,70,20],...
    'fontsize',10,'backgroundcolor',bcmealpanel,...,
    'foregroundcolor',fcmealpanel);
meal = uicontrol('parent',mealpanel,'Style','text','String',...
'Meal [mg/dL/min]:','Position',[20,30,110,20],...
    'fontsize',10,'backgroundcolor',bcmealpanel,...,
    'foregroundcolor',fcmealpanel);
meal2 = uicontrol('parent',mealpanel,'Style','text','String',...
    'Meal [mmol/L/min]:','Position',[20,30,110,20],...
    'fontsize',10,'backgroundcolor',bcmealpanel,...,
    'foregroundcolor',fcmealpanel,'visible','off');
meal3 = uicontrol('parent',mealpanel,'Style','text','String',...
    'The numbers given in the meal fields are initial rates which decays...
exponentially','Position',[20,10,400,12],...
    'fontsize',8,'backgroundcolor',bcmealpanel,...,
    'foregroundcolor',fcmealpanel);
%*************************************************************************
%EDIT TEXT BOXES USED FOR MEALS
%*************************************************************************
mealtime1edit = uicontrol('parent',mealpanel,'Style','edit','String',tmeals(1),...
    'Position',[140,70,60,20],...
    'Callback',{@mealtime1_Callback});
mealtime2edit = uicontrol('parent',mealpanel,'Style','edit','String',tmeals(2),...
    'Position',[210,70,60,20],...
    'Callback',{@mealtime2_Callback});
mealtime3edit = uicontrol('parent',mealpanel,'Style','edit','String',tmeals(3),...
    'Position',[280,70,60,20],...
    'Callback',{@mealtime3_Callback});
mealtime4edit = uicontrol('parent',mealpanel,'Style','edit','String',tmeals(4),...
    'Position',[350,70,60,20],...
    'Callback',{@mealtime4_Callback});
meal1edit = uicontrol('parent',mealpanel,'Style','edit','String',mealsam(1),...
    'Position',[140,30,60,20],...
    'Callback',{@meal1_Callback});
meal2edit = uicontrol('parent',mealpanel,'Style','edit','String',mealsam(2),...
    'Position',[210,30,60,20],...
    'Callback',{@meal2_Callback});
meal3edit = uicontrol('parent',mealpanel,'Style','edit','String',mealsam(3),...
    'Position',[280,30,60,20],...
    'Callback',{@meal3_Callback});
meal4edit = uicontrol('parent',mealpanel,'Style','edit','String',mealsam(4),...
    'Position',[350,30,60,20],...
    'Callback',{@meal4_Callback});


%*************************************************************************
% CHECKBOX FOR CONTROLLER CHOICE
%*************************************************************************
checkbox1 = uicontrol('parent',controllerpanel,'Style','checkbox',...
'String','Use a controller to control insulin injection','Position',...
[25,6,300,20],...
    'fontsize',10,'backgroundcolor',bccontrollerpanel,...,
    'foregroundcolor',fccontrollerpanel,'Callback',{@Checkbox1_Callback});
```

```
%*************************************************************************
% STATIC TEXT FOR INFUSION
%*************************************************************************
time = uicontrol('parent',infusionpanel,'Style','text','String',...
'Time [min]:','Position',[20,45,70,20],...
    'fontsize',10,'backgroundcolor',bcinfusionpanel,...,
    'foregroundcolor',fcinfusionpanel);
infuse = uicontrol('parent',infusionpanel,'Style','text','String',...
'Infusion [mU]:','Position',[10,10,110,20],...
    'fontsize',10,'backgroundcolor',bcinfusionpanel,...,
    'foregroundcolor',fcinfusionpanel);
%*************************************************************************
%EDIT TEXT BOXES USED FOR INFUSION
%*************************************************************************
infusetime1edit = uicontrol('parent',infusionpanel,'Style','edit',...
'String',tin(1),...
    'Position',[140,45,60,20],...
    'Callback',{@infusetime1_Callback});
infusetime2edit = uicontrol('parent',infusionpanel,'Style','edit',...
'String',tin(2),...
    'Position',[210,45,60,20],...
    'Callback',{@infusetime2_Callback});
infusetime3edit = uicontrol('parent',infusionpanel,'Style','edit',...
    'String',tin(3),...
    'Position',[280,45,60,20],...
    'Callback',{@infusetime3_Callback});
infusetime4edit = uicontrol('parent',infusionpanel,'Style','edit',...
    'String',tin(4),...
    'Position',[350,45,60,20],...
    'Callback',{@infusetime4_Callback});
infuse1edit = uicontrol('parent',infusionpanel,'Style','edit',...
    'String',inam(1),...
    'Position',[140,10,60,20],...
    'Callback',{@infuse1_Callback});
infuse2edit = uicontrol('parent',infusionpanel,'Style','edit',...
    'String',inam(2),...
    'Position',[210,10,60,20],...
    'Callback',{@infuse2_Callback});
infuse3edit = uicontrol('parent',infusionpanel,'Style','edit',...
    'String',inam(3),...
    'Position',[280,10,60,20],...
    'Callback',{@infuse3_Callback});
infuse4edit = uicontrol('parent',infusionpanel,'Style','edit','String',inam(4),...
    'Position',[350,10,60,20],...
    'Callback',{@infuse4_Callback});

