DESIGN AND BASIC VERIFICATION OF A DISCRETE EVENT SIMULATOR FOR GLUCOSE METABOLISM IN HUMAN BEINGS

by

Elizabeth Andrews

A Thesis Submitted in
Partial Fulfillment of the
Requirements for the Degree of

Master of Science
in Computer Science

at

The University of Wisconsin-Milwaukee

December 2016

ABSTRACT

DESIGN AND BASIC VERIFIATION OF A DISCRETE EVENT SIMULATOR FOR GLUCOSE METABOLISM IN HUMAN BEINGS

by

Elizabeth Andrews

The University of Wisconsin-Milwaukee, 2016

Under the Supervision of Professor Mukul Goyal

This thesis describes the design and basic verification of a discrete event simulator for glucose metabolism in human beings. The simulator implements the glucose metabolism related behavior of various organs in the human body and tracks the blood plasma glucose level as the human body goes through a sequence of diet and exercise events. The simulator can mimic insulin resistance in various organs as well as the loss of insulin production in the pancreas and the adverse impact of these changes on the metabolic behavior of various organs. Thus, the simulator can serve as a model for people with diabetes. Such a model can be immensely useful to study the impact of specific life style changes on a person with diabetes. This thesis describes the simulator design as well as the results of simulations that verify

the basic correctness of the simulator. This simulator represents the result of a multi-year collaborative effort involving the author, her MS thesis advisor and several other students working with the thesis advisor.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Diabetes refers to a set of metabolic disorders characterized by persistently high blood plasma glucose levels. Recent decades have seen an explosive growth in the number of people afflicted by diabetes. Sedentary life style and bad food habits are the main reasons behind this growth. A person with Type 2 Diabetes (the most common diabetes variant) can significantly improve the daily life quality as well as prospects of avoiding long-term diabetes complications (limb amputations, kidney/heart failure, blindness etc.) by following a healthy life style that keeps the blood plasma glucose level under control. However, modifying life-long habits is hard and hence a number of Internet and smart-phone based tools have been designed that help diabetic people keep their blood plasma glucose level under control. The blood plasma **G**lucose level **S**imulator described in this thesis, henceforth called the **GS** simulator, can be used to design the next generation of these tools. While the current generation tools essentially allow the user to record the diet/exercise activities and plasma glucose levels measured using glucose meters and view the accumulated information in a variety of ways, the next generation tools will be able to create user-specific models of plasma glucose level variation using the accumulated information and use these models to suggest life-style changes and even real-time guidance regarding diet and exercise activities. The GS simulator can simulate the variation in plasma glucose level as the human body goes through a sequence of diet and exercise activities. Thus, the simulator can be used to determine the life style changes that will allow the user to achieve good control over the plasma glucose level. The GS simulator can also be used to implement a diabetes self-management tool that provides real-time guidance regarding diet and exercise to a diabetic user.

PREVIOUS WORK

Energy metabolism in human beings is a well-researched field. While there exists more than a hundred years of research in understanding various metabolic processes taking place in the human body, new insights are being gained even today. Basic tenets of energy metabolism in humans are well understood and a number of textbooks describe this material. Our primary source for understanding energy metabolism in human beings has been Keith Frayn's "Metabolic Regulation: A Human Perspective," third edition [1]. The glucose metabolism in human beings, as implemented in the current version of GS simulator, is largely based on the Frayn's text. In addition, the simulator's implementation of the behavior of various organs (including the values of various parameters affecting this behavior) has been heavily influenced by a number of well-cited papers [1] [2] [3] [4] [5] [6] [7] [8] [9].

Glucose plays an essential role in human body's energy metabolism. Due to its relevance in maintaining the proper functioning of the human body, over the years, several studies have been conducted to understand blood glucose dynamics. Mathematical modelling has been a popular focus in this area of research and has been used to study several different aspects of glucose metabolism such as glucose-insulin sensitivity, prediction of glucose concentrations in plasma, the effect of exercise on glucose metabolism and many others. One of the most popular models developed to aid understanding of glucose metabolism - particularly the glucose-insulin system – is Bergman's minimal model [10]. This model is based on differential equations and consists of two parts – glucose kinetics and insulin kinetics. Over the years, this model has been examined and revised many times to incorporate additional physiological effects of glucose and insulin.

Additionally, the impact of mild-to-moderate exercise has also been studied and incorporated into the Bergman's minimal model by different researchers [11] [12].

Surprisingly, it appears that discrete event simulation of energy metabolism in human beings is not yet a well-explored area. Our literature survey revealed only two simulators that are somewhat similar to our efforts. One of them is DiMSim simulator [13], where metabolic pathways are viewed as bipartite graphs consisting of metabolites and reactions linked by unidirectional or bidirectional arcs, while the other is Discrete Metabolic Simulation System (DMSS) [14]. In both cases, the discrete event simulation framework was built based on the biochemical interactions among various organs in the human body.

CHAPTER 2

SIMULATOR DESIGN OVERVIEW

The core of the GS simulator consists of a **HumanBody** object and a **Simulator Controller** (**SimCtl**) object. The HumanBody object reads the files describing various diet and exercise activities and values for parameters affecting the behavior of various organs. This information is then stored inside appropriate data structures inside the HumanBody object. The SimCtl object reads a timed sequence of diet/exercise events the body will go through. These events are stored in the order in which they will be fired in an event queue maintained in the SimCtl object.

The GS simulator has been designed as a single-threaded application. Since the actual functioning of the human body resembles a multi-threaded application, early work did consider a multi-threaded approach where classes representing each individual organ ran on its own distinct thread. However, further research established that a 'tick' based sequential design or polled design - where predefined functions (representing different metabolic processes in different organs) execute at each tick - could mimic the required metabolic behavior of the human body without any loss of functionality. In this design, the current time is stored in a member variable called **tick** inside the SimCtl object. At the beginning of each tick, the SimCtl object fires all the diet/exercise events whose firing time has arrived and lets the HumanBody object know. The SimCtl object then calls the **processTick()** function on the HumanBody object, which in turn allows all organs to do the work they are supposed to do during this tick. After this, the SimCtl

object increments the tick value (i.e. advances time by one unit) and repeats the whole process again. A high level simplified class diagram of the simulator is given in Figure 1.

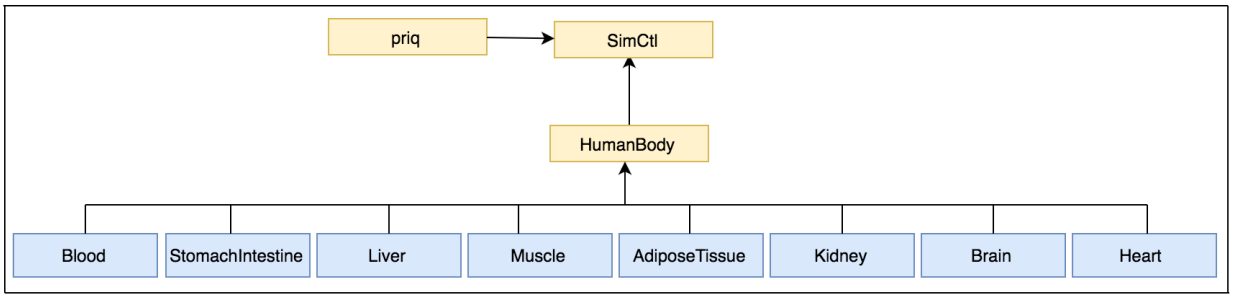

Figure 1: Top level class diagram of simulator

The simulator consists of primarily ten classes that represent the human body and its internal organs. The class HumanBody is designed to model the metabolic behavior of the actual human body. In order to do so, we have modeled the HumanBody class as an envelope class that contains member objects representing various organs. The HumanBody object not only maintains global information relevant to each organ, but also regulates the interdependencies among various organs.

Each organ class in turn implements the behavior of individual organ of the human body (except **StomachIntestine** which represents the functionality of the digestive system consisting of the stomach, the small intestine and the portal vein). These classes model the metabolic pathways related to glucose metabolism in their corresponding organ, communicating with each other using a pointer to the HumanBody object. Additionally, the bloodstream (or the blood plasma) is represented by the **Blood** class. This class is used primarily to maintain substrates related to glucose metabolism in the blood.

In order to represent different metabolic states of a human being, this study defines 6 states - **FED_RESTING, FED_EXERCISING, FED_POSTEXERCISE, POSTABSORPTIVE_RESTING, POSTABSORPTIVE_EXERCISING, POSTABSORPTIVE_POSTEXERCISE.** Various organs have different values for the parameters governing their behavior for different metabolic states. The next section provides a detailed description of the top-level classes defined in this simulator.

DETAILED DESIGN DESCRIPTION

THE MAIN () FUNCTION

The main( ) function serves as the starting point of the simulator by creating new instances of the SimCtl and HumanBody classes and invoking methods defined in these classes to process all the configuration and event description files used by the simulator. The algorithm for main( ) is given in Figure 2.

1. Declare an instance of the class SimCtl.
2. Declare an instance of the class HumanBody.
3. Invoke HumanBody methods to process Food, Exercise and Parameter description files.
4. Invoke a SimCtl method to process the Events file.
5. Invoke a SimCtl method to start the simulation.

Figure 2: Algorithm for main( ) function

The Food and Exercise files are used to store information about the different food types and different exercises. Each line in these files represent an individual food type or exercise type respectively. The Parameter file contains all the configurable fields related to each organ. These parameters are dependent on the metabolic state of the body. Similar to the Food and Exercise files, each line of the Parameter file (i.e. the configuration file) corresponds to a single parameter (corresponding to a particular metabolic state and organ). Each line of these files should adhere to their respective format detailed in Figure 3.

```
Food file         -      id name servingSize RAG SAG protein fat
Exercise file     -      id name intensity postExerciseDuration
Parameter file    -      BodyState BodyOrgan ParameterName ParameterValue
```

Figure 3: File Format - Food, Exercise and Parameter

Please note that food intake - which includes the serving size along with the quantity of Rapidly Available Glucose, Slowly Available Glucose, protein and fat in this serving - is specified in grams. Exercise intensity is measured in MET (abbreviated from Metabolic Equivalent of a Task), where 1 MET is equal to 1 kcal per kg per hour and is nominally defined as the normal resting metabolic rate (i.e. whole body energy expenditure) [1].

