September 27, 2017

MASTER THESIS - APPLIED MATHEMATICS

TOWARDS THE AUTOMATIC CONTROL OF BLOOD GLUCOSE IN PATIENTS WITH T1DM

Niels Middelhuis (s1008560)

Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS)
Chair of Hybrid Systems

Assessment committee:
Prof. dr. H.J. Zwart
Dr. J.W. Polderman
H. Blauw Msc.
Dr. Ir. R. Langerak
Dr. Ir. J.R. Buitenweg

UNIVERSITY OF TWENTE.

Abstract

The regulation of the blood glucose concentration is of vital importance for patients with type 1 diabetes mellitus (T1DM). To improve the quality of life of patients with T1DM, a control algorithm is needed that is able to automatically regulate the blood glucose concentration. This thesis presents an example of such a control algorithm. The control algorithm of interest can be characterized as an individualized fuzzy logic controller. Several different type of injections are used to imitate the reasoning of patients with T1DM. The goal of this thesis was to develop a mathematical model that can be used for the analysis of the control algorithm. The mathematical model is based on the theory of timed automata extended with the additional features provided by the tool Uppaal. In order to analyze the control algorithm as (extended) timed automaton in Uppaal, a linear ARX model was estimated on available clinical data. This linear ARX model was used to simulate the blood glucose concentration based on the ingested amount of carbohydrates, the administered amount of insulin and the administered amount of glucagon. Both the model of the control algorithm and the blood glucose concentration were implemented in Uppaal and were used for the analysis of the control algorithm. This analysis showed that the control algorithm is initialized correctly, that it satisfies certain requirements for this initial setting and what the influence of a change in this initial setting is on different characteristics.

Keywords: blood glucose concentration, control algorithm, type 1 diabetes mellitus, system identification, ARX, timed automata, Uppaal

Contents

List of Abbreviations, Nomenclature and Medical Terms

Abbreviations

ARMAX	Auto Regressive Moving Average with eXogenous input
ARX	Auto Regressive with eXogenous input
BW	Body Weight
CGM	Continuous Glucose Monitoring
CLT	Central Limit Theorem
D	Decrease
EG	Extra injection Glucagon
EI	Extra injection Insulin
FD	Flat Decrease
FI	Flat Increase
FPE	Final Prediction Error
G	Blood glucose concentration
GUI	Graphical User Interface
HR	Heart Rate
I	Increase
ISF	Insulin Sensitivity Factor
MISO	Multiple Input Single Output
MPC	Model Predictive Control
PC	Percentage Curve
PG	Pre-injection Glucagon
PI	Pre-injection Insulin
PID	Proportional-Integral-Derivative
PIN	Patient Identification Number
S	Glucose rate of change, slope
SD	Slight Decrease
SI	Slight Increase
SISO	Single Input Single Output
STD	Strong Decrease
STI	Strong Increase
SMC	Statistical Model Checking
T1DM	Type 1 Diabetes Mellitus
T2DM	Type 2 Diabetes Mellitus
VSD	Very Strong Decrease
VSI	Very Strong Increase

Nomenclature

$A(q)$	A-polynomial of linear model
a_i	Coefficients of the insulin curve
$B(q)$	B-polynomial of linear model
b_i	Coefficients of the glucagon curve
$C(q)$	C-polynomial of linear model
d	Number of parameters to be estimated
$\mathcal{D}_{\mathcal{M}}$	Dimension of the parameter vector θ

$\varepsilon(t)$	Additive noise
$e(t)$	Estimation error
$e_{\text{pred}}(t)$	One-step ahead prediction error
$e_{\text{sim}}(t)$	Simulation error
G	Blood glucose concentration
$G(q,\theta)$	Transfer function from input to output
$H(q,\theta)$	Transfer function from noise to output
k	Discrete time instants
N	Number of measurements
n_a	Order of the polynomial $A(q)$
n_b	Order of the polynomial $B(q)+1$
n_c	Order of the polynomial $C(q)$
n_k	Time-delay between input and output
S	Glucose rate of change, slope
S_g	Glucose rate of change the moment a glucagon injection is administered
T_i	Value of the counter corresponding to patient status i
$u(t)$	Input signal
u_M	The ingested amount of carbohydrates
u_I	The adminsitered amount of insulin
u_H	The administered amount of glucagon
V_N	Loss-function
$x_{i,\text{new}}$	Value of the last entry of the array corresponding to patient status i
$x_{i,j}$	Value of entry j of the array corresponding to patient status i
$y(t)$	Output signal
$\hat{y}(t\|\theta)$	Estimate of the output signal
Z^N	Data set
θ	Set of parameters to be estimated
$\hat{\theta}_N$	Estimate of the parameter vector θ
μ_0	Current mean blood glucose concentration in the night
μ_{-1}	Previous mean blood glucose concentration in the night
σ_0^2	Current variance of the blood glucose concentration in the night
σ_{-1}^2	Previous variance of the blood glucose concentration in the night

Medical Terms

Artificial pancreas	System that helps patients with T1DM to automatically control their blood glucose concentration
Euglycemia	The situation in which the blood glucose concentration is normal, i.e. situation in which the blood glucose concentration is between 3.90 and 10.0 mmol/L
Glucose	A simple sugar that circulates in the blood plasma and which is the most important energy source that is needed by all the cells and organs of our body
Glucagon	Hormone secreted by the α-cells of the pancreas which increases the blood glucose concentration
Glycemic index	Number that indicates the effect of the amount of carbohydrates on the blood glucose concentration
Hyperglycemia	The situation in which the blood glucose concentration is too high, i.e. situation in which the blood glucose concentration is above 10.0 mmol/L
Hypoglycemia	The situation in which the blood glucose concentration is too low, i.e. situation in which the blood glucose concentration below 3.90 mmol/L
Insulin	Hormone secreted by the β-cells of the pancreas which decreases the blood glucose concentration
Insulin sensitivity	Estimate of how much the blood glucose concentration will drop in response of 1U of insulin
Inter-patient variability	Variability of glucose metabolism between different patients. Examples of inter-patient variability factors are weight, age, gender, physical activity, and insulin resistance.
Intra-patient variability	Variability of glucose metabolism for the same patient over time. Examples of intra-patient variability factors are stress, illness, physical activity and meal consumption.

Chapter 1

Introduction

In this thesis we present a control algorithm for the regulation of the blood glucose concentration in patients with type 1 diabetes mellitus (T1DM). This chapter discusses the importance of such a control algorithm for patients with T1DM, the goal and outline of this thesis.

1.1 Context and motivation

Diabetes mellitus is a widespread disease. According to the World Health Organization (WHO), in 2017 approximately 422 million people suffered from diabetes world-wide [38], from which 1.2 million in the Netherlands [11].

In healthy individuals, the blood glucose concentrations are tightly controlled by the pancreas through the secretion of the counter-regulatory hormones: insulin and glucagon. Insulin is secreted by the pancreatic β-cells and reduces the blood glucose concentration, whereas glucagon is secreted by the pancreatic α-cells and increases the blood glucose concentration. By producing and releasing these two hormones the blood glucose concentration can be stabilized within the euglycemic range (3.90 - 10.0 mmol/L). However, in patients with T1DM there is a dysfunction of the pancreas, often in combination with reduced insulin sensitivity, due to the autoimmune destruction of the pancreatic β-cells.

Due to this dysfunction of the pancreas T1DM patients may experience high blood glucose concentrations ($>$ 10.0 mmol/L), better known as hyperglycemia. Symptoms of a high blood glucose concentration include excessive urination, excessive drinking, blurred vision and fatigue. Hyperglycemia may not be immediately life-threatening, but on the long term it can lead to damage to the eyes (retinopathy), kidneys (nephropathy) and nerves (neuropathy).

Nowadays, T1DM patients face the daily challenge of manually controlling their blood glucose concentration. The only way T1DM can be treated is with use of insulin. Multiple times a day the blood glucose concentration needs to be measured to determine the appropriate amount of insulin to be administered. In order to determine the appropriate amount of insulin, external disturbances must be taken into account. These include physical activity and the intake of carbohydrates. These actions are a major challenge for the patients, because it is difficult to take all external disturbances into consideration. A high dose of insulin or any other disturbance may cause low blood glucose concentrations ($<$ 3.90 mmol/L), better known as hypoglycemia. The severity of hypoglycemia increases with lower blood glucose concentrations. Nonsevere hypoglycemia may lead to anxiety, nausea, confusion, blurred vision, and difficulty in speaking, while severe hypoglycemia leads to coma or seizure. Therefore, the main objective of diabetes management is to reduce the variations in blood glucose concentration and to keep the blood glucose concentration within the euglycemic range. This can be achieved with use of an artificial pancreas system.

1.2 Problem statement

Artificial pancreas systems are designed to continuously measure and regulate the blood glucose concentration of T1DM patients. The most important part of the artificial pancreas is the control algorithm, which controls the patients blood glucose concentration based on the measured blood glucose concentration. In other words, the artificial pancreas takes over the blood glucose control from the patient. A

suitable closed-loop algorithm for blood glucose regulation should provide safe and effective glycemic control in T1DM patients in the presence of exercise, meals and other disruptions.

The control algorithm of interest in this thesis can be characterized as a fuzzy logic controller. The main elements of a fuzzy logic controller are fuzzy sets of multiple inputs and single or multiple outputs and fuzzy rules of the form IF (input) THEN (output) [2]. These elements are used to determine the output of the controller based on the input. The basic idea of this approach is that these fuzzy rules can be used to imitate the reasoning of T1DM patients. This reasoning is based on medical knowledge and traditional treatment. In the control algorithm of interest, this reasoning is implemented with use of several type of injections, where each injection corresponds to a different glucose circumstance, i.e. to a different blood glucose concentration in combination with the glucose rate of change. The control algorithm is initialized with the weight of the patient according to a certain formula.

Once the control algorithm is initialized, it needs to satisfy certain requirements with respect to the standard insulin pump therapy. It needs to outperform the standard insulin pump therapy. However, it is not exactly clear whether the control algorithm satisfies these requirements once it is initialized. In addition, it is not exactly clear whether there is a better initial setting and/or what the influence of a change in this initial setting is on the performance of the control algorithm.

The performance of the control algorithm and the influence of a change in the initial setting on the performance of the control algorithm can only be tested with use of clinical trials. However, these clinical trials are very expensive and they take up a lot of time. If the results of these clinical trials indicate that the initial setting needs to be adjusted, then in advance, it is not clear what the influence of this change is on the performance of the control algorithm. A mathematical model that describes the functioning of the control algorithm and that can be used to investigate the performance and the influence of a change in the initial setting on the performance of the control algorithm is needed. Therefore, the goal of this thesis is to model and analyze the control algorithm with the use of a modelling language to be specified.

1.3 Thesis outline

In Chapter 2 some background information relevant to this thesis is presented. Firstly, the glucoregulatory system is described. The three main parts of the glucoregulatory system are glucose, insulin and glucagon. Glucose is a major source of energy for the body and enters the body through the intake of carbohydrates. Insulin and glucagon are the two hormones that are needed to keep the amount of glucose in the blood within the desired range. Secondly, some information is presented regarding diabetes mellitus, current insulin therapies and how the artificial pancreas can replace these therapies.

In Chapter 3 the control algorithm of interest in this master thesis is presented. The control algorithm can be characterized as an individualized fuzzy logic controller, which consists of several type of injections. The combination of blood glucose concentration and glucose rate of change, together with some timing constraints, determine the type of injection that needs to be administered to maintain the blood glucose concentration within the desired range.

In Chapter 4 the system identification problem is discussed. In order to be able to model and analyze the control algorithm, a simple linear black-box model is needed. This black-box model needs to simulate the blood glucose concentration based on the ingested amount of carbohydrates, the administered amount of insulin and the administered amount of glucagon.

In Chapter 5 the theory of timed automata is presented. This theory is used together with the additional features provided by the tool Uppaal to model the control algorithm described in chapter 3 as (extended) timed automaton. Uppaal is used for the implementation and analysis of the control algorithm as (extended) timed automaton.

In Chapter 6 some general points regarding the control algorithm, the system identification and the implementation and analysis in Uppaal are discussed. The chapter ends with some recommendations for future research.

Finally, in Chapter 7 the main conclusions of this master thesis are presented.

Chapter 2

Medical background information

This chapter gives some background information regarding the glucoregulatory system and diabetes mellitus and is based on [21].

2.1 The glucoregulatory system

The three main parts of the glucoregulatory system are glucose, insulin and glucagon.

2.1.1 Glucose

Glucose is a simple sugar that circulates in the blood plasma. It is one of the most important energy sources that is needed by all the cells and organs of our body, such as our muscles, adipose tissues and our brain. The muscles and adipose tissues need insulin to be able to absorb glucose. The brain cells are the only cells that do not need insulin to absorb glucose. To guarantee the required amount of glucose in the brain cells, it is very important that the blood glucose concentration stays above a certain level. Without glucose, the brain is unable to work well.

Glucose enters the glucoregulatory system by the intake of food rich in carbohydrates. After ingestion, food travels down the esophagus to the stomach. In the stomach, acids and enzymes break down the food into tiny pieces. During this process, glucose is released into the intestines where it is absorbed into the blood. This released glucose can then be used by the cells and body organs. Glucose that is not directly required by the body is stored in the liver in the form of glycogen. In case the blood glucose concentration drops below a certain level, this stored glycogen is released back into the blood as glucose to increase the blood glucose concentration.

In fasting conditions, the blood glucose concentration is normally between 3.9 and 5.5 mmol/L. After the ingestion of carbohydrates, the blood glucose concentration increases to approximately 7.8 mmol/L. The blood glucose concentration should be within the normal range within one or two hours after the ingestion of carbohydrates.

Every body needs a different amount of carbohydrates. The amount of carbohydrates that is required for the body depends for example on age, sex, height, weight, level of physical activity, current blood glucose level, and the target blood glucose concentration.

The effect of a meal on the blood glucose concentration does not only depend on the amount of carbohydrates, but also on the glycemic index. The glycemic index gives an estimate of how fast the blood glucose concentration increases after a meal. Two meals with the same amount of carbohydrates, but different glycemic indices may have a different influence on the blood glucose concentration. Carbohydrates with a high glycemic index (more than 70) are quickly digested and released into the blood, whereas carbohydrates with a low glycemic index (less than 55) are slowly digested and released into the blood. This means that the peak of the blood glucose concentration is higher in case of carbohydrates with high glycemic index than in case of carbohydrates with low glycemic index. In the latter case, the response of the blood glucose concentration is slower and flatter. Figure 2.1 shows an example of the effect of the glycemic index on the blood glucose concentration. Factors that tend to affect the glycemic index are the amount of fats, proteins and fibers in the meals.

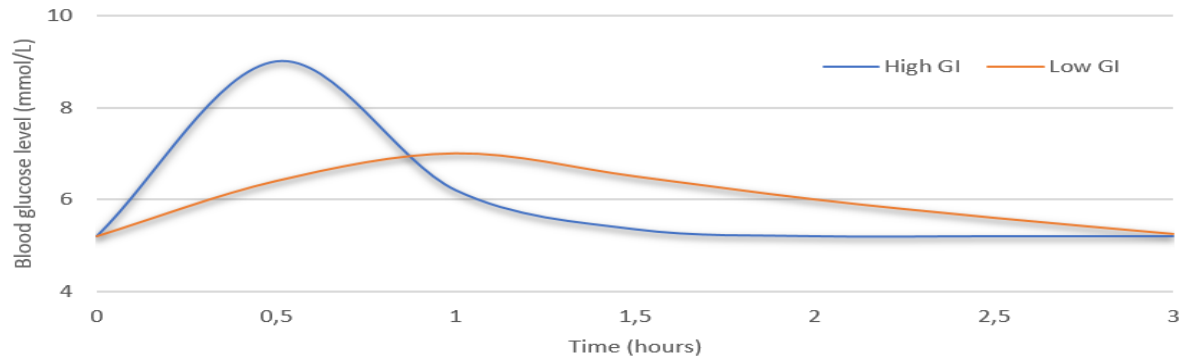


Figure 2.1: *The influence of the glycemic index on the blood glucose concentration. The blue line represents a high glycemic index and the red line represents a low glycemic index [4].*

2.1.2 Insulin

The cells inside the human body need glucose for energy. However, glucose cannot be absorbed by most of the cells directly. After the intake of carbohydrates, the blood glucose concentration increases. Whenever the blood glucose concentration increases, the β-cells in the pancreas are signaled to release the hormone insulin into the blood. The higher the increase of the blood glucose concentration, the more insulin is secreted by the pancreas. This insulin then attaches to the cells, which signals the cells to absorb the glucose from the blood. In other words, insulin can be described as a key which can be used to unlock the cells such that glucose is allowed to enter the cells and can be used for energy.

Glucose that is not directly required by the body is stored in the liver in the form of glycogen. Insulin is needed to convert the glucose into glycogen and to store it inside the liver. This storage causes a decrease of the blood glucose concentration.

2.1.3 Glucagon

When the blood glucose concentration is low or when the body needs glucose, the α-cells in the pancreas are signaled to release the hormone glucagon into the blood. This glucagon helps the liver to convert the glycogen back to glucose and to release it into the blood. This released glucose causes an increase of the blood glucose concentration.

2.2 Diabetes Mellitus

Diabetes mellitus is a group of metabolic diseases that are characterized by high blood glucose concentrations over a prolonged period of time. There are two main types of diabetes, namely type 1 diabetes mellitus (T1DM) and type 2 diabetes mellitus (T2DM). Approximately 90% of the diabetes patients are diagnosed with T2DM and the remaining 10% are diagnosed with T1DM.

T2DM is characterized by insulin resistance, which may be combined with relatively reduced insulin section of the pancreas. This means that the β-cells of the pancreas still secrete some insulin, however, the body does not respond properly anymore. T1DM is an autoimmune disease where the immune system destroys the β-cells of the pancreas that produce insulin. Due to the destruction of the β-cells, the human body does not or hardly produce insulin, which causes increasing blood glucose concentrations.

Hyperglycemia is the situation in which the blood glucose concentration is too high ($>$ 10.0 mmol/L) over a prolonged period of time. Hyperglycemia arises due to the absence of insulin. In this case, the liver, the muscles and the adipose tissues are no longer able to take glucose out of the blood, thereby increasing the blood glucose concentration. Patients with diabetes can also experience hypoglycemia. Hypoglycemia is the situation in which the blood glucose concentration is too low ($<$ 3.90 mmol/L). Hypoglycemia may be caused even in the presence of an intact counter-regulatory response of glucagon [31]. As mentioned earlier, hyperglycemia and hypoglycemia can lead to serious health problems in both the short and long term.

Therefore, it is important that the blood glucose concentration can be maintained within the euglycemic range, which corresponds to a blood glucose concentration between 3.90 and 10.0 mmol/L.

2.2.1 Current therapies

The treatment of diabetes depends on the type of diabetes the patient is diagnosed with. T2DM can be treated by means of a healthy lifestyle and with use of various medications that lower the blood glucose concentration. T1DM can only be treated by the administration of insulin. Multiple times a day the blood glucose concentration needs to be measured to determine how much insulin needs to be administered. Insulin is administered subcutaneously (i.e. under the skin) and can be administered with an insulin pen or with an insulin pump. An insulin pump provides continuous administration of insulin to keep the blood glucose concentration in the desired range between meals and in the night. An extra insulin shot, called bolus, can be administered before the intake of a meal. The use of an insulin pump reduces the occurrence of severe hypoglycemia by a factor of four [20].

The current insulin therapies for patients with T1DM are all patient-controlled, which means that in each case the patient needs to determine the amount of insulin that needs to be administered. To determine the right amount of insulin, the patient must take into account physical activity and the intake of carbohydrates. These actions are a major challenge for the patient and also require considerable efforts. In other words, diabetes has a great impact on a patient's life. Therefore, there is a need for an alternative therapy for T1DM patients which reduces the disease burden and which improves the quality of life.

2.2.2 Artificial pancreas

The artificial pancreas is a system that is designed to continuously measure and regulate the blood glucose concentration. The blood glucose concentration can be regulated with insulin or with insulin and glucagon. The main components of a single-hormone artificial pancreas are a continuous glucose monitoring (CGM) system, an insulin infusion pump and a control algorithm that determines the appropriate amount of insulin that needs to be administered to reduce the blood glucose concentration. The blood glucose concentration can also be regulated with use of a bi-hormonal artificial pancreas, which also consists of a glucagon infusion pump to increase the blood glucose concentration when needed.

An example of such a bi-hormonal artificial pancreas can be found in [5]. This artificial pancreas is able to regulate the blood glucose concentration of T1DM patients by the administration of insulin and glucagon. The system consists of a portable device, which contains the CGM system, control algorithm, insulin and glucagon infusion pump, and two wireless transmitters to obtain the continuous glucose measurements. In combination with the patient, this bi-hormonal artificial pancreas is a fully closed-loop system, i.e. it provides automated glucose control without meal and exercise announcements.

The control algorithm is the most important part of the artificial pancreas and regulates the blood glucose concentration. A suitable closed-loop algorithm should provide safe and effective glycemic control in T1DM patients in the presence of physical activity, meals and other disruptions. One of the major challenges in achieving this goal is the wide variability of glucose metabolism between patients (inter-patient variability) and for the same patient over time (intra-patient variability). Examples of inter-patient variability factors are weight, age, gender, physical fitness, and insulin resistance. Intra-patient variability can be caused by stress, illness, physical activity and meal consumption. Some factors can be both inter- and intra-patient. Other challenges are delays related to the subcutaneous insulin/glucagon infusion and its action in lowering/increasing the blood glucose concentration and the time-delay between the measured subcutaneous glucose concentration and the plasma blood glucose concentration.

Several control algorithms can be used to regulate the blood glucose concentration of T1DM patients. Examples of control algorithms are Proportional-Integral-Derivative (PID) [34], [37], [39], Model Predictive Control (MPC) [13], [18], and fuzzy logic control [2], [30]. The next chapter discusses a possible control algorithm that can be used to regulate the blood glucose concentration of T1DM patients.

Chapter 3

Control algorithm

This chapter discusses the control algorithm of interest in this master thesis. The control algorithm can be characterized as an individualized fuzzy logic controller and uses both insulin and glucagon to regulate the blood glucose concentration of patients with type 1 diabetes to a certain target blood glucose concentration.

The control algorithm consists of three types of injections, which are called curve injections, pre-injections and extra injections. Figure 3.1 shows the relationship between the three types of injections, where blue represents insulin and orange represents glucagon. This figure shows at which combination of blood glucose concentration and glucose rate of change which type of injection will be administered. Each category corresponds to a certain range of this glucose rate of change.

Section 3.1 discusses the three types of injections. First, the insulin and glucagon curve injections are discussed. Second, the insulin and glucagon pre-injections are discussed. Pre-injections are administered in case the increase or decrease of the blood glucose concentration is too strong. Third, the insulin and glucagon extra injections are discussed. Extra injections are administered when the effect of the curve injections and the pre-injections on the blood glucose concentration is not satisfactory enough. Extra injections have priority over the pre-injections, whereas pre-injections have priority over the curve injections.

Section 3.2 discusses the method used to determine whether a curve, pre- or extra injection needs to be administered.

Section 3.3 discusses the initialization of the control algorithm. The size of the three injections depend on the initial setting of the control algorithm.

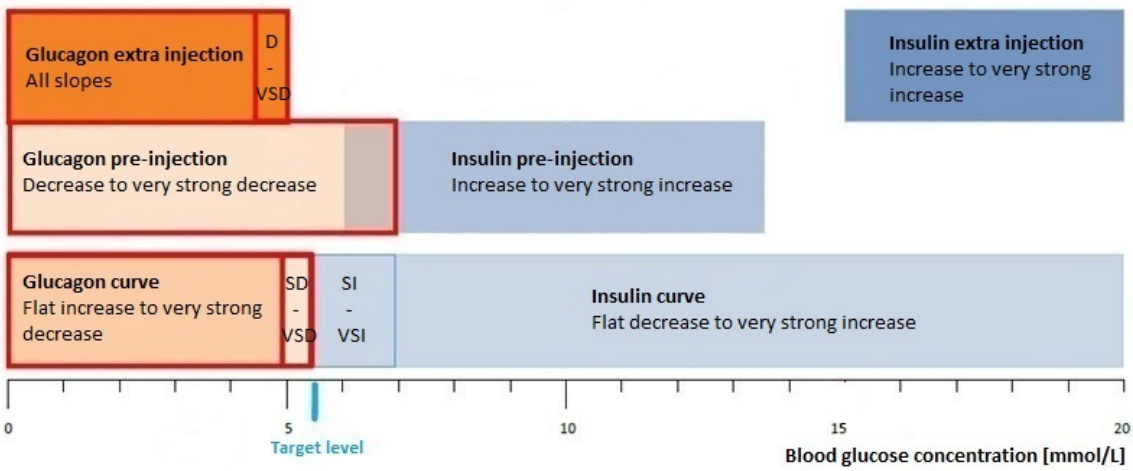


Figure 3.1: *The different types of injections based on the blood glucose concentration and glucose rate of change. Blue represents insulin and orange represents glucagon (VSD = Very Strong Decrease, D = Decrease, SD = Slight Decrease, SI = Slight Increase, VSI = Very Strong Increase)*

3.1 Injection types

3.1.1 Curve injections

Curve injections are administered to reach the target blood glucose concentration. The curve injections are based on two three-degree polynomials: one for insulin and one for glucagon. Both polynomials are used to determine the amount of insulin or glucagon that needs to be administered at a certain blood glucose concentration to reach the target blood glucose concentration. The glucose rate of change determines the moment these curve injections are administered. The glucose rate of change is defined as the difference between the most recent measured blood glucose concentration and the blood glucose concentration measured ten minutes before.

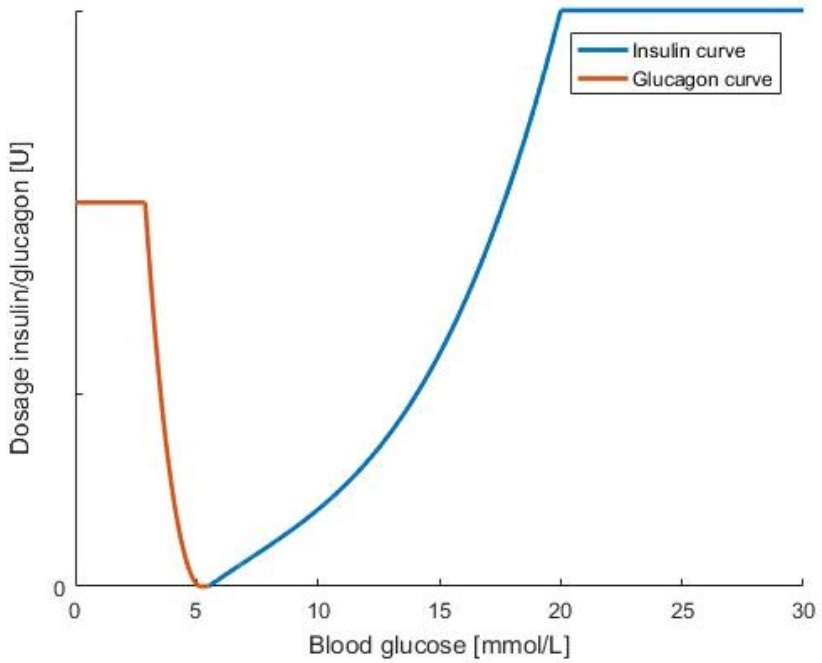


Figure 3.2: *Plot of the insulin curve (blue) and the glucagon curve (orange). Both curves are cut off at a certain level.*

The insulin and glucagon curves are three-degree polynomials as shown in figure 3.2. Both the insulin and glucagon curve are cut off at a certain value. The polynomials are given by:

$$\text{Curve Insulin} = a_0 + a_1 G + a_2 G^2 + a_3 G^3 \tag{3.1}$$

$$\text{Curve Glucagon} = b_0 + b_1 G + b_2 G^2 + b_3 G^3 \tag{3.2}$$

where the coefficients a_i and b_i for $i = 1, 2, 3$ are given constants. These constants are determined by fitting a three-degree polynomial through certain specified data points. The blood glucose concentration and the glucose rate of change determine whether an insulin or glucagon injection needs to be administered. Table 3.1 shows when an insulin injection will be administered and when a glucagon injection will be administered.