%*************************************************************************
% STATIC TEXT FOR SETPOINTS
%*************************************************************************
setpoint1 = uicontrol('parent',setpointpanel,'Style','text','String',...
'G setpoint:','Position',[12,8,100,20],...
    'fontsize',10,'backgroundcolor',bctspanpanel,...,
    'foregroundcolor',fctspanpanel);
setpoint2 = uicontrol('parent',setpointpanel,'Style','text','String',...
```

```
'I setpoint:','Position',[210,8,100,20],...
    'fontsize',10,'backgroundcolor',bctspanpanel,...,
    'foregroundcolor',fctspanpanel);
%**************************************************************************
% EDIT TEXT FOR SETPOINTS
%**************************************************************************
setpoint1edit = uicontrol('parent',setpointpanel,'Style',...
'edit','String',a(1),...
    'Position',[120,8,60,20],...
    'Callback',{@setpoint1_Callback});
setpoint2edit = uicontrol('parent',setpointpanel,'Style',...
'edit','String',a(2),...
    'Position',[320,8,60,20],...
    'Callback',{@setpoint2_Callback});


%**************************************************************************
% PUSH BUTTON
%**************************************************************************
simulate1 = uicontrol('parent',simulatorpanel,'Style','pushbutton',...
'String','SIMULATE',...
    'Position',[2,20,200,30],'fontsize',12,'fontweight','bold',...
    'backgroundcolor',[0,0.3,0.6],'foregroundcolor',[1,1,1],...
    'Callback',{@simulate1_Callback});

%**************************************************************************
% STATIC TEXT FOR DATA
%**************************************************************************
ge1 = uicontrol('parent',datapanel,'Style','text','String',...
'Glucose effectiveness:','Position',[1,30,150,20],...
    'fontsize',10,'backgroundcolor',bcinfusionpanel,...,
    'foregroundcolor',fcinfusionpanel);
is1 = uicontrol('parent',datapanel,'Style','text','String',...
'Insulin sensitivity:','Position',[10,10,110,20],...
    'fontsize',10,'backgroundcolor',bcinfusionpanel,...,
    'foregroundcolor',fcinfusionpanel);
ge2 = uicontrol('parent',datapanel,'Style','text','String',p(1),...
    'Position',[140,30,50,20],...
    'fontsize',10,'backgroundcolor',bcinfusionpanel,...,
    'foregroundcolor',fcinfusionpanel);

is2 = uicontrol('parent',datapanel,'Style','text','String',p(3)/p(1),...
'Position',[140,10,50,20],...
    'fontsize',10,'backgroundcolor',bcinfusionpanel,...,
    'foregroundcolor',fcinfusionpanel);




%**************************************************************************
% CALLBACKS FOR TSPAN
%**************************************************************************
    function tminimum_Callback(hObject,eventdata)
        user_entry = str2double(get(hObject,'string'));
        if isnan(user_entry)
```

```
        errordlg('You must enter a numeric value in field starttime',...
        'Bad Input','modal')
        set(tminimumedit,'String',tspan(1))
        return
    end
    if user_entry >= tspan(2)
        errordlg('Not a valid time span','Bad Input','modal')
        set(tminimumedit,'String',tspan(1))
        return
    end
    if user_entry < 0
        errordlg(...
        'The value in the field starttime must be positive or 0',...
    'Bad Input','modal')
        set(tminimumedit,'String',tspan(1))
        return
    end
    tspan(1) = user_entry;
    % changing the meal and infusion time, so the default values fit
    % inside the new timespan
    newt = linspace(tspan(1),tspan(2),6);
    tmeals(1) = newt(2);tmeals(2) = newt(3);tmeals(3)=newt(4);
    tmeals(4) = newt(5);
    tin(1) = newt(2);tin(2) = newt(3);tin(3)=newt(4);tin(4) = newt(5);
    set(mealtime1edit,'String',tmeals(1))
    set(mealtime2edit,'String',tmeals(2))
    set(mealtime3edit,'String',tmeals(3))
    set(mealtime4edit,'String',tmeals(4))
    set(infusetime1edit,'String',tin(1))
    set(infusetime2edit,'String',tin(2))
    set(infusetime3edit,'String',tin(3))
    set(infusetime4edit,'String',tin(4))
end


function tmaximum_Callback(hObject,eventdata)
    user_entry = str2double(get(hObject,'string'));
    if isnan(user_entry)
        errordlg('You must enter a numeric value in field end time',...
            'Bad Input','modal')
        set(tmaximumedit,'String',tspan(2))
        return
    end
    if user_entry <= tspan(1)
        errordlg('Not a valid time span','Bad Input','modal')
        set(tmaximumedit,'String',tspan(2))
        return
    end
    tspan(2) = user_entry;

    newt = linspace(tspan(1),tspan(2),6);
    tmeals(1) = newt(2);tmeals(2) = newt(3);tmeals(3)=newt(4);
    tmeals(4) = newt(5);
    tin(1) = newt(2);tin(2) = newt(3);tin(3)=newt(4);tin(4) = newt(5);
    set(mealtime1edit,'String',tmeals(1))
```