The Event file is used to store the sequence of events we want to simulate using this simulator. Each line in this file represents an individual event and should adhere to the format detailed in Figure 4.

```
Event file    -    day hour minutes type subtype howmuch
```

Figure 4: File Format - Event

Here, **day**, **hour** and **minutes** refer to the number of days, (residual) hours and (residual) days the simulator has been running. It is used to calculate the tick count at which the event should be fired (i.e. the time at which this event occurs). The three remaining fields depend on the event itself. At the time of writing this document, the simulator has the capability to handle 3 types of events – Food, Exercise and Halt. **Type** indicates the type of event being fired, **subtype** indicates the particular ID of the food eaten or exercise done, and **howmuch** is used to specify the amount of food eaten or the duration of the exercise event. The Event file is processed by loading all the events specified in the file into a priority queue. (The class **priq** mentioned in Figure 1 is an implementation of a priority queue itself. Specific details regarding this implementation are beyond the scope of this document.)

The following classes have been defined to aid event handling.

1. **Event**

   The **Event** class is a base class to **FoodEvent**, **ExerciseEvent** and **HaltEvent** classes, which represent the different types of events that can be given as an input to the simulator. This class defines variables to store the time at which an event occurred and the type of the event that occurred.

2. **FoodEvent**

   The FoodEvent class inherits the Event class and contains additional information to handle a food event - such as variables to store the quantity of food intake and the ID of food that has been eaten.

3. **ExerciseEvent**

   The ExerciseEvent class inherits the Event class and contains additional information to handle an exercise event – such as variables to store the duration of the exercise in minutes and the ID of the exercise done.

4. **HaltEvent**

   The HaltEvent class inherits from the Event class and contains no additional information. It is used to halt the simulation.

Once all the pre-processing and initializations have been completed, the simulator starts the simulation of glucose metabolism in the human body. It enters a repetitive pattern using the

concept of ‘ticks’, where predefined tasks are performed during each tick. This functionality is implemented in the SimCtl method **run_simulation( )**.

SIMCTL

The class SimCtl is responsible for initializing and maintaining the ‘tick’ counter - the unit time for this simulator. In our design, each tick is defined as a single minute. Although changing the definition of ‘tick’ will not affect the overall architecture of the GS simulator, it is important to note that doing so will require updates to the value of several configuration parameters. Furthermore, if each ‘tick’ is defined to be a much larger time frame (than a minute), the order of execution of functions - representing metabolic processes in various organs- in each tick may become relevant. At the moment, when the ‘tick’ definition is one minute, the order in which various organs perform their tasks is not relevant.

The **run_simulation( )** method of the SimCtl class uses an infinite loop, where each iteration specifies the tasks done during a single tick. During each tick, the simulator first determines if the event queue contains any events that must be fired at this time. For each such event, the simulator calls the corresponding event processing method defined in the HumanBody class. Finally, the method **processTick( )** is called on the HumanBody object and the tick counter is updated. The processTick( ) and event processing methods (mentioned above) defined in the HumanBody class are discussed next. It is important to note that processTick( ) is a method present in every organ class, and the HumanBody class itself. This method is responsible for executing all the tasks defined for each tick for that particular organ.

HUMANBODY

The class HumanBody is designed to model the metabolic behavior of the actual human body. In order to mimic the biological structure and metabolic pathways of the human body, the GS simulator models the HumanBody class as an envelope class containing objects representing individual organs of the human body. This class also maintains variables - such as **bodyState** and **bodyWeight_** - that store information relevant to all the organ classes.

The primary responsibilities of the HumanBody class includes alerting different organs that a new tick has started, facilitating the interdependencies between various organs, and maintaining the metabolic state (and corresponding parameters) of the simulator. Additionally, this class also has several methods that process Food and Exercise events. The configuration parameters of the HumanBody class are as follows:

Parameter	**Default Value**	**Comments**
insulinResistance_	0	This parameter will have a value between 0 and 1. It controls the reaction of individual organs to the presence of insulin in the bloodstream.
insulinPeakLevel_	1	This parameter will have a value between 0 and 1. It controls the amount of insulin the pancreas produces (in comparison to the insulin produced in a healthy person.
bodyWeight_	65	Weight of the simulated body in kilograms

Table 1: Configuration parameters - HumanBody

As mentioned earlier in this documentation, processTick() is the method responsible for executing all the tasks defined for each tick for that particular organ. In the HumanBody class, this method has two essential tasks. The first task is to notify each organ class about the occurrence of a new tick by calling their respective processTick( ) methods, while the second task is to maintain the metabolic state of the body after the execution of exercise related events. The latter is done by tracking the current metabolic state of the GS simulator. If the simulator is in one of the exercise states i.e. FED_EXERCISING or POSTABSORPTIVE_EXERCISING, the state is changed to the corresponding post-exercise state, FED_POSTEXERCISE or POSTABSORPTIVE_POSTEXERCISE if the duration of the exercise event has completed. Similarly, the state is changed from post-exercise to the corresponding resting state based on the value of postExerciseDuration, which you may recall is a value specified in the Exercise file. This value is used in the method **processExerciseEvent( )** – method used to process exercise events – to calculate the tick count at which this change of state must occur.

The algorithm for processExerciseEvent( ) is given in Figure 5.

1. If the body is already in an Exercise state, error out and exit the simulator.
2. Calculate the current energy expenditure based on body weight and intensity of exercise.
3. Change the current metabolic state to its corresponding Exercise state.
4. Calculate the tick at which Exercise event is completed.
5. Calculate the amount of time, the human body will remain in a post-exercise state.

Figure 5: Algorithm for processExerciseEvent( ) function

The method processExerciseEvent( ) is called by the class SimCtl when an Exercise event is fired by the GS simulator. This method first checks if the simulator is already in an Exercise state, in which case it causes the simulator to exit without any further operation. This was done in order to force the user of this simulator to change the 'Event' file because realistically, a human being does not perform an exercise within another exercise. Once the event has been validated, the additional energy expenditure caused by this exercise is calculated based on the individual's body weight and the intensity of exercise as (body weight * exercise intensity) / 60, where exercise intensity is measured in MET. Since the body weight is specified in kilograms and intensity is measured in MET, where 1 MET is 1kcal / (kg * hour), the energy expenditure calculated using this formula is specified in kcal/minute. This calculated amount is then stored in a variable **currEnergyExpenditure** to be used as required by the organ classes.

A post-exercise state was defined in this study to accommodate for the fact that in actuality, the human body remains at an elevated metabolic state for some period after exercising. In this study, the duration for this period is modeled as a fraction of the duration of the exercise itself. The tick count at which the simulator has to switch from the exercise state to a post-exercise state is calculated by simply adding the duration of this exercise event (specified using subtype in Event file) to the current value of simulator tick count. This count is tracked by the method processTick( )to initiate the change of state when required.

In order to process food events, i.e. a person eating, the HumanBody class contains a method **processFoodEvent()**. This method has two responsibilities – the first one is model eating by adding the 'nutrients' to their corresponding variables in StomachIntestine class and the second

one is to switch the metabolic state to a fed state. The former responsibility is carried out by calling an **addFood( )** method defined in the StomachIntestine class.

The HumanBody class also defines a method **stomachEmpty()** to handle the change of state from fed to post-absorptive. The fed state is characterized by the presence of nutrients in the stomach. In this simulator, this presence is tracked by the StomachIntestine class in its processTick( )method. If the stomach is found to be empty, the simulator transitions to the post-absorptive state by executing the method stomachEmpty() (called by the StomachIntestine class). This method the changes the metabolic state of the GS simulator to either POSTABSORPTIVE_RESTING, POSTABSORPTIVE_EXERCISING or POSTABSORPTIVE_POSTEXERCISE based on the current state.

It is important to note that a change in the metabolic state of the GS simulator necessitates the updating of configuration parameters associated with that change. For example, the rate at which glycogen breakdown occurs in the muscle varies between the rest and exercise state. Therefore, the HumanBody class contains the method **setParams()**, which is called every time a state change occurs in the simulator. Similar to the method processTick(), this method is present in every organ class and the HumanBody class itself. It is used to set the new parameter values - associated with the new state - for each organ. In this HumanBody class this method is primarily used to call the setParams()methods in all organ classes. Additionally, it also sets the value for the HumanBody configuration parameters.

LIVER

The **Liver** class models the metabolic pathways of the liver body organ. These pathways primarily include glycogenolysis (glycogen breakdown), gluconeogenesis (glucose generation from lactate, glycerol and amino acids), glycolysis (glucose breakdown to form lactate and fatty acids), and glycogen synthesis. In reality, all these metabolic processes are multi-step processes with several intermediate products. For example, glucose breakdown to glycogen involves the formation and subsequent breakdown of glucose-6-phosphate (G6P) and pyruvate. A part of this G6P is then used to synthesize glycogen while a part of the pyruvate is used to synthesize lactate. In order to simplify the simulator, this study does not model intermediate products, i.e. for the given example - in this simulator - G6P and pyruvate are not modeled; only glucose, glycogen and lactate are included. Additionally, although in reality these processes occur in a concurrent manner, in the GS simulator, these metabolic processes are modeled sequentially in the **processTick( )** method of the Liver class. For its functioning, the processTick( )method interacts with the **PortalVein** class (defined as part of the digestive system in StomachIntestine) and the **Blood** class. It retrieves required data from these classes and then performs a sequence of tasks which simulate the metabolic processes.