Table 3.1: *The moments an insulin or glucagon curve injections will be administered.*

Curve	Range blood glucose	Range glucose rate of change
Insulin	$5.5 < G < 7$	Slight increase to very strong increase
	$G \geq 7$	Flat decrease to very strong increase
Glucagon	$5 \leq G < 5.5$	Flat increase to very strong decrease
	$G < 5$	Slight decrease to very strong decrease

The insulin and glucagon curves as described by the equations (3.1) and (3.2) are the so-called 100% curves. The actual amount of insulin to be administered depends on the value of a parameter called ”Percentage curve” (PC). This parameter depends on the weight of the patient and is set during initialization of the control algorithm. Since the value of PC depends on the weight of the patient, the value of

PC is different for each patient and therefore the individual curve injections are different for each patient. The individual insulin curve injection can be determined with use of the following formula:

$$\text{Individual curve injection} = \text{PC} \cdot \text{Curve injections}$$

Due to the limited knowledge about the influence of glucagon on the blood glucose concentration of a patient and different patients, the individual glucagon curve injections are the same for each patient. This means that the individual glucagon curve injections do not depend on the value of PC.

The actual amount of insulin to be administered does not only depend on the value of percentage curve, but also on the physical activity of the patient. The physical activity of the patient is monitored with the use of a heart rate sensor and an accelerometer. The activity correction of the insulin curve can be found in table 3.2. In this table it can be seen that the more active the patient the greater the correction. The reason for this is that the insulin sensitivity is increased during exercise. A higher insulin sensitivity means that the cells, inside the human body, are better able to use any available insulin to take up glucose during and after activity and therefore less insulin is needed.

Table 3.2: *Activity correction of the insulin curve. The percentages are the percentage of the initial determined insulin curve injection.*

	Description	**Insulin curve injection**
1	Rest and light exercise	100%
2	Moderate exercise	85%
3	Moderate/heavy exercise	75%
4	Heavy exercise	55%

The curve injections are only administered when a certain amount of time has been passed since the last curve, pre- or extra injection. The time from the last injection is monitored with use of a timer. The moment the timer reaches a certain threshold it will be determined whether, according to table 3.1, a curve injection needs to be administered or not. This threshold depends on the size of the glucose rate of change and is set the moment a curve, pre- or extra injection is administered. The higher the absolute value of the glucose rate of change, the shorter the length of the timer will be. The minimum value of this threshold is equal to 6 minutes, whereas the maximum value of this threshold is equal to 15 minutes. Thus, the administration of a curve injection insulin or glucagon ranges from once every 6 minutes to once every 15 minutes.

3.1.2 Pre-injections

A pre-injection insulin or glucagon is administered when there is a strong increase or decrease of the blood glucose concentration. A strong increase of blood glucose concentration could be due to the intake of a certain amount of carbohydrates, where a strong decrease in blood glucose concentration could be due to physical activity. A pre-injection can only be administered when the corresponding patient status is inactive (see section 3.2). The moment a pre-injection is administered, the corresponding patient status becomes active. If the patient status is active, then it needs to be reset before the corresponding pre-injection can be administered again. The reset depends on time, the blood glucose concentration and the glucose rate of change. The size of the corresponding pre-injection depends on the size of the glucose rate of change. Table 3.3 shows the size of the pre-injections, where each of the categories correspond to a different range of the glucose rate of change. The parameters PI (Pre-injection Insulin) and PG (Pre-injection Glucagon) are set during initialization of the controller.

Table 3.3: *The size of the insulin and glucagon pre-injections according to the size of the glucose rate of change (PI = Pre-injection Insulin, PG = Pre-injection Glucagon).*

Type of pre-injection	**Category**	**Injection size**
Insulin pre-injection	Increase	$0.75PI$
	Strong increase	$0.85PI$
	Very strong increase	$1.00PI$
Glucagon pre-injection	Decrease	$0.75PG$
	Strong decrease	$0.85PG$
	Very strong decrease	$1.00PG$

The size of an insulin pre-injection does not only depend on the size of the glucose rate of change, but also whether in the last 45 minutes a successful glucagon pre- or extra injection has been administered. If this is the case, the size of the insulin pre-injection will be reduced in accordance to the formula:

$$\text{Corrected VI} = \text{VI} \cdot ((S_g/10) + 1)$$

where S_g is the value of the glucose rate of change the moment of the glucagon pre- or extra injection. The value of S_g is always negative. This means that, for example, if the glucose rate of change has a value of -4 mmol/L/h, the insulin pre-injection size will be reduced with 40%. If the glucose rate of change is less than -10 mmol/L/h, the size of the pre-injection becomes negative. In that case, no insulin pre-injection will be administered.

It could be possible that the pre-injection insulin needs to be supplemented with an additional amount of insulin. This could be the case when an insulin pre-injection of size $0.75PI$ or $0.85PI$ is administered, the glucose rate of change still increases and then falls into a higher category. Therefore, five and ten minutes after a successful pre-injection insulin it is checked whether the glucose rate of change falls into a higher category than the moment the pre-injection was administered. If this is the case, the pre-injection will be supplemented with the difference between the two pre-injections. In case of a full pre-injection $(1.00PI)$, the pre-injection cannot be supplemented because the glucose rate of change already falls into the highest category.

As mentioned earlier, the threshold of the curve injection timer is set the moment a pre-injection is administered. Consequently, it is not possible to administer pre-injections and curve injections at the same time. A supplementation of the insulin pre-injection does not cause a reset of the curve injection timer.

3.1.3 Extra injections

An extra injection insulin or glucagon is administered when the curve and pre-injections are not able to cope with the strong increase or decrease of the blood glucose concentration. Just like the pre-injections, an extra injection can only be administered when the corresponding patient status is inactive. The moment an extra injection is administered, the corresponding patient status becomes active. If the patient status is active, it needs to be reset before the corresponding extra injection can be administered again. The reset depend on time, the current blood glucose concentration and the glucose rate of change. Table 3.4 shows for which combination of blood glucose concentration and glucose rate of change which extra injection will be administered. In addition, it shows that the size of the corresponding extra injection is the same for each combination of blood glucose concentration and glucose rate of change. The parameters EI (Extra injection Insulin) and EG (Extra injection Glucagon) are set during initialization of the controller.

As mentioned earlier, the threshold of the curve injection timer is set the moment an extra injection is administered. Consequently, it is not possible to administer extra injections and curve injections at the same time.

Table 3.4: *The type of extra injection to be administered based on the blood glucose concentration and glucose rate of change (EI = Extra injection Insulin, EG = Extra injection Glucagon)*

Type of extra injection	Category		Injection size
Insulin extra injection	Increase to very strong increase		$1.00EI$
Glucagon extra injection	$G \geq 4.5$	Decrease to very strong decrease	$1.00EG$
	$G < 4.5$	All glucose rate of changes	$1.00EG$

3.2 Patient statuses

In the previous section the three type of injections of the control algorithm were discussed. The moment a curve injection needs to be administered depends on the curve injection timer, the corresponding threshold value and the combination of blood glucose concentration and glucose rate of change. The moment a pre- or extra injection needs to be administered depends on the value of the counter of the corresponding patient status and whether the patient status is active or inactive. Table 3.5 shows the four patient statuses with their corresponding blood glucose concentration, glucose rate of change and waiting times.

Table 3.5: *The four patient statuses with their corresponding combination of blood glucose concentration, glucose rate of change and waiting time.*

Nr.	Patient status	Range glucose level	Range glucose rate of change	Waiting time
1	Insulin pre-injection	$6 < G < 13.5$	Increase to very strong increase	n.v.t.
2	Insulin extra injection	$G > 15$	Increase to very strong increase	30 minutes after insulin pre-injection
3	Glucagon pre-injection	$G < 7$	Decrease to very strong decrease	10 minutes after glucagon extra injection
4	Glucagon extra injection	$4.5 \leq G < 5$ $G < 4.5$	Decrease to very strong decrease All slopes	10 minutes after glucagon pre-injection

Each patient status has its own counter of which the value is computed based on the last twenty blood glucose measurements. Each counter has a corresponding array with 20 available entries. The value of the corresponding counter is equal to the sum of these entries. The value of each entry can be 0 or 1 and depends on the combination of blood glucose concentration and glucose rate of change. Every time a new blood glucose measurement comes available, each entry is moved one to the left and the value of the last entry is set to 0 or 1. If the blood glucose concentration and the glucose rate of change meet one of the conditions mentioned in table 3.5, the value of the entry of the corresponding array is set to 1. Otherwise, the value of that entry is set 0. Formally, this could be described as:

$$\begin{bmatrix} x_{i,1}(k) \\ x_{i,2}(k) \\ \vdots \\ x_{i,19}(k) \\ x_{i,20}(k) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & 1 \\ 0 & \cdots & \cdots & \cdots & 0 \end{bmatrix} \begin{bmatrix} x_{i,1}(k-1) \\ x_{i,2}(k-1) \\ \vdots \\ x_{i,19}(k-1) \\ x_{i,20}(k-1) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} x_{i,\text{new}}(k)$$

$$x_{i,\text{new}}(k) = \begin{cases} 1 & \text{if condition of patient status } i \text{ is satisfied} \\ 0 & \text{otherwise} \end{cases}$$

$$T_i(k) = \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} x_{i,1}(k) \\ \vdots \\ x_{i,20}(k) \end{bmatrix}$$

where $x_{i,j} \in [0, 1]$ is the value of entry j of the array corresponding to patient status i, $x_{i,\text{new}}$ is the value of the last entry the moment a new blood glucose measurement comes available and T_i is the value of the counter of patient status i for discrete instants of time $k \in \mathbb{N}$, $i = 1, \ldots, 4$ and $j = 1, \ldots, 20$.

As soon as one of the arrays have 15 out of 20 entries filled with a 1 (i.e. $T_i = 15$ for $i = 1$, 2, 3 or 4), the threshold has been reached, patient status i will become active and the corresponding injection will be administered. The purpose of these counters is to prevent that there is a direct reaction on the combination of blood glucose concentration and glucose rate of change. Only when there is some certainty about the current blood glucose concentration and glucose rate of change, the corresponding patient status will become active.

Once a certain patient status becomes active, it will stay active until the corresponding reset takes place. The reset depend on time, blood glucose concentration and glucose rate of change. This means that if a certain patient status is already active and the corresponding counter becomes equal to 15, the patient status cannot become active again and the corresponding injection will not be administered.

From table 3.5 it follows that it is possible that the patient status corresponding to the glucagon pre- and extra injection become active at the same moment. In that case, the patient status corresponding to the

extra injection becomes active and the patient status corresponding stays inactive. This means that the glucagon extra injection will be administered instead of the glucagon pre-injection.

3.3 Initialization

For each patient the controller is individualized by specifying the size of the curve, pre- and extra injections. In case of insulin, the size of these injections are based on the weight of the patient. In this manner, the controller is able to deal with the interpatient variability.

Insulin

For the insulin injections, individualization takes place by setting the parameters: PC (Percentage Curve), PI (Pre-injection Insulin) and EI (Extra injection Insulin). The values of these parameters are set with use of the following formulas:

- $PC = a \cdot BW + b$
- $PI = c \cdot PC + d$
- $EI = (e \cdot (PC/100))$

where BW is the body weight of the patient in kg and a, b, c, d and e are given constants determined with use of patient data. The equation for PC only holds for patients with a weight higher than 55kg. For patients with a weight lower than 55kg the value of PC is set to 20%. The formulas above show that the value of PC depends on the weight of the patient, whereas the size of the pre- and extra injections insulin depend on the value of PC.

Glucagon

For the glucagon injections, individualization takes places by setting the parameters PC (Percentage Curve), PG (Pre-injection Glucagon) and EG (Extra injection Glucagon). The values of these parameters are set with use of the following formulas:

- $PC = x$ %
- $PG = y$ U
- $EG = z$ U

where x, y and z are known constants determined with use of some patient data. The values of these parameters do not depend on the weight of the patient and therefore these values will be the same for each patient. In the future, these values may depend on patient characteristics.

Chapter 4

System identification from clinical data

In order to be able to model and analyze the control algorithm, a (simple) linear model is needed that can be used to simulate the blood glucose concentration based on the ingested amount of carbohydrates, the administered amount of insulin and the administered amount of glucagon. This chapter describes the identification of such a linear model based on clinical data. The theory and notation discussed in this chapter is based on the book of Ljung [27].

Section 4.1 discusses the system identification problem. The basic description for a linear system with additive noise is given and how the estimated model output can be represented in terms of the measured input and output.

Section 4.2 discusses the two linear models that are the most common in the empirical modelling of blood glucose concentrations. For simplicity, the single-input-single-output (SISO) versions of these models are discussed.

Section 4.3 discusses the parameter estimation methods that can be used to identify the two linear models. The parameter estimation methods minimize the weighted sum of squares of the error between the model output and the measured response. The error to be minimized does not only depends on the model structure, but also on how the error is computed.

Section 4.4 discusses methods that can be used to determine the delay and the model order of the linear models. These need to be specified before the linear models can be identified.

Section 4.5 discusses a method that can be used to validate the residuals on whiteness and independence. The identified model should pass both the whiteness and the independence test.

Section 4.6 discusses the clinical data that were available for the identification of the linear models. The data are obtained from [5].

Section 4.7 discusses the results of the identification of the linear models obtained with the methods and data described in the previous sections. The MATLAB System Identification Toolbox [29] was used for identification. This toolbox provides functions for constructing mathematical models from measured input-output data.

4.1 System identification problem

System identification deals with the problem of building mathematical models of dynamical systems based on measured input-output data. A mathematical model describes the relationship between an input signal and a measured output signal. The output is not only a direct result of the input, but also has a component that consists of uncontrollable inputs and measurement errors. This component is better known as noise and is assumed to be additive at the output. Figure 4.1 shows a schematic overview of the relationship between the input signal, $u(t)$, the additive noise, $\varepsilon(t)$, and the output signal, $y(t)$.

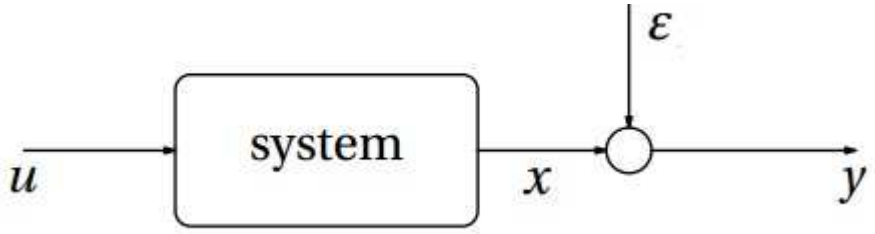


Figure 4.1: *Schematic overview of the relationship between the input signal $u(t)$, the additive noise $\varepsilon(t)$ and the output $y(t)$ [32].*

The basic description for a linear system with additive noise is given by:

$$y(t) = G(q)u(t) + H(q)\varepsilon(t) \tag{4.1}$$

where q is the backward shift operator defined as $(q^{-1}u)(t) = u(t-1)$, $G(q)$ is the transfer function from input to the output and $H(q)$ is the transfer function from the noise to the output. The noise term is assumed to be white noise, i.e. a sequence of independent identically distributed random variables. In general, this random variable is assumed to be normally distributed with mean zero and variance λ.

The system identification problem deals with the problem of how to determine the transfer functions in (4.1). In general, these transfer functions cannot be determined a priori from knowledge of the systems behaviour. Therefore, the coefficients of these transfer functions need to be estimated and enter the model as a set of parameters θ to be estimated:

$$y(t) = G(q,\theta)u(t) + H(q,\theta)\varepsilon(t) \tag{4.2}$$

Equation (4.2) now denotes a set of models $\mathcal{M}$. Parameter estimation methods can be used to select that member in the set that appears to be the most suitable for the particular situation.

The measured output is represented by $y(t)$. However, the model output, denoted as $\hat{y}(t|\theta)$, is an estimate of the measured output and depends on the estimated parameters θ and the model structure $\mathcal{M}$, which will be discussed in the next section. The error between the model output and the measured output, denoted as $e(t)$, is given by:

$$e(t) = y(t) - \hat{y}(t|\theta) \tag{4.3}$$

Assuming that $H(q,\theta)$ is nonzero, equation (4.2) can be rewritten as:

$$e(t) = H^{-1}(q,\theta)\left[y(t) - G(q,\theta)u(t)\right] \tag{4.4}$$

The estimated model output can be found by inserting (4.4) into (4.3) and gives:

$$\hat{y}(t|\theta) = H^{-1}(q,\theta)G(q,\theta)u(t) + \left[1 - H^{-1}(q,\theta)\right]y(t) \tag{4.5}$$

The question now is how to parametrize the transfer functions $G(q,\theta)$ and $H(q,\theta)$. One way is to parametrize these transfer functions as rational functions, i.e. a function from which both the numerator and denominator are polynomials. The set of parameters θ then contains the coefficients of the numerator and denominator. These models structures are also known as black-box models.

4.2 Black-box models

Black-box models are also called empirical models, whereas there are also physiological, first-principles-based models. Physiological models are more attractive than empirical models in the sense that the former contain more physiological meaning than the latter. The structures of empirical models are not derived from the underlying physiology. This means that their parameters usually do not correspond directly to physiological parameters such as insulin sensitivity. However, in some situations empirical models are more attractive than physiological models since they are often much simpler and their parameters can be estimated more easily [16].

Two of the most commonly used linear dynamic models for the prediction or simulation of blood glucose concentrations are ARX and ARMAX models [14], [15], [16]. These two linear dynamic models are discussed below. For simplicity, the SISO versions of these models will be discussed.

4.2.1 ARX

Autoregressive models with exogenous input (ARX) are model structures described as:

$$A(q)y(t) = B(q)u(t - n_k) + \varepsilon(t) \tag{4.6}$$

with

$$A(q) = 1 + a_1 q^{-1} + \cdots + a_{n_a} q^{-n_a} \tag{4.7}$$

$$B(q) = b_0 + b_1 q^{-1} + \cdots + b_{n_b} q^{-n_b+1} \tag{4.8}$$

The parameters n_a and n_b are the orders of the ARX model, where n_a is equal to the number of poles and $n_b - 1$ is equal to the number of zeros. The parameter n_k is the time-delay and is equal to the number of samples that elapse before the effect of an input change is observed in the output.

Comparison with (4.2) shows that the transfer functions $G(q, \theta)$ and $H(q, \theta)$ are given by:

$$G_{ARX}(q, \theta) = \frac{B(q)}{A(q)} \qquad H_{ARX}(q, \theta) = \frac{1}{A(q)} \tag{4.9}$$

The estimate of the model output can be found by inserting (4.9) into (4.5):

$$\hat{y}(t|\theta) = B(q)u(t - n_k) + \left[1 - A(q)\right] y(t) \tag{4.10}$$

In the ARX model structure the parameters to be estimated are the coefficients of the polynomials (4.7) and (4.8):

$$\theta = \begin{bmatrix} a_1 & \cdots & a_{n_a} & b_0 & \cdots & b_{n_b} \end{bmatrix}^T \tag{4.11}$$

If we now introduce the vector:

$$\varphi(t) = \begin{bmatrix} -y(t-1) & \cdots & -y(t-n_a) & u(t-n_k) & \cdots & u(t-n_k-n_b+1) \end{bmatrix}^T \tag{4.12}$$

Then (4.10) can be written as:

$$\hat{y}(t|\theta) = \theta^T \varphi(t) = \varphi^T(t)\theta \tag{4.13}$$

Thus, the estimated model output is a scalar product between a known vector $\varphi(t)$ and the parameter vector θ. Such a model is called a linear regression. Linear regression is an important property since powerful and simple estimation methods can be used to determine the parameter vector θ [27].

4.2.2 ARMAX

Autoregressive moving average with exogenous input (ARMAX) models are model structures described as:

$$A(q)y(t) = B(q)u(t - n_k) + C(q)\varepsilon(t) \tag{4.14}$$

with $A(q)$ and $B(q)$ as defined in (4.7) and (4.8) and:

$$C(q) = 1 + c_1 q^{-1} + \cdots + c_{n_c} q^{-n_c} \tag{4.15}$$

The parameter n_c defines the order of the C polynomial.

Comparison with (4.2) shows that the transfer function $G(q, \theta)$ and $H(q, \theta)$ are given by:

$$G_{ARMAX}(q, \theta) = \frac{B(q)}{A(q)} \qquad H_{ARMAX}(q, \theta) = \frac{C(q)}{A(q)} \tag{4.16}$$

The estimate of the model output can be found by inserting (4.16) into (4.5):

$$\hat{y}(t|\theta) = \frac{B(q)}{C(q)} u(t - n_k) + \left[1 - \frac{A(q)}{C(q)}\right] y(t) \tag{4.17}$$

In this case, the parameter vector to be estimated is:

$$\theta = \begin{bmatrix} a_1 & \cdots & a_{n_a} & b_0 & \cdots & b_{n_b} & c_1 & \cdots & c_{n_c} \end{bmatrix}^T \tag{4.18}$$

If we now introduce the vector:

$$\varphi(t,\theta) = \begin{bmatrix} -y(t-1) & \cdots & -y(t-n_a) & u(t-n_k) & \cdots & u(t-n_k-n_b+1) \\ & & & & & e(t-1,\theta) & \cdots & e(t-n_c,\theta) \end{bmatrix}^T \tag{4.19}$$

where $e(t)$ is the estimation error as defined in (4.3). Then (4.17) can be written as:

$$\hat{y}(t|\theta) = \varphi^T(t,\theta)\theta \tag{4.20}$$

Notice the similarity with the linear regression (4.13). However, due to the nonlinear effect of θ in the vector $\varphi(t,\theta)$ the equation (4.20) itself is no linear regression. To stress the relationship with (4.13), (4.20) is called a *pseudolinear regression* [27].

Comparison of (4.9) with (4.16) shows the main difference between the ARX model and the ARMAX model. In both cases, the input term and disturbance term are modelled with the same set of poles. However, the ARMAX model allows the disturbance term to have numerator dynamics C which results in a moving average of recent values of $e(t)$ [16] and a better flexibility to model the noise dynamics.

4.3 Parameter estimation methods

Parameter estimation methods are methods that can be used to estimate parameters in a certain model structure $\mathcal{M}$. The main goal is to find those numerical values of the unknown parameters for which the estimation error, as defined in (4.3), is minimal in some sense.

The estimation error is a sequence which can be seen as a vector in $\mathbb{R}^N$, where N is the number of measurements. The size of this vector could be measured using any norm in $\mathbb{R}^N$. The most common choice is the norm defined by:

$$V_N(\theta, Z^N) = \frac{1}{N}\sum_{t=1}^{N}\frac{1}{2}e^2(t,\theta) \tag{4.21}$$

The function $V_N(\theta, Z^N)$ is, for a given dataset Z^N, a well-defined scalar-valued function of the parameter vector θ and is a natural measure of the validity of the model. The estimate of the parameter vector, denoted as $\hat{\theta}_N$, can be found by minimization of (4.21):

$$\hat{\theta}_N = \hat{\theta}_N(Z^N) = \underset{\theta\in\mathcal{D}_\mathcal{M}}{\operatorname{argmin}}\, V_N(\theta, Z^N) \tag{4.22}$$

where $\mathcal{D}_\mathcal{M}$ is the dimension of the parameter vector θ and argmin denotes the argument of the minimum. If the minimum is not unique, argmin denotes the set of minimizing arguments. The general term for the family of approaches that is used to find (4.22) is *prediction-error identification methods*.

4.3.1 ARX

It was shown that the estimated model output of the ARX model could be written as a linear regression. The error that corresponds to this linear regression can be found by inserting (4.13) into (4.3):

$$e(t,\theta) = y(t) - \varphi^T(t)\theta \tag{4.23}$$

Inserting (4.23) into (4.21) gives:

$$V_N(\theta, Z^N) = \frac{1}{N}\sum_{t=1}^{N}\frac{1}{2}\left[y(t) - \varphi^T(t)\theta\right]^2 \tag{4.24}$$

This is the least-squares criterion for the linear regression (4.13). The main advantage of this least-squares criterion is that it is a quadratic function in θ. Therefore, the global minimum can be found analytically, which gives the least-squares estimate $\hat{\theta}_N$ given by:

$$\hat{\theta}_N = \left[\varphi^T\varphi\right]^{-1}\varphi^T y \tag{4.25}$$

The MATLAB System Identification Toolbox contains the `arx` function to find the least-squares estimate defined by (4.25).

4.3.2 ARMAX

It was shown that the estimated model output of the ARMAX model is called a pseudolinear regression. The error that corresponds to this pseudolinear regression can be found by inserting (4.20) into (4.3):

$$e(t,\theta) = y(t) - \varphi^T(t,\theta)\theta \tag{4.26}$$

Inserting (4.26) into (4.21) gives:

$$V_N(\theta, Z^N) = \frac{1}{N}\sum_{t=1}^{N}\frac{1}{2}\left[y(t) - \varphi^T(t,\theta)\theta\right]^2 \tag{4.27}$$

The pseudolinear estimate $\hat{\theta}_N$ can be found by minimizing this criterion function. However, because this minimization problem is a non-linear minimization problem the minimum cannot be found analytically. Therefore, the estimate needs to be found with use of an iterative search algorithm to find the local minima. The MATLAB System Identification Toolbox contains the `armax` function to find the pseudolinear estimate.

4.3.3 Choice of estimation error

The form of $V(\theta, Z^N)$ does not only depends on the model structure (e.g. ARX or ARMAX), but also on how the estimation error is computed. In MATLAB, the estimation error can be computed in two ways, namely as the one-step ahead prediction error or as the simulation error. The way in which the estimation error is computed can be specified with the `Focus` option available in both the `arx` and `armax` function. The default uses the one-step ahead prediction error.

One-step ahead prediction error

When `Focus` is set to `prediction`, the estimation error represents the one-step ahead prediction error. The one-step ahead prediction error is given by:

$$e_{\text{pred}}(t) = y(t) - \hat{y}_{\text{pred}}(t|\theta) \tag{4.28}$$

where

$$\hat{y}_{\text{pred}}(t|\theta) = H^{-1}(q,\theta)G(q,\theta)u(t-n_k) + \left[1 - H^{-1}(q,\theta)\right]y(t) \tag{4.29}$$

In case of ARX and ARMAX it can be shown that the product transfer function $H^{-1}G$ is always proper, i.e. the degree of the numerator does not exceed the degree of the denominator. This means that the estimated model output is computed from past input and output data:

$$u(t-n_k), u(t-n_k-1), \cdots, u(1), y(t-1), \cdots, y(1)$$

Thus, prediction means that the estimated model output is computed using the current and past values of measured input and output values one-step ahead into the future. In other words, in case the `Focus` is set to `prediction`, the estimation focuses on producing a good predictor model.