```
        set(mealtime2edit,'String',tmeals(2))
        set(mealtime3edit,'String',tmeals(3))
        set(mealtime4edit,'String',tmeals(4))
        set(infusetime1edit,'String',tin(1))
        set(infusetime2edit,'String',tin(2))
        set(infusetime3edit,'String',tin(3))
        set(infusetime4edit,'String',tin(4))
    end
%**************************************************************************
% CALLBACKS FOR INITVAL
%**************************************************************************
function G0_Callback(hObject,eventdata)
        user_entry = str2double(get(hObject,'string'));
        if isnan(user_entry)
            errordlg('You must enter a numeric value in field G0',...
            'Bad Input','modal')
            set(G0edit,'String',initval(1))
            return
        end
        if user_entry < 0
            errordlg('You must have a positive G0','Bad Input','modal')
            set(G0edit,'String',initval(1))
            return
        end
        initval(1) = user_entry;
end

function I0_Callback(hObject,eventdata)
        user_entry = str2double(get(hObject,'string'));
        if isnan(user_entry)
            errordlg('You must enter a numeric value in field I0',...
                'Bad Input','modal')
            set(I0edit,'String',initval(3))
            return
        end
        if user_entry < 0
            errordlg('You must have a positive I0','Bad Input','modal')
            set(I0edit,'String',initval(3))
            return
        end
        initval(3) = user_entry;
end

function Gsc0_Callback(hObject,eventdata)
        user_entry = str2double(get(hObject,'string'));
        if isnan(user_entry)
            errordlg('You must enter a numeric value in field Gsc0',...
            'Bad Input','modal')
            set(X0edit,'String',initval(4))
            return
        end
        if user_entry < 0
            errordlg('You must have a positive Gsc0','Bad Input','modal')
            set(Gsc0edit,'String',initval(4))
            return
```

```
        end
        initval(4) = user_entry;
    end

%*************************************************************************
% CALLBACKS FOR PARAMETERS
%*************************************************************************
    function popup_Callback(hObject,eventdata)
        str = get(popup, 'String');
        val = get(popup,'Value');
        switch str{val};
            case 'Default'
                p(1) = 0; p(2) = 0; p(3) = 0; p(4) = 0;
                b(1) = 92; b(2) = 15; b(3) = 12; a(1) = 92; a(2) = 15;
                set(p1edit,'string',p(1));
                set(p2edit,'string',p(2));
                set(p3edit,'string',p(3));
                set(p4edit,'string',p(4));
                set(Gbedit,'string',b(1));
                set(Ibedit,'string',b(2));
                set(setpoint1edit,'string',a(1));
                set(setpoint2edit,'string',a(2));
            case 'Type 1 (use control)'
                p(1) = 0.03; p(2) = 0.03; p(3) = 0.00005; p(4) = 0.1;
                b(1) = 81.3; b(2) = 15; b(3) = 12; a(1) = 81.3; a(2) = 15;
                set(p1edit,'string',p(1));
                set(p2edit,'string',p(2));
                set(p3edit,'string',p(3));
                set(p4edit,'string',p(4));
                set(Gbedit,'string',b(1));
                set(Ibedit,'string',b(2));
                set(setpoint1edit,'string',a(1));
                set(setpoint2edit,'string',a(2));
            case 'Type 1 (No control)' % User selects Membrane.
                p(1) = 0.001; p(2) = 0.03; p(3) = 0.00005; p(4) = 0.1;
                b(1) = 81.3; b(2) = 0; b(3) = 12; a(1) = 81.3; a(2) = 15;
                set(p1edit,'string',p(1));
                set(p2edit,'string',p(2));
                set(p3edit,'string',p(3));
                set(p4edit,'string',p(4));
                set(Gbedit,'string',b(1));
                set(Ibedit,'string',b(2));
                set(setpoint1edit,'string',a(1));
                set(setpoint2edit,'string',a(2));
            case 'Last example' %
                p(1) = 0.03; p(2) = 0.03; p(3) = 0.00005; p(4) = 0.1;
                b(1) = 81.3; b(2) = 15; b(3) = 12; a(1) = 81.3; a(2) = 15;
                set(p1edit,'string',p(1));
                set(p2edit,'string',p(2));
                set(p3edit,'string',p(3));
                set(p4edit,'string',p(4));
                set(Gbedit,'string',b(1));
                set(Ibedit,'string',b(2));
                set(setpoint1edit,'string',a(1));
                set(setpoint2edit,'string',a(2));
```