The configuration parameters of the Liver class are as follows:

Parameter	**Default Value**	**Comments**
glucoseToGlycogen_	glycogenToGlucose_	Rate at which glucose is converted to glycogen in mg/kg/min
glycogenToGlucose_	2*0.9007795	Rate at which glycogen is converted to glucose in mg/kg/min

glycolysisMin_	0.297	The minimum rate at which glycolysis occurs in the liver in mg/kg/min
glycolysisMax_	2.972	The maximum rate at which glycolysis occurs in the liver in mg/kg/min
glycolysisToLactateFraction_	1	By default, all lactate generated during glycolysis is released into the bloodstream
gluconeogenesisRate_	1.8*0.45038975	Rate at which glucose is generated from gluconeogenesis substrates in mg/kg/min
gngFromLactateRate_	gluconeogenesisRate_	Rate at which glucose is generated from lactate when plasma level of lactate is high (measured in mg/kg/min)
normalGlucoseLevel_	100	Amount of glucose in liver measured in mg/dL
fluidVolume_	10	Volume of liver in dL
Glut2Km_	20*180.1559/10.0	Michaelis constant for GLUT2 measured in mg/deciliter
Glut2VMAX_	50	Maximum rate of movement measured in mg/kg/min

Table 2: Configuration parameters - Liver

As outlined in the algorithm given in Figure 6, the source of glucose for the liver is the portal vein. The rate and direction of movement of glucose between the liver and portal vein is determined by their relative concentrations of glucose. Glucose will move from the environment with higher concentration to the one with lower concentration. This can be explained using *Michaelis-Menten kinetics.* A detailed explanation of Michaelis-Menten kinetics and glucose transporters is beyond the scope of this documentation. However, a brief summary of the same is given below. Please note that this summary is based on information from *Frayn's* textbook [1].

```
1. Compare the concentrations of glucose in the portal vein and the
   liver, and transfer glucose from the organ/vein with higher
   concentration to vein/organ with lower concentration.
2. Release all the glucose remaining in the portal vein to the
   bloodstream.
3. Synthesize glycogen from glucose based on insulin and glucose level.
4. Perform glycogenolysis to form glucose based on insulin and glucose
   level.
5. Perform glycolysis based on insulin level, and release the lactate
   formed into the bloodstream.
6. Consume gluconeogenesis substrates from the blood to perform
   gluconeogenesis based on insulin level and substrate concentration.
7. Process amino acids in portal vein.
8. Based on glucose concentration inside the liver and blood glucose
   concentration, release glucose into the bloodstream.
```

Figure 6: Algorithm for processTick( ) function - Liver

Movement of glucose across membranes is facilitated by glucose transporters. Broadly speaking, there are two types of glucose transporters – active and passive. This explanation touches upon only passive transporters, also known as the GLUTn transporters. Passive transporters mediate the movement of glucose down a concentration gradient. Michaelis-Menten kinetics is often used to explain the properties of this movement. In the GS simulator, the rate of movement across membranes, $V_{transport}$, is modeled using Michaelis-Menten law i.e. $V_{max} * ([S]/ (K_m +[S]))$. Here, V_{max} refers to the maximal rate of transport while [S] refers to the difference in relative concentrations across membranes. In this context, Michaelis-Menten constant or K_m indicates the affinity of a transporter for glucose molecules. A low value for K_m suggests high affinity. i.e. the rate of uptake is independent of glucose concentrations within the normal plasma glucose concentration range (i.e. uptake is more or less constant). On the other hand, when affinity is low, the rate of glucose uptake is calculated using Michaelis-Menten law.

Liver cells have predominantly GLUT2 type of glucose transporter [1]. GLUT2 has low affinity for glucose molecules [1]. As a result, the rate of movement of glucose between the liver and portal vein is dependent on their relative concentrations of glucose. During each tick, the current glucose concentrations in liver and portal vein determine the amount of glucose transferred between the two. Once the transfer has been simulated, all the glucose remaining in the portal vein is released into the bloodstream. This is done in order to simulate the fact that after food intake, most of the ingested glucose passes through the liver (without being absorbed or stored) initially to enter the blood stream.

```
double glInPortalVein = body->portalVein->getConcentration();
double glInLiver = glucose/fluidVolume_;

if (glInLiver < glInPortalVein)
{
    double diff = glInPortalVein - glInLiver;
    x = (double)(Glut2VMAX__(SimCtl::myEngine()));
    double g = x * diff/ (diff + Glut2Km_);

    if (g > body->portalVein->getGlucose() )
    {
        g = body->portalVein->getGlucose();
    }

    body->portalVein->removeGlucose(g);
    glucose += g;
}

//release all portalVein glucose to blood
body->portalVein->releaseAllGlucose();
```

Figure 7: Glucose absorption by the liver

Following this, the GS simulator moves on to model the remaining metabolic processes performed by the liver organ.

1. Glycogen synthesis

In the Liver class, the configuration parameter **glucoseToGlycogen_** determines the amount of glucose converted to glycogen, for a healthy individual, in each tick. The actual amount of glycogen synthesized is a random amount averaged on this parameter. Furthermore, glycogen synthesis in the liver organ depends on the current concentration of glucose in the liver, and the current insulin level of the individual [1]. Insulin suppresses glucose production while it stimulates glycogen synthesis [1] [2]. The GS simulator models the impact of these factors by calculating a fractional value '**scale**' which is then multiplied to glucoseToGlycogen_ to calculate the absolute amount of glycogen synthesized during each tick by the liver. 'Scale' is positively related to both the current concentration of glucose in the liver and the current insulin level of the individual (maintained by the **Blood** class using **insulinPeak_**).

```
 glInLiver = glucose/fluidVolume_;
double scale = glInLiver/normalGlucoseLevel_;
scale *= body->blood->insulin;
x = (double)(glucoseToGlycogen__(SimCtl::myEngine()));
double toGlycogen = scale * x;
if (toGlycogen > glucose)
    toGlycogen = glucose;
glycogen += toGlycogen;
glucose -= toGlycogen;
```

Figure 8: Glycogen synthesis in the liver

2. Glycogenolysis

Hepatic glycogenolysis or glycogen breakdown is brought about by a drop in blood glucose level [1]. This occurs both during the post-absorptive state and during an exercise

event. In the GS simulator, the configuration parameter **glycogenToGlucose_** determines the amount of glycogen converted to glucose, for a healthy individual, in each tick. Similar to glycogen synthesis, this process is also dependent on the current insulin level of the individual [1] [2]. Glycogenolysis also depends on the ability of the liver to respond to insulin fluctuation in the bloodstream. The GS simulator models this dependency using the configuration parameter **insulinResistance_**. As mentioned earlier, this parameter can take a value between 0 and 1, where 0 indicates a healthy person who responds to insulin as expected, while 1 indicates a person with extreme Type 2 diabetes. Similar to glycogen synthesis, the GS simulator models the impact of these factors by calculating a fractional value '**scale**' which is then multiplied to glycogenToGlucose_ to calculate the absolute amount of glycogen broken down during each tick by the liver. In order to suppress glycogenolysis when insulin level is high and stimulate it when the insulin level is low, 'scale' is inversely dependent on the insulin level. The impact of insulin is restricted by multiplying 'scale' with insulinResistance_, to model Type 2 diabetes.

```
scale = 1 - (body->blood->insulin)*(1 - (body>insulinResistance_));
glInLiver = glucose/fluidVolume_;
if( glInLiver > normalGlucoseLevel_ )
    scale *= normalGlucoseLevel_/glInLiver;
x = (double)(glycogenToGlucose__(SimCtl::myEngine()));
double fromGlycogen = scale * x;
if( fromGlycogen > glycogen )
    fromGlycogen = glycogen;
glycogen -= fromGlycogen;
glucose += fromGlycogen;
```

Figure 9: Glycogen breakdown in the liver

3. Glycolysis

Glycolysis is the process by which glucose is converted to lactate. Similar to glycogenolysis, glycolysis is also dependent on the current insulin level and insulin resistance of the individual [1] [2]. Insulin stimulates glycolysis [1]. The GS simulator mimics glycolysis in the liver using the configuration parameters **glycolysisMin_** and **glycolysisMax_**. In this model, irrespective of other factors glycolysis always occurs at a basal rate with an average value of glycolysisMin_. Additionally, depending on the current insulin level and insulin sensitivity of the individual, glycolytic rate can increase. This insulin dependency is also modeled by the GS simulator and the additional amount of glucose converted to lactate is calculated as a fraction of the peak value glycolysisMax_, based on 'scale' - the variable used to model the impact of insulin using the same logical concepts described above. In the GS simulator, lactate is the only product of glycolysis. This lactate is released into the bloodstream once formed.

```
    scale = (1.0 - body->insulinResistance_)*(body->blood->insulin);

    x = (double)(glycolysisMin__(SimCtl::myEngine()));
    if( x > glycolysisMax_*(body->bodyWeight_))
        x = glycolysisMax_*(body->bodyWeight_);

    double toGlycolysis = x + scale*((glycolysisMax_*(body-
>bodyWeight_)) - x);

    if( toGlycolysis > glucose)
        toGlycolysis = glucose;

    glucose -= toGlycolysis;
    body->blood->lactate += toGlycolysis*glycolysisToLactateFraction_;
```

Figure 10: Glycolysis in the liver

4. Gluconeogenesis

Gluconeogenesis is the process by which glucose is generated from non carbohydrate substrates - primarily lactate, alanine and glycerol. The pathway of gluconeogenesis is primarily controlled by two factors – substrate concentration and the hormonal regulation of concerned enzymes [1]. The effect of enzymes is out of the scope of this study and is hence not currently modeled in the simulator. Additionally, gluconeogenesis is also dependent on the insulin level (i.e. it is inhibited by insulin) and insulin resistance of the individual [1] [2].

```
scale = 1 - (body->blood->insulin)*(1 - (body->insulinResistance_));
x = (double)(gngRate__(SimCtl::myEngine()));
double gng = x *scale;
glucose += body->blood->consumeGNGSubstrates(gng);

x = (double)(gngFromLactateRate__(SimCtl::myEngine()));
glycogen += body->blood->gngFromHighLactate(x);
```

Figure 11: Gluconeogenesis in the liver

The simulator models the process of gluconeogenesis by consuming the required substrates (maintained as a single entity) from the Blood class and adding it to the glucose store in the Liver class. The amount of substrates consumed is controlled by configuration parameter **gngRate_**. At each tick, the liver consumes a random amount of substrates - with an average value of gngRate - from the bloodstream. The impact of insulin level and insulin resistance is modeled based on ‘scale’ as described above.

Additionally, hepatic gluconeogenesis can also be stimulated by an increased concentration of lactate in the bloodstream. For example, after an exercise event. In this

scenario however, gluconeogenesis will contribute to glycogen stored in the liver rather than generate glucose [1]. This metabolic pathway has been modeled in this simulator, by calling a method **gngFromHighLactate()** of the Blood class. This method calculates the amount of glycogen produced from lactate based on the current concentration of lactate in the blood.

In addition to these four metabolic processes, during every tick the relative concentrations of glucose in the liver and bloodstream is used to determine the movement of glucose between the two. Similar to the movement of glucose between the portal vein and liver, this flow is also modeled - in the GS simulator - using Michaelis-Menten kinetics of GLUT2 transporter.

Finally, the liver also consumes unbranched amino acids from the bloodstream. A majority of the consumed amino acids is converted to alanine and transferred back to the blood stream, while the remainder is oxidized or used for the synthesis of glucose, fatty acids and ketone bodies. In the simulator the latter is not modeled. The former is modeled as part of the **PortalVein** subclass of the **StomachIntestine** class, and will be discussed later in this documentation.