Simulation error

When `Focus` is set to `simulation`, the estimation error represents the simulation error. The simulation error is given by:

$$e_{\text{sim}}(t) = y(t) - \hat{y}_{\text{sim}}(t|\theta) \tag{4.30}$$

where

$$\hat{y}_{\text{sim}}(t|\theta) = G(q,\theta)u(t) \tag{4.31}$$

The simulated model output is also called an infinite-step ahead predictor. Since the model is given no information about the measured output data, infinite-step ahead predictions are noise-free [16]. This means that $\varepsilon(t)$ in (4.2) is ignored.

In case of ARX and ARMAX the transfer function $G(q,\theta)$ is defined as $B(q)/A(q)$. Multiplying both sides of (4.31) with $A(q)$ shows that the simulated output $\hat{y}_{\text{sim}}(t|\theta)$ only uses the available input data and previous simulated outputs to generate the current model output. In other words, the simulated model output is defined recursively as:

$$\hat{y}_{\text{sim}}(t|\theta) = g(t, Z^{t-1}, \hat{\theta}_N) \tag{4.32}$$

where

$$Z^{t-1} = \{\hat{y}_{\text{sim}}(t-1), u(t-1), \hat{y}_{\text{sim}}(t-2), u(t-2), \cdots, \hat{y}_{\text{sim}}(1), u(1)\} \tag{4.33}$$

Since the simulated model output is defined recursively, setting `Focus` to `simulation` leads to a non-linear optimization problem.

Thus, simulation means that the estimated model output is computed using the current and past values of measured input data and initial conditions. In other words, in case the `Focus` is set to `simulation`, the estimation focuses on producing a good simulation model for simulation with the current inputs.

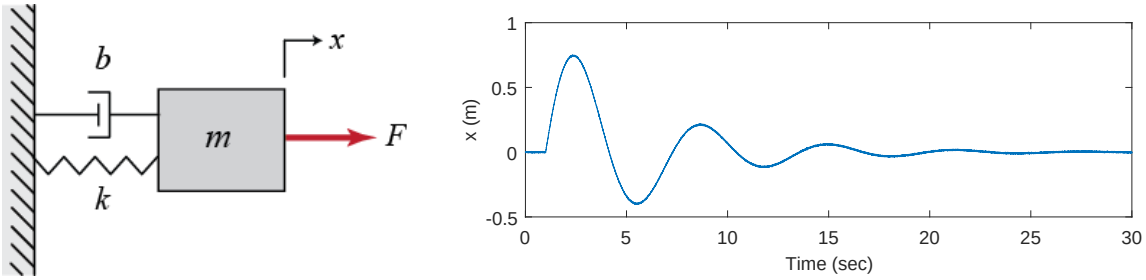
Comparison between prediction and simulation
Comparison of (4.29) and (4.31) shows that the two methods are equivalent in case $H(q, \theta) = 1$. An example of such a model is the output error model. In that case, it does not matter if `Focus` is set to `prediction` or `simulation`.

Comparison of (4.29) and (4.31) also shows the conceptual difference between prediction and simulation. In case of prediction, $y(t-1)$ and earlier y-values are measured values, and can therefore give a model output that "looks good" even though the model may be bad [27]. In case of simulation, $\hat{y}(t-1)$ and earlier $\hat{y}$-values are a result of computation using inputs and initial conditions. The next example illustrates the difference between prediction and simulation.

Example 1. ***Mass-Spring-Damper System***
Consider the damped spring-mass system, shown in figure 4.2a, with mass $m = 1$ kg, damping constant $b = 0.4$ Ns/m and spring constant $k = 1.02$ N/m. This system can be described with the following second order differential equation:

$$\ddot{x}(t) + 0.4\dot{x}(t) + 1.02x(t) = F(t) \tag{4.34}$$

where $F(t)$ is a certain force exerted on the mass at time t. Suppose that the mass is at rest at equilibrium until time $t = 1$ when it is hit by a very intense force, i.e. $F(t) = \delta(t)$. Figure 4.2b then shows the impulse response of the system. In this figure it can be seen that the system is underdamped and that it oscillates around its equilibrium. At $t = 30$ the system comes back at rest at its equilibrium.



(a) *Mass-Spring-Damper System* **(b)** *Impulse response*

Figure 4.2: *Mass-Spring-Damper System with the impulse response of (4.34)*

Suppose we don't know the dynamics of the system, as described by (4.34), and we want to estimate a second order linear ARX model on measured data with a certain sampling interval. The sampling interval determines the time between consecutive measurements. Suppose we measure the position of the mass every 1 second, every 0.1 second and every 0.01 second and use these measurements to estimate, for each sampling interval, two second order ARX models: one with focus set to prediction and one with focus set to simulation. If we then simulate the output of both models for each sampling interval, figure 4.3 is obtained.

In this figure it can be seen that in case we sample every 1 second, the output of both models is identical and that both models are able to describe the measured data very well. In case we sample every 0.1 seconds, the simulation model is still able to describe the measured data very well, whereas the output of the prediction model is less accurate. In case we sample every 0.01 seconds, the output of the prediction model is almost equivalent to zero, whereas the simulation model is still able to describe the measured data.

The reason for this is that in case the focus is set to prediction the previous measured values are used to estimate the model output. In case the data is sampled too fast, the variation in the data between two consecutive points is very small. This means that the model predicts the current output to be equal to the previous two outputs, i.e. this model is perfect for prediction. However, in case this model is used for simulation, the measured output values are not available and the model output is computed using measured inputs and the initial conditions only. In case the initial conditions are equal to zero, the simulated output will be almost equivalent to zero. In other words, the model is useless for simulation. In case the focus is set to simulation, the model output will be estimated based on the measured inputs and initial conditions only. If this model is then used for simulation, the same measured inputs and initial conditions are used to simulate the model output. Thus, the model is estimated for its desired purpose.

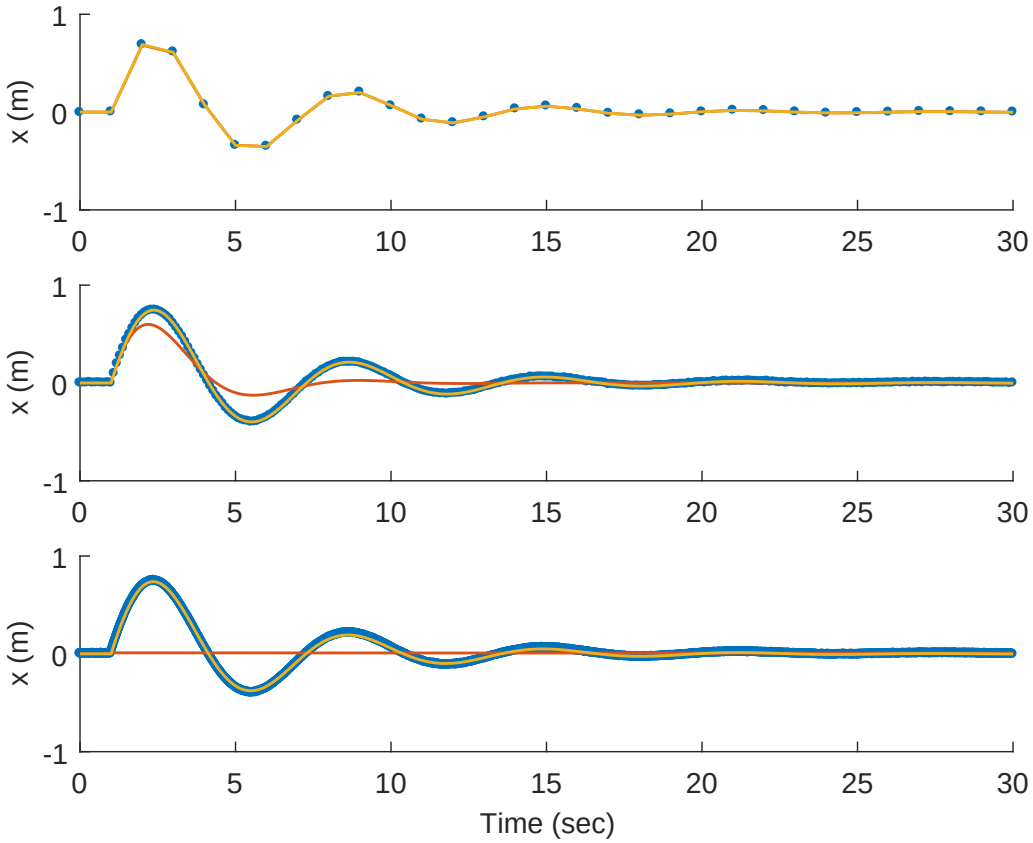


Figure 4.3: *Results of the estimation of a second order linear ARX model with focus set to prediction (red line) and simulation (yellow line) for different sampling intervals. The blue dots represent the measurements. Top:* $T = 1$ *sec. Middle:* $T = 0.1$ *sec. Bottom:* $T = 0.01$ *sec.*

4.4 Model determination

The identification of the linear models involve specifying the model orders n_a and n_b, the delay n_k and estimating the coefficients of the transfer functions $G(q, \theta)$ and $H(q, \theta)$. The model orders determine how many previous samples are used to compute the current output. The model order needs to be specified before the corresponding parameters can be estimated. Physiological insight and the intended model application could be useful to tell which range of model orders should be considered. When this is not possible, the System Identification Toolbox offers some tools to assist in the task of model order selection. Since the choice of model order is also influenced by the size of the delay, first some methods will be discussed that can be used to determine the size of the delay.

4.4.1 Delay determination

The System Identification Toolbox offers two standard methods that can be used to determine the delay n_k from the input to the output. These two methods are discussed in Appendix A. Appendix A also shows that, in this thesis, these two methods were not able to determine the delay. Two other methods

that can be used to determine the delay is with use of visual inspection or with use of two information criteria.

The method of visual inspection suggests to plot the input and output data and then to read the time difference between the first change in the input and the first change in the output. This method is practical for SISO systems, however, in case of MISO systems it might be difficult to tell which input caused the initial change in the output or if the change is caused by external factors.

The information criteria method is not a standard method to estimate the delay from the inputs to the outputs. However, the results discussed in section 4.7 showed that this method can be used to estimate the delays. The following two criteria are used: (1) Akaike's Final Prediction Error (FPE) and (2) FIT. The two criteria are explained in more detail in the next subsection. The idea is to compute, for a set of low order models with a certain range of delays, the corresponding FPE and FIT values and to return the delays corresponding to the models with the minimum FPE and maximum FIT.

The impulse responses of the low order models and estimated delays can be used to determine which delay is the "best" delay and need to be used in the model order determination. The requirement is that the impulse responses need to be physiologically meaningful, i.e. the impulse responses should not show any behaviour that is physiologically impossible.

4.4.2 Order determination

Once the delay of the model is estimated, the orders of the corresponding polynomials n_a and n_b need to be determined. In general, the aim is to use a model order not higher than necessary. If the order is higher than necessary, the extra parameters are basically used to model the measurement noise.

The order of the polynomials can be determined with the use of several selection criteria. In this thesis the following two criteria were used:

- Akaike's Final Prediction Error (FPE)
  Akaike's Final Prediction Error (FPE) is defined as:

$$FPE = \frac{1 + (\frac{d}{N})}{1 - (\frac{d}{N})} \frac{1}{N} \sum_{t=1}^{N} e^2(t, \hat{\theta}_N) \tag{4.35}$$

  where N is the number of measurements in the estimation data set, $e(t)$ the estimation error at time t, $\hat{\theta}_N$ the estimated parameters and d the number of estimated parameters.

  The FPE contains two terms. The first term describes the model complexity, while the second term describes the model accuracy. After identification of several different models, the models can be compared using this criteria. The "best" model is the model with the smallest FPE value and gives the "best" trade-off between accuracy and complexity [27].

- FIT
  The FIT of a model estimation is a statistical metric that quantifies how much of the variation in the data is explained by the model estimation [16] and is defined as:

$$FIT = \left(1 - \frac{\|y - \hat{y}\|}{\|y - \bar{y}\|}\right) \times 100\% \tag{4.36}$$

  where y is the measured output, $\hat{y}$ is the model output, $\bar{y}$ is the mean of the measured output, and $\|.\|$ is the 2-norm of the corresponding vector.

  The value of FIT varies between $-\infty$, which indicates very poor model estimation, and 100, which is obtained for a perfect estimation. A value of zero indicates that the model is no better at fitting the measured data than a straight line equal to the mean of the data. After identification of several different models, the models can be compared using this criteria. The "best" model is the model with the highest FIT value.

Two different information criteria are described, because the quantity that is minimized/maximized depends on the choice of `Focus`. For example, if `Focus` is set to `simulation`, the FIT criterion is obtained for the simulation error $e_{\text{sim}}(t)$ given by (4.30). If `Focus` is set to `prediction`, the FIT criterion is obtained for the one-step ahead prediction error $e_{\text{pred}}(t)$ given by (4.28). However, the estimation error in the FPE criterion is computed as the one-step ahead prediction error $e_{\text{pred}}(t)$, regardless of the choice of `Focus`. Thus, even when the model is obtained by minimizing the simulation error, the FPE criterion is computed with use of the one-step ahead prediction error. This means that, when a certain model is identified with the `Focus` set to `simulation`, the FIT criterion needs to be used instead of the FPE criterion. In case the model is identified with the `Focus` set to `prediction`, both criteria can be used.

4.5 Residual analysis

Model validation is the problem of determining whether the estimated model agrees sufficiently with the observed data. Residual analysis is an useful technique to validate estimated models.

The residuals are defined as the differences between the one-step ahead predicted model output and the measured output. In other words, the residuals are those parts of the data that could not be explained by the model. Hence, these residuals contain information about the quality of the model. The residuals associated with the data and a given model should be a realization of white noise. Therefore, residual analysis consists of two tests: the whiteness test and the independence test.

The whiteness test is used to check whether the residuals are uncorrelated. In this test the autocorrelation function for the residuals is computed together with the confidence interval of the corresponding estimates. For a good model, it is required that the autocorrelation function stays inside the confidence interval for nonzero time shifts k.

The independence test is used to check whether the residuals are uncorrelated with past inputs. In this test the cross-correlation of the residuals with the input signals is computed together with the confidence interval of the corresponding estimates. For a good model, it is required that the cross-correlation function stays inside the confidence interval for nonzero time shifts k. If there is a peak outside the confidence interval for nonzero time shifts, it means that the effect of the input $u(t-k)$ on the output $y(t)$ is not properly described by the model. Hence, the model could be improved.

The model should pass both the whiteness and the independence test, except when `Focus` is set to `simulation`. When the focus is set to `simulation`, the model output is assumed to be noise-free, which means that the modelling focuses on the estimation of $G(q,\theta)$ and not on the estimation of $H(q,\theta)$. In this case, the model only needs to pass the independence test and not the whiteness test [33].

4.6 Clinical data

The data used for the identification of the linear models are obtained from [5]. In this study home treatment with a bi-hormonal artificial pancreas was compared with standard insulin pump therapy in patients with type 1 diabetes. Patients were treated with the artificial pancreas for four days, started on a Saturday. The first day the patients were admitted to a clinical research centre and received training on the use of the artificial pancreas. After an overnight stay at the research centre, the patients went home to use the system for three days. During the treatment period the patients were allowed to carry out their normal activities and there were no restrictions on meals. The patients were asked to keep a diary to record self-monitored blood glucose measurements, meals and activities. The carbohydrate content of the meals were estimated by the patients or by the research team based on the notes in the diary.

The study included 16 patients, of whom 10 completed the study. The data of these 10 patients (5 women en 5 men) were available for identification. The patients were labelled with a patient identification number (PIN), ranging from 401 to 416. Figure 4.4 shows the data from one particular patient (PIN413) for day 2, 3 and 4, where the days are separated by the black dotted vertical lines. The first day was excluded because it was considered as a start-up day. The first plot is the blood glucose concentration (mmol/L) measured every ten seconds with use of two glucose sensors. The second plot is the amount of carbohydrates (gram) estimated based on the patients diaries. The third and fourth plot are the amount of insulin and glucagon (Units) administered by the artificial pancreas, respectively.

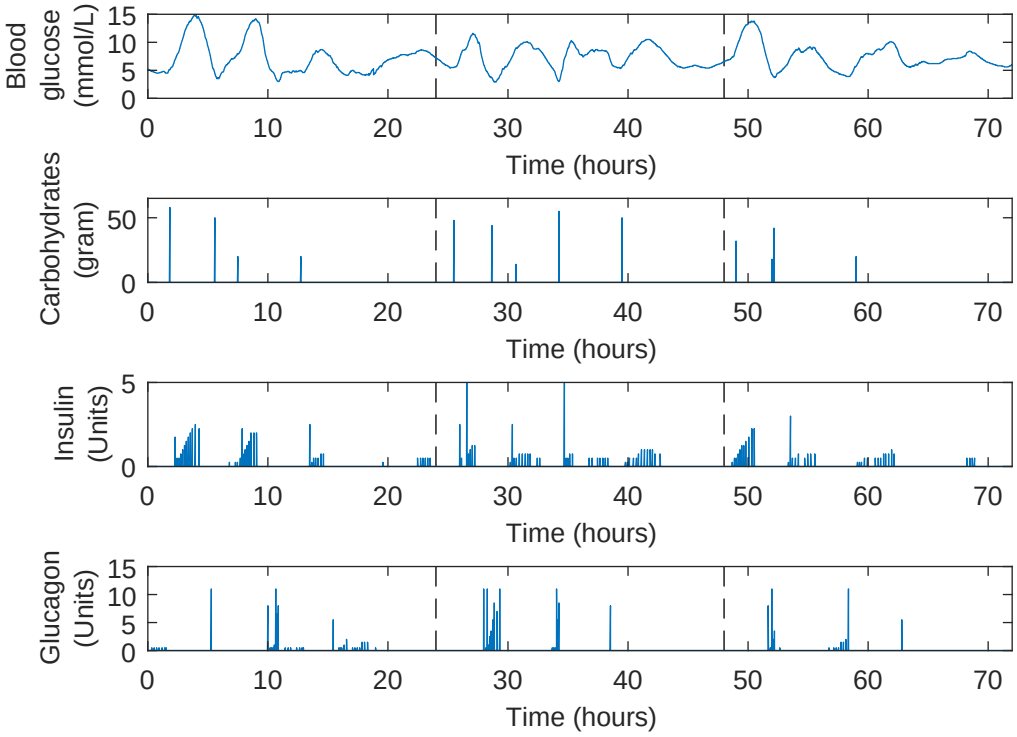


Figure 4.4: *The data of one patient (PIN413) for day 2, 3 and 4. The days are separated by the black dotted vertical lines. Top: Glucose concentration [mmol/L]; Upper-Center: Carbohydrate intake [gram]; Lower-Center: Insulin doses [Units]; Bottom: Glucagon doses [Units]*

4.7 Results and discussion

This section discusses the results of the system identification performed with the clinical data. The results below have been obtained with use of the clinical data of patient PIN413. The linear models that were identified are the MISO versions of (4.6) and (4.14). The models have three inputs, the amount of carbohydrates u_M, the administered amount of insulin u_I and the administered amount of glucagon u_H, and one output, the blood glucose concentration G. This means that the linear models are described by:

ARX:

$$A(q)y(t) = B_1(q)u_M(t - n_{k1}) + B_2(q)u_I(t - n_{k2}) + B_3(q)u_H(t - n_{k3}) + \varepsilon(t) \tag{4.37}$$

ARMAX:

$$A(q)y(t) = B_1(q)u_M(t - n_{k1}) + B_2(q)u_I(t - n_{k2}) + B_3(q)u_H(t - n_{k3}) + C(q)\varepsilon(t) \tag{4.38}$$

where n_{ki} is the time-delay of the corresponding input and $A(q)$, $B_i(q)$ and $C(q)$ are defined as (4.7), (4.8) and (4.15).

4.7.1 Pre-treatment of the data

For patient PIN413, four days of data were available, from which only the last three days were used for identification. Each day was partitioned into three parts in order to deal with the uncertainties in the meal information and changing insulin sensitivity during the day. The two main uncertainties in the meal information are the carbohydrate content of the meals and the moment a certain meal was ingested. The carbohydrate content of the meals was estimated by the patients or by the research team based on the notes in the diary. However, Brazeau et al. [6] showed that 63% of the meals are underestimated by approximately 20% of the total carbohydrate content per meal. In addition, in case the research team had to estimate the size of the meals, the research team had to rely on the information written down in the diaries. There were also some cases in which a patient discovered, for example, in the evening that he/she forgot to write down the moment he/she eat his/her lunch. At the time this was discovered, the patient wrote down the moment of his/her lunch. However, this can lead to incorrect information, because he/she may not be sure at what time the lunch was ingested.

The three days were partitioned as follows:

- Day 2 - Sunday
  - Morning + afternoon: 07:00 – 18:18
  - Evening: 18:18 – 23:00
  - Night: 23:00 – 08:00
- Day 3 - Monday
  - Morning + afternoon: 08:00 – 17:00
  - Evening: 17:00 – 22:00
  - Night: 22:00 – 07:00
- Day 4 - Tuesday
  - Morning + afternoon: 07:00 – 17:00
  - Evening: 17:00 – 23:00
  - Night: 23:00 – 07:00

By using the partitions defined above, the three meals were divided over the several parts of the day. The ”morning + afternoon” part contained the breakfast and the lunch, whereas the ”evening” part contained the dinner. The night part was not used for identification, since in the night no meals were taken and in some cases also no glucagon was administered. The morning and afternoon had to be taken together, because else there was no sufficient amount of glucagon available in each part, which could lead to identifiability problems.

Before the parameters of the linear models were estimated, some pre-treatment was performed. First, the sampling time of the output and the three inputs were matched. The artificial pancreas, used in the data collection, measures the blood glucose concentration every ten seconds. The moment one of the corresponding injections need to be administered is also determined every ten seconds. However, the corresponding injections are administered in multiple shots and therefore can be administered every second. In addition, the meals can also be ingested every second. Therefore, the three inputs were resampled such that the sampling time of each input was equal to ten seconds.

Second, offsets in the data were removed. Offsets need to be removed in order to estimate more accurate models, because linear models can only describe the relationship between the change in the input and the change in the output. In addition, linear models cannot capture arbitrary differences between the input and the output. Offsets in the blood glucose concentration were removed by direct subtraction of the physiological equilibrium. The physiological equilibrium was assumed to be the same as the first value of each partition. Therefore, this value was subtracted from the other blood glucose concentrations in the same partition. This was possible because the partitions were chosen in such a way that at the beginning of each partition the blood glucose concentration was at a certain equilibrium value and not in an increase or decrease of blood glucose concentration. This is also the reason why the partitions defined above start and end at different time instants.

Third, the data were checked on missing data points. The missing data points, which could be due to malfunctions in the CGM sensors or in the communication link, were reconstructed by piece-wise linear interpolation. On day 3 the most data points were missing, namely 2.28% from the total of 8640 measurements. The maximum amount of consecutive missing data points was 36, which corresponds to 6 minutes of consecutive data points missing.

4.7.2 Identification of the linear models

During identification, the data of one of the partitions of one of the three days was used for estimation. Initially, it was the idea to validate the estimated model on the same partition of the other two days and to choose the model with minimum FPE or maximum FIT on the validation days. In this case, the FIT value was required to be higher than 50% on the validation data. A minimum FPE value was not specified since the FPE of a model always needs to be compared with the FPE of other models. However, in that case, it was impossible to identify one model that met that criterion. For example, in case the

models were estimated (with `Focus` set to `simulation`) on the "morning + afternoon" part of day 3 and validated on the "morning + afternoon" part of day 2 and day 4, the maximum FIT achieved on day 2 was -17.53%, whereas the maximum FIT on day 4 was 15.62%. Therefore, the models were estimated and "validated" on the partition of one particular day and the model with minimum FPE or maximum FIT was chosen, where the FIT was required to be at least 50% or higher.

The linear models were identified on the "morning + afternoon" part of one of the days in order to deal with the meal uncertainties mentioned above. Brazeua et al. [6] showed that fewer errors were made when estimating the amount of carbohydrates in the breakfast. The reason for this is that there is generally less variation in its composition, fewer ingredients are used and therefore fewer errors can be made.

The results discussed below were obtained on the "morning + afternoon" part of day 3. The models (with the same order and delays) were also identified on the "morning + afternoon" part of day 2 and day 4. However, the estimated model with the highest FIT was obtained for estimation on the "morning + afternoon" part of day 3.

The results discussed below were obtained with focus set to simulation. The models were also estimated with the focus set to prediction (default). This method was also used in [32]. However, with the clinical data in this thesis, it was impossible to estimate a low order model with a FIT of 50% or higher on the estimation data. In case of prediction, a FIT higher than 50% could only be achieved for the model orders $n_a = 2$, $n_{bi} \geq 500$ and $n_k = \begin{bmatrix} 104 & 380 & 96 \end{bmatrix}$. The main reason for this is the high sampling frequency. A high sampling frequency leads to very small variations between consecutive data points. In case a low order model is identified wit the focus set to prediction, the model predicts the current output to be equal to the previous outputs. In other words, the model produces an excellent one-step ahead prediction with minimum prediction error. However, this model is useless for simulation. The same was observed in Example 1 in section 4.3. In [32], the sampling frequency was equal to two minutes and therefore more variations between consecutive data points were present.

First, the model orders and time-delay of each input were estimated. The model orders and time-delays were only estimated for the ARX model, because ARMAX models differ from ARX models only in the description of the noise. The estimated model orders and delays were used as a starting point for the order determination of the ARMAX model.

Delay determination
The two methods explained in section 4.4 were used to estimate the time-delay of each input. The two methods discussed in Appendix A were also used, however, these were unable to estimate realistic values for the delays (see Appendix A for the results). The estimated time-delays were compared with the time-delays mentioned in literature. In literature, the following time-delays are mentioned:

- The blood glucose concentration begins to rise within 15 to 30 minutes after ingestion of carbohydrates [23], i.e. the first input has a minimum delay of 90 to 180 time units.
- Fast-acting insulin starts to work about 15 minutes after administration, peaks in about 1 hour, and keeps working for 2 to 4 hours [24], i.e. the second input has a minimum delay of 90 time units.
- The effect of glucagon starts within 10 to 15 minutes and lasts for 30 to 60 minutes [22], i.e. the third input has a minimum delay of 60 to 90 time units.