```
    end
end



function p1_Callback(hObject,eventdata)
    user_entry = str2double(get(hObject,'string'));
    if isnan(user_entry)
        errordlg('You must enter a numeric value in field p1'...
        ,'Bad Input','modal')
        set(p1edit,'String',p(1))
        return
    end
    p(1) = user_entry;
    set(ge2,'string',p(1))
end

function p2_Callback(hObject,eventdata)
    user_entry = str2double(get(hObject,'string'));
    if isnan(user_entry)
        errordlg('You must enter a numeric value in field p2',...
        'Bad Input','modal')
        set(p2edit,'String',p(2))
        return
    end
    p(2) = user_entry;
    set(is2,'string',p(3)/p(2))
end

function p3_Callback(hObject,eventdata)
    user_entry = str2double(get(hObject,'string'));
    if isnan(user_entry)
        errordlg('You must enter a numeric value in field p3',...
        'Bad Input','modal')
        set(p3edit,'String',p(3))
        return
    end
    p(3) = user_entry;
    set(is2,'string',p(3)/p(2))
end

function p4_Callback(hObject,eventdata)
    user_entry = str2double(get(hObject,'string'));
    if isnan(user_entry)
        errordlg('You must enter a numeric value in field p4',...
            'Bad Input','modal')
        set(p4edit,'String',p(4))
        return
    end
    p(4) = user_entry;
end

function Gb_Callback(hObject,eventdata)
    user_entry = str2double(get(hObject,'string'));
    if isnan(user_entry)
```

```
            errordlg('You must enter a numeric value in field Gb','Bad Input',...
            'modal')
            set(Gbedit,'String',b(1))
            return
        end
        if user_entry < 0
            errordlg('You must have a positive Gb','Bad Input','modal')
            set(Gbedit,'String',b(1))
            return
        end
        b(1) = user_entry;
    end

    function Ib_Callback(hObject,eventdata)
        user_entry = str2double(get(hObject,'string'));
        if isnan(user_entry)
            errordlg('You must enter a numeric value in field Ib',...
                'Bad Input','modal')
            set(Ibedit,'String',b(2))
            return
        end
        if user_entry < 0
            errordlg('You must have a positive Ib','Bad Input','modal')
            set(Ibedit,'String',b(2))
            return
        end
        b(2) = user_entry;
    end

    function VI_Callback(hObject,eventdata)
        user_entry = str2double(get(hObject,'string'));
        if isnan(user_entry)
            errordlg('You must enter a numeric value in field VI',...
                'Bad Input','modal')
            set(VIedit,'String',b(3))
            return
        end
        if user_entry < 0
            errordlg('You must have a positive VI','Bad Input','modal')
            set(VIedit,'String',b(3))
            return
        end
        b(3) = user_entry;
    end
%**************************************************************************
%**************************************************************************
% CALLBACKS FOR MEALS
%**************************************************************************
    function mealtime1_Callback(hObject,eventdata)
        user_entry = str2double(get(hObject,'string'));
        if isnan(user_entry)
            errordlg('You must enter a numeric value in field time ...
            (meal 1)','Bad Input','modal')
            set(mealtime1edit,'string',tmeals(1))
            return
```

```
    end
    if tspan(2) <= user_entry
        errordlg('You must enter a time inside the timespan in...
        field time (meal 1)','Bad Input','modal')
        set(mealtime1edit,'string',tmeals(1))
        return
    end
    if tspan(1) >= user_entry
        errordlg('You must enter a time inside the timespan in field ..
        time (meal 1)','Bad Input','modal')
        set(mealtime1edit,'string',tmeals(1))
        return
    end
    tmeals(1) = user_entry;
end

function mealtime2_Callback(hObject,eventdata)
    user_entry = str2double(get(hObject,'string'));
    if isnan(user_entry)
        errordlg('You must enter a numeric value in field time
        (meal 2)','Bad Input','modal')
        set(mealtime2edit,'string',tmeals(2))
        return
    end
    if tspan(2) <= user_entry
        errordlg('You must enter a time inside the timespan in
        field time (meal 2)','Bad Input','modal')
        set(mealtime2edit,'string',tmeals(2))
        return
    end
    if tmeals(1) >= user_entry
        errordlg('You must enter a time larger than the one in
        the field time (meal 1)','Bad Input','modal')
        set(mealtime2edit,'string',tmeals(2))
        return
    end
    tmeals(2) = user_entry;
end

function mealtime3_Callback(hObject,eventdata)
    user_entry = str2double(get(hObject,'string'));
    if isnan(user_entry)
        errordlg('You must enter a numeric value in field time
        (meal 3)','Bad Input','modal')
        set(mealtime3edit,'string',tmeals(3))
        return
    end
    if tspan(2) <= user_entry
        errordlg('You must enter a time inside the timespan in
        field time (meal 3)','Bad Input','modal')
        set(mealtime3edit,'string',tmeals(3))
        return
    end
    if tmeals(2) >= user_entry
        errordlg('You must enter a time larger than the one in
```