MUSCLE

The **Muscle** class models the metabolic pathways of the skeletal muscles. There are two main processes that the GS simulator needs to capture with respect to skeletal muscles. These include glucose absorption from the bloodstream and glycolysis. Identical to the processes explained in the Liver class, the metabolic processes of the skeletal muscles are also multi-step processes with several intermediate products that the simulator does not model. It is important to note that, even

though these processes occur in a concurrent manner in actuality (like in the liver organ), the simulator models them sequentially in the Muscle class method **processTick()**. Additionally, the metabolism of skeletal muscles changes dramatically during an exercise event. The impact of exercise has also been modeled in the processTick()method.

The configuration parameters of the Muscle class are as follows:

Parameter	**Default Value**	**Comments**
basalGlucoseAbsorbed_	0.344	Glucose absorbed by the muscle at basal rate due to the presence of GLUT1 (mg/kg/min)
glucoseOxidationFraction_	0.5	The fraction of absorbed glucose that is oxidized
bAAToGlutamine_	0	The amount of glutamine produced from branched amino acids in 1 tick
glycolysisMin_	0.4	The minimum rate of glycolysis in the muscle measured in mg/kg/min
glycolysisMax_	9*glycolysisMin_	The peak value for glycolysis in the muscle, expressed in mg/kg/min
Glut4Km_	5*180.1559/10.0	Michaelis constant for GLUT4
Glut4VMAX_	10	Maximum rate of movement measured in mg/kg/min

Table 3: Configuration parameters - Muscle

A detailed explanation of the metabolic processes - with respect to glucose metabolism - of the muscle organ is given below:

1. Glucose absorption

In the muscle, glucose uptake is controlled by the glucose transporters GLUT1 and GLUT4 [1]. While GLUT1 plays a role in the uptake of glucose at a "basal" rate, GLUT4 is insulin-sensitive [1]. Insulin increases the rate at which muscle takes up glucose from the blood. The glucose absorbed by the muscle may either be oxidized, used for glycogen synthesis, or processed via the pathway of glycolysis.

```
   // Absorption via GLUT1
   x = (double)(basalAbsorption__(SimCtl::myEngine()));
   body->blood->removeGlucose(x);
   glycogen += x;

   // Absorption via GLUT4
   double bgl = body->blood->getBGL();
   double scale = (1.0 - body->insulinResistance_)*(body->blood-
>insulin);
   x = (double)(Glut4VMAX__(SimCtl::myEngine()));
   double g = scale*x*bgl/(bgl + Glut4Km_);
   body->blood->removeGlucose(g);
   glycogen += (1.0 - glucoseOxidationFraction_)*g;
```

Figure 12: Glucose absorption in the muscle

In the GS simulator, basal absorption of glucose has been modeled using the configuration parameter **basalAbsorption_** as a more or less constant uptake. The amount of glucose absorbed is calculated as a random value with an average of basalAbsorption_. This is due to the fact that GLUT1 - the transporter which mediates basal absorption in the muscle - has a high affinity for glucose and is therefore independent of the concentration of glucose in the bloodstream. On the other hand, glucose uptake by the glucose transporter GLUT4 has been modeled based on glucose concentration using Michaelis-Menten kinetics. Since GLUT4 is insulin-sensitive, the

impact of current insulin level and insulin resistance of the individual has also been modeled by calculating their 'scale' of impact (on glucose absorption) as discussed in the section 'Liver' above. In both these processes of glucose uptake, glucose is removed from the blood stream and stored internally as glycogen in the Muscle class.

2. Glycolysis

In the muscle, glucose absorbed from the bloodstream is either oxidized or stored locally as glycogen. The GS simulator thus models glycolysis by breaking down stored muscle glycogen to form lactate. The process of glycolysis in the muscle in similar to the corresponding process in the liver, and is therefore modeled in a similar manner using the configuration parameters parameters **glycolysisMin_** and **glycolysisMax_**.

```
scale = (1.0 - body->insulinResistance_)*(body->blood->insulin);

x = (double)(glycolysisMin__(SimCtl::myEngine()));
if( x > glycolysisMax_*(body->bodyWeight_))
   x = glycolysisMax_*(body->bodyWeight_);

g = x + scale* ( (glycolysisMax_*(body->bodyWeight_)) - x);

if( glycogen >= g )
{
     glycogen -= g;
     body->blood->lactate += g;
}
else
{
     body->blood->lactate += glycogen;
     glycogen = 0;
}
```

Figure 13: Glycolysis in the muscle

Skeletal muscles also participate in amino acid metabolism by converting branched amino acids in the blood to glutamine [1]. This metabolic pathway has also been modeled in the processTick() method by consuming branched amino acids from the **Blood** class and releasing it back to the Blood class as glutamine.

Carbohydrate metabolism in the skeletal muscle during exercise

The Muscle class plays a vital role in glucose metabolism during an exercise event. In reality, exercise is broadly categorized into aerobic and anaerobic exercise. Anaerobic exercise refers to high intensity short duration exercise while aerobic exercise on the other hand involves prolonged lower intensity exercise. An example of anaerobic exercise is sprinting, while running a marathon would be an example of aerobic exercise. In this simulator, only the impact of aerobic exercise on glucose metabolism has been modeled. Anaerobic exercise is beyond the scope of this study.

In order to meet the energy requirements of aerobic exercise, it is necessary for skeletal muscles to use stored fuels in addition to what is found in the muscle itself. Therefore, the muscle must be supplied with the required substrates through the circulatory system. Additionally, these fuels must be almost completely oxidized in order to prevent the build up of lactic acid, which causes the onset of muscle fatigue [1] [15].

The two sources of fuel are carbohydrates and fat, and their relative usage as energy substrates in the muscle vary based on the intensity and duration of the exercise event. In relatively light exercise most of the required energy comes from non-esterified fatty acids [1]. At higher

intensities, carbohydrate tends to predominate early on, with the predominance of fat increasing as glycogen stores get depleted [1]. The oxidation of glucose is a major source of energy for exercising human beings. The importance of carbohydrates as an energy source increases with the intensity of exercise. For low intensity aerobic exercise, carbohydrate oxidation accounts for 10-15% of total energy production, increasing progressively to 80-100% of total energy production during high intensity exercises [16]. It is important to note that in this context, exercise intensity refers to VO_{2max} or maximal aerobic capacity of the individual.

The skeletal muscle has access to two sources of glucose – plasma glucose and glycogen stored in the muscle itself. Plasma glucose is an important source of energy during exercise, with its importance increasing with the intensity of exercise [17]. However, research shows that the total contribution of plasma glucose to energy production during exercise remains approximately at around 10% without significant variation across exercise of different intensities [1] [17]. The GS simulator thus models plasma uptake by the skeletal muscle during exercise - in the processTick() method of the Muscle class - by calculating the amount of glucose required to support 10% of the current exercise event. This glucose is then consumed from the bloodstream.

On the other hand, the contribution of stored glycogen to total energy production varies based on the intensity of the exercise event. During low intensity exercise, stored glycogen does not contribute significantly to energy production, and carbohydrate oxidation appears to be met solely by glucose uptake [17]. From earlier discussion we know that the contribution of carbohydrate metabolism to total energy production increases with the intensity of exercise but, the contribution of plasma glucose uptake remains relatively constant across different levels of

exercise intensity. This means that at moderate to higher intensities, the contribution of stored muscle glycogen to total energy production becomes predominant. In order to simulate the contribution of stored glycogen, this study categorizes an exercise event into 3 broad classifications based on their intensity. The break points for this categorization are 3 MET and 6 MET. All exercises below 3 MET are considered low intensity exercises while those above 6 MET are considered high intensity exercises. The contribution of glycogen to total energy production is then calculated based on the category of the exercise event. For low intensity exercises, the contribution of glycogen is assigned to 0. In case of high intensity exercises, 30% of the total energy production comes from glycogen [1]. For moderate exercises (between 3 MET and 6 MET), the contribution is glycogen is calculated as proportional to the exercise intensity, with a maximum contribution of 30%.

Additionally, the rate of glycolysis is also affected by an exercise event. During light exercise, plasma taken up from the bloodstream is almost completely oxidized and therefore does not contribute to glycolytic flux. However, as the intensity of exercise increases, glycolytic flux increases, thereby resulting in a higher rate of glycolysis. The GS simulator models this increase based on the intensity of the exercise event.

STOMACHINTESTINE

The **StomachIntestine** class models the digestive system of the human body. It includes functionality to mimic the metabolic processes of the stomach, intestine and portal vein. Primarily, this class is responsible for modeling all the functionality related to glucose metabolism, between the time you ingest food and its corresponding nutrients appear in the

blood stream. In this study, ingested food has been modeled using its component nutrients - RAG, SAG, protein and fat. Although the simulator does not currently model fat and protein metabolism, these nutrients have an impact on the digestion of dietary carbohydrates (RAG and SAG) [18] [19]. This impact has been considered and modeled in this simulator.

The configuration parameters of the StomachIntestine class are as follows:

Parameter	**Default Value**	**Comments**
fluidVolumeInEnterocytes_	1	Configuration parameter of enterocytes - Expressed in dL
fluidVolumeInLumen_	1	Configuration parameter of enterocytes - Expressed in dL
Glut2Km_In_	20*180.1559/10.0	Configuration parameter of enterocytes - Michaelis-Menten constant for GLUT2 movement in mg/dL
Glut2VMAX_In_	1	Configuration parameter of enterocytes - Maximum rate of movement measured in mg/kg/min
Glut2Km_Out_	20*180.1559/10.0	Configuration parameter of enterocytes - Michaelis-Menten constant for GLUT2 movement in mg/dL
Glut2VMAX_Out_	1	Configuration parameter of enterocytes - Maximum rate of movement measured in mg/kg/min
sglt1Rate_	1	Configuration parameter of enterocytes - Constant rate of movement due to active transport measured in mg/min
glycolysisMin_	0.1801559	Configuration parameter of enterocytes - Basal rate of glycolysis in the enterocytes measured in mg/kg/min

glycolysisMax_	5*glycolysisMin_	Configuration parameter of enterocytes - The peak value for glycolysis in the enterocytes expressed in mg/kg/min
aminoAcidsAbsorptionRate_	1	Configuration parameter of enterocytes - Rate at which amino acids are released into the portal vein per tick (measured in mg/min)
glutamineOxidationRate_	1	Configuration parameter of enterocytes - Rate at which glutamine in the blood is oxidized (measured in mg/min)
glutamineToAlanineFraction_	0.5	Configuration parameter of enterocytes - Fraction of glutamine converted to alanine
RAG_Mean_	5	Configuration parameter of StomachIntestine - To control the rate at which RAG is available for absorption
RAG_StdDev_	5	Configuration parameter of StomachIntestine - To control the rate at which RAG is available for absorption
SAG_Mean_	60	Configuration parameter of StomachIntestine - To control the rate at which SAG is available for absorption
SAG_StdDev_	20	Configuration parameter of StomachIntestine - To control the rate at which SAG is available for absorption
FatDelayMax_	300	Configuration parameter of StomachIntestine - Maximum allowed delay caused in availability of glucose due to presence of fat in food

ProteinEffectMin_	0.5	Configuration parameter of StomachIntestine - Minimum delay caused in availability of glucose due to presence of protein in food
fluidVolume_	1	Configuration parameter of PortalVein - Expressed in dL

Table 4: Configuration parameters - Digestive system (StomachIntestine)

The process of digestion begins in the mouth followed by the stomach. However, since the role of both these organs is primarily limited to mechanical digestion [1] i.e. the breakdown and liquefaction of food particles, they do not play a significant role in this simulator. The mouth organ has not been modeled, while the functionality of the stomach organ in this simulator is limited to storing the ingested nutrients (in response to a food event).