First, the delay of the three inputs were estimated with the method of visual inspection. As mentioned before, this method is difficult to use in case of MISO systems, since it might be difficult to tell which input caused the initial change in the output or if this change is caused by external factors. Figure 4.4 shows this difficulty. In this figure can be seen that in some cases meals were ingested at the same moment glucagon was administered. This makes it very difficult to distinguish which input caused the initial change in the output, because both inputs an increase of the blood glucose concentration. Nevertheless, this method was applied on the data in order to estimate the time delays of the three inputs. The following results were obtained with use of this method:

- In case of meals, an average delay of 94 time units ($\approx$ 15 minutes) was estimated
- In case of insulin, an average delay of 237 time units ($\approx$ 39 minutes) was estimated
- In case of glucagon, an average delay of 81 time units ($\approx$ 13 minutes) was estimated

Second, the delay of the three inputs were estimated with use of the FPE and FIT criteria. To estimate the delay of each input, the model orders n_a, n_b and a certain range of delays needed to be specified. After some trial-and-error, and keeping the time-delays mentioned in literature in mind, the following model orders and range were chosen: $n_a = 2$, $n_b = \begin{bmatrix}1 & 1 & 1\end{bmatrix}$, $70 \leq n_{k1} \leq 130$, $180 \leq n_{k2} \leq 500$ and $60 \leq n_{k3} \leq 110$. For each model-order combination the corresponding ARX models were identified, the FPE and FIT values of these identified models were computed and the delays corresponding to the models with the lowest FPE and highest FIT were returned. The following results were obtained:

- Minimum FPE (FPE = 0.0018781, FIT = 75.49%)
  - In case of meals, a delay of 104 time units ($\approx$ 17 minutes) was estimated
  - In case of insulin, a delay of 380 time units ($\approx$ 63 minutes) was estimated
  - In case of glucagon, a delay of 96 time units (16 minutes) was estimated
- Maximum FIT (FIT = 82.48%, FPE = 0.0018827)
  - In case of meals, a delay of 78 time units (13 minutes) was estimated
  - In case of insulin, a delay of 244 time units ($\approx$ 41 minutes) was estimated
  - In case of glucagon, a delay of 84 time units (14 minutes) was estimated

Compared to the time-delays mentioned in literature, it can be concluded that with both methods the estimated delays are realistic values. The only value that is questionable is the time-delay of insulin, because the influence of insulin on the blood glucose concentration is much earlier visible.

The delay used in the rest of the model identification is determined with use of the impulse responses of the models with orders and delays estimated above. The requirement is that the impulse responses should not show any behaviour that is physiologically impossible. The impulse responses for the three models are shown in figure 4.5. The columns correspond to the impulse response of the meal intake, insulin administration and glucagon administration, respectively. The top row corresponds to the impulse responses of the delay determined by visual inspection, the center row to the impulse responses of the delay determined by minimizing FPE and the bottom row to the impulse responses of the delay determined by maximizing FIT. In this figure can be seen that the impulse responses of the first and third model show some oscillations, which are not visible in the impulse responses of the second model. These oscillations are not expected for patients with type 1 diabetes, since ideally a certain amount of carbohydrates, insulin or glucagon cannot cause an increase of the blood glucose concentration followed by a decrease of the blood glucose concentration. Therefore, the delay that was obtained by minimizing FPE was used in the model identification, i.e. $n_k = \begin{bmatrix}104 & 380 & 96\end{bmatrix}$.

Order determination
The orders of the ARX model were estimated with the delay mentioned above. For the estimation of the model orders only the FIT criterion was applied to assess the estimated orders. As mentioned in section 4.4, the FPE criterion computes the estimation error as the one-step ahead prediction error, even when the focus is set to simulation. The FIT criterion is computed with use of the actual quantity that is minimized during the estimation. In our case, the focus is set to simulation, which means that the FIT criterion is needed and not the FPE criterion. In order to determine the model orders, the following steps were taken. First, models with order $2 \leq n_a \leq 10$, $1 \leq n_{bi} \leq 10$, $n_{k1} = 104$, $n_{k2} = 380$ and $n_{k3} = 96$ were identified and their corresponding FIT was computed. Second, the models were ranked in decreasing order according to their FIT. Third, the models were compared with the FIT of the ARX model with the lowest order, i.e. $n_a = 2, n_{b1} = n_{b2} = n_{b3} = 1$. For this model the FIT is equal to 75.49%.

Table 4.1 shows the five models that were ranked the highest according to their FIT. In this table it can be seen that the ARX model with order $n_a = 9, n_b = \begin{bmatrix}4 & 9 & 9\end{bmatrix}$ is the model with the highest FIT. Compared to the FIT of the ARX model with the lowest order, it can be concluded that an increase in model order does not cause a significant increase in FIT. Thus, in order to keep the model as simple as possible, the ARX model with lowest model order is chosen for identification.

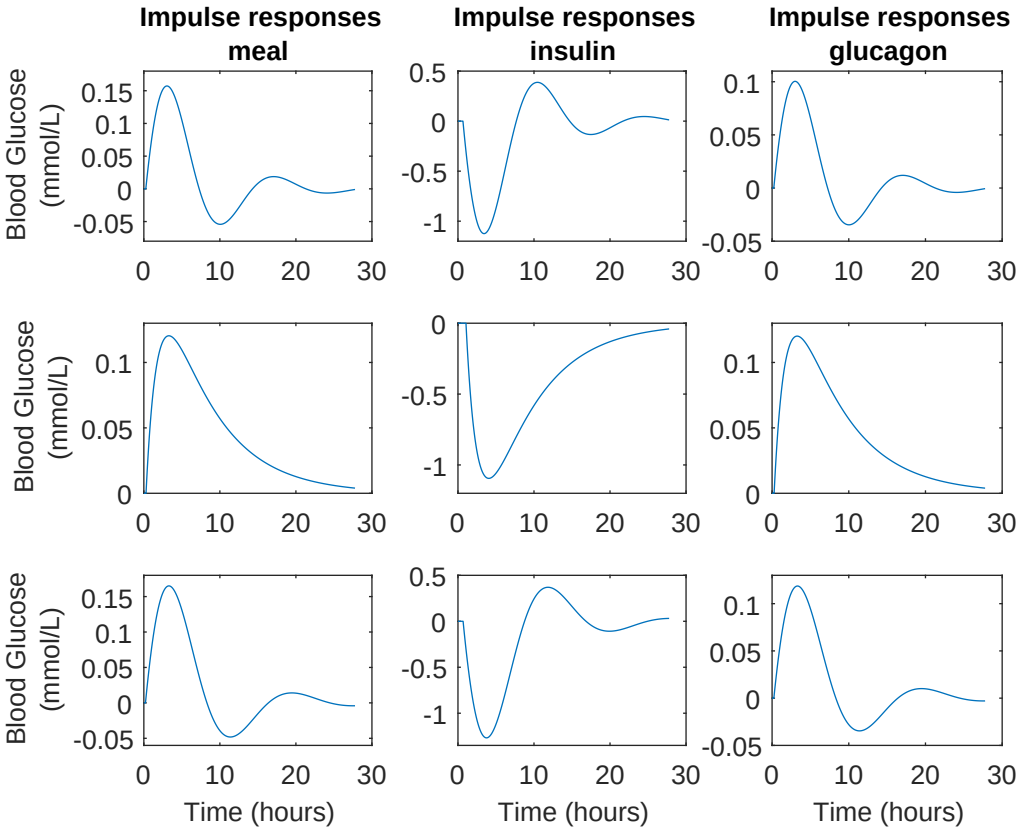


Figure 4.5: *Impulse responses of the meals, insulin and glucagon for three low order ARX models with different values for n_k. Top row: Impulse responses for the model with delay estimated by visual inspection; Center: Impulse responses for the model with delay estimated by minimizing FPE; Bottom row: Impulse responses for the model with delay estimated by maximizing FIT*

Table 4.1: *The five highest ranked models based on FIT (decreasing order)*

n_a	n_{b1}	n_{b2}	n_{b3}	FIT (%)
9	4	9	9	76.30
9	5	9	9	76.29
9	4	9	10	76.29
9	3	9	9	76.29
9	4	9	8	76.29

Parameter estimation - ARX

The ARX model, with the orders and delays determined above, was identified with use of the MATLAB function `arx`. With use of this function the following polynomials were estimated:

$$A(q) = 1 - 1.9978519q^{-1} + 0.9978526q^{-2} \tag{4.39a}$$

$$B_1(q) = 0.000327445q^{-104} \tag{4.39b}$$

$$B_2(q) = -0.002977359q^{-380} \tag{4.39c}$$

$$B_3(q) = 0.000326746q^{-96} \tag{4.39d}$$

The poles of this model are located at $z_1 = 0.9996$ and $z_2 = 0.9983$, i.e. the poles are inside the unit circle and therefore the system is asymptotically stable.

Figure 4.6 shows the output of the ARX model applied to the ”morning + afternoon” part of day 2, day 3 and day 4. The blue lines indicate the data, the red lines indicate the output of the model and the black hexagrams indicate the moment of the meals. This figure shows the problem discussed at the beginning of this section, namely that the identified model is able to reproduce the estimation data (day 3), but is unable to reproduce the data of the other two days.

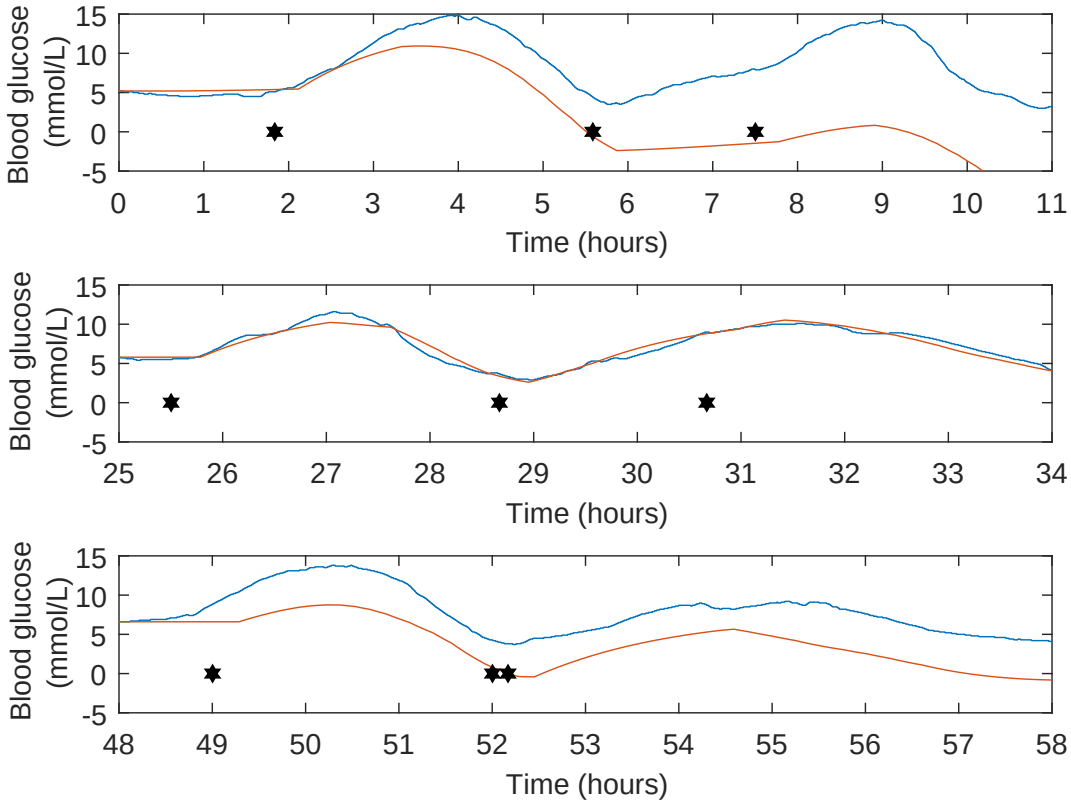


Figure 4.6: *The output of the ARX model applied to the "morning + afternoon" part of day 2 (top), day 3 (middle) and day 4 (bottom). The blue lines indicate the data, the red lines indicate the output of the model and the black hexagrams indicate the moment of the meals.*

One possible reason why the model is unable to reproduce the data of day 2 could be the non-linearity of the meals. In section 2.1.1 it was explained that the effect of a meal on the blood glucose concentration depends on the glycemic index. A different meal composition leads to a different glycemic index and therefore a different influence on the blood glucose concentration. At $t = 1.8$ a breakfast of 58 grams of carbohydrates was ingested. However, this amount of carbohydrates is not able to describe the increase in the blood glucose concentration. This could indicate that the composition of this breakfast is different from the composition of the meals used in the model identification. The same holds for the meals at $t = 5.5$ and $t = 7.5$. At $t = 5.5$ a lunch with 44 grams of carbohydrates was ingested, whereas at $t = 7.5$ candies with 20 grams of carbohydrates were ingested. The increase of the blood glucose concentration is larger after the candies than after the lunch. Probably, the candies contain more fast carbohydrates than the lunch. Meals with fast carbohydrates are processed faster by the human body than meals with slower carbohydrates. This non-linearity cannot be modelled with a linear model.

Another reason why the model is unable to reproduce the data of day 2 could be that the insulin sensitivity of the patient on day 2 was different from the insulin sensitivity on day 3. The insulin sensitivity gives an estimate of how much your blood glucose concentration will drop in response to 1U of insulin. The "1800 rule" can be used to calculate the insulin sensitivity factor (ISF) for patients who use rapid-acting insulin [12]. The ISF can be computed by dividing 1800 by the total daily dose of rapid-acting insulin. On day 2, a total amount of 32U of insulin was administered, which corresponds to an ISF of 56.25 mg/dl per 1U of insulin or 3.12 mmol/L per 1U of insulin. On day 3, a total amount of 20.75U of insulin was administered, which corresponds to an ISF of 86.75 mg/dl per 1U of insulin or 4.81 mmol/L per 1U of insulin. The ISF of day 3 was approximately 50% higher than the ISF of day 2. In other words, the patient was more sensitive for insulin on day 3 than on day 2. This means that on day 3 less insulin was needed to obtain the same decrease in blood glucose concentration as on day 2. This difference in ISF could be the reason why the decrease in the model output on day 2 is larger than the decrease in measured blood glucose concentration.

One possible reason why the model is unable to reproduce the data of day 4 could be the dawn phenomenon. The dawn phenomenon is an early-morning increase of the blood glucose concentration relevant to people with diabetes. On day 4 at $t = 49$, when the breakfast was ingested, the blood glucose concentration was already increased. The moment the breakfast was ingested, the blood glucose concentration was already equal to 8.80 mmol/L, whereas the blood glucose concentration on the other two days at that moment was equal to 5.20 mmol/L and 5.80 mmol/L, respectively. The identified model is only able to describe changes in the output based on changes in the input. However, in absence of meals and/or glucagon, the identified model is unable to describe this increase of the blood glucose concentration.

The above showed that the identified model is not able to reproduce the data of two other two days. Figure 4.7 shows the output of the ARX model applied to the data of the three partitions of day 3 together with the ingested carbohydrates, the administered insulin and the administered glucagon. The three partitions are indicated by the black dotted vertical lines. The blue line indicates the data and the red line indicates the output of the model. In this figure can be seen that the model is able to reproduce the ”morning + afternoon” part, but unable to reproduce the rest of the day. The moment the ”evening” part starts a dinner was ingested. The increase of the model output is slower then the increase of the measured blood glucose concentration. One possible reason could be the non-linearity of the meals as discussed above. Another possible reason could be that the patient has been active during the day, whereas in the morning the body has been in rest during the night. In this figure it can also be seen that, from $t = 36$, the model output drifts away from the measured blood glucose concentration. At this moment, the measured blood glucose concentration starts to increase and then it stabilizes around 9 mmol/L. However, in this area no carbohydrates were ingested and no glucagon was administered. It could be possible that the patient forgot to write down a meal. Without a meal or glucagon, the identified model is unable to describe this increase of the blood glucose concentration and therefore the output of the model decreases until the moment carbohydrates or glucagon was taken. The model is unable to describe the rest of the ”evening” part due to this drift.

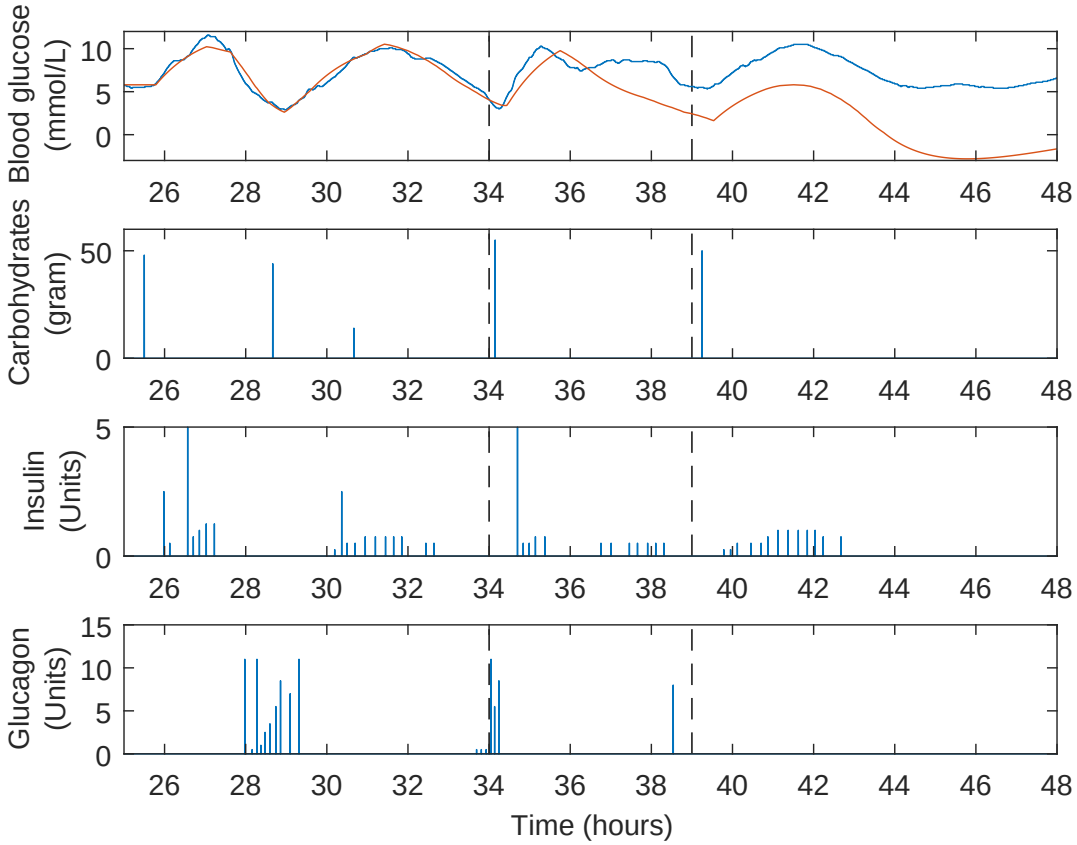


Figure 4.7: *The output of the ARX model applied to the data of day 3. The blue lines indicate the data and the red line indicates the output of the model. The model is able to simulate the ”morning + afternoon” part, but unable to simulate the rest of the data.*

Parameter estimation - ARMAX
The model orders of the ARMAX model were estimated in the same manner as in case of the ARX model. First, models with order $n_a = 2$, $1 \leq n_{bi} \leq 5$, $n_k = \begin{bmatrix}104 & 380 & 96\end{bmatrix}$ and $1 \leq n_c \leq 5$ were identified and their corresponding FIT was computed. The value of n_a was set to 2, because during identification of the ARX model it turned out that the coefficients $a_3, a_4, \cdots, a_{10}$ of the A-polynomial were not significantly different from 0. To save computation time, the values of n_b and n_c were set to a maximum of 5 instead of 10. Second, the models were ranked in decreasing order according according to their FIT. Third, the models were compared with the FIT of the ARMAX model with the lowest order, i.e. $n_a = 2$, $n_b = \begin{bmatrix}1 & 1 & 1\end{bmatrix}$ and $n_c = 1$. For this model the FIT is equal to 74.84%.

The maximum FIT was obtained for $n_a = 2$, $n_b = \begin{bmatrix}3 & 5 & 5\end{bmatrix}$ and $n_c = 1$ and is equal to 83.89%. Compared with the low order ARMAX model, it can be seen that the the FIT increases with almost 10%. Due to this increase, the ARMAX model with maximum FIT is preferred above the low order ARMAX model.

The ARMAX model, obtained with the highest FIT, was identified with use of the MATLAB function `armax`. With use of this function the following polynomials were estimated:

$$A(q) = 1 - 1.99934q^{-1} + 0.99934q^{-2} \tag{4.40a}$$

$$B_1(q) = 0.002647q^{-104} + 0.00236q^{-105} - 0.004724q^{-106} \tag{4.40b}$$

$$B_2(q) = -0.1197q^{-380} + 0.04246q^{-381} - 0.03024q^{-382} + 0.004456q^{-383} + 0.1008q^{-384} \tag{4.40c}$$

$$B_3(q) = 0.00977q^{-96} - 0.004187q^{-97} - 0.005622q^{-98} + 0.002433q^{-99} - 0.002203q^{-100} \tag{4.40d}$$

$$C(q) = 1 + 0.9452q^{-1} \tag{4.40e}$$

The poles of this model are located at $z_1 = 0.9992$ and $z_2 = 0.9987$, i.e. the poles are inside the unit circle and therefore the system is asymptotically stable.

Figure 4.8 shows the output of the ARMAX model applied to the ”morning + afternoon” part of day 2, day 3 and day 4. The blue lines indicate the data, the red lines indicate the output of the model and the black hexagrams indicate the moment of the meals. This figure shows that the ARMAX model is able to reproduce the estimation data (day 3), but unable to reproduce the data of the other two days. This could be explained with the same reasons as mentioned in the case of ARX.

Comparison ARX and ARMAX
Figure 4.9 shows the output of both the ARX and ARMAX model applied to the ”morning + afternoon” part of day 3. The blue line indicates the data, the red line indicates the output of the ARX model and the yellow line indicates the output of the ARMAX model. This figure shows that even though the ARMAX model obtained a higher FIT, the output of the ARX is more realistic than the output of the ARMAX model. At $t = 27$, the output of the ARMAX model drops 1 mmol/L within 40 seconds. This corresponds to a slope of 90 mmol/L/h, which is an unrealistic value. At $t = 27.6$, this decrease is even 2 mmol/L within 40 seconds. Therefore, the ARX model is preferred above the ARMAX model.

4.7.3 Residual analysis
Residual analysis consists of two test: the whiteness test and the independence test. Figure 4.10 shows the result of both the whiteness test and independence test of the three inputs obtained with the MATLAB function `resid`. This function generates the correlation functions for lags -25 to 25. However, the negative lags are omitted because there is feedback in the control loop. Since the ARX model is obtained with focus set to simulation, the model only needs to pass the independence test. In this figure it can be seen that for all the three inputs there are no peaks outside the confidence interval for nonzero time shifts. This means that the model is a good description of the system.

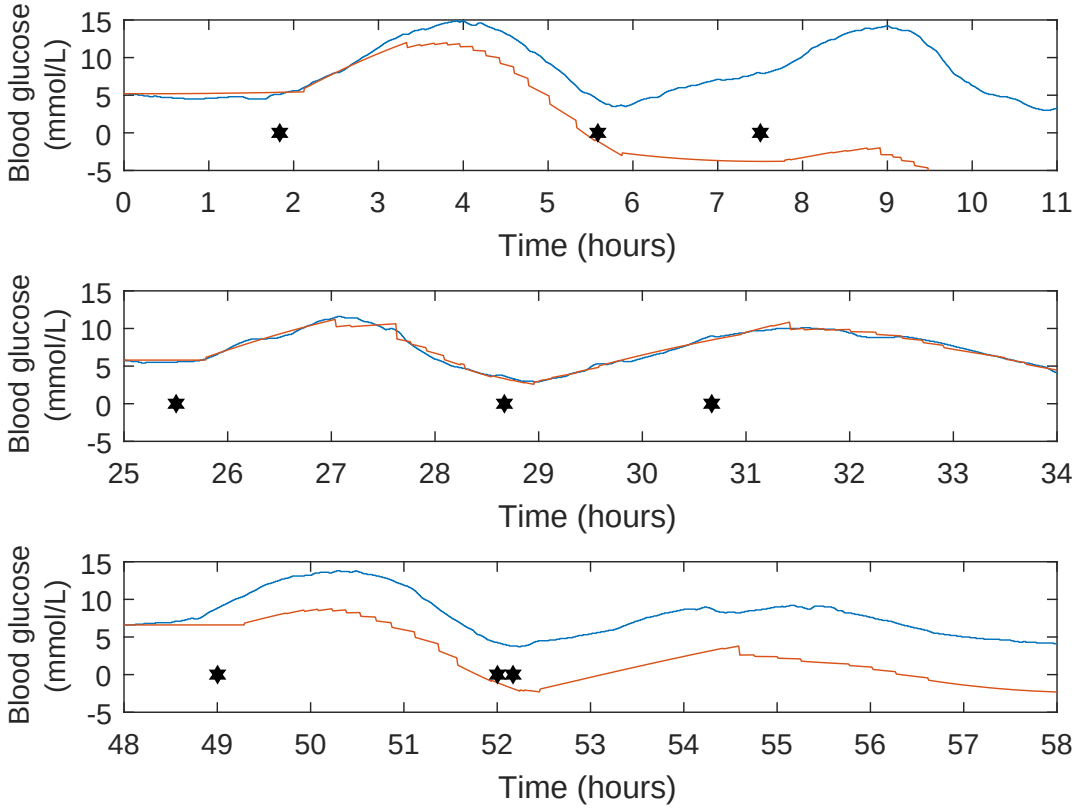


Figure 4.8: *The output of the ARMAX model applied to the "morning + afternoon" part of day 2 (top), day 3 (middle) and day 4 (bottom). The blue lines indicate the data, the red lines indicate the output of the model and the black hexagrams indicate the moment of the meals.*

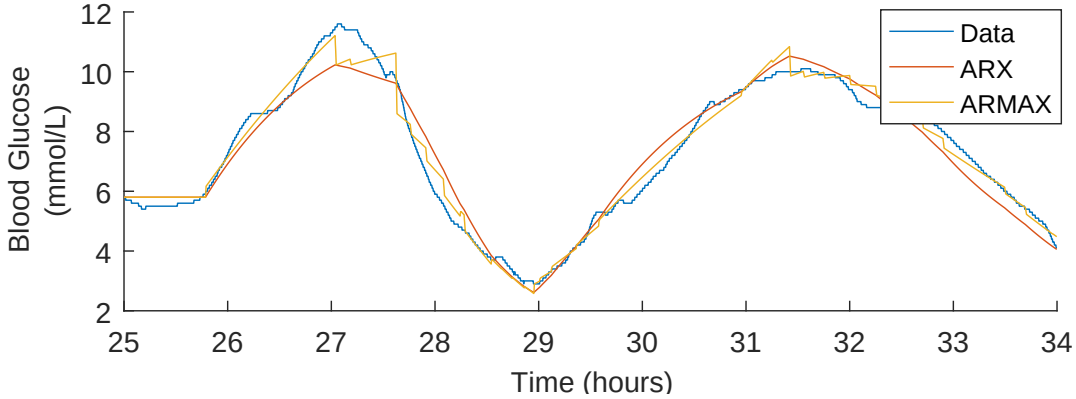


Figure 4.9: *The output of the ARX model (red) and the ARMAX model (yellow) applied to the "morning + afternoon" part of day 2.*

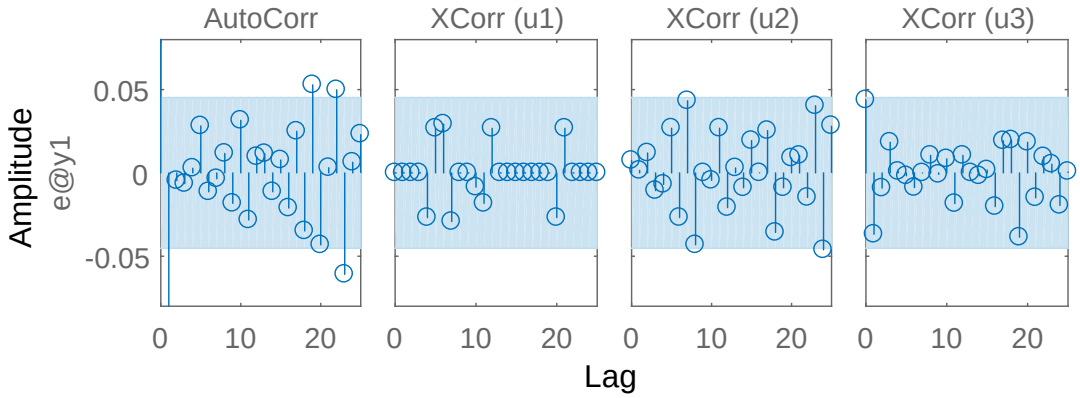


Figure 4.10: *Residual analysis of the identified ARX model. From left to right: autocorrelation function of residuals, cross-correlation between residuals and input 1, cross-correlation between residuals and input 2 and cross-correlation between residuals and input 3.*

Chapter 5

Modelling and analysis of the control algorithm

In Chapter 3 the control algorithm of interest was presented. The control algorithm consists of three types of injections of which the moment these injections need to be administered depend on timing constraints. As explained earlier, without a mathematical model that describes the functioning of the control algorithm it is difficult to investigate the performance of the control algorithm and the effect of a change in initial setting on the performance of the control algorithm. The theory of timed automata is widely used to model real-time systems of which the dynamics depend on timing constraints. In addition, the theory of timed automata can be used to analyze such real-time systems on reachability, safety and liveness [3].