```
        the field time (meal 2)','Bad Input','modal')
        set(mealtime3edit,'string',tmeals(3))
        return
    end
    tmeals(3) = user_entry;
end

function mealtime4_Callback(hObject,eventdata)
    user_entry = str2double(get(hObject,'string'));
    if isnan(user_entry)
        errordlg('You must enter a numeric value in field time
        (meal 4)','Bad Input','modal')
        set(mealtime4edit,'string',tmeals(4))
        return
    end
    if tspan(2) <= user_entry
        errordlg('You must enter a time inside the timespan in
        field time (meal 4)','Bad Input','modal')
        return
    end
    if tmeals(3) >= user_entry
        errordlg('You must enter a time larger than the one in
        the field time (meal 3)','Bad Input','modal')
        set(mealtime4edit,'string',tmeals(4))
        return
    end
    tmeals(4) = user_entry;
end

function meal1_Callback(hObject,eventdata)
    user_entry = str2double(get(hObject,'string'));
    if isnan(user_entry)
        errordlg('You must enter a numeric value in field meal 1',...
        'Bad Input','modal')
        set(meal1edit,'string',mealsam(1))
        return
    end
    if user_entry < 0
        errordlg('You must have a positive initial meal rate
        in field meal 1','Bad Input','modal')
        set(meal1edit,'string',mealsam(1))
        return
    end
    mealsam(1) = user_entry;
end

function meal2_Callback(hObject,eventdata)
    user_entry = str2double(get(hObject,'string'));
    if isnan(user_entry)
        errordlg('You must enter a numeric value in field meal 2',...
        'Bad Input','modal')
        set(meal2edit,'string',mealsam(2))
        return
    end
    if user_entry < 0
```

```
            errordlg('You must have a positive initial meal rate in
            field meal 2','Bad Input','modal')
            set(meal2edit,'string',mealsam(2))
            return
        end
        mealsam(2) = user_entry;
    end

    function meal3_Callback(hObject,eventdata)
        user_entry = str2double(get(hObject,'string'));
        if isnan(user_entry)
            errordlg('You must enter a numeric value in field meal
            3','Bad Input','modal')
            set(meal3edit,'string',mealsam(3))
            return
        end
        if user_entry < 0
            errordlg('You must have a positive initial meal rate in
            field meal 3','Bad Input','modal')
            set(meal3edit,'string',mealsam(3))
            return
        end
        mealsam(3) = user_entry;
    end

    function meal4_Callback(hObject,eventdata)
        user_entry = str2double(get(hObject,'string'));
        if isnan(user_entry)
            errordlg('You must enter a numeric value in field meal 4'...
            ,'Bad Input','modal')
            set(meal4edit,'string',mealsam(4))
            return
        end
        if user_entry < 0
            errordlg('You must have a positive initial meal rate in
            field meal 4','Bad Input','modal')
            set(meal4edit,'string',mealsam(4))
            return
        end
        mealsam(4) = user_entry;
    end
%**************************************************************************

%**************************************************************************
% CALLBACKS FOR CHECKBOX1 /controller
%**************************************************************************

function Checkbox1_Callback(hObject,eventdata)
if (get(hObject,'Value') == get(hObject,'Max'))
    set(infusionpanel,'visible','off')
    set(setpointpanel,'visible','on')
    decider = 1;
else
    set(infusionpanel,'visible','on')
     set(setpointpanel,'visible','off')
```

```
   decider = 0;
end
end

%**************************************************************************
% CALLBACKS FOR INFUSION
%**************************************************************************
    function infusetime1_Callback(hObject,eventdata)
        user_entry = str2double(get(hObject,'string'));
        if isnan(user_entry)
            errordlg('You must enter a numeric value in field time
            (infusion 1)','Bad Input','modal')
            set(infusetime1edit,'string',tin(1))
            return
        end
        if tspan(2) <= user_entry
            errordlg('You must enter a time inside the timespan in
            field time (infusion 1)','Bad Input','modal')
            set(infusetime1edit,'string',tin(1))
            return
        end
        if tspan(1) >= user_entry
            errordlg('You must enter a time inside the timespan in
            field time (infusion 1)','Bad Input','modal')
            set(infusetime1edit,'string',tin(1))
            return
        end
        tin(1) = user_entry;
    end

    function infusetime2_Callback(hObject,eventdata)
        user_entry = str2double(get(hObject,'string'));
        if isnan(user_entry)
            errordlg('You must enter a numeric value in field
            time (infusion 2)','Bad Input','modal')
            set(infusetime2edit,'string',tin(2))
            return
        end
        if tspan(2) <= user_entry
            errordlg('You must enter a time inside the timespan
            in field time (infusion 2)','Bad Input','modal')
            set(infusetime2edit,'string',tin(2))
            return
        end
        if tin(1) >= user_entry
            errordlg('You must enter a time larger than the one
            in the field time (infusion 1)','Bad Input','modal')
            set(infusetime2edit,'string',tin(2))
            return
        end
        tin(2) = user_entry;
    end

    function infusetime3_Callback(hObject,eventdata)
        user_entry = str2double(get(hObject,'string'));
```