The third stage (following mouth and stomach organ) in the process of digestion is intestinal absorption - the primary activity modeled in the StomachIntestine class. The algorithm for the **processTick()** method of the StomachIntestine class is given in Figure 14.

```
1. Calculate the amount of RAG digested in 1 tick, dependent on effect of
   protein.
2. Calculate the amount of SAG digested in 1 tick, dependent on effect of
   protein and fat.
3. Model the consumption of digested glucose by enterocytes i.e. the
   absorptive cells of intestine. (This glucose then either enters the
   portal vein or is used as a substrate for glycolysis)
4. Model the consumption of protein by enterocytes.
```

Figure 14: Algorithm for processTick( ) function - StomachIntestine

Englyst et al. [20] proposed a classification of dietary carbohydrates - as Rapidly Available Glucose (RAG) and Slowly Available Glucose (SAG) - based on their likely site, rate, and extent of digestion. Englyst's experiments concluded that RAG was available for intestinal absorption 20 minutes after food ingestion while SAG was available after 120 minutes. In order to accurately model the process of glucose digestion and absorption from the small intestine, it is necessary to take this into consideration. In the simulator, the appearance of RAG and SAG for intestinal absoprtion is modeled separately using a normal distribution with configuration parameters - **RAG_Mean_**, **RAG_StdDev_**, **SAG_ Mean _** and **SAG_StdDev_** -for mean and standard deviation.

At this point, it is necessary to note that the presence of fat and protein in diet can significantly impact the metabolism of glucose in the gastrointestinal tract by reducing the rate at which glucose from diet is available for digestion [18] [19]. One of the reasons suggested for this change is a delay in gastric emptying caused by the presence of protein and fat [19]. The simulator models this impact by reducing the amount of RAG and SAG available for intestinal absorption (at each tick) based on the amount of protein and fat present in the ingested food. Since no significant link was found between their respective impacts on glucose metabolism [19], the simulator models the effects of fat and protein on digestion separately. With respect to protein, the impact has been modeled dependent on the percentage of protein found in the diet. As the percentage of protein increases, the model widens the number of ticks over which glucose gets absorbed. The impact of fat has also been modeled in a similar manner. However, since the impact of fat plateaus once the amount of fat in the diet is high [21], the simulator limits the

delay caused by fat using a configuration variable **FatDelayMax_** when the percentage of fat in the diet is over 65%.

Once the ingested carbohydrates have been processed and is finally available for intestinal absorption, they enter the enterocytes (absorptive cells of intestinal mucosa). According to Frayn's textbook [1], this absorption is mediated by both active and facilitated diffusion. During the initial stages of digestion, the high concentration of glucose available for absorption enables passive (facilitated) diffusion via GLUT2. As you may recall, GLUT2 has low affinity for glucose molecules. Therefore, the rate of movement of glucose is dependent on the concentration of glucose available for absorption and the amount of glucose currently stored in the enterocytes. In the GS simulator, this movement has been modeled based on Michaelis-Menten kinetics. In the later stages of digestion, once the concentration of glucose available for absorption reduces, active transport - mediated by SGLT-1, ensures that dietary glucose is completely absorbed [1]. This has been modeled in the GS simulator using a configuration variable **sglt1_**. Once the glucose available for absorption has dropped, the GS simulator loads all remaining dietary glucose into the enterocytes at a basal rate averaged on sglt1_, during each tick.

Once the glucose has been loaded into the enterocytes, it either enters the portal vein, mediated by GLUT2 or it acts as a substrate for glycolysis [1]. The simulator models the former using Michaelis-Menten kinetics. The latter depends on the insulin level and insulin resistance of the individual and results in lactate in the bloodstream. Glycolysis in the enterocytes is modeled using configuration parameters **glycolysisMin_** and **glycolysisMax_**. The impact of insulin has been modeled using 'scale' as described in earlier sections.

It is important to note that enterocytes also absorb dietary protein, resulting in the addition of amino acids in the bloodstream. This has been modeled using a method of the **PortalVein** subclass of StomachIntestine. Specific details about amino acid metabolism is out of the scope of this document. Although (some) amino acid metabolism is considered and implemented in this simulator, the research and implementation of this portion was done by other researchers and is hence not included my thesis documentation. The occurrence of this metabolic process is however mentioned for the purpose of completeness.

BLOOD

With the exception of the portal vein, the Blood class represents the entire bloodstream of the human body. This class is primarily used to maintain substrates related to glucose metabolism in the blood. Several methods, in addition to the **processTick( )** method, have been defined in the Blood class for this purpose. Additionally, it is relevant to note that red blood cells consume glucose every minute in order to produce lactate via glycolysis. This process has been captured in the simulator and is modeled in the processTick( ) method of the Blood class.

The configuration parameters of the Blood class are as follows:

Parameter	**Default Value**	**Comments**
fluidVolume_	50	Expressed in dL
glycolysisMin_	0.1801559	The basal rate for glycolysis in the blood mxeasured in mg/kg/min
glycolysisMax_	5*glycolysisMin_	The peak value for glycolysis in the bloodstream expressed in mg/kg/min
normalGlucoseLevel_	100	Measured in mg/dL
highGlucoseLevel_	200	Measured in mg/dL

minGlucoseLevel_	40	Measured in mg/dL
highLactateLevel_	4053.51	Measured in mg

Table 5: Configuration parameters - Blood

The two main processes modeled in processTick( ) includes glycolysis (conversion of glucose to lactate) and the maintenance of insulin level in the human body. The GS simulator models glycolysis in the blood using the same logical concepts used for other organs, i.e., it is modeled using configuration parameters **glycolysisMin_** and **glycolysisMax_**, and the impact of insulin is modeled by calculating 'scale'.

```
    double scale = (1.0 - body->insulinResistance_)*(body->blood->insulin);

    x = (double)(glycolysisMin__(SimCtl::myEngine()));
    if( x > glycolysisMax_*(body->bodyWeight_))
        x = glycolysisMax_*(body->bodyWeight_);

    double toGlycolysis = x + scale * ( (glycolysisMax_*(body->bodyWeight_))
- x);

    if( toGlycolysis > glucose)
        toGlycolysis = glucose;

    glucose -= toGlycolysis;
    body->blood->lactate += toGlycolysis;
```

Figure 15: Glycolysis in the bloodstream

Following the pathway of glycolysis, the simulator moves on to update the insulin level of the human body based on the current plasma glucose level. This process depends on the diabetic health of the individual. The GS simulator models this using **insulinPeakLevel_**, a configuration parameter of HumanBody class. This parameter (which takes a value between 0 and 1) is configured to 1 for a normal healthy individual and to 0 for an individual suffering from extreme Type 1 Diabetes (0 indicates that the pancreas is unable to produce any insulin.)

The GS simulator updates the insulin level of the individual at each tick by comparing the current plasma glucose level with the configuration parameters **highGlucoseLevel_** and **normalGlucoseLevel_** (configuration parameters of the Blood class). The insulin level is assigned as 0 when plasma glucose is lesser than normal glucose level (configuration parameter of the Blood class), and is assigned to **insulinPeakLevel_** when the plasma glucose level is at the configured high value. In between these extremes, the insulin level is calculated as a fraction of peak glucose level, based on current plasma glucose concentration.

```
    double bgl = glucose/fluidVolume_;

    if( bgl >= highGlucoseLevel_)
        insulin = body->insulinPeakLevel_;
    else
    {
        if( bgl < normalGlucoseLevel_)
            insulin = 0;
        else
        {
            insulin = (body->insulinPeakLevel_)*(bgl -
normalGlucoseLevel_)/(highGlucoseLevel_ - normalGlucoseLevel_);
        }
    }
```

Figure 16: Maintenance of insulin level in the bloodstream

As mentioned earlier, several methods in addition to the processTick( ) method have been defined in the Blood class to maintain additional substrates in the bloodstream. These methods are called by the organ classes every time the organ consumes or releases the corresponding substrate from/into the bloodstream. The methods present are:

1. **consumeGNGSubstrates()**
2. **removeGlucose()**

3. **addGlucose()**
4. **gngFromHighLactate()**

In each of these functions, the organ that consumes/adds the substrate will pass the amount to be processed as a parameter to these functions. Since the names of the functions are self-explanatory, I will not be explaining their functionality further.

ADIPOSETISSUE

Biologically speaking, there are two types of adipose tissue in the human body. In an adult human, the adipose tissue is almost all "white", and its major metabolic role is the controlled storage and release of fat [1]. There are two sources for the fat stored in the adipose tissue - triacylglycerol from plasma and *de novo lipogenesis*, the synthesis of lipid from other sources, mainly glucose [1]. Since a typical western diet contains contains significant amount of fatty acids, the former is more important. Based on this information, in this study, we have not modeled the uptake of glucose to form TAG in adipose tissue in our simulator.

The release of fat from adipose tissue results in the formation of glycerol. Since glycerol is one of the substrates utilized for gluconeogenesis, we have modeled this pathway in our simulator. Fat mobilization is suppressed by insulin [1]. In the simulator, this has been taken into consideration by limiting the occurrence of the fat breakdown (lipolysis) to when the the plasma glucose level is lesser than the normal glucose level of the human body. Additionally, since the simulator does not keep track of the amount of TAG stored internally in the adipose tissue, the

amount of glycerol entering the bloodstream at each tick has been modeled using configurable parameter, loaded from the configuration file.