Section 5.1 discusses the theory of timed automata. The section starts with some examples of timed automata, then the theory is described and the section ends with the Train-Gate-Controller example. The theory discussed in this section is based on [1].

Section 5.2 discusses the extensions of the tool Uppaal to the theory of timed automata. These extensions are needed to model the control algorithm as a timed automaton. In addition, the modelling and query language of Uppaal is presented, especially the query language of Uppaal SMC. This query language is necessary for analysis of the control algorithm.

Section 5.3 discusses the modelling and implementation of the control algorithm as timed automaton in Uppaal. Some additional timed automata were modelled to compute certain statistics of the control algorithm.

Section 5.4 discusses the validation of the identified linear model from chapter 4 and the analysis of the control algorithm in Uppaal. The control algorithm is analyzed with respect to certain requirements. In addition, the influence of a change in the initial setting of the control algorithm on the performance of the control algorithm was investigated.

5.1 Timed Automata

5.1.1 Examples

Figure 5.1 shows an example of a timed automaton. A timed automaton is a graph that contains a finite set of states or locations and a finite set of labelled edges extended with real-valued variables. These real-valued variables, called clocks, are used to model the time in the system. The clocks are initialized with zero when the system is started. After the system is started, these variables increase synchronously with the same rate. The edges can be associated with action symbols and with clock constraints to restrict the behaviour of the automaton. These constraints on edges are called guards. The guard specifies that the transition represented by an edge may be taken only if the current value of the clocks satisfy this constraint. Clocks may be reset to zero when a transition is taken. A location can be associated with a clock constraint, called an invariant. The invariant specifies that time can elapse in a location only as long as the condition is satisfied. In other words, an invariant describes when a transition *must* be taken, whereas the guard describes when a particular transition *may* be taken [17].

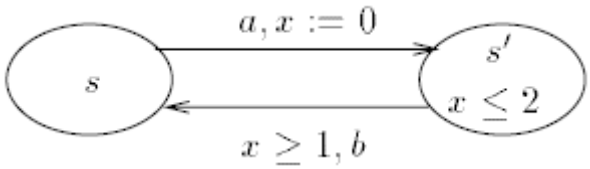


Figure 5.1: *Example of a timed automaton [1]*

The timed automaton of figure 5.1 consists of two locations s and s', where s is the initial location. The variable x is the clock of the automaton and a and b are symbols. The initial location has no invariant. This means that the automaton can spend arbitrary amount of time in this location. The system can switch to location s' on symbol a. In that case, the clock x gets reset to zero. In this location, the clock x shows the time elapsed since the occurrence of the last switch. The invariant $x \leq 2$ associated with location s' specifies that the system can stay in this location for at most 2 units and that a switch must occur before the invariant is violated. The switch from location s' to s is enabled only when the value of the clock is greater than 1. This means that the delay between a and b is always between 1 and 2.

Figure 5.2 shows an example of a timed automaton with two clocks x and y and four symbols a, b, c and d. The automaton consists of four locations, where s_0 is the initial location. Each time the system makes the transition from s_0 to s_1 on symbol a the clock x gets reset to zero. The invariant $x < 1$ associated with the locations s_1 and s_2 ensures that the transition from s_2 to s_3 happens within 1 unit after the transition from s_0 to s_1. The reset of clock y together with the guard associated with the transition from s_3 to s_0 ensures that the delay between b and d is always greater than 2. However, because the location s_3 has no invariant, the symbol d can be postponed indefinitely. This means that the transition from s_3 to s_0 is not guaranteed to happen.

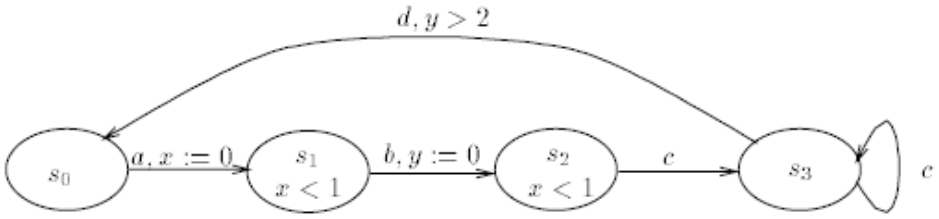


Figure 5.2: *Example of a timed automaton with two clocks [1]*

An automaton can be composed into a network of timed automata. Figure 5.3 shows an example of a timed automaton that is composed of two separate timed automata. The left automaton is the automaton as discussed in figure 5.1. The right automaton is analogous to the left automaton, but with a different clock and different symbols. The only thing that is the same is symbol b. Symbols that belong to both the automata are used for synchronization. This means that the left automaton can make the transition from location s' to s only if the right automaton can make the transition from location t to t' and vice versa. In other words, when the left automaton is in location s', but the right automaton is in location t', then the left automaton cannot make the transition from s' to s, because the right automaton is not able to make the transition from t to t'. In this definition, synchronization is blocking.

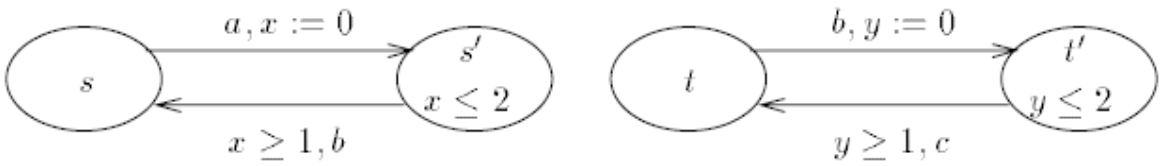


Figure 5.3: *Example of a timed automaton composed of two separate timed automata [1]*

5.1.2 Formal syntax

As explained above, locations and edges can be associated with clock constraints. Assume a finite set of real-valued clock variables C. The set $\Phi(C)$ of clock constraints φ is defined as:

$$\varphi := x \leq c \mid c \leq x \mid x < c \mid c < x \mid \varphi_1 \wedge \varphi_2$$

where $x \in C$ is a clock and $c \in \mathbb{Q}$ is a constant.

To keep track of the changes of clock values, we use functions known as clock valuations. A clock valuation is a function $\nu : C \to \mathbb{R}_{\geq 0}$ that assigns a real value to each clock in C. A clock valuation ν for C satisfies a clock constraint φ over C if and only if φ evaluates to true according to the values given by ν. For $d \in \mathbb{R}_{\geq 0}$, $\nu + d$ denotes the clock valuation that sets each clock $x \in C$ to the value $\nu(x) + d$. For $Y \subseteq C$, $\nu[Y := 0]$ denotes the clock valuation which resets each clock in Y to 0 and keeps the value of the rest of the clocks $C \backslash Y$ the same.

A timed automaton can be formally defined as follows. A timed automaton A is a tuple $\langle L, L^0, \Sigma, C, I, E \rangle$, where L is a finite set of locations, $L^0 \subseteq L$ is a set of initial locations, Σ is a finite set of labels, C is a finite set of clocks, $I : L \to \Phi(C)$ is the set of invariants, and $E \subseteq L \times \Sigma \times 2^C \times \Phi(C) \times L$ is the set of edges. An edge $\langle l, a, g, r, l' \rangle$ represents a transition from location l to location l' on symbol a, with guard g and a set of clocks r to be reset with this transition.

Let $A = \langle L, L^0, \Sigma, C, I, E \rangle$ be a timed automaton. The semantics of a timed automaton is defined as a labelled transition system T_A. A state of T_A is a pair (l, ν) such that l is a location of A and ν is a clock valuation for C such that ν satisfies the invariant $I(l)$. S_A denotes the set of all states of A. If l is an initial location of A and $\nu(x) = 0$ for all clocks $x \in C$, then the state (l, ν) is called an initial state. In the transition system T_A two types of transitions are possible:

1. The state (l, ν) can change due to elapse of time
   For a state (l, ν) and a real-valued time increment $d \geq 0$, $(l, \nu) \xrightarrow{d} (l, \nu + d)$ if for all $0 \leq d' \leq d$, $\nu + d'$ satisfies the invariant $I(l)$
2. The state (l, ν) can change due to an action transition
   For a state (l, ν) and an edge $\langle l, a, g, r, l' \rangle \in E$ such that ν satisfies g, $(l, \nu) \xrightarrow{a} (l', \nu[r := 0])$

Timed automata can be composed into a network of timed automata. Let $A_1 = \langle L_1, L_1^0, \Sigma_1, C_1, I_1, E_1 \rangle$ and $A_2 = \langle L_2, L_2^0, \Sigma_2, C_2, I_2, E_2 \rangle$ be two timed automata and assume that $C_1 \cap C_2 = \emptyset$. Then, the product, denoted $A_1 \parallel A_2$, is defined as the timed automaton $\langle L_1 \times L_2, L_1^0 \times L_2^0, \Sigma_1 \cup \Sigma_2, C_1 \cup C_2, I, E \rangle$ where $I(s_1, s_2) = I(s_1) \wedge I(s_2)$ and the edges are defined by:

1. For $a \in \Sigma_1 \cap \Sigma_2$, for every $\langle l_1, a, g_1, r_1, l_1' \rangle \in E_1$ and $\langle l_2, a, g_2, r_2, l_2' \rangle \in E_2$, E contains $\langle (l_1, l_2), a, g_1 \wedge g_2, r_1 \cup r_2, (l_1', l_2') \rangle$,
2. For $a \in \Sigma_1 \backslash \Sigma_2$, for every $\langle l, a, g, r, l' \rangle \in E_1$ and every $m \in L_2$, E contains $\langle (l, m), a, g, r, (l', m) \rangle$,
3. For $a \in \Sigma_2 \backslash \Sigma_1$, for every $\langle l, a, g, r, l' \rangle \in E_2$ and every $m \in L_1$, E contains $\langle (m, l), a, g, r, (m, l') \rangle$.

Thus, locations of the product are pairs of the component locations, and the invariant of a compound location is the conjunction of the invariants of the component locations [1]. The transitions are obtained by synchronizing the transitions with identical labels. The system synchronize not only on transitions with identical labels, but also on amount of time elapsed.

5.1.3 Example: Train-Gate-Controller

A classic example from the timed automata literature is the Train-Gate-Controller. This example considers an automatic controller that opens and closes a gate at a railroad crossing. The system is composed of three components: Train, Gate and Controller as shown in figure 5.4. The entire system is denoted as Train $\parallel$ Gate $\parallel$ Controller. The initial location of the system is $L^0 = (s_0, t_0, u_0)$.

Synchronization between the train and the controller takes place through the two labels *approach* and *exit*. The entry and exit of the train from the railroad crossing are marked with the two events *in* and *out*. At least two minutes and at most five minutes before the train enters the crossing, the train is required to send the signal *approach*. This requirement is expressed by the guard $x > 2$, associated with the

event *in*, and the invariant $x \leq 5$ in location s_1. The invariants in the locations s_1, s_2 and s_3 indicate that the maximum delay between the labels *approach* and *exit* is five minutes.

Location t_0 represents the situation in which the gate is open, whereas location t_2 represents the situation in which the gate is closed. Synchronization between the gate and the controller takes place through the two events *lower* and *raise*. The opening and the closing of the gate are marked with the two labels *up* and *down*. The gate responds to the label *lower* by closing within 1 minute, expressed with the invariant $y \leq 1$ in location t_1. The gate responds to the label *raise* within 1 to 2 minutes, expressed by the guard $y \geq 1$ associated with the event *up*, and the invariant $y \leq 2$ in location t_3.

Location u_0 represents the situation in which the controller is idle. In this location the controller waits till it receives the label *approach* or *exit* from the train. The controller sends the label *lower* to the gate 1 minute after it receives the label *approach*. This is expressed by the guard $z = 1$ associated with the label *lower* and the invariant $z \leq 1$ in location u_1. The controller sends the label *raise* to the gate at most 1 minute after it receives the label *exit*. This is expressed by the invariant $z \leq 1$ in location u_2.

The safety requirement for the system is that whenever the train is inside the gate, the gate should be closed. This means that whenever the train is in location s_2, the gate should be in location t_2. However, we could ask ourselves the question if it is possible that the train is inside the gate, but that the gate is still open. In other words, can we reach a state in which the train is in location s_2 while the gate is in location t_0? From figure 5.3 it cannot immediately be verified whether this is possible. For example, consider the scenario in which the event *approach* is immediately followed by the event *in*. This corresponds to the transition from the initial location (s_0, t_0, u_0) to (s_1, t_0, u_1), and from (s_1, t_0, u_1) to (s_2, t_0, u_1). However, if we consider the timing information, it can be established that the event *approach* cannot be immediately followed by the event *in*. This is because in the location (s_1, t_0, u_1) both clocks x and z have the same value, and hence the event *lower* with guard $z = 1$ is guaranteed to precede the event *in* with guard $x > 2$. A more extensive analysis would show that whenever the train is in location s_2, the gate is in location t_2. In other words, whenever the train is inside the gate, the gate is closed.

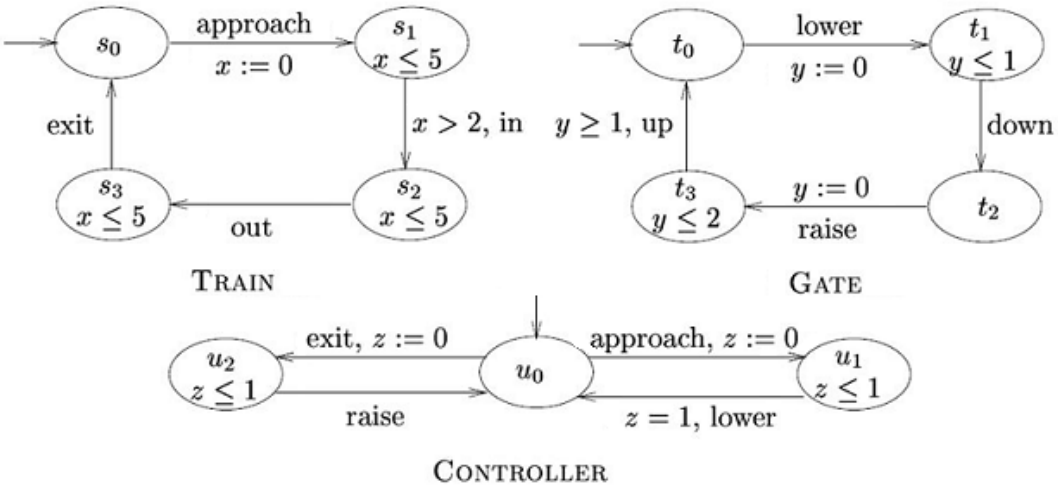


Figure 5.4: *The train-gate-controller example [1]. The system is composed of three components: the train, the gate and the controller which communicate with each others through the several labels.*

5.1.4 Tools

A variety of tools exist for the modelling and verification of real-time systems. Verification is a method used to check the correctness of a system. The three tools that are most closely related to the theory of timed automata are: Timed COSPAN, KRONOS and Uppaal [1]. For the modelling and analysis of the control algorithm the tool Uppaal will be used, because this tool extends the theory of timed automata with variables, structured data types, and channel synchronization. In addition, there is experience with Uppaal at the University of Twente.

5.2 Uppaal

The theory of timed automata discussed above is not rich enough to model the control algorithm as a timed automaton. The reason for this is that the type of injection that needs to be administered does not only depend on timing constraints, but also on the current blood glucose concentration and the glucose rate of change. Uppaal extends the theory of timed automata with variables, structured data types, and channel synchronization. These extensions are necessary to model these dependencies.

5.2.1 Extensions to the theory of Timed Automata

The Uppaal modelling language extends timed automata with the following additional features [3]:

- **Constants, bounded integer variables and floating-point variables**
  Variables can be used as part of the system. These variables can be constants, bounded integer variables or floating-point variables (also called doubles). By definition, a constant is a well-defined real number of which the value cannot be modified. Bounded integer variables are variables with an integer value with a lower bound and upper bound, whereas doubles are variables that represent numbers that have a fractional part. Both integer and double variables can be updated on edges. Guards, invariants and resets may contain expressions ranging over these variables. These variables can be initialized with a value unequal to zero.
- **Binary synchronization and broadcast channels**
  Communication between two or more automata can be achieved through the use of synchronizations via so-called channels. Channels can be binary synchronization channels or broadcast channels. In case of binary synchronization channels, an edge labelled with channel! (called the sender) synchronizes with another edge labelled with channel? (called the receiver). If several edges labelled with channel? are enabled, a synchronization pair is chosen non-deterministically. However, if there is no receiver, the automaton cannot take the edge. In other words, synchronization is blocking. Broadcast channels can be used to synchronize one sender (broadcast!) with an arbitrary number of receivers (broadcast?). The sender can still execute the broadcast! action, even if there are no receivers, i.e. broadcast sending is never blocking. If there are one or multiple receivers that have an enabled edge, these receivers will synchronize simultaneously.
- **Normal, urgent and committed locations**
  Locations can be declared as normal, urgent and committed. In normal locations time is allowed to pass and the automaton can stay in a certain location as long as the invariant is satisfied. When the automaton is in an urgent or a committed location, time is not allowed to pass. This means that an enabled outgoing edge needs to be taken without any delay. These locations are equivalent to a normal location that has an invariant $x \leq 0$, where x is a clock variable. The clock x is reset on all incoming edges. Committed locations are even more restrictive than urgent locations. In a committed location, time is not allowed to pass and the next transition must involve an outgoing edge of at least one of the committed locations.
- **Arrays**
  An array is a structure that consists of a collection of elements. These arrays can be used for clocks, channels, constants and integer variables.
- **User-functions**
  Custom user-functions can be defined in a C-like language to perform a certain action. Resets and guards on edges can refer to these user-functions.

5.2.2 Overview of the tool Uppaal

Uppaal is an integrated tool environment based on the theory of timed automata as discussed above. The tool is jointly developed by Uppsala University in Sweden and Aalborg University in Denmark and was first released in 1995. Since then it has been in constant development. In 2011 an extended version of Uppaal was released which supports the new statistical model-checker (SMC). A complete overview of the functionality of Uppaal and Uppaal SMC can be found in [3] and [10].

Uppaal consists of a graphical user interface (GUI) and a verification engine. The GUI, shown in figure 5.5, is divided into three main parts: the editor, the simulator and the verifier, accessible via the three tabs. The editor can be used for modelling, the simulator can be used for validation and the verifier can be used for verification of real-time systems modelled as a network of timed automata.

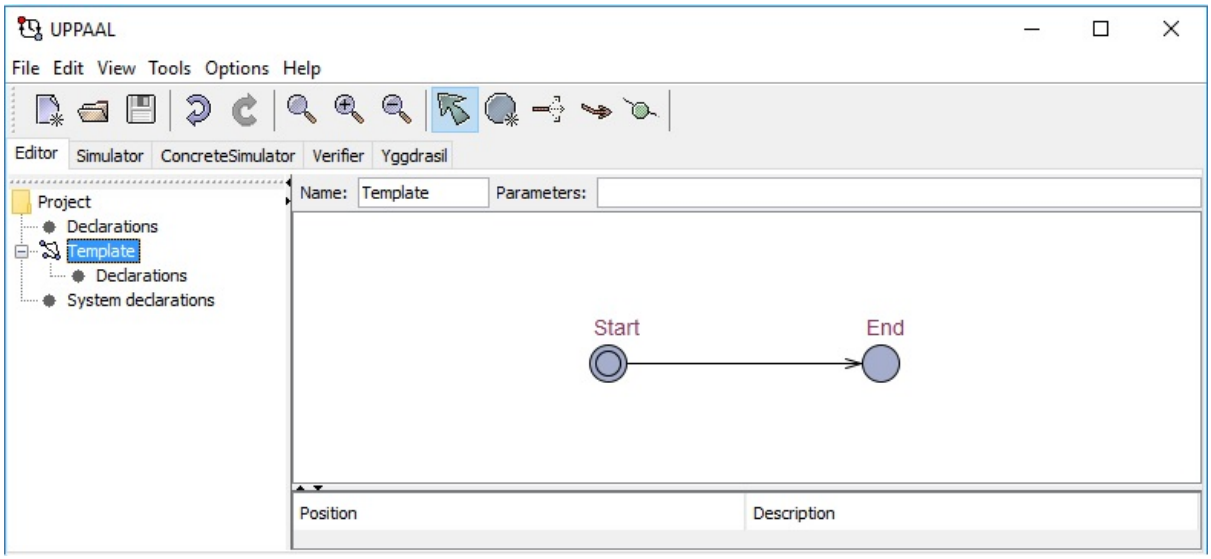


Figure 5.5: *Overview of Uppaal*

Editor
The graphical editor can be used to construct the system as a network of timed automata, called processes in the tool, working in parallel. Each automaton is constructed and instantiated from a parameterised template. The parameters in these templates can be substituted for a certain parameter value in the system declarations. In this way, multiple instances of the same automaton with different variable values can be created.

The graphical syntax in Uppaal is directly inspired from the description of timed automata discussed in the previous section. This means, that an automaton is represented as a graph consisting of locations and edges extended with clock variables. Locations can be labelled with names and invariants and edges can be labelled with guard conditions, synchronizations and assignments.

The declaration of data variables, clocks and synchronization channels are done with use of a C-like language. These declarations can be declared to be global (accessible for each automaton) or local (accessible for the automaton where it is declared).

Simulator
The simulator can be used to validate the system constructed with the graphical editor. The simulator can be used in multiple ways: the transitions in the constructed system can be executed manually to test its functionality; a random mode can let the system run on its own; or a trace (saved or imported from the verifier) can be checked to see how certain states are reached or why certain states cannot be reached. The simulator supports only integer data types (no floating-point types like doubles) and continuous ranges over clocks (specified by constraints in guards and invariants).

Verifier
The main purpose of Uppaal is to verify the model with respect to certain properties. The verifier can be used to verify these properties with use of the query language of Uppaal. This query language uses a simplified version of Timed Computation Tree Logic (TCTL) [28]. The query language contains standard model checking queries and SMC queries. The standard model checking queries are not used in this thesis. For information about these queries, the reader is referred to [3]. The SMC queries make use of statistical model checking. The main idea of the SMC queries is to monitor several simulations of the system, and then to use results from statistics to decide whether the system satisfies the property with some degree of confidence. The SMC queries can be divided into the following categories:

- **Simulation**
  Simulation can be used to visualize the values of expressions (evaluating to integers, doubles or clocks) along a number of simulated runs. This query can give insight on the behaviour of the system. To use this query the number of simulations to be performed, the time bound on the

simulations and the expressions to be monitored and visualized need to be specified. The plotted data can be exported as either a picture or a text file.

- **Probability estimation**
  Probability estimation can be used to compute the probability of a specific property. The probability estimation algorithm computes the number of runs needed to produce an approximation interval $[p-\varepsilon, p+\varepsilon]$ with a confidence $1-\alpha$. The approximation interval is derived using Clopper-Pearson exact method [8]. This method uses the fact that the measurements are always binary, i.e. the property is satisfied or not, and thus the result follows a binomial distribution [10]. To use this query the time bound and the property to be tested need to be specified.
- **Hypothesis testing**
  Hypothesis testing can be used to verify if the probability of a specified property is greater than or equal to a certain threshold. Wald's sequential hypothesis testing is used to test the null-hypothesis against the alternative hypothesis [36]. It compares the probability of a specified property with a specified threshold value, and returns a boolean result that indicates whether the property is greater or smaller than the threshold. To use this query the time bound on the simulations, the property to be tested and the threshold value need to be specified.
- **Probability comparison**
  Probability comparison can be used to check whether the probability of a certain property is greater than the probability of another property. This is done with an extended Wald testing algorithm [9]. A boolean result is returned that indicates whether the specified inequality of both properties holds or not. To use this query the time bound on the simulations and the property to be tested need to be specified.
- **Expected value of min or max**
  Expected value can be used to evaluate the expected values of the minimum or maximum of an expression that evaluates to a clock, integer or double value. A confidence interval is returned using the fact that measurements follow the Student's t-distribution, i.e. the measurements approach a normal distribution when $N \to \infty$ [26]. To use this query the time bound of the simulations, the number of runs, whether the minimum or maximum need to be evaluated and the property to be tested need to be specified.

For the analysis of the control algorithm the queries simulation, probability estimation and expected value of min or max are used. For these queries two statistical parameters need to be set. The first parameter is α and defines the confidence interval. The second parameters is ε and defines the width of the approximation interval.

5.3 Control algorithm as (extended) timed automaton

This section discusses the implementation of the identified linear model from chapter 4 and the modelling of the control algorithm as a timed automaton. The identified linear model is in the following referred to as the blood glucose model. Also some additional automata were implemented to compute certain statistics regarding the control algorithm.

The entire system consists of the blood glucose model and the controller. The blood glucose model consists of one automaton, whereas the controller consists of five timed automata: one for the control algorithm and one automaton for each of the four patient statuses. The network of timed automata communicates with each other through several synchronization channels. These channels are declared as broadcast channels, since the SMC queries require the channels to be broadcast. In this case, synchronization is never blocking. The declared variables, arrays, clocks, channels and self-defined functions, mentioned in the following subsections, can be found in Appendix B.

5.3.1 Blood glucose model

The implementation of the identified linear model as a timed automaton can be found in figure 5.6. The variables G1, G2 and W are the parameters of the corresponding template and are used for initialization of the blood glucose model and the control algorithm. The automaton consists of three locations: Init, P1 and P2. The location Init is added to initialize the blood glucose model. The location P1 corresponds to the situation in which the system waits for a new blood glucose measurement. The location P2 corresponds to the situation in which a new blood glucose measurement comes available and the control

algorithm needs to determine whether an insulin, glucagon or no injection needs to be administered. The initial location is specified as an urgent location, whereas the other two locations are declared as normal locations. This means that in the initial location, time may not pass and that the outgoing edge needs to be taken as soon as possible. In the other two locations time is allowed to pass.