```
    if isnan(user_entry)
        errordlg('You must enter a numeric value in field
        time (infusion 3)','Bad Input','modal')
        set(infusetime3edit,'string',tin(3))
        return
    end
    if tspan(2) <= user_entry
        errordlg('You must enter a time inside the timespan
        in field time (infusion 3)','Bad Input','modal')
        set(infusetime3edit,'string',tin(3))
        return
    end
    if tin(2) >= user_entry
        errordlg('You must enter a time larger than the one
        in the field time (infusion 2)','Bad Input','modal')
        set(infusetime3edit,'string',tin(3))
        return
    end
    tin(3) = user_entry;
end

function infusetime4_Callback(hObject,eventdata)
    user_entry = str2double(get(hObject,'string'));
    if isnan(user_entry)
        errordlg('You must enter a numeric value in
        field time (infusion 4)','Bad Input','modal')
        set(infusetime4edit,'string',tin(4))
        return
    end
    if tspan(2) <= user_entry
        errordlg('You must enter a time inside the
        timespan in field time (infusion 4)','Bad Input','modal')
        set(infusetime4edit,'string',tin(4))
        return
    end
    if tin(3) >= user_entry
        errordlg('You must enter a time larger than
        the one in the field time (infusion 3)','Bad Input','modal')
        set(infusetime4edit,'string',tin(4))
        return
    end
    tin(4) = user_entry;
end

function infuse1_Callback(hObject,eventdata)
    user_entry = str2double(get(hObject,'string'));
    if isnan(user_entry)
        errordlg('You must enter a numeric value in field
        infusion 1','Bad Input','modal')
        set(infuse1edit,'string',inam(1))
        return
    end
    if user_entry < 0
        errordlg('You must enter a positive infusion size
        in field infusion 1','Bad Input','modal')
```

```
        set(infuse1edit,'string',inam(1))
        return
    end
    inam(1) = user_entry;
end

function infuse2_Callback(hObject,eventdata)
    user_entry = str2double(get(hObject,'string'));
    if isnan(user_entry)
        errordlg('You must enter a numeric value in field
        infusion 2','Bad Input','modal')
        set(infuse2edit,'string',inam(2))
        return
    end
    if user_entry < 0
        errordlg('You must enter a positive infusion size in field infusion 2','Bad Input','modal')
        set(infuse2edit,'string',inam(2))
        return
    end
    inam(2) = user_entry;
end

function infuse3_Callback(hObject,eventdata)
    user_entry = str2double(get(hObject,'string'));
    if isnan(user_entry)
        errordlg('You must enter a numeric value in field meal 3','Bad Input','modal')
        set(infuse3edit,'string',inam(3))
        return
    end
    if user_entry < 0
        errordlg('You must enter a positive infusion size in field
        infusion 3',
        'Bad Input','modal')
        set(infuse4edit,'string',inam(3))
        return
    end
    inam(3) = user_entry;
end

function infuse4_Callback(hObject,eventdata)
    user_entry = str2double(get(hObject,'string'));
    if isnan(user_entry)
        errordlg('You must enter a numeric value in field infusion 4',
        'Bad Input',
        'modal')
        set(infuse4edit,'string',inam(4))
        return
    end
    if user_entry < 0
        errordlg('You must enter a positive infusion size in field
        infusion 4','Bad Input','modal')
        set(infuseedit,'string',inam(4))
        return
    end
    inam(4) = user_entry;
```

```
    end


%**************************************************************************
% SETPOINT CALLBACKS
%**************************************************************************

function setpoint1_Callback(hObject,eventdata)
        user_entry = str2double(get(hObject,'string'));
        if isnan(user_entry)
            errordlg('You must enter a numeric value in field G setpoint',...
            'Bad Input','modal')
            set(setpoint1edit,'string',a(1))
            return
        end
        if user_entry < 0
            errordlg('You must enter a positive value in field G setpoint',...
            'Bad Input','modal')
            set(setpoint1edit,'string',a(1))
            return
        end
        a(1) = user_entry;
end

function setpoint2_Callback(hObject,eventdata)
        user_entry = str2double(get(hObject,'string'));
        if isnan(user_entry)
            errordlg('You must enter a numeric value in field I setpoint '
            ,'Bad Input','modal')
            set(setpoint2edit,'string',a(2))
            return
        end
        if user_entry < 0
            errordlg('You must enter a positive value in field I setpoint '
            ,'Bad Input','modal')
            set(setpoint2edit,'string',a(2))
            return
        end
        a(2) = user_entry;
end

%**************************************************************************
% CALLBACKS FOR checkboxunits
%**************************************************************************

function checkboxunits_Callback(hObject,eventdata)
if (get(hObject,'Value') == get(hObject,'Max'))
    set(meal,'visible','off')
    set(meal2,'visible','on')
    set(Gbunit,'visible','off')
    set(Gbunit2,'visible','on')
    b(1) = b(1)/18;
    set(Gbedit,'string',b(1))
    decider2 = 1;
```

```
else
    set(meal2,'visible','off')
    set(meal,'visible','on')
    set(Gbunit2,'visible','off')
    set(Gbunit,'visible','on')
    b(1) = b(1)*18;
    set(Gbedit,'string',b(1))
    decider2 = 0;
end
end