The adipose tissue also participates in amino acid metabolism in a manner similar to that of the muscle organ i.e. adipose tissue converts branched amino acids in the blood to glutamine. This metabolic pathway has also been modeled in the **processTick( )** method.

The configuration parameters of the AdiposeTissue class are as follows:

Parameter	**Default Value**	**Comments**
bAAToGlutamine_	0	The amount of glutamine produced from branched amino acids in 1 tick
lipolysisRate_	0	The amount of gluconeogenesis substrates generated from fatty acids in 1 tick

Table 6: Configuration parameters - AdiposeTissue

KIDNEY

The **Kidney** class models the metabolic pathways of the kidney body organ. With respect to glucose metabolism, these pathways primarily include glycolysis, gluconeogenesis and glucose reabsorption from the kidney.

Kidney cells have both GLUT1 and GLUT2 glucose transporters [1]. While GLUT1 results in a basal rate of movement of glucose across the membrane, the movement mediated by GLUT2 is dependent on the relative concentrations of glucose in the kidney and the bloodstream. The GS

simulator models the basal rate of movement using a configuration parameter **basalAbsorption_**. During each tick, a random value (with an average of basalAbsoprtion_) is calculated to simulate basal glucose movement between the kidney and the bloodstream. On the other hand, the movement mediated by GLUT2 is modeled using Michaelis-Menten kinetics in the GS simulator. During each tick, the relative glucose concentrations in the kidney and the bloodstream determines the rate and direction of glucose transfer between the two. Additionally, renal glucose uptake also depends on the insulin-sensitivity of the individual [8]. The GS simulator models this dependency using the configuration variable **insulinResistance_**.

```
    double bgl = body->blood->getBGL();
    double glInKidney = glucose/fluidVolume_;

    x = (double)(Glut2VMAX__(SimCtl::myEngine()));
    double y = (double)(basalAbsorption__(SimCtl::myEngine()));

    if( glInKidney < bgl )
    {
        double diff = bgl - glInKidney;
        double g = (1 + body->insulinResistance_)*x*diff/(diff + Glut2Km_);
        g += y;

        body->blood->removeGlucose(g);
        glucose += g;
    }
    else
    {
        double diff = glInKidney - bgl;
        double g = (1 + body->insulinResistance_)*x*diff/(diff + Glut2Km_);
        g += y;

        if( g > glucose )
        {
           exit(-1);
        }

        glucose -= g;
        body->blood->addGlucose(g);
}
```

Figure 17: Glucose uptake by the kidney

The simulator also models glycolysis and gluconeogenesis, occurring in the kidney organ, during each tick. Based on data from Gerich's paper [2], we know that both these processes - in the kidney - are dependent on the insulin level and insulin resistance of the individual. The simulator thus models these processes as explained earlier (for other organs) in this documentation. Glycolysis results in lactate in the bloodstream while gluconeogenesis consumes substrates from the bloodstream to produce glucose, which is stored internally in the Kidney class. Similar to the liver, gluconeogenesis is also stimulated in the presence of high lactate content in the blood, and is thus modeled in a similar manner.

```
    double scale = (1.0 - body->insulinResistance_)*(body->blood->insulin);

    x = (double)(glycolysisMin__(SimCtl::myEngine()));
    if( x > glycolysisMax_*(body->bodyWeight_))
        x = glycolysisMax_*(body->bodyWeight_);

    double toGlycolysis = x + scale* ( (glycolysisMax_*(body->bodyWeight_))
- x);

    if( toGlycolysis > glucose)
        toGlycolysis = glucose;
    glucose -= toGlycolysis;
    body->blood->lactate += toGlycolysis;
```

Figure 18: Glycolysis in the kidney

```
   scale = 1 - (body->blood->insulin)*(1 - (body->insulinResistance_));
   x = (double)(gngRate__(SimCtl::myEngine()));
   double gng = x *scale;
   glucose += body->blood->consumeGNGSubstrates(gng);

   x = (double)(gngFromLactateRate__(SimCtl::myEngine()));
   glucose += body->blood->gngFromHighLactate(x);
```

Figure 19: Gluconeogenesis in the kidney

Kidney is also responsible for reabsorbing (almost) all the glucose from the glomerular filtrate, since the human body does not 'want' to excrete glucose [1]. Since the GS simulator does not have a separate class representing the blood in the glomerular filtrate, this process has been modeled using the common bloodstream or Blood class. In order to model this process, the GS simulator uses two configuration parameters - **glucoseExcretionRate_** and **reabsorptionThreshold_**. The amount of glucose to be excreted in urine - for each tick - is specified using the configuration parameter glucoseExcretionRate_, while reabsorptionThreshold_ is used to indicate how much of this glucose is reabsorbed into the bloodstream. This process has been modeled as part of the **processTick()** method.

Finally, Frayn's textbook [1] states that while glutamine is not a good substrate for hepatic uptake, it is particularly removed by the kidney for further processing. The simulator models this metabolic pathway by consuming a constant amount of glutamine consume from the bloodstream during each tick. This amount is specified using the configuration parameter **glutamineConsumed_**.

The configuration parameters of the Kidney class are as follows:

Parameter	**Default Value**	**Comments**
fluidVolume_	1.5	Measured in dL
Glut2VMAX_	5	Michaelis constant for GLUT2
Glut2Km_	20*180.1559/10.0	Maximum rate of movement measured in mg/kg/min
Glut1Rate_	1	Constant rate of movement measured in mg/kg/min
glycolysisMin_	0.1801559	The minimum rate of glycolysis in the kidney measured in mg/kg/min

glycolysisMax_	5*glycolysisMin_	The peak value for glycolysis in the kidney expressed in mg/kg/min
gluconeogenesisRate_	1.8*0.45038975	Rate at which gluconeogenesis occurs in each tick measured in mg/kg/min
gngFromLactateRate_	gluconeogenesisRate_	Rate at which glucose is generated from lactate when plasma level of lactate is high (measured in mg/kg/min)
glutamineConsumed_	0	Amount of glutamine consumed by the kidney at each tick
reabsorptionThreshold_	11*180.1559/10	Amount of glucose reabsorbed from the glomerular filtrate measured in mg/dl
glucoseExcretionRate_	100/(11*180.1559/10)	Amount of glucose excreted in urine measured in mg/dl

Table 7: Configuration parameters - Kidney

BRAIN

The **Brain** class models the metabolic pathways of the brain body organ. With respect to glucose, metabolism within the brain is rather straightforward. Under normal circumstances, the brain does not appear to use any metabolic fuel other than glucose. The glucose transporter expressed in the brain is GLUT3. GLUT3 has a high affinity for glucose and therefore, the consumption of glucose by the brain is not dependent on plasma glucose concentration. GLUT3 is also not sensitive to insulin.

The glucose consumed by the brain is primarily completely oxidized. A portion of it is used to form alanine, which it then releases back into the blood stream. This consumption of glucose by the brain has been modeled by the simulator in the **processTick()** method of the Brain class. In

addition to glucose consumption, the brain also consumes branched amino acids from the blood. The amino acids are converted to glutamine and released back to the bloodstream. This process has also been modeled in the simulator. The configuration parameters of the Brain class are as follows:

Parameter	Default Value	Comments
glucoseOxidized_	1.08	Measured in mg/kg/min
glucoseToAlanine_	0	The amount of alanine produced from plasma glucose in 1 tick
bAAToGlutamine_	0	The amount of glutamine produced from branched amino acids in 1 tick

Table 8: Configuration parameters - Brain

The algorithm for the processTick() method of the Brain class is given in Figure 20.

```
1. Consume glucose from the blood based on body weight.
2. Convert a portion of consumed glucose to alanine (based on
   configuration parameter)
3. Release alanine to blood
4. Consume branched amino acids from the blood.
5. Generate glutamine from branched amino acids
6. Release glutamine to blood
```

Figure 20: Algorithm for processTick( ) function - Brain

HEART

The **Heart** class models the metabolic pathways of the heart body organ. In reality, the heart can use a number of fuels, including fatty acids, glucose and blood [1]. The rate at which it uses each fuel is dependent on their concentration in the blood.

With respect to uptake of glucose, the heart has two glucose transporters, GLUT1 and GLUT4 While GLUT1 results in a basal rate of movement of glucose across the membrane, the movement mediated by GLUT4 is insulin-sensitive and depends on the relative concentrations of glucose in the heart and the bloodstream. The GS simulator models the basal rate of movement using a configuration parameter **basalAbsorption_**. On the other hand, the movement mediated by GLUT4 is modeled using Michaelis-Menten kinetics. The impact of insulin on GLUT4 absorption is calculated using the same logical concepts as described for other organs. Insulin stimulates glucose uptake mediated by GLUT4. Glucose absorption by the heart has been modeled in the **processTick()** method of the Heart class. Additionally, the heart also consumes glucose from the blood to form lactate. This metabolic pathway has been modeled using the configurartion parameter **lactateOxidized_**.

The configuration parameters of the Heart class are as follows:

Parameter	**Default Value**	**Comments**
basalGlucoseAbsorbed_	14	Measured in mg/kg/min
Glut4Km_	5*180.1559/10.0	Michaelis constant for GLUT4
Glut4VMAX_	0	Maximum rate of movement measured in mg/kg/min
lactateOxidized_	0	The amount of plasma lactate oxidized by the heart in 1 tick

Table 9: Configuration parameters – Heart

The algorithm for the processTick() method of the Heart class is given in Figure 21.

1. Consume glucose at the basal rate (dependent on body weight)
2. Consume additional glucose based on plasma glucose concentration and insulin level of the individual.
3. Consume glucose from the blood to form lactate.
4. Oxidize a portion of this lactate based on configuration parameters.
5. Release remaining lactate to blood stream.

Figure 21: Algorithm for processTick( ) function - Heart

CHAPTER 3

SIMULATION RESULTS AND MODEL VERIFICATION

In this chapter, the results of simulating glucose metabolism in the human body, for 300 minutes, is presented. 3 set of simulations were conducted in order to verify different aspects of glucose metabolism modeled by the GS simulator.