At the transition from Init to P1, the blood glucose model is initialized with use of the function `initP()`. This function computes the blood glucose concentration at t $= 0$ based on the values of G1 and G2, which represent the blood glucose concentrations at time t $= -1$ and t $= -2$. The computed blood glucose concentration is stored in the first entry of the array Ga. This array contains the last sixty blood glucose concentrations and is used to compute the size of the glucose rate of change. The weight of the patient is used to compute the initial settings of the control algorithm. The global variables G and S contain the last computed value of the blood glucose concentration and glucose rate of change, respectively, and are available for the other automata every time the creq! action is send. The invariant $t <= 1$, associated with the location P1, in combination with the guard $t >= 1$, associated with the transition from P1 to P2, represents the discrete time steps of the blood glucose model. At each time step the automaton makes the transition from P1 to P2 and synchronizes with the control algorithm through the channel creq. At this transition the new blood glucose concentration and slope are computed, with use of the functions `updateG()` and `calculateS()`. The function `updateUM()` is used to update the array um that contains the three main meals: breakfast, lunch and dinner. The moment these meals are ingested is determined with use of the variable it. In location P2 the automaton waits for the controller to send the cresp! action. At the transition from P2 to P1 the clock t is reset.

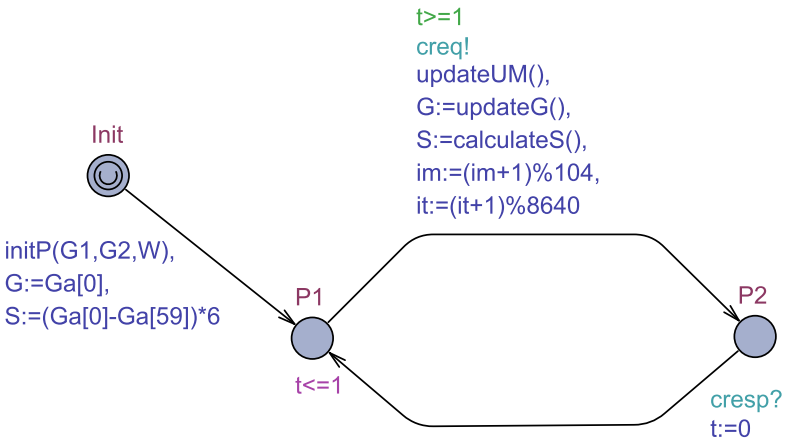


Figure 5.6: *Implementation of the blood glucose model as timed automaton, where G1, G2 and W are the parameters of the template. The automaton consists of three locations. The initial location is declared as urgent, whereas the other two locations are normal locations.*

5.3.2 Control algorithm

Control algorithm

The implementation of the control algorithm as timed automaton can be found in figure 5.7. The control algorithm consists of 15 locations of which the locations InsulinCurve, GlucagonCurve, PreInsulin, ExtraInsulin, PreGlucagon and ExtraGlucagon correspond to the different type of injections as displayed in figure 3.1. All locations are specified as committed, except for location Init. In this way, location Init is the only location in which time is allowed to pass. In the other locations, time is not allowed to pass and the next transition must involve an outgoing edge of one of these committed locations. This means that the controller determines the size of the injection to be administered in zero time.

The automaton starts in location Init. The moment the patient sends the creq! action, the controller synchronizes with the patient and makes the transition to C1. At the transition from C1 to C2 the new blood glucose concentration G and glucose rate of change S are used to update the entries of the arrays corresponding to each patient status and to compute the value of T_i as explained in section 3.2. These arrays represent the last twenty blood glucose concentrations and glucose rate of changes and are used to determine when a certain type of injection needs to be administered. At C2 the automaton can make one of the five possible transitions, depending on which patient status becomes active:

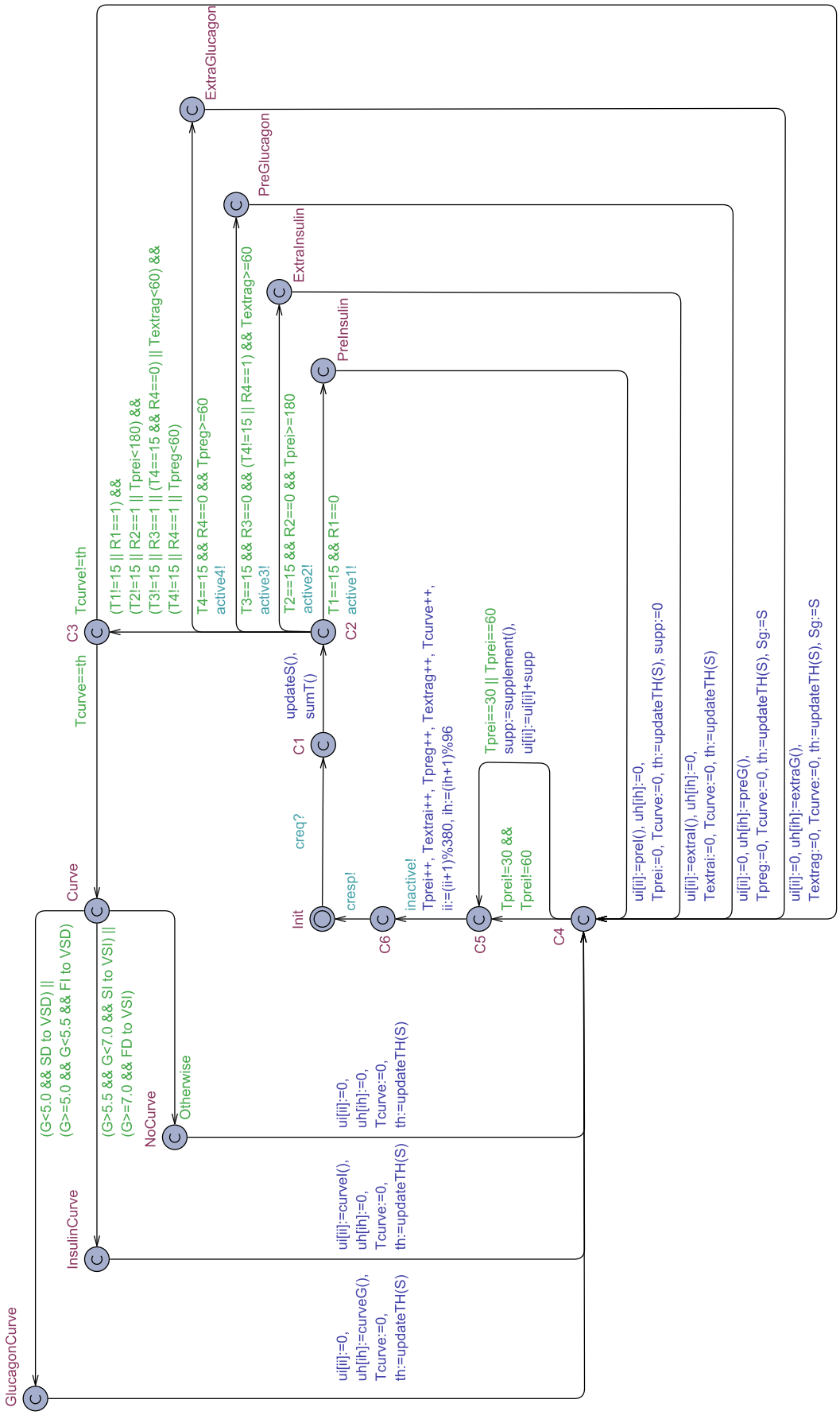


Figure 5.7: *Implementation of the control algorithm as timed automaton. The locations PreInsulin, ExtraInsulin, PreGlucagon, ExtraGlucagon, InsulinCurve and GlucagonCurve correspond to the different coloured blocks in figure 3.1.*

1. Transition from C2 to PreInsulin to C4
   This transition is taken when patient status 1 becomes active, i.e. T1 $== 15$ and R1 $== 0$. In this case, the size of the pre-injection insulin is determined with the function `preI()`, the timers Tprei and Tcurve are reset and the threshold of the curve injection timer is set based on the size of the glucose rate of change.

2. Transition from C2 to ExtraInsulin to C4
   This transition is taken when patient status 2 becomes active, i.e. T2 $== 15$, R2 $== 0$ and Tprei $>= 180$. In this case, the size of the extra injection insulin is determined with the function `extraI()`, the timers Textra and Tcurve are reset and the threshold of the curve injection timer is set based on the size of the glucose rate of change.

3. Transition from C2 to PreGlucagon to C4
   This transition is taken when patient status 3 becomes active and patient status 4 does not become active or is already active, i.e. T3 $== 15$, R3 $== 0$, T4! $= 15$ or R4 $== 1$, and Textrag $>= 60$. In this case, the size of the pre-injection glucagon is determined with the function `preG()`, the timers Tpreg and Tcurve are reset and the threshold of the curve injection timer is set based on the size of the glucose rate of change. In case both patient status 3 and 4 become active, the next transition is taken.

4. Transition from C2 to ExtraGlucagon to C4
   This transition is taken when patient status 4 becomes active, i.e. T4 $== 15$, R4 $== 0$ and Tpreg $>= 180$. In this case, the size of the extra injection glucagon is determined with the function `extraG()`, the timers Textra and Tcurve are reset and the threshold of the curve injection timer is set based on the size of the glucose rate of change.

5. Transition from C2 to C3
   This transition is taken when none of the patient statuses become active. In location C3 it is checked whether a curve injection needs to be administered or not:

   (a) Transition from C3 to Curve to GlucagonCurve/InsulinCurve/NoCurve to C4
   This transition is taken when the curve timer equals the earlier specified threshold, i.e. Tcurve $== th$. The blood glucose concentration and glucose rate of change determine whether a glucagon, insulin or no curve injection needs to be administered. In each case, the size of the corresponding injection is determined with the function `curveG()` or `curveI()`, the timer Tcurve is reset and the threshold of the curve injection timer is set based on the size of the glucose rate of change.

   (b) Transition from C3 to C4
   This transition is taken when the curve timer is not equal to the earlier specified threshold, i.e. Tcurve! $= th$. This means that no type of insulin or glucagon injection needs to be administered.

The transition from C4 to C5 can be performed in two ways, namely via the edge with guard Tprei! $= 30$ $\&\&$ Tprei! $= 60$ or via the edge with guard Tprei==30 $\|$ Tprei==60. Five and ten minutes after a successfull pre-injection insulin the second edge is taken to check whether the pre-injection insulin needs to be supplemented or not. This is done with the function `supplement()`, which checks whether the glucose rate of change falls into a higher category compared to the glucose rate of change five or ten minutes before. If the glucose rate of change falls into a higher category and the pre-injection needs to be supplemented, the function determines the size of this supplement. If this is not the case, the function returns zero. The first edge is taken at the other time units.

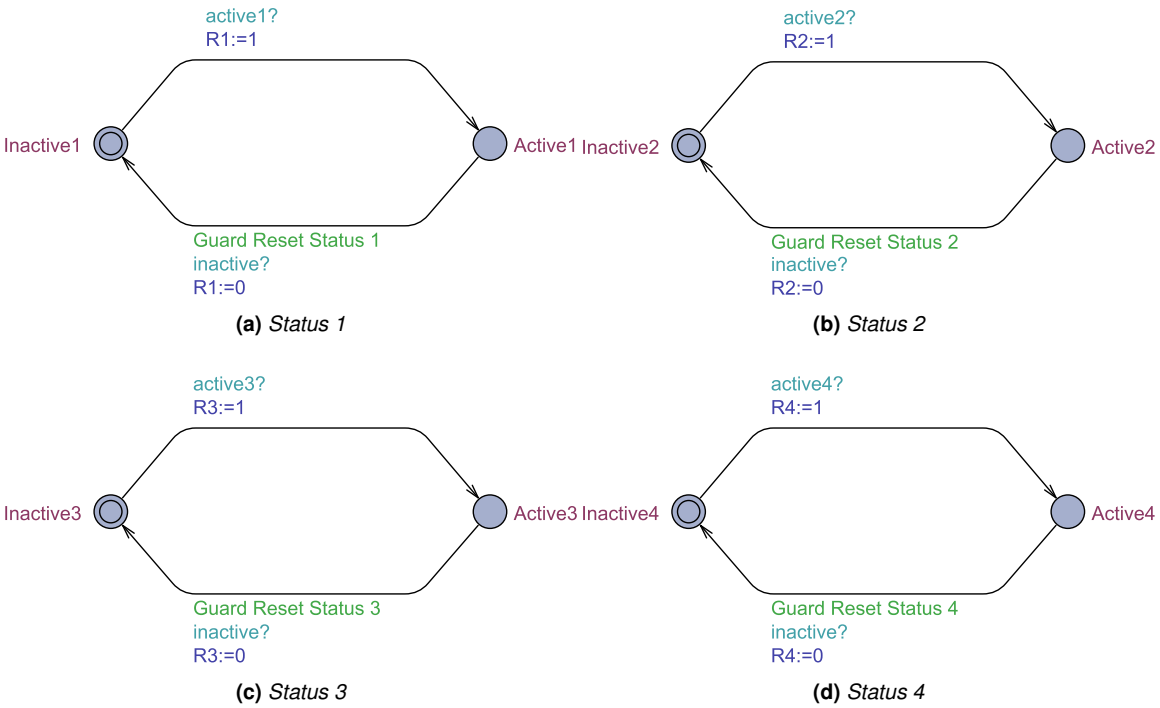
The transition from C5 to C6 is used to update the timers of the several injections and the pointers of the insulin and glucagon arrays. At this transition the control algorithm also synchronizes with the four patient statuses through the channel `inactive` to check whether a certain patient status needs to be reset. At the transition from C6 back to Init the control algorithm synchronizes with the blood glucose model by sending the cresp! action, causing the automaton of the blood glucose model to make the transition from P2 back to P1.

The timers Tcurve, Tprei, Textrai, Tpreg and Textrag are implemented as integers which are updated at the transition from location C5 to C6. The reason for this is that the guard corresponding to an edge that

synchronizes on broadcast channels may not contain clock constraints. Since these timers are used in the guards of the automata of the four patient statuses and the channels are declared as broadcast, these timers had to be implemented as integers.

Patient statuses

The implementation of the four patient statuses as timed automata can be found in figure 5.8. As can be seen in this figure, the four patient statuses are implemented in the same manner. Each automaton consists of two locations. Consider, for example, the automaton corresponding to status 1. The initial location of the automaton is Inactive1 and corresponds to the situation in which patient status 1 is inactive. The second location of the automaton is Active1 and corresponds to the situation in which patient status 1 is active, which means that the pre-injection insulin just has been administered. The transition from Inactive1 to Active1 takes place through the channel `active1`, which is send by the control algorithm the moment the size of the pre-injection insulin is determined. At this transition the value of the variable R1 is set to 1, indicating that the automaton is in the second location. As long as the automaton is in Active1, the controller is unable to administer another pre-injection insulin. The transition from Active1 to Inactive1 takes place through the channel `inactive`, which is send by the controller at the end of each cycle. This transition can only take place when the corresponding guard is enabled. In this case, the variable R1 is set back to 0, indicating that the automaton is in location Inactive1 and that a new pre-injection insulin can be determined. In case the automaton does not satisfy the property set by the guard, the channel is ignored and the transition does not take place. The same holds for the other three automata.



(a) *Status 1* **(b)** *Status 2*

(c) *Status 3* **(d)** *Status 4*

Figure 5.8: *Implementation of the four patient statuses as timed automata.*

5.3.3 Additional automata

As discussed in the problem statement we want to analyze the performance of the control algorithm. Some performance metrics of a control algorithm for the regulation of blood glucose concentration in T1DM patients are: the mean blood glucose concentration in the night and the total amount of time in hyper-, hypo- and euglycemia. However, the query language of Uppaal does not contain any function that can be used to compute these metrics directly. Therefore, some additional timed automata were constructed to compute these metrics.

Mean blood glucose concentration in the night

The mean blood glucose concentration in the night is computed with the two timed automata in figure 5.9, where T0 and T1 are integer variables instantiated in the system declaration.

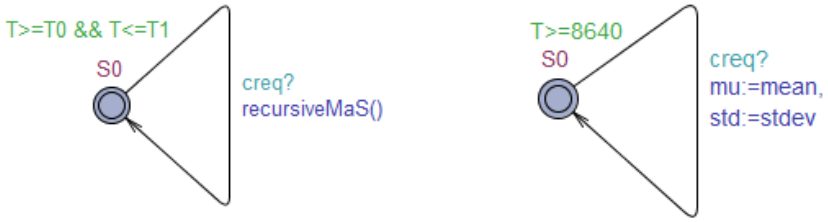


Figure 5.9: *The two timed automata that were used to compute and to return the mean blood glucose concentration and corresponding standard deviation. The left automaton computes the mean blood glucose concentration and corresponding standard deviation between T0 and T1. The right automaton writes the last computed values of the mean and standard deviation to the variables mu and std.*

The left automaton is used to compute the mean and the standard deviation of the blood glucose concentration. Since the night is defined as the time between 00:00 and 07:00, T0 and T1 were instantiated as 5760 and 8280, respectively. Every time a new blood glucose concentration is computed, the blood glucose model sends the creq! action. At that moment, the left automaton checks whether the total elapsed time T is between T0 and T1. If this is the case, the blood glucose concentration is used to compute the mean and corresponding standard deviation with use of the function `recursiveMaS()`. This function computes the mean and the variance in a recursive manner with use of the following two formulas:

$$\mu_0 = \frac{p-1}{p} \cdot \mu_{-1} + \frac{1}{p} \cdot G \tag{5.1}$$

$$\sigma_0^2 = \frac{p-1}{p} \cdot \sigma_{-1}^2 + \frac{1}{p-1} \cdot (G - \mu_0)^2 \tag{5.2}$$

where μ_0 is the current mean blood glucose concentration, μ_{-1} is the previous mean blood glucose concentration, σ_0^2 is the current variance of the blood glucose concentration, σ_{-1}^2 is the previous variance of the blood glucose concentration, G is the current blood glucose concentration and p is the total number of blood glucose concentrations used so far. The value of p is increased by one every time the automaton computes the mean and variance. The standard deviation is computed by taking the square root of the variance.

The right automaton is used to return the last computed values, because the query language of Uppaal does not contain any function that can return the last computed value of a certain variable. At the last time instant,T $= 8640$, the last computed values of the mean and standard deviation are stored into the variables mu and std (which are zero till that moment). The mean and standard deviation can then be returned by Uppaal with use the expected values queries with the number of runs equal to 1.

Total amount of time in hyper-, hypo- and euglycemia

The total amount of time in hyper-, hypo- and euglycemia are computed with the use of the three automata in figure 5.10. The left automaton is used for computing the total amount of time in hyperglycemia, whereas the middle and right automata are used for computing the total amount of time in hypoglycemia and euglycemia, respectively. Each automaton has its own local clocks x and y. This means that the clocks x and y in the left automaton are different from the clocks x an y in the middle and right automaton. The clock x is used to compute the total amount of time in a certain location, whereas the clock y is used to compute the maximum amount of time in a certain location.

One of the features of Uppaal SMC is that it is possible to change the rate of the clocks. This can be done with use of the invariant of the corresponding location. Consider, for example, the left automaton from figure 5.10. In the initial location M1, the invariant is set to $x' == 0$ and $y' == 0$, which means that both clocks are stopped. The moment the patient sends the creq! action and the blood glucose concentration is higher than 10 mmol/L, the automaton makes the transition from M1 to M2. In this location, the invariant is set to $x' == 1$ and $y' == 1$, which means that the clocks are started. The automaton stays in this location until the moment the patient sends the creq! action and the blood glucose concentration is lower than 10 mmol/L. The automaton then switches back to location M1. At the transition from M2 to M1 the value of clock y is reset. In this way it computes the maximum amount of time in location M2. Timed automata in which the clock is stopped and started are called *Stopwatch Timed Automata* [7].

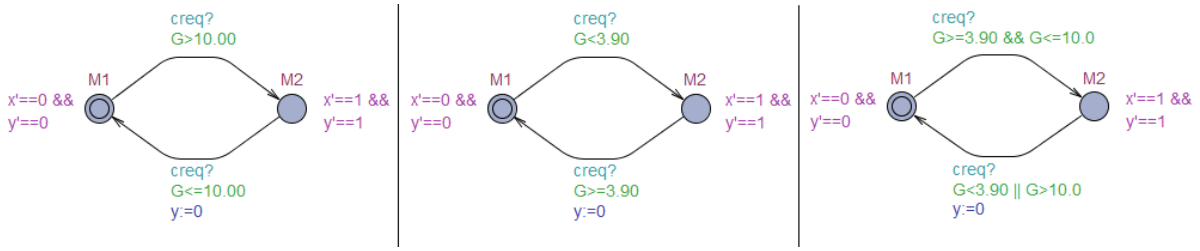


Figure 5.10: *The three automata used for computing the total amount of time in hyperglycemia (left), hypoglycemia (middle) and euglycemia (right). Clock x computes the total time in the second location and clock y computes the maximum time in the second location.*

5.4 Analysis of the control algorithm

This section discusses the validation of the blood glucose model and the analysis of the control algorithm. The blood glucose model is validated by simulating the day on which the blood glucose model was identified in Uppaal. Then the results of this simulation can be compared with the data available in MATLAB.

5.4.1 Validation blood glucose model

The blood glucose model was validated by reproducing the day in Uppaal on which the model was identified. The control algorithm in Uppaal was initialized with the same values for PC, PI, EI, PG and EG as used in the clinical study. The control algorithm was validated by comparing the following statistics:

- The mean blood glucose concentration during the day
- The maximum blood glucose concentration during the day
- The minimum blood glucose concentration during the day
- The total amount of insulin that was administered during the day
- The total amount of glucagon that was administered during the day

The results can be found in table 5.1. In this table it can be seen that there is a small difference between the blood glucose concentration values computed in MATLAB and in Uppaal. However, this is not surprising since Uppaal uses a simplified linear model to describe the blood glucose concentration based on the ingested carbohydrates, administered amount of insulin and the administered amount of glucagon only, whereas the blood glucose concentration in real-life is also dependent on other factors. In Uppaal a lower total amount of insulin was administered. This can be explained by the fact that the control algorithm used in real-life administered some anti-occlusion injections that prevented the cannulas to occlude. In Uppaal these injections were not administered. The same holds for the total amount of glucagon administered. A lower/higher amount of insulin/glucagon administered in Uppaal could also indicate an incorrect value of the coefficient corresponding to these inputs.

Table 5.1: *Comparison between some statistics computed in MATLAB and Uppaal to validate the blood glucose model.*

	MATLAB	**Uppaal**
Mean blood glucose concentration (mmol/L)	7.30	6.74
Maximum blood glucose concentration (mmol/L)	11.60	11.22
Minimum blood glucose concentration (mmol/L)	2.90	3.60
Total amount of insulin (Units)	41.75	36.40
Total amount of glucagon (Units)	96	107

5.4.2 Analysis of the control algorithm

Addition of some stochastics

In order to be able to use the Uppaal SMC queries to analyze the performance of the control algorithm, some stochastics need to be added to the entire system. This needs to be done because both the blood glucose model and the control algorithm are deterministic. This means that if we would simulate two similar days with the same amount of carbohydrates for the three main meals, the result of these simulations would be the same. Therefore, the size of the ingested meals will be drawn from a distribution with a certain mean and standard deviation. The clinical data of the ten patients, as discussed

in chapter 4, were used to determine the mean and the standard deviation of each meal. Table 5.2 shows the minimum, maximum, mean and standard deviation of each meal computed from the clinical data. In Uppaal, the samples can be drawn from an exponential or uniform distribution. The underlying

Table 5.2: *Minimum, maximum, mean and standard deviation of the three main meals computed with use of the clinical data*

Meal	Minimum (gram)	Maximum (gram)	Mean (gram)	St.dev. (gram)
Breakfast	15.0	65.0	37.4	15.6
Lunch	7.0	96.0	43.5	20.4
Dinner	15.0	110.0	43.5	21.9

distribution of the samples determines which distribution needs to be used in Uppaal. However, it is unknown what the underlying distribution of the samples is. This is not problematic because the Central Limit Theorem (CLT) namely states that the distribution of the average of a sufficiently large number of independent, identically distributed random variables will be approximately normal, regardless of the underlying distribution [26]. The term ”sufficiently large” is relative and depends on the underlying distribution. The rule of thumb is that a sample size of at least 30 will usually suffice, although for many distributions smaller sample sizes will do [26]. It can be shown that, in case of the uniform distribution, the difference between the normal approximation and the exact distribution is often already negligible for a sample size of four [19], [25]. However, in case of the exponential distribution, a sample size of forty is required for the difference to be negligible [19]. During analysis the uniform distribution was used, because the difference between the normal approximation and the exact distribution is negligible for a smaller sample size than in case of the exponential distribution. The mean and standard deviation of a uniform distribution with minimum value a and maximum value b are defined as:

$$\mu = \frac{1}{2}(a + b) \tag{5.3}$$

$$\sigma = \frac{1}{\sqrt{12}}(b - a) \tag{5.4}$$

Table 5.2 can be used in combination with equations (5.3) and (5.4) to estimate the values of a and b. The results can be found in table 5.3. In this table it can be seen that the lower bounds of the distributions are rather low. Therefore, the lower bounds were set to 15, because in the clinical data the minimum carbohydrate content for both breakfast and dinner was equal to 15 grams. In this case, the mean values become slightly higher and the standard deviations slightly lower.

Table 5.3: *The values of the uniform distributions estimated from table 5.2 and equations (5.3) and (5.4)*

Meal	a (gram)	b (gram)
Breakfast	10.4	64.4
Lunch	8.2	78.8
Dinner	5.6	81.4

Analysis of the control algorithm

As mentioned in the problem statement, once the control algorithm is initialized, it needs to satisfy certain requirements with respect to the standard insulin pump therapy used by T1DM patients nowadays. The control algorithm could replace this standard insulin pump therapy only when it is able to regulate the blood glucose concentration better than the standard insulin pump therapy does. The requirements the control algorithm needs to satisfy are as follows:

- The total time in euglycemia should be equal or higher than 68.5%
- The total time in hypoglycemia should be equal or lower than 2.4%
- The total time in hyperglycemia should be equal or lower than 24.3%
- The mean blood glucose in the night should be lower than 7.8 mmol/L

The values mentioned in the requirements specified above are the values for the standard insulin pump therapy computed from and used in [5].

The control algorithm is initialized based on the weight of the patient. The patient on which the linear model was identified weighed 73 kg. Once the controller is initialized, it should satisfy the requirements specified above. The requirements specified above were computed with the Uppaal SMC query expected value of the maximum, where the number of runs was set to 30 to ensure that the CLT holds. One whole day consisting of 24 hours was simulated. Breakfast, lunch and dinner were drawn from the uniform distributions mentioned in table 5.3, with a minimum of 15 grams of carbohydrates, and were given at 08:00, 12:00 and 18:00, respectively. The result of the analysis can be found in table 5.4. In this table it can be seen that the controller, initialized with the weight of the patient, satisfies the requirements specified above.