%**************************************************************************
% PUSHBUTTON CALLBACKS
%**************************************************************************


    function simulate1_Callback(hObject,eventdata)
        limit2 = 180;
        limit1 = 60;
        if decider2 == 1
            limit2 = limit2/18;
            limit1 = limit1/18;
        end
        [SG,SI,RES,T] = modbersim(0,decider,tspan,initval,p,b,a,tmeals,...
        mealsam,tin,inam);
        Gbvec = b(1)*ones(size(T));
        Ibvec = b(2)*ones(size(T));
        limithigh = limit2*ones(size(T));
        limitlow = limit1*ones(size(T));
        plot(ax1,T,RES(:,1),'-b',T,RES(:,4),'-r',T,Gbvec,'--black',T,...
        limithigh,'--magenta',T,limitlow,'--magenta','linewidth',3)
        plot(ax2,T,RES(:,2),'-b','linewidth',3)
        plot(ax3,T,RES(:,3),'-b',T,Ibvec,'--black','linewidth',3)
        set(ax1,'xcolor','white')
        set(ax1,'ycolor','white')
        set(ax2,'xcolor','white')
        set(ax2,'ycolor','white')
        set(ax3,'xcolor','white')
        set(ax3,'ycolor','white')
        legend(ax1,'Level','Subcutaneous level','Baseline')
        legend(ax2,'Level')
        legend(ax3,'Level','Baseline')
        xlabel(ax1,'[min]','color','white')
        ylabel(ax1,'[mg/dL]','color','white')
        xlabel(ax2,'[min]','color','white')
        ylabel(ax2,'[1/min]','color','white')
        xlabel(ax3,'[min]','color','white')
        ylabel(ax3,'[mU/L]','color','white')
        axis(ax1,'tight')
        axis(ax2,'tight')
        axis(ax3,'tight')
    end
```