NORMAL GLUCOSE HOMEOSTASIS - HEALTHY INDIVIDUAL

The first set of simulations (Figure 22 and Figure 23) verify the maintenance of normal glucose homeostasis in a healthy individual. Multiple simulations were run varying the initial blood glucose and lactate concentrations. Figure 22 illustrates the variation of plasma glucose concentration (measured in mg/dL) with time, while Figure 23 depicts the corresponding fluctuation of glycogen (measured in mg) in the liver. As expected, each curve in Figure 22 (each representing a different initial blood glucose level) converged to the normal blood glucose level within 2 hours, irrespective of the initial glucose/lactate concentrations. This mimics the maintenance of normal glucose homeostasis in a healthy individual. In the GS simulator, normal plasma glucose level can be modified by changing the appropriate configuration parameters.

In the human body, the required plasma glucose concentration is maintained by primarily controlling the metabolic processes in the liver, muscle and kidney. In case of liver metabolism - broadly speaking - if the blood glucose level is higher than normal, the liver takes in glucose and stores it in the form of glycogen, thereby increasing the local glycogen storage; and if the blood glucose level is lower than normal, glycogen breakdown in the liver increases, resulting in a

reduced amount of glycogen in the liver. Figure 23 rightly illustrates this fluctuation in liver glycogen storage with respect to plasma glucose concentration. As you can see, the graph for Initial BGL 300 increases rapidly for the first 1 hour (due to high plasma glucose concentration). The liver glycogen then decreases once the plasma glucose level has stabilized. This decrease illustrates the maintenance of plasma glucose level at 100-120mg/dL when organs consume glucose from the bloodstream.

Similarly, the graph for Initial BGL 50 portrays the rapid breakdown of liver glycogen at low plasma glucose concentrations. The graph correctly shows a steep decline in liver glycogen till the plasma glucose level has stabilized. Beyond this the slope of the graph reduces to illustrate the breakdown of liver glycogen at normal plasma glucose levels.

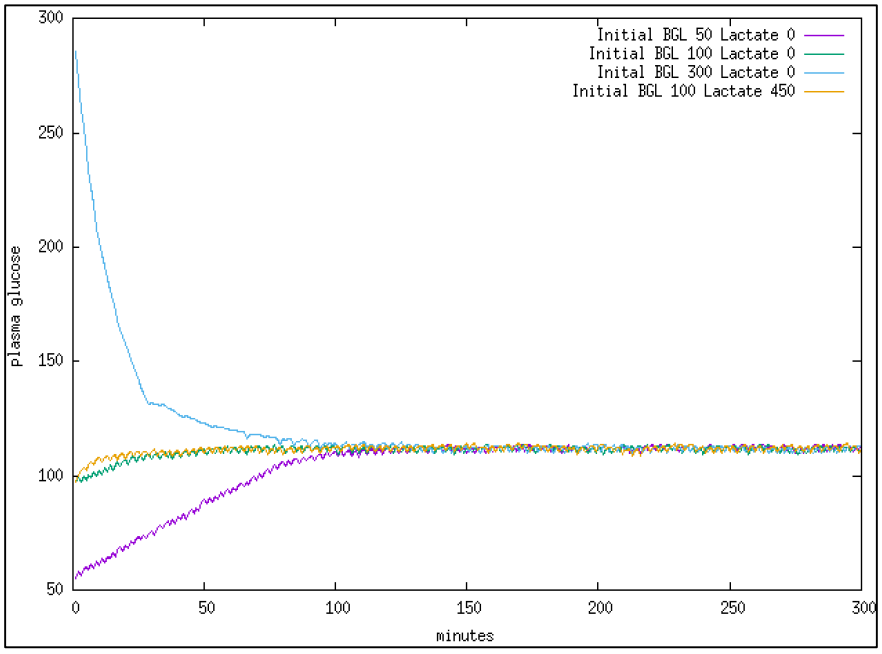

Figure 22: Blood Glucose Level vs time - Healthy individual

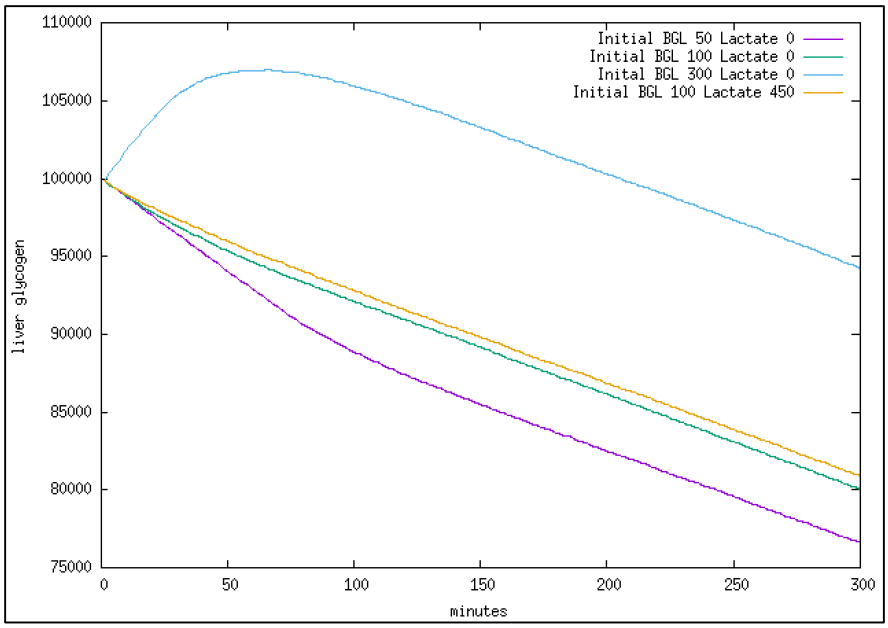

Figure 23: Liver Glycogen Storage vs time - Healthy individual

IMPACT OF DIABETES ON NORMAL GLUCOSE HOMEOSTASIS

Figure 24 through Figure 38 illustrates the effect of Type 1 and Type 2 diabetes on glucose metabolism. In this set of simulations, we verify the effect of insulin peak, i.e. the capacity of the pancreas to produce insulin, on plasma glucose concentration and the corresponding fluctuation of liver and muscle glycogen level when the insulin resistance of the individual is fixed. Please note that insulin resistance is a fraction between 0 and 1 used in the simulator to control the reaction of various organs to the presence of insulin in the blood (i.e. it models the extent of Type 2 diabetes), while insulin peak is used to indicate the maximum amount of insulin that the individual is capable of producing in comparison to a healthy individual. A perfectly healthy individual will have insulin resistance 0 and insulin peak 1. A value of 0 for insulin peak level indicates Type 1 Diabetes (the diabetes variant where the body does not have the ability to produce any insulin). People with Type 2 Diabetes also over time partially lose their ability to produce insulin.

Simulations - Blood Glucose Level

The initial blood glucose level is set at a high value of 300 mg/dL. Figures 24 though 28 illustrates the variation of plasma glucose with time. As expected, for each level of insulin resistance, the insulin peak value determines the rate at which blood glucose level returns to the normal value. The GS simulator rightly demonstrates that the plasma glucose level returns to the normal level at an increasingly slower rate when the extent of Type 1 diabetes increases.

Figure 24 illustrates glucose metabolism in an individual with insulin resistance 0, i.e. a person whose organs respond to insulin in the bloodstream as expected. Each curve in turn represents an

individual with a different value for insulin peak i.e. individuals with varying extents of Type 1 diabetes. When insulin peak is 1 and insulin resistance is 0, we see that the blood glucose level stabilizes rapidly. This mimics the maintenance of normal glucose homeostasis in a healthy individual. However, as the value for insulin peak decreases (indicating a reduction in the ability of the pancreas to produce insulin) we see that time taken for the curves to stabilize (i.e. time taken to return to normal plasma glucose concentration) increases. This is because the GS simulator correctly models the insulin sensitivity of metabolic processes in various organs. For example, consider an individual with insulin peak 0, i.e. an individual who cannot produce insulin. In this case, glucose metabolism in several organs - especially the liver, muscle and kidney - is severely impacted, resulting in increased levels of glucose in the bloodstream. In the liver, glycogen breakdown, glycogen synthesis, gluconeogenesis and glycolysis is not regulated by insulin as expected, when the insulin peak value is 0. As a result, glycogen breakdown occurs at all times irrespective of plasma concentrations, while glycogen synthesis is always restricted. This in turn increases the local concentration of glucose in the liver, thereby reducing absorption of glucose by the liver organ. Similarly, in the muscle, glucose absorption is restricted since GLUT4 is insulin dependent. In all organs, glycolysis is restricted and gluconeogenesis continues unrestricted, both contributing to glucose in the bloodstream.

Figure 28 illustrates glucose metabolism in an individual with insulin resistance 1, i.e. a person whose organs do not respond as expected to insulin. In this case, we see that even if pancreas produce insulin as expected (insulin peak 1), the stabilization of plasma glucose concentration is less rapid. This is a because the organs do not respond to insulin in the bloodstream as expected, and this sensitivity is modeled in the GS simulator. For example, consider an individual with

insulin peak 1 and insulin resistance 0. In this case, insulin sensitive pathways - such as glycogen breakdown in the liver - are not regulated by insulin level of the bloodstream. Glycogen breakdown in the liver happens in an unrestricted manner (since liver does not respond to high insulin level in body), while glucose absorption by the muscle is also restricted. Additionally, in all organs, glycolysis is restricted and gluconeogenesis continues unrestricted, both contributing to glucose in the bloodstream.