Table 5.4: *Computation of the requirements for the initial settings of the controller with use of the expected value query, where the number of runs was set to 30 with an 95% confidence interval.*

Requirement	**Result**
Percentage of time in euglycemia	$93.97 \pm 1.15\%$
Percentage of time in hypoglycemia	$0.78 \pm 0.34\%$
Percentage of time in hyperglycemia	$5.03 \pm 0.79\%$
Mean blood glucose concentration in the night	5.87 ± 0.08

In case the control algorithm is initialized with the weight of the patient, the value of PC is equal to 46%. The requirements were also computed for other values of PC to investigate whether there is a ”better” initial setting for this specific patient and what the influence is of a change in the value of PC on the requirements. The requirements were computed for the following values of PC: 10%, 15%, 20%, 25%, 30%, 35%, 40%, 46%, 50%, 55%, 60%, 70% and 80%. The results can be found in figure 5.11.

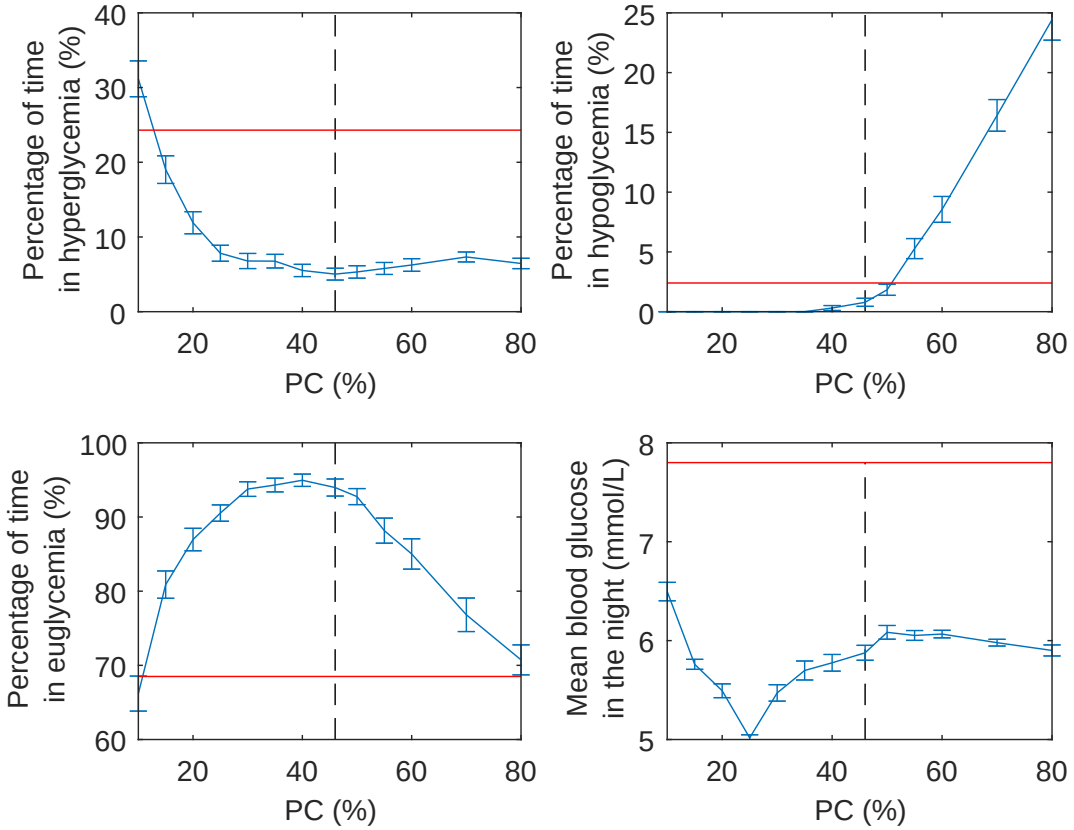


Figure 5.11: *Requirements computed for different values of percentage curve (PC). The errorbars indicate the 95% confidence interval, the vertical dotted lines indicate the value of PC based on the weight of the patient and the red lines indicate the requirements for the standard therapy. Top left: percentage of time in hyperglycemia. Top right: percentage of time in hypoglycemia. Bottom left: percentage of time in euglycemia. Bottom right: Mean blood glucose concentration in the night.*

From figure 5.11 the following results can be obtained:

- The percentage of time in hyperglycemia is less than 24.3% for a PC value of 13% or higher. When the value of PC increases, the percentage of time in hyperglycemia decreases and then even stagnates. The minimum percentage of time in hyperglycemia is achieved for a value of 46%.
- The percentage of time in hypoglycemia is less than 2.4% for a PC value of 51% or lower. The percentage of time in hypoglycemia increases when the value of PC increases. The percentage of time in hypoglycemia is zero for a PC value of 35% or lower.
- The percentage of time in euglycemia is higher than 68.5% for a PC value of 11% or higher. The percentage of time in euglycemia increases when the value of PC increases. However, after 40% the percentage of time in euglycemia decreases when the value of PC increases. The maximum percentage of time in euglycemia is achieved for a value of 40%.
- The mean blood glucose concentration in the night is less than 7.8 mmol/L for each value of PC. The mean blood glucose concentration in the night decreases for increasing PC values between 10 and 25%. Between 25% and 50%, the mean blood glucose concentration increases for increasing PC. After 50%, the mean blood glucose concentration decreases again for increasing PC.

These results can be explained as follows. When the value of PC increases, more insulin will be administered which will cause the blood glucose concentration to decrease faster. Hence, less time will be spent in hyperglycemia and more time will be spent in euglycemia. At a certain moment increasing the value of PC does not lead to a decrease of the percentage of time spent in hyperglycemia. This is because the effect of insulin on the blood glucose concentration has a certain delay and this delay does not decrease when more insulin will be administered. However, when more insulin will be administered the minimum blood glucose concentration will be lower and possibly enters the hypoglycemic range. At a certain moment too much insulin will be administered and the control algorithm will not be able to compensate this excess amount of insulin with the administration of glucagon. Hence, more time will be spent in hypoglycemia and less time will be spent in euglycemia. Based on these results it can be concluded that there are not indications to adjust the initial value of PC, because all four requirements are satisfied.

The effect of a change in PC value on other characteristics was also investigated. The characteristics of interest are: the total administered amount of insulin and glucagon, the maximum and minimum blood glucose concentration and the standard deviation of the blood glucose concentration in the night. The results can be found in figure 5.12.

In this figure it can be seen that in case the value of PC increases, the total administered amount of insulin and glucagon increases. The increase of the total administered amount of glucagon is caused by the increase of the total administered amount of insulin. In case more insulin is administered, the minimum blood glucose concentration will become lower and therefore more glucagon will be administered to compensate for this decrease. Another requirement obtained from [5] is that the total administered amount of glucagon per day should be lower than 90U. However, for a PC value of 55% or higher the total administered amount of glucagon will exceed this value. This indicates that too much insulin will be administered during the day. In addition, this amount of glucagon is unable to compensate the excess amount of insulin, because the minimum blood glucose concentration keeps decreasing for increasing PC. The maximum blood glucose concentration first decreases for increasing PC and after a certain value the maximum blood glucose concentration increases again. The decrease of the maximum blood glucose concentration can be explained by the fact that more insulin will be administered. The increase of the maximum blood glucose concentration is caused by the increase of the total administered amount of glucagon, because glucagon causes an increase of the blood glucose concentration. If, in addition, at the same moment of a glucagon injection a meal is ingested, the blood glucose concentration will increase even faster and insulin will be unable to compensate for this increase. An increase of PC value also causes an increase of the standard deviation of the blood glucose concentration in the night. This is due to the increase of the total administered amount of insulin and glucagon. An increase of PC value causes a yo-yo effect of the blood glucose concentration in the night.

Uppaal can also be used to compute other interesting information regarding the performance of the control algorithm, such as the probability that a certain type of injection will be administered, the probability that the patient experiences hyper- or hypoglycemia and the expected maximum time in hyper- or hypoglycemia. These values were computed for the case the control algorithm was initialized with the weight of the patient. The following statistical parameters were used: $\alpha = 0.05$, $\varepsilon = 0.025$ and $N = 30$. The result can be found in table 5.5.

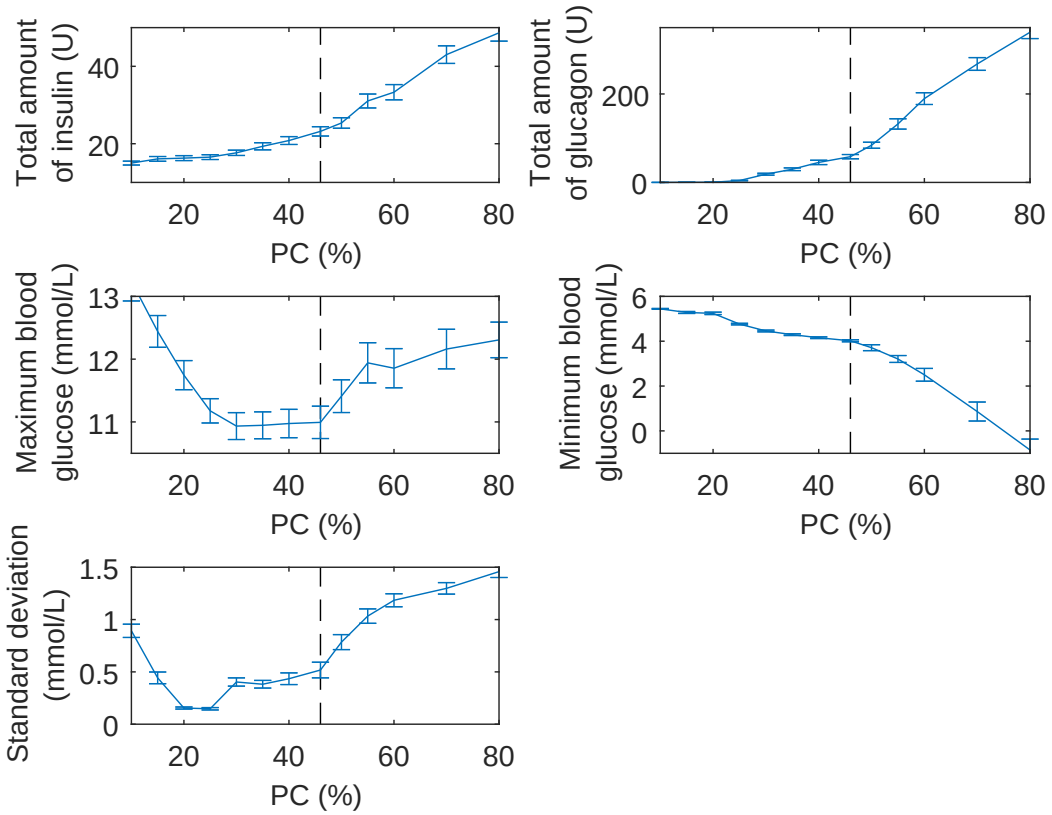


Figure 5.12: *Some characteristics of the patient computed for different values of percentage curve (PC). The errorbars indicate the 95% confidence interval and the vertical dotted lines indicate the value of PC based on the weight of the patient. Top left: total amount of insulin. Top right: total amount of glucagon. Middle left: maximum blood glucose concentration. Middle right: minimum blood glucose concentration. Bottom left: standard deviation of the blood glucose concentration in the night.*

Table 5.5: *Some interesting information regarding the performance of the control algorithm, where the control algorithm is initialized with the weight of the patient. The following statistical parameters were used:* $\alpha = 0.05$, $\varepsilon = 0.025$ *and* $N = 30$

Description	**Result**
The probability that a pre-injection insulin will be administered	$91.5\% - 96.5\%$
The probability that an extra injection insulin will be administered	$0.0\% - 5.0\%$
The probability that a pre-injection glucagon will be administered	$48.5\% - 53.5\%$
The probability that an extra injection glucagon will be administered	$95.0\% - 100.0\%$
The probability that a patient experiences hyperglycemia	$80.5\% - 85.5\%$
The probability that a patient experiences hypoglycemia	$18.3\% - 23.3\%$
The expected maximum time in hyperglycemia per 24 hours (in hours)	0.92 ± 0.11
The expected maximum time in hypoglycemia per 24 hours (in hours)	0.12 ± 0.06

Chapter 6

Discussion and recommendations

This chapter discusses some general points regarding the control algorithm, the system identification on clinical data and the modelling and analysis of the control algorithm in Uppaal. The chapter ends with some recommendations.

6.1 Discussion

The goal of this thesis was to model and analyze the control algorithm of interest in order to be able to investigate the performance of the control algorithm and the influence of a change in the initial setting on the performance of the control algorithm. To achieve this goal, the theory of timed automata was used together with the additional features provided by the tool Uppaal. The modelling and analysis discussed in Chapter 5 showed that the theory of timed automata together with the additional features of Uppaal was perfectly suitable to model and analyze the control algorithm, because all the elements of the control algorithm could be modelled. The only part of the control algorithm that was not modelled in Chapter 5, but that was discussed in Chapter 3, was the influence of physical activity on the size of the injections to be administered. This part could be included in future versions of the model, but in this thesis it was not the main part of the control algorithm.

The analysis in Uppaal showed that, once the control algorithm is initialized with the weight of the patient, it satisfies the four requirements specified in Chapter 5. Analysis also showed what the influence of a change in the initial setting is on the performance of the control algorithm. However, the accuracy of this analysis is mainly determined by the accuracy of the identified linear model. The accuracy of the identified linear model determines how well the blood glucose concentration can be simulated based on the ingested amount of carbohydrates, the administered amount of insulin and the administered amount of glucagon. If the identified model is not able to describe, for example, the hyper- or hypoglycemic range well, this will have influence on the computation of the percentage of time in hyper- and hypoglycemia. On the estimation data the identified linear model was able to describe the hyper- and hypoglycemic range.

The accuracy of the identified linear model is questionable, because it was identified on only one small part of clinical data of one specific day of one specific patient. In case the excitation of the input signals is not rich enough this could lead to identifiability problems or to an incorrect value of one of the coefficients of the linear model. However, these problems could not be investigated because we were able to estimate only one linear model which performed well on the estimation data. In addition, the analysis of the control algorithm did not show any inexplicable results. In future research, the accuracy of the identified linear model should be investigated.

The accuracy of the identified linear model is also questionable, because the results discussed in Chapter 4 showed that the identification of the linear models on the clinical data was problematic. The identification of the linear models was difficult mainly due to two reasons. Firstly, this was due to the high sampling frequency. From an information theoretic point of view it is advantageous to sample fast. Slower sampling leads to data sets that are subsets of the maximal one, and hence is less informative [27]. However, from a system identification point of view it can be disadvantageous to sample fast. Ljung [27] mentions two aspects that may prevent us from sampling as fast as technically possible: (1) identifying sampled models with very small sampling interval compared to the natural time constants is

a numerically sensitive procedure, i.e. all poles cluster around the point $z = 1$; and (2) the model fit may be concentrated to the high-frequency band, which may lead to curious results especially for the low frequencies [35]. The results showed that the poles of the identified linear model clustered around the point $z = 1$, which indicates that the sampling interval is small compared to the natural time constants of the system. The second reason why the identification of the linear models was difficult are the meal uncertainties. The meal uncertainties were mainly present in the composition of the meals, the estimated amount of carbohydrates in the various meals and the time of ingestion of these meals. In the study of [5] the patients were allowed to carry out their normal activities and there were no restrictions on meals. Thus, patients were allowed to eat anything they wanted. This led to a wide variation between the composition of the meals. The composition of a meal determines the influence it has on the blood glucose concentration. The carbohydrate content and the timing of the meals were recorded in a diary. However, the accuracy of the estimated carbohydrate content depends on the estimation skills of the patient. Some patients are used to estimate the carbohydrate content in their meals and some do not. As mentioned earlier, Brazeau et al. [6] showed that inaccurate carbohydrate counting is frequent and that fewer errors were made at breakfast, since there is generally less variation in its composition. In addition, the timing of some meals were inaccurate, because in some cases it happened that a patient forgot to write down, for example, the amount of carbohydrates and the moment of his/her lunch. In the evening the patient wrote down the amount of carbohydrates and the moment of ingestion. However, this could lead to inaccurate information because the patient may forget what was eaten and at which time. Due to the high sampling frequency there is a great difference between a lunch ingested at 12:00 or at 12:15, namely 90 time samples.

The high sampling frequency could be reduced with resampling. However, in this thesis, resampling was not possible because the sample time of the identified linear model should match the sample time of the control algorithm. The control algorithm needs blood glucose concentrations every 10 seconds to compute the size of the corresponding injection and therefore the sample time of the linear model needs to be 10 seconds.

Another reason why identification of the linear models was difficult could be the use of impulses for each of the three inputs. The meals were represented as one impulse for each meal, whereas each insulin and glucagon injection was represented as multiple impulses of which the length corresponded to the total time it took to administer these injections in practice. The use of impulses for each of the three inputs could cause identifiability problems. In addition, some meals were ingested at the same moment glucagon was administered. Because both meals and glucagon lead to an increase of the blood glucose concentration it might be difficult to determine which input caused the first change of the blood glucose concentration and/or which input had the biggest influence on this change. In [14] the glucose rate of appearance in the blood plasma was estimated for the meal data and considered as the input for a meal to avoid these identifiability problems. Identifiability problems could lead to incorrect estimated parameter values and a wrong description of the blood glucose concentration.

During identification only linear models were examined. The main reason for this was because these models are also used in literature to describe the blood glucose concentration based on some inputs. In addition, these linear models can be implemented easily in Uppaal. However, it is known that the dynamics of the blood glucose concentration are non-linear, especially in the hyper- and hypoglycemic range. In order to deal with these non-linearities additional inputs could be used. The linear models identified in this thesis had three inputs, namely the ingested amount of carbohydrates, the administered amount of insulin and the administered amount of glucagon. However, in practice, the blood glucose concentration also depends on other inputs. Examples are physical activity, stress and illness. In future research, the use of additional inputs could be examined.

It is also important to note that the clinical data obtained from [5] was not obtained for the purpose of identification of the linear models. The study was designed to asses the performance and safety of an artificial pancreas system and to compare this performance with current insulin pump therapy. During this master thesis we were looking for data that could be used for the identification of the linear models and this data was the best data available. The results discussed in Chapter 4 showed that identification of the linear models was possible, however, that the data contained the uncertainties discussed above.

Before the data was used for identification, some pre-treatment was performed. One of the things was the removal of offsets in the measured blood glucose concentration. This was done to estimate more accurate models, since linear models cannot capture arbitrary differences between the input and output. The reason for this is that linear models describe the relationship between the change in the input and the change in the output. In this thesis, the physiological equilibrium was removed from the blood glucose concentration. By removing this physiological equilibrium, the ”best” linear ARX model was estimated on the ”morning + afternoon” part of day 3, which resulted in a FIT of 75.49%. In that case, the physiological equilibrium was equal to 5.80 mmol/L. Instead of removing the physiological equilibrium, the mean blood glucose concentration could be removed. The mean blood glucose concentration was equal to 7.30 mmol/L. However, in that case, a model with a FIT of 63.93% would be obtained on the same data. In case the offsets would not be removed at all, a model with a FIT of -52.06% would be obtained. Thus, removal of the offsets resulted in a more accurate model.

For the analysis of the control algorithm some stochastiscs were included in the blood glucose model. This was achieved by using a uniform distribution with a certain minimum and maximum value for the amount of carbohydrates in the breakfast, lunch and dinner. Stochastics can also be implemented in different manners. The first manner could be to take the coefficients of the blood glucose model from a distribution with a certain mean and standard deviation. In order to compute this mean and standard deviation we have to identify more than one linear model. However, in this thesis it was very difficult to identify more linear models and therefore this mean and variance could not be computed. In addition, because the poles of the linear model were close to the unit circle, a small variation in the coefficients of the A-polynomial could lead to instability. Therefore, this method was not used. A second manner could be to include a model of the CGM sensors, i.e. a model of the measurement noise. However, this model is not available yet and therefore could not be implemented.

The query expected value of min or max was used to analyze the performance of the control algorithm. For this query the number of runs was set equal to $N = 30$. In this case, the Central Limit Theorem states that the mean of the samples of the uniform distribution approaches a normal distribution with the specified mean and variance. The sample mean and sample variance of the meals generated by Uppaal were compared to the desired mean and variance in order to verify the value of N. This comparison showed that the sample mean and sample variance were very close to the desired mean and variance, which indicates that the value of N seems to be correct.

The simulation query of Uppaal can be used to visualize the values of expressions along simulated runs. When a simulation is computed with use of the simulation query, Uppaal filters out the data and retains points that are distinguishable with respect to a certain resolution when plotted on a screen [10]. This filtered data can then be exported as either a picture or a text file. However, even for the minimum resolution Uppaal filtered out some data that was of interest. For example, when we tried to plot the administered amount of insulin and glucagon for one day, it happened that most of the impulses were filtered out. It seemed that no insulin or glucagon was administered, even though this was the case. Therefore, this query could not be used to verify the identified blood glucose model with the data available in MATLAB.

During analysis, we were only able to analyze the performance of the control algorithm on one patient. In the future, we are interested in the analysis of the control algorithm on more than one patient, so that we can consider more realistic scenarios. In this case we can verify whether the control algorithm can deal with the interpatient variability.

6.2 Recommendations

The first recommendation relates to the identification of the linear models discussed in Chapter 4. The linear models were identified on the clinical data of only one patient, because there were a lot of uncertainties in the meal information. Therefore, it is recommended to investigate whether it is possible to minimize these uncertainties by revising the meal information reported in the diaries. If it is possible to reduce these meal uncertainties, it could be possible that more accurate linear models can be obtained. In addition, it could be possible that more than just one linear model can be obtained. In that case, the ability of the control algorithm to deal with interpatient variability could be investigated.

The second recommendation relates to the acquisition of the data. Perhaps, in the future a clinical trial could be designed with the purpose of obtaining data specifically intended for the identification of the linear models. An idea is to invite a certain amount of patients to a clinical research centre for one day and to provide meals with the same composition at certain specified time instants (for example at 08:00, 12:00 and 17:00). In this way there is less variation in the composition of the meals and the moment of the meals are more certain.

The third recommendation relates to the models to be identified. In this thesis, simple linear models were identified based on three inputs. However, it is known that the dynamics of the blood glucose concentration are non-linear. Therefore, it is recommended to investigate whether is it possible to estimate models that are somewhat more advanced, but still easy to implement in Uppaal. It could also be investigated whether it is possible to use additional inputs such as the glycemic index or the heart rate, because these inputs also have substantial influence on the blood glucose concentration.

The fourth recommendation also relates to the models to be identified. Instead of using additional inputs, it could also be investigated whether it is possible to modify the shape of the inputs. In this thesis, the inputs were modelled as impulses at certain time points. For each meal, only one impulse at a certain time point was available. However, the influence of a meal on the blood glucose concentration is much longer visible. So, perhaps the meals could also be implemented as multiple consecutive impulses to imitate this influence. Another implementation could be the method used in [14]. In that study, an equation for the glucose rate of appearance in the blood plasma was used modify the shape of the meal input.

The fifth recommendation relates to the control algorithm. The control algorithm deals with most of the challenges mentioned in Chapter 2, such as interpatient variability, physical activity and meals. In order to deal with the interpatient variability, the control algorithm is individualized based on the weight of the patient. However, in the future it may be needed to take more factors into account, because the interpatient variability also depends on other factors such as gender, age and duration of diabetes. In order to deal with the intrapatient variability, the control algorithm needs to be extended with a self-learning algorithm that is able to modify the settings of the control algorithm based on some metrics. This self-learning algorithm can then be verified in Uppaal with use of the timed automaton of the control algorithm, as discussed in this thesis, and some additional timed automata that compute the specified metrics.

The last recommendation is for the developers of Uppaal. It is recommended to improve the tool Uppaal. Uppaal was perfectly suitable for the modelling and the analysis of the control algorithm. However, the tool can be made more user friendly. For example, the simulation query could be improved. Due to the filtering of the tool, in some cases information was filtered out and was not shown in the plot. Therefore, it would be recommended to include some feature that can be used to export the data obtained in Uppaal to MATLAB before it is filtered. In this way, MATLAB could be used for plotting the results. This is advantageous, because in Uppaal it is also not possible to zoom in or zoom out or to read the value of selected points from a graph.

Chapter 7

Summary and conclusions

In this thesis, a control algorithm for the regulation of the blood glucose concentration in patients with type 1 diabetes mellitus was presented. The control algorithm can be characterized as a fuzzy logic controller that used fuzzy rules to imitate the reasoning of T1DM patients. The control algorithm is initialized with the weight of the patient. The performance of this control algorithm, once it is initialized, and the influence of a change in the initial setting on the performance of the control algorithm can only be tested with use of clinical trials. However, these clinical trials take up a lot of time. Therefore, the goal of this thesis was to model and analyze the control algorithm with use of a modelling language to be specified.

Before the control algorithm was modelled and analyzed, a linear ARX model was identified on available clinical data. The purpose of this linear model was to simulate the blood glucose concentration based on the ingested amount of carbohydrates, administered amount of insulin and the administered amount of glucagon. Results showed that it was very difficult to identify an accurate linear model that can be used to describe both the estimation and validation data. The main reason for this was the high sampling frequency, which leads to a small variation between consecutive data points. Therefore, only the simulation method could be used, because the prediction method does not provide a good simulation model. Another reason why identification was difficult was the uncertainties in the ingested meals. There were uncertainties in both the size of the meals and the moment of the meals. By partitioning each day intro three parts we were able to estimate a linear ARX model that could be used to simulate the blood glucose concentration in Uppaal.

The control algorithm was modelled with use of the theory of timed automata and the extensions provided by the tool Uppaal. The reason for this was because the functioning of the control algorithm depends on some timing constraints which can be modelled perfectly with the theory of timed automata. In addition, this theory is perfectly suitable for the analysis of the control algorithm in Uppaal. From the analysis it can be concluded that, in case the control algorithm is initialized with the weight of the patient, it satisfies the following four requirements:

- The total time in euglycemia should be equal or higher than 68.5%
- The total time in hypoglycemia should be equal or lower than 2.4%
- The total time in hyperglycemia should be equal or lower than 24.3%
- The mean blood glucose concentration in the night should be lower than 7.8 mmol/L

In other words, the control algorithm is initialized correctly. From the analysis it can also be concluded that in case the value of PC is increased, the percentage of time in hyperglycemia decreases, the percentage of time in hypoglycemia increases, the percentage of time in euglycemia first increases and then decreases and that the mean blood glucose concentration in the night first decreases and then increases.

References

[1] R. Alur. Timed automata. In *International Conference on Computer Aided Verification*, pages 8–22. Springer, 1999.

[2] E. Atlas, R. Nimri, S. Miller, E.A. Grunberg, and M. Phillip. MD-logic artificial pancreas system. *Diabetes care*, 33(5):1072–1076, 2010.

[3] G. Behrmann, A. David, and K. Larsen. A tutorial on UPPAAL. *Formal methods for the design of real-time systems*, pages 33–35, 2004.

[4] R.E.M. Bekhuis. In silico modeling of the GI tract in Type 1 Diabetes Mellitus patients to simulate the glucose regulation for different meals. *University of Twente, the Netherlands*, 2016.

[5] H. Blauw, A.C. van Bon, R. Koops, and J.H. DeVries. Performance and safety of an integrated bihormonal artificial pancreas for fully automated glucose control at home. *Diabetes, Obesity and Metabolism*, 18(7):671 – 677, 2016.

[6] A.S. Brazeau, H. Mircescu, K. Desjardins, C. Leroux, I. Strychar, J.M. Ekoé, and R. Rabasa-Lhoret. Carbohydrate counting accuracy and blood glucose variability in adults with type 1 diabetes. *Diabetes research and clinical practice*, 99(1):19–23, 2013.