```
end
```

Appendix B

ESIM - A SIMULATOR

B.1 Introduction

Esim is the user interface making the modified model easier to use. Esim is a Matlab program. However you must have some knowledge about this model and the parameters, to use the program.

B.2 Using Esim

In Esim you first define the timespan. Then you define the subject to simulate. There are two different kinds of simulation implemented in Esim, namely simulation with and without a controller

B.2.1 Simulation without a controller

By standard no controller is used, which means no insulin response is given automatically. In this simulation part you are able to see the subjects reaction to injections and meals. The results are shown in the right of the screen, by the three graphs

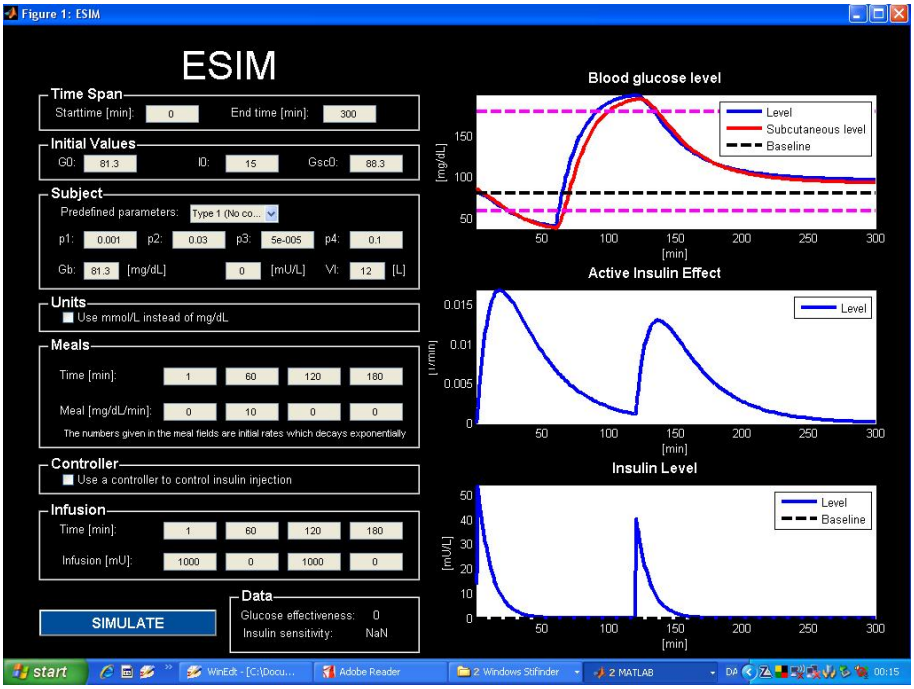

Figure B.1: ESIM, WHEN CONTROLLER NOT IS CHOSEN

blood glucose concentration, Effect of active insulin and blood insulin concentration. A screenshot taken from this simulation part is given in figure B.1

B.2.2 Simulation with a controller

By checking the box 'Use a controller to control insulin' is checked you are in the other simulation part. Here you can see how the controller react to a mealdisturbance. In figure B.2 a screenshot from this simulation part is given.

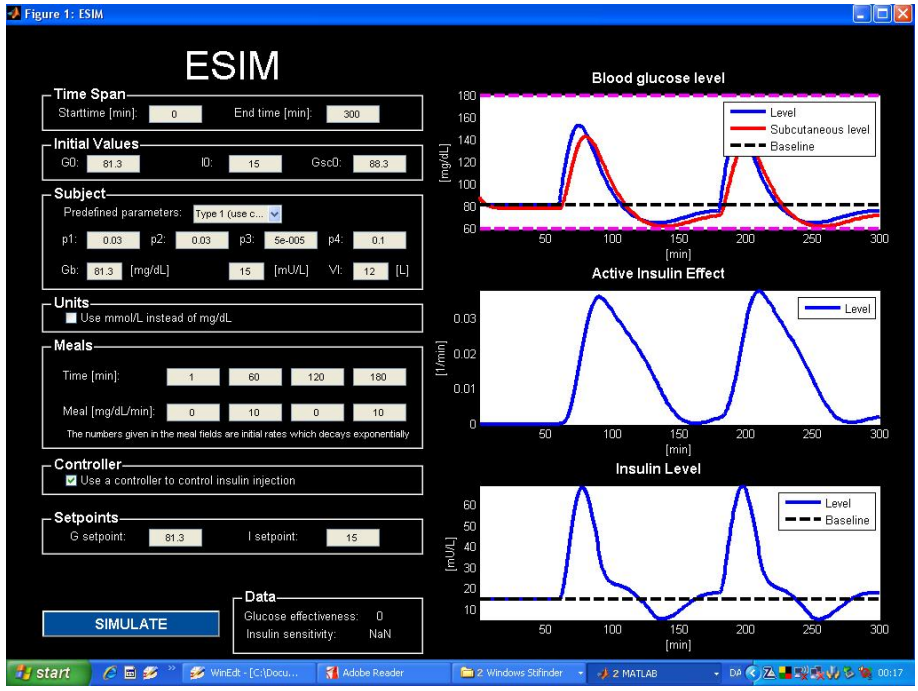

Figure B.2: ESIM, WHEN CONTROLLER IS CHOSEN

Bibliography

[1] Diabetes type 1. *www.umm.edu*.

[2] Yang Kuang Athena Markroglou, Jiaxu Li. Mathematical models and software tools for the glucose-insulin regulatory system and diabetes: an overview. *Applied numerical mathematics*, 56:559–573, may 2006.

[3] Andrea Caumo Claudio Cobelli and Matteo Omenetto. Overestimation of minimal model glucose effectiveness in presence of insulin response is due to undermodeling. *American Journal of Physiology*, 278:E481–E488, 1999.

[4] Andrea Caumo Claudio Cobelli, Francesca Bettini and Michael J. Quon. Overestimation of minimal model glucose effectiveness in presence of insulin response is due to undermodeling. *American Journal of Physiology*, 275:E1031–E1036, 1998.

[5] Steno Diabetes Center Dorte Hammelev, Novo Nordisk. diabetes og insulin..... *Novo Nordisk*, 1:1–42, april 2006.

[6] Michael E. Fisher. A semiclosed-loop algorithm for the control of blood glucose levels in diabetics. *IEEE Transactions on biomedical engineering*, 38(1):57–61, January 1991.

[7] Andrea De Gaetoano and Ovide Arino. Mathematical modelling of the intravenous glucose tolerance test. *Journal of Mathematical Biology*, 40:136–168, Revised version February 1999 2000.

[8] Bud Clark M.A Sami Kanderian M.S.E G.M Steil, Ph.D. and Ph.D K. Rebrin, M.D. Modeling insulin action for development of a closed-loop artificial pancreas. *Diabetes technology and therapeutics*, 7(1):94–108, 2005.

[9] S.B Jørgensen K.J Åstrøm, B.Wittenmark. Chapter 5 pid controller. *Chemical process control, notes published at department of chemical engineering, technical university of Denmark.*, 1994.

[10] Sandra M. Lynch and B. Wayne Bequette. Model predictive control of blood glucose in type 1 diabetics using subcutaneous glucose measurements. *Proceeding of the American Control Conference, Anchorage.*, pages 4039–4043, May 2002.

[11] Giovanni Pacini and Richard N. Bergman. Minmod: a computer program to calculate insulin sensitivity and pancreatic responsivity from the frequently sampled intravenous glucose tolerance test. *Computer methods and programs in biomedicine*, (23):113–122, 1986.

[12] Andrea Caumo Paolo Vicini and Claudio Cobelli. The hot ivgtt two compartment minimal model: indexes of glucose effectiveness and insulin sensitivity. *American Journal of Physiolology*, 273:1024–1032, 1997.

[13] Charles R. Bowden Richard N. Bergman, Gianna Toffolo and Claudio Cobelli. Minimal modeling, partition analysis, and identification of glucose disposal in animals and man. *IEEE Transactions on biomedical engineering*, pages 129–135, 1980.

[14] Lawrence S. Phillips Richard N. Bergman and Claudio Cobelli. Physiologic evaluation of factors controlling glucose tolerance in man. *The American society for Clinical investigation*, 68:1456–1467, December 1981.

[15] unknown. Glucose. *Lab Tests online*.