[7] F. Cassez and K. Larsen. The impressive power of stopwatches. In *International Conference on Concurrency Theory*, pages 138–152. Springer, 2000.

[8] C.J. Clopper and E.S. Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, pages 404–413, 1934.

[9] A. David, K. Larsen, A. Legay, M. Mikučionis, D. Poulsen, J. Van Vliet, and Z. Wang. Statistical model checking for networks of priced timed automata. *Formal Modeling and Analysis of Timed Systems*, pages 80–96, 2011.

[10] A. David, K.G. Larsen, A. Legay, M. Mikučionis, and D.B. Poulsen. Uppaal SMC tutorial. *International Journal on Software Tools for Technology Transfer*, 17(4):397–415, 2015.

[11] Diabetes Fonds. `www.diabetesfonds.nl/over-diabetes/diabetes-in-het-algemeen/diabetes-in-cijfers`. Accessed: 2017-09-06.

[12] R.S. Dinsmoor. Insulin Sensitivity Factor. `www.diabetesselfmanagement.com/diabetes-resources/definitions/insulin-sensitivity-factor/`, 2013. Accessed: 2017-09-16.

[13] F.H. El-Khatib, S.J. Russell, D.M. Nathan, R.G. Sutherlin, and E.R. Damiano. A bihormonal closed-loop artificial pancreas for type 1 diabetes. *Science Translational Medicine*, 2(27):27ra27–27ra27, 2010.

[14] D.A. Finan, H. Zisser, L. Jovanovic, W.C. Bevier, and D.E. Seborg. Identification of linear dynamic models for type 1 diabetes: a simulation study. *IFAC Proceedings Volumes*, 39(2):503–508, 2006.

[15] D.A. Finan, C.C. Palerm, F.J. Doyle, H. Zisser, L. Jovanovic, W.C. Bevier, and D.E. Seborg. Identification of empirical dynamic models from type 1 diabetes subject data. In *American Control Conference, 2008*, pages 2099–2104. IEEE, 2008.

[16] D.A. Finan, C.C. Palerm, F.J. Doyle, D.E. Seborg, H. Zisser, W.C. Bevier, and L. Jovanovic. Effect of input excitation on the quality of empirical dynamic models for type 1 diabetes. *American Institute of Chemical Engineers journal*, 55(5):1135–1146, 2009.

[17] P. Frasca. Lectures notes in modeling & control of hybrid dynamical systems. University of Twente, the Netherlands, April 2015.

[18] R. Gondhalekar, E. Dassau, and F.J. Doyle III. Tackling problem nonlinearities & delays via asymmetric, state-dependent objective costs in mpc of an artificial pancreas. *IFAC-PapersOnLine*, 48 (23):154–159, 2015.

[19] T. Greytak. Lecture 4: Sums of random variables. In *Statistical Physics I—MIT Course No. 8.044*. Cambridge MA, 2013. URL `ocw.mit.edu/courses/physics/8-044-statistical-physics-i-spring-2013/readings-notes-slides/MIT8_044S13_ProbabilityCh4.pdf`. MIT OpenCourseWare.

[20] A. Haidar. The artificial pancreas: How closed-loop control is revolutionizing diabetes. *IEEE Control Systems*, 36(5):28 – 47, 2016.

[21] J.E. Hall and A.C. Guyton. *Textbook of Medical Physiology (11th edition)*. Elsevier Saunders, 2006.

[22] R. Hanas. *Type 1 diabetes in children, adolescents, and young adults: how to become an expert on your own diabetes*. Class Publishing Ltd, 2007.

[23] Healthy Eating. How soon after ingestion of food does blood sugar rise? `healthyeating.sfgate.com/soon-after-ingestion-food-blood-sugar-rise-1399.html`. Accessed: 2017-08-08.

[24] Humalog (Insulin Lispro). `www.drugs.com/humalog.html`. Accessed: 2017-08-08.

[25] F. Killmann and E. von Collani. A note on the convolution of the uniform and related distributions and their use in quality control. *Economic Quality Control*, 16(1):17–41, 2001.

[26] R.J. Larsen and M.L. Marx. *An introduction to mathematical statistics and its applications*, volume 2. Prentice-Hall Englewood Cliffs, NJ, 1986.

[27] L. Ljung. *System Identification: Theory for the User (2nd edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1999. ISBN 0-13-656695-2.

[28] G. Logothetis and K. Schneider. Symbolic model checking of real-time systems. In *Temporal Representation and Reasoning, 2001. TIME 2001. Proceedings. Eighth International Symposium on*, pages 214–223. IEEE, 2001.

[29] MATLAB System Identification Toolbox, R2016B. The MathWorks, Natick, MA, USA.

[30] R. Mauseth, Irl B. Hirsch, J. Bollyky, R. Kircher, D. Matheson, S. Sanda, and C. Greenbaum. Use of a fuzzy logic controller in a closed-loop artificial pancreas. *Diabetes technology & therapeutics*, 15(8):628–633, 2013.

[31] R.J. McCrimmon and R.S. Sherwin. Hypoglycemia in type 1 diabetes. *Diabetes*, 59(10):2333–2339, 2010.

[32] N. Middelhuis. Internship report - Estimation of the parameters in the glucagon dynamics with use of available data. *University of Twente, the Netherlands*, 2016.

[33] Residual Analysis. `nl.mathworks.com/help/ident/residual-analysis.html`. Accessed: 2017-08-04.

[34] G.M. Steil, C.C. Palerm, N. Kurtz, G. Voskanyan, A. Roy, S. Paz, and F.R. Kandeel. The effect of insulin feedback on closed loop glucose control. *The Journal of Clinical Endocrinology & Metabolism*, 96(5):1402–1408, 2011.

[35] B. Wahlberg and L. Ljung. Design variables for bias distribution in transfer function estimation. *IEEE Transactions on Automatic Control*, 31(2):134–144, 1986.

[36] A. Wald. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16(2): 117–186, 1945.

[37] S.A. Weinzimer, G.M. Steil, K.L. Swan, J. Dziura, N. Kurtz, and W.V. Tamborlane. Fully automated closed-loop insulin delivery versus semiautomated hybrid control in pediatric patients with type 1 diabetes using an artificial pancreas. *Diabetes care*, 31(5):934–939, 2008.

[38] World Health Organization (WHO). www.who.int/diabetes/en/. Accessed: 2017-07-26.

[39] J.E. Youssef, J.R. Castle, D.L. Branigan, R.G. Massoud, M.E. Breen, P.G. Jacobs, B.W. Bequette, and W.K. Ward. A controlled study of the effectiveness of an adaptive closed-loop algorithm to minimize corticosteroid-induced stress hyperglycemia in type 1 diabetes, 2011.

Appendices

Appendix A

Delay determination methods

A.1 Description

In section 4.4 two methods were explained to estimate the delay(s) corresponding to the input(s) of a certain model. However, there are two other methods that can be used to estimate the delay(s) of a certain model. These methods are the standard methods offered by the MATLAB System Identification Toolbox.

The first method is to use the MATLAB function `delayest`. To use this function, the model orders, the smallest and largest delays to be tested need to be specified. This function then estimates the delay that each input most likely has by evaluating ARX models with the specified range of delays and chooses the delay of the model that minimizes the loss function defined by (4.21).

The second method is to use the MATLAB function `impulseest`. This function returns a non-parametric estimate of the impulse response. The model order is determined automatically using persistence of excitation analysis on the input data. This impulse response can then be plotted with a confidence interval represented by three standard deviations. The time delay is then the time when the impulse response is first outside the confidence region.

A.2 Results

A.2.1 First method

In general, the two methods discussed above can be used to estimate the delays of the corresponding model. However, the following results show that, in this thesis, these two methods were unable to estimate the delays of the linear models discussed in chapter 4.

The first method was applied on the data of day 2, 3 and 4 of patient PIN413. This function was used for an ARX model with model orders `na=2`, `nb=[1 1 1]` and different values of `nkmin` and `nkmax`. The results can be found in table A.1.

Table A.1: *Some of the results of the MATLAB function `delayest` for different values of `nkmin` and `nkmax`, where `na=2` and `nb=[1 1 1]`*

nkmin	nkmax	delayest
[0 0 0]	[30 30 30]	[0 28 0]
[50 50 50]	[70 70 70]	[50 50 70]
[50 50 50]	[100 100 100]	[50 100 92]
[50 50 50]	[100 120 100]	[50 120 100]
[0 100 90]	[100 200 150]	[0 198 108]
[0 100 90]	[100 300 150]	[0 100 90]
[0 190 100]	[100 300 150]	[0 273 129]
[0 270 120]	[100 400 160]	[0 372 142]
[0 370 130]	[100 500 160]	[0 489 142]
[0 0 0]	[100 500 190]	[0 0 0]

From table A.1 it can be concluded that this MATLAB function is unable to estimate the delays of the three inputs. In case of the first input, the delay is always estimated as the lower bound specified by `nkmin`. In case of the second input, the function is able to estimate some value for the delay that is unequal to the lower or upper bound specified by `nkmin` and `nkmax`. However, this value varies enormously. In addition, some values are questionable. A delay of 489 would mean a delay of 1 hour and 22 minutes. In case of the third input, the function estimates the delay as the lower bound, upper bound or some value in between. The same results were obtained when the data of only one particular day was used instead of all days or when the data of a different patient was used.

The frequency response corresponding to the used data could be used to explain why, for example, the function always estimates the delay of the first input to be equal to the lower bound of the specified delay range. In the Laplace domain, the "time delay property" given by:

$$f(t-a)\cdot u(t-a) \overset{\mathcal{L}}{\longleftrightarrow} e^{-as}F(s)$$

states that the Laplace Transform (LT) of a time delayed function is the product of the LT of the function $f(\cdot)$ and e^{-as}, where a is the time delay and $s = j\omega$. In the frequency domain the magnitude and the phase are the two quantities of importance. The magnitude of the time delay in the frequency domain is:

$$\begin{aligned}|e^{-j\omega a}| &= |\cos(\omega a) - j\sin(\omega a)| = \sqrt{(\cos(\omega a))^2 + (-\sin(\omega a))^2} \\ &= \sqrt{\cos^2(\omega a) + \sin^2(\omega a)} = 1\end{aligned}$$

This shows that the magnitude of a time delay is unity and is independent of frequency. This makes sense because a time delay does not change the magnitude of the signal, only the timing of the signal. The phase associated with a time delay is:

$$\begin{aligned}\angle e^{-j\omega a} &= \arctan\left(\frac{-\sin(\omega a)}{\cos(\omega a)}\right) = -\arctan\left(\frac{\sin(\omega a)}{\cos(\omega a)}\right) = -\arctan\big(\tan(\omega a)\big) \\ &= -\omega a\end{aligned}$$

This shows that the phase shift associated with a time delay decreases linearly with a slope of $-\omega a$, where a is the time delay.

The frequency response of the used data can be estimated with use of the MATLAB function `spa` and is shown in figure A.1. The upper row shows the magnitude of the response of the three inputs to the output and the lower row shows the phase shift of the three inputs to the output. In this figure it can be seen that the phase diagrams only show a phase roll off in case of the second input. The other two inputs do not have such a roll off. Thus, this figure suggests that there is a time-delay for input two only and not for input one and three.

A.2.2 Second method

The second method was also used on the data of day 2, 3 and 4 of patient PIN413. Figure A.2 shows the result of the function `impulseest`. In this figure it can be seen that the estimated impulse response of the first input is (almost) identical to zero. The impulse response is always inside the confidence region, just like the impulse response of the third input. This would mean that both the meals and glucagon have a delay of 0 time units. The estimated impulse response of the second input leaves the uncertainty region after 80 samples. This would mean that insulin has a delay of 80 time units, i.e. 13 minutes and 20 seconds. This seems to be a realistic value since insulin starts to work about 15 minutes after injection [24]. However, since the estimated delays of the first and third input are not realistic, the result cannot be used for further identification.

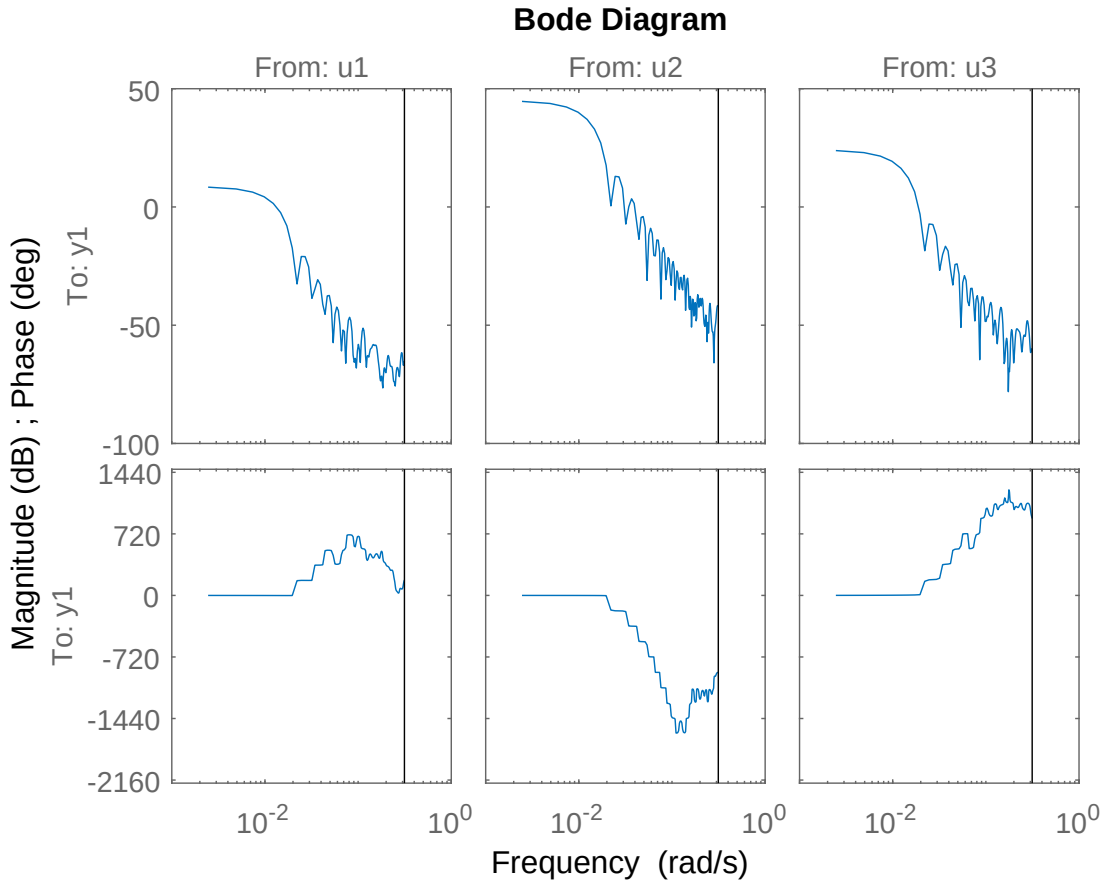


Figure A.1: *The estimated frequency response of the data of patient PIN413. Top row: magnitude of the system response as a function of frequency. Bottom row: phase shift of the system response as a function of frequency. The columns (from left to right) correspond to the first, second and third input, respectively.*

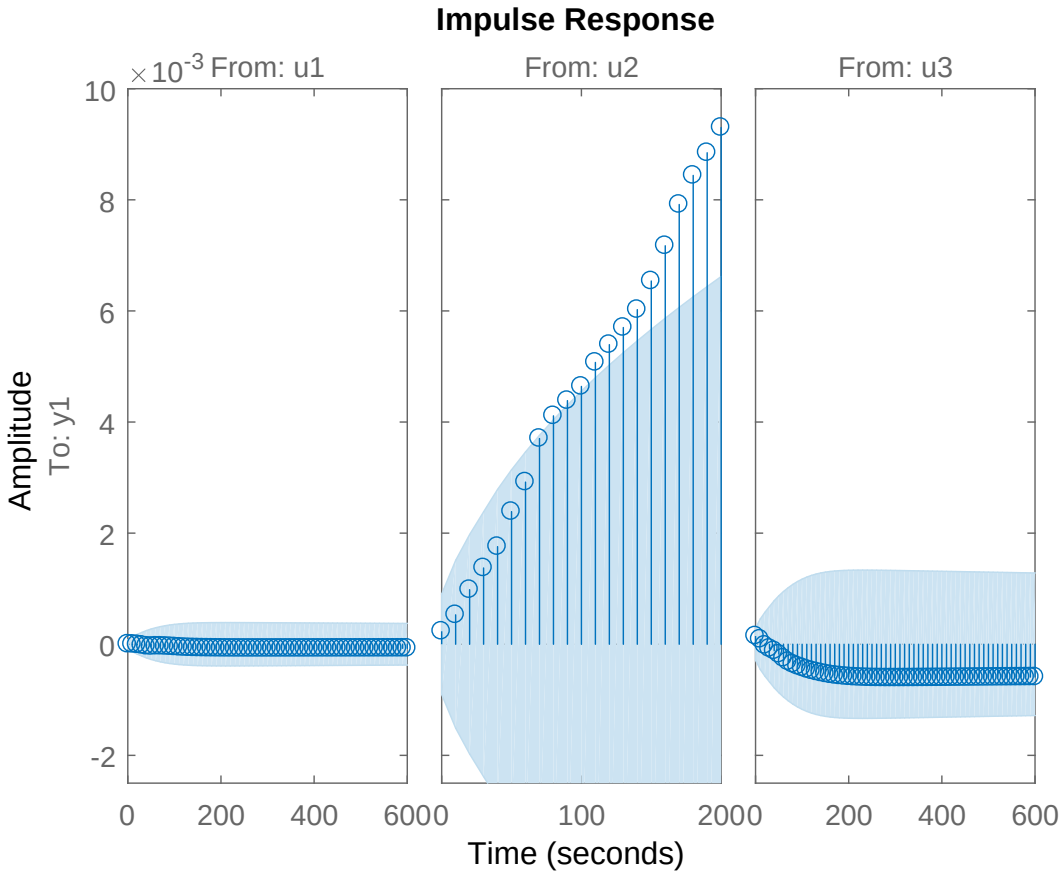


Figure A.2: *Result of the function* `impulseest`*. The function returns a non-parametric estimate of the impulse response together with a confidence interval represented by three standard deviations (filled light-blue region). The columns (from left ro right) correspond to the first, second and third input, respectively.*

Appendix B

Uppaal

This appendix contains the declared variables, clocks, channels, and self-defined functions as mentioned in chapter 5.

B.1 Declarations

B.1.1 Variables

Table B.1: *Variables used in Uppaal for the implementation of the control algorithm as timed automaton*

Description	Variable	Type	Initial Value	Global/ Local
Current blood glucose concentration	G	double	5.50	Global
Current slope	S	double	0.00	Global
Slope at the moment of a glucagon injection	Sg	double	0.00	Global
Percentage curve	PC	double	0.00	Global
Size of the pre-injection insulin	VI	double	0.00	Global
Size of the extra injection insulin	EI	double	0.00	Global
Size of the pre-injection glucagon	VG	double	0.00	Global
Size of the extra injection glucagon	EG	double	0.00	Global
Mean blood glucose concentration	mean	double	0.00	Global
Variance of blood glucose concentration	variance	double	0.00	Global
Standard deviation of blood glucose concentration	std	double	0.00	Global
Pointer to determine the moment of the three main meals	it	int	0	Global
Pointer of the meal array um	im	int	0	Global
Pointer of the insulin array ui	ii	int	0	Global
Pointer of the glucagon array uh	ih	int	0	Global
Variable used for patient status 1	R1	int	0	Global
Variable used for patient status 2	R2	int	0	Global
Variable used for patient status 3	R3	int	0	Global
Variable used for patient status 4	R4	int	0	Global
Time from the last curve injection	Tcurve	int	0	Global
Time from the last pre-injection insulin	Tprei	int	180	Global
Time from the last extra injection insulin	Textrai	int	0	Global
Time from the last pre-injection glucagon	Tpreg	int	270	Global
Time from the last extra injection glucagon	Textrag	int	270	Global
Ingested amount of carbohydrates	meal	double	0.00	Local
Size of the administered pre-injection insulin	pre_size	double	0.00	Local
Size of the supplement of the pre-injection insulin	supp	double	0.00	Local
Number of values used for calculation of mean	p	double	1.00	Local
Last computed mean	mu	double	0.00	Local
Last computed standard deviation	std	double	0.00	Local
Pointer of the arrays S1, S2, S3 and S4	p	int	0	Local
Counter of patient status 1	T1	int	0	Local
Counter of patient status 2	T2	int	0	Local
Counter of patient status 3	T3	int	0	Local
Counter of patient status 4	T4	int	0	Local

B.1.2 Arrays

Table B.2: *Arrays used in Uppaal for the implementation of the control algorithm as timed automaton*

Description	Variable	Type	Size	Global/ Local
Array containing the ingested amounts of carbohydrates	um	double	104	Global
Array containing the administered amount of insulin	ui	double	380	Global
Array containing the administered amount of glucagon	uh	double	96	Global
Array used for calculating the value of counter T1	S1	int	20	Local
Array used for calculating the value of counter T2	S2	int	20	Local
Array used for calculating the value of counter T3	S3	int	20	Local
Array used for calculating the value of counter T4	S4	int	20	Local

B.1.3 Clocks

Table B.3: *Clocks used in Uppaal for the implementation of the control algorithm as timed automaton*

Description	Variable	Global/Local
Total time elapsed	T	Global
Time used for defining discrete time points	t	Local
Total amount of time in hyperglycemia	x	Local
Maximum amount of time in hyperglycemia	y	Local
Total amount of time in hypoglycemia	x	Local
Maximum amount of time in hypoglycemia	y	Local
Total amount of time in euglycemia	x	Local

B.1.4 Channels

Table B.4: *Channels used in Uppaal for the implementation of the control algorithm as timed automaton*

Description	Variable	Type
Channel used for communication between patient and controller	creq	Broadcast
Channel used for communication between controller and patient	cresp	Broadcast
Channel used for inactivation of the four patient statuses	inactive	Broadcast
Channel used for activation of patient status 1	active1	Broadcast
Channel used for activation of patient status 2	active2	Broadcast
Channel used for activation of patient status 3	active3	Broadcast
Channel used for activation of patient status 4	active4	Broadcast

B.1.5 User-defined functions

This section describes the user-defined functions mentioned in chapter 5. This section also contains the source code of the functions as used in Uppaal.

initP(G1,G2,W)
This function is used to initialize the blood glucose model and the settings of the control algorithm. The parameters used for initialization are the blood glucose concentration on time $t = -2$ (*G2*) and $t = -1$ (*G1*) and the weight of the patient (*W*). The parameters *G1* and *G2* are used to initialize the array *Ga*. This array contains the last sixty blood glucose concentrations and is used to determine the new blood glucose concentration and the size of the glucose rate of change. Initialization of this array is needed to avoid large fluctuations of the glucose rate of change at the beginning of a simulation day. The last 58 entries of this array are set to the value of *G2*, whereas the second entry is set to the value of *G1*. The first entry is set to the blood glucose concentration on time $t = 0$, which is computed based on the values of *G1* and *G2*. The weight of the patient is used to initialize the settings of the control algorithm. If the weight of the patient is higher than 55kg, the value of *PC* is set according to the formula mentioned in Chapter 3. If the weight of the patient is less than or equal to 55kg, the value of PC is set to 20. The values of *PI*, *EI*, *PG* and *EG* are set based on the value of *PC* (see Chapter 3 for the formulas).

updateUM()
This function is used to update array *um*. This array contains the ingested amounts of carbohydrates in the three main meals: breakfast, lunch and dinner. The pointer *it* determines the moment one of these meals is ingested, whereas the pointer *im* determines the entry of the array *um* that needs to be updated. Breakfast, lunch and dinner are ingested at time points 0 (i.e. 08:00), 1440 (i.e. 12:00) and 3600 (i.e. 18:00), respectively. At these time points, the size of the meal is drawn from a uniform distribution with a specified minimum and maximum value. For breakfast, lunch and dinner the minimum values are set to 15, whereas the maximum values are set to 64.4, 78.8 and 81.4, respectively. On the other time points the value is set to 0, i.e. no meal is ingested.

updateG()
This function is used to compute the new blood glucose concentration with use of the identified linear model. This function first moves each entry of the array *Ga* one entry to the right. Then the first entry is set to the new blood glucose concentration. This blood glucose concentration is computed from the previous two blood glucose concentrations, the ingested amount of carbohydrates 104 time steps before,

the administered amount of insulin 380 time steps before and the administered amount of glucagon 96 time steps before.

calculateS()
This function is used to compute the size of the glucose rate of change. The glucose rate of change is defined as the difference between the blood glucose concentration at this moment and the blood glucose concentration ten minutes before. The unit of this value is mmol/L/10 sec. However, the unit of the glucose rate of change needs to be mmol/L/h and therefore this value is multiplied with 6.

updateS()
This function is used to update the arrays *S1*, *S2*, *S3* and *S4*. These arrays are used to determine which patient status needs to become active. The entries of the corresponding arrays represent the last twenty blood glucose concentrations and glucose rate of changes. If at a certain moment, the blood glucose concentration and glucose rate of change meet one of the conditions mentioned in table 3.5, the entry of the corresponding array is set to 1. Else, the entry of the corresponding array is set to 0. The pointer *p* is used to determine which entry of the four arrays need to be updated. This function implements the idea discussed in Section 3.2.

sumT()
This function is used to compute the value of the counters *T1*, *T2*, *T3* and *T4*. The value of these counters are equal to the sum of the entries of the corresponding arrays *S1*, *S2*, *S3* or *S4*.

updateTH(S)
This function is used to compute the new threshold of the curve injection timer. The value of this threshold depends on the size of the glucose rate of change.

curvel()
This function is used to compute the size of the curve injection insulin. The size of the curve injection insulin is determined with respect to a third-degree polynomial. The amount of insulin is cut-off at a certain value.

curveG()
This function is used to compute the size of the curve injection glucagon. The size of the curve injection glucagon is determined with respect to a third-degree polynomial. The amount of insulin is cut-off at a certain value.

prel()
This function is used to compute the size of the pre-injection insulin. The size of the pre-injection insulin depends on the size of the glucose rate of change and whether in the last 45 minutes a pre- or extra injection glucagon has been administered. The higher the glucose rate of change, the higher the pre-injection insulin. In case within the last 45 minutes a pre- or extra injection glucagon has been administered, the size of the pre-injection insulin is reduced according to a certain formula.

extral()
This function is used to compute the size of the extra injection insulin.

preG()
This function is used to compute the size of the pre-injection glucagon. The size of the pre-injection glucagon depends on the size of the glucose rate of change. The higher (in absolute value) the glucose rate of change, the higher the pre-injection glucagon.

extraG()
This function is used to compute the size of the extra injection glucagon.

supplement()
This function is used to compute the size of the supplement of the pre-injection insulin. Five and ten minutes after a successful pre-injection insulin, it is checked whether the pre-injection needs to be supplemented. If, at these moments, the slope falls into a higher category, the pre-injection insulin is

supplemented with the difference between the two pre-injections. This can only be the case when an insulin pre-injection of size $0.75PI$ or $0.85PI$ is administered. In case of $1.00PI$, the pre-injection cannot be supplemented.

recursiveMaS()
This function is used to compute the (recursive) mean blood glucose concentration and the corresponding standard deviation of the blood glucose concentration. These values are computed with equations 5.1 and 5.2.