1. INSULIN RESISTANCE 0

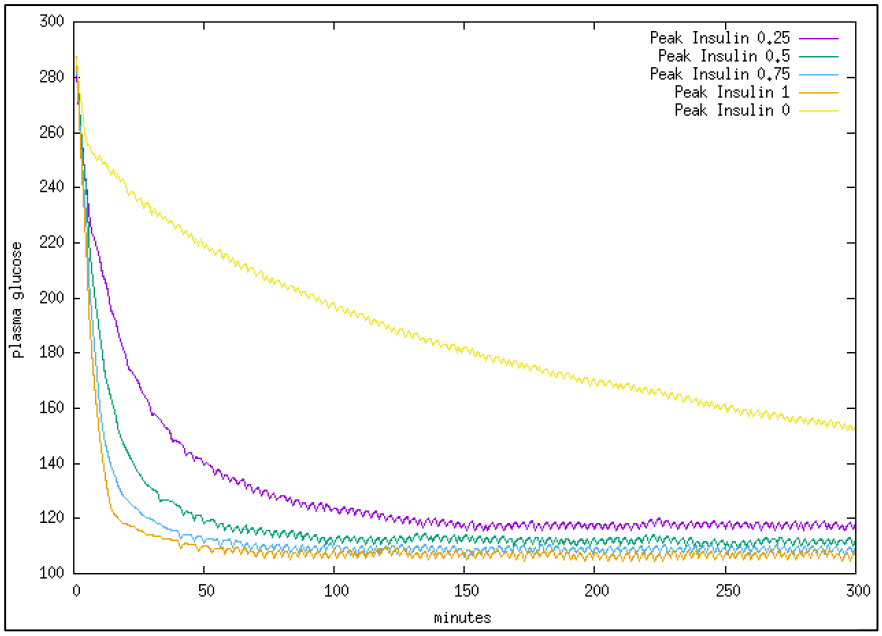

Figure 24: Blood Glucose Level - Insulin Resistance 0

2. INSULIN RESISTANCE 0.25

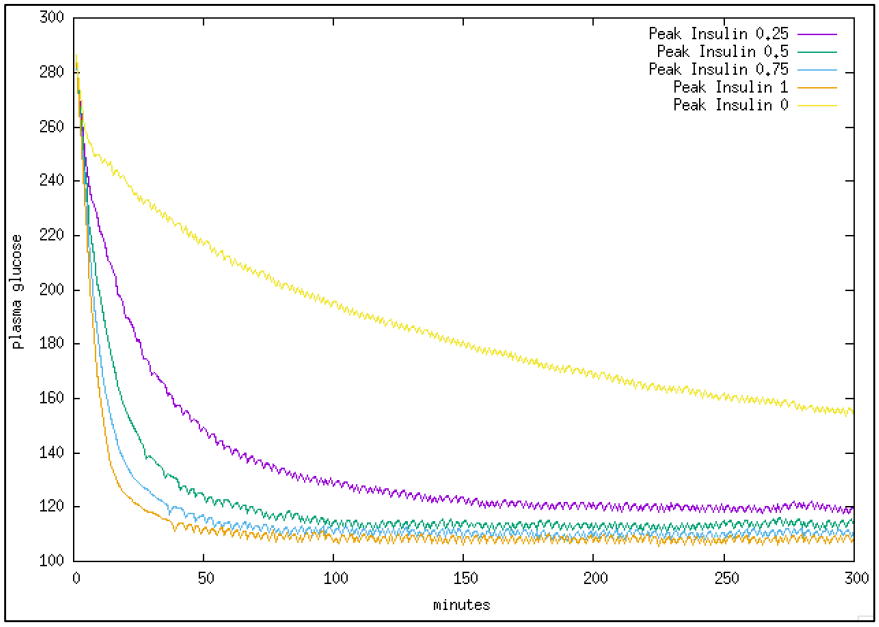

Figure 25: Blood Glucose Level - Insulin Resistance 0.25

3. INSULIN RESISTANCE 0.5

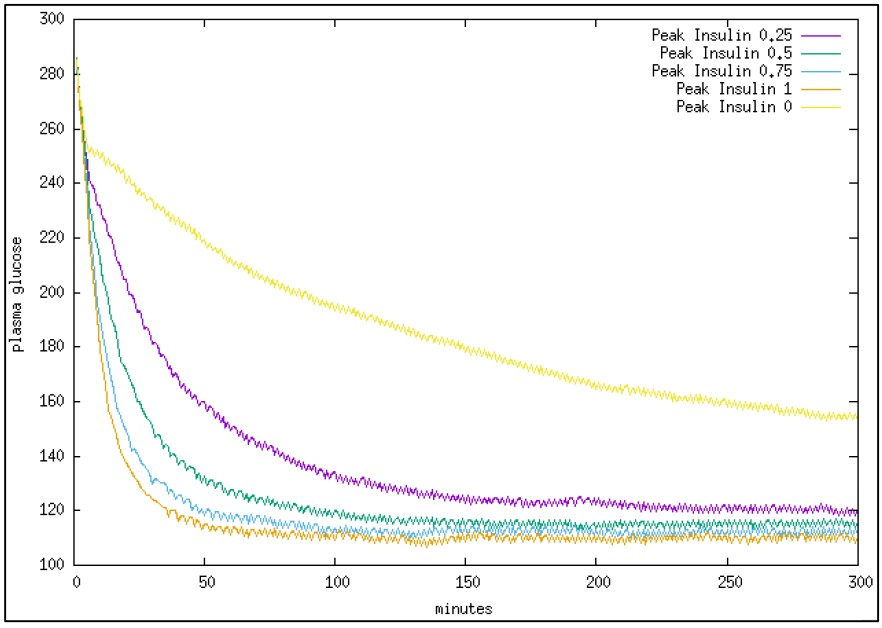

Figure 26: Blood Glucose Level - Insulin Resistance 0.5

4. INSULIN RESISTANCE 0.75

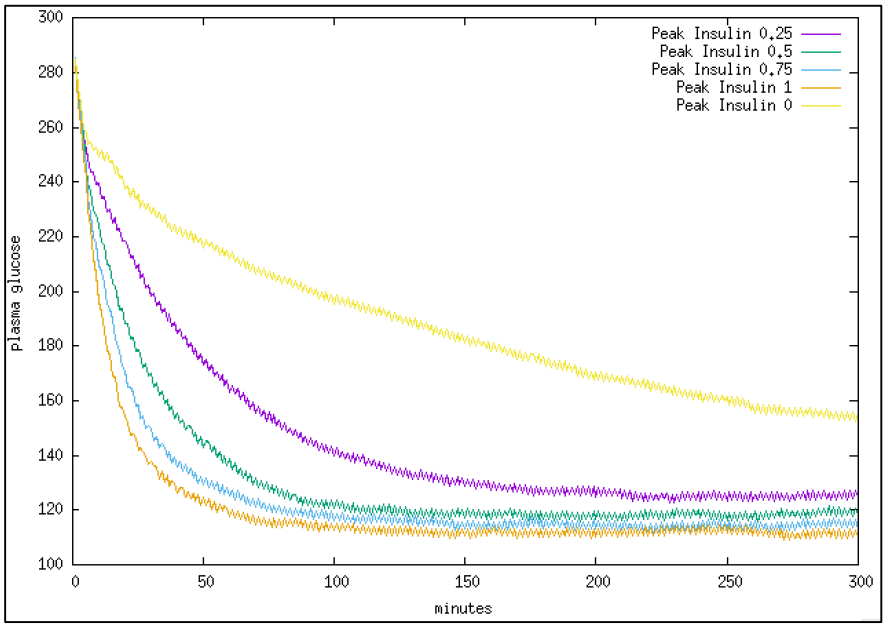

Figure 27: Blood Glucose Level - Insulin Resistance 0.75

5. INSULIN RESISTANCE 1

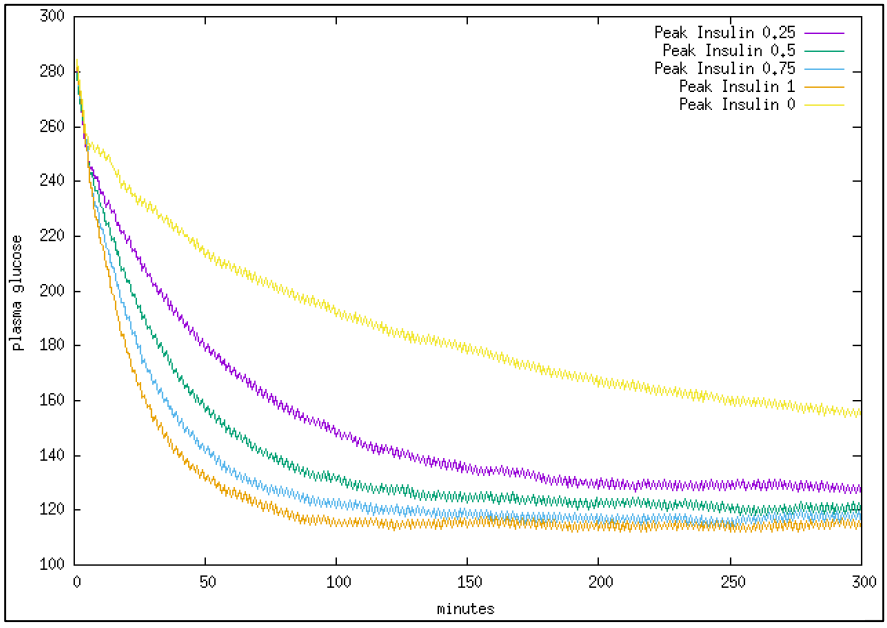

Figure 28: Blood Glucose Level - Insulin Resistance 1

Simulations - Liver Glycogen

Figures 29 though 33 illustrate the change in liver glycogen levels in response to high plasma glucose concentration in individuals with varying extents of Type 1 diabetes. In healthy humans, when the plasma glucose concentration is high, the pancreas generate insulin to prompt the organs - primarily the liver, muscle and kidney - to absorb excess glucose from the bloodstream. In the liver, most of this glucose is stored locally as glycogen. When a person suffers from diabetes, the ability of the pancreas to produce the required insulin is impaired. This in turn impacts the response of organs to high plasma glucose concentrations.

In figures 24 through 28, we saw that the blood glucose level stabilizes with time. The liver organ plays an important role in this process. When the plasma concentration of glucose is high, the liver organ responds to the corresponding high insulin level, to absorb excess glucose from the blood and store it locally as glycogen. Figures 29 through 33 verifies that the GS simulator correctly mimics this process. In each graph, we see that when the individual is capable of producing insulin the glycogen levels in the liver increases initially. This models the uptake of glucose - by the liver - in response to the high plasma glucose concentration. Once the plasma glucose levels have stabilized, glycogen breakdown occurs in order to support the energy requirements of various organs and maintain normal glucose levels.

Additionally, the GS simulator models the effect of insulin on glucose metabolism in the liver. As discussed in this documentation, glycogen breakdown is restricted by insulin while glycogen synthesis is stimulated by insulin. Similarly, glycolysis is stimulated and gluconeogenesis is restricted by the presence of insulin. When insulin is not present (or present in reduced amounts)

in the body, this expected response for each of these processes is affected. For example, when insulin peak is 0, glycogen break occurs in an unrestricted manner irrespective of plasma glucose concentration. Similarly, glycogen synthesis does not occur. As a result, the liver glycogen in the liver does not increase in response to high plasma glucose concentrations. The curve representing Insulin Peak 0 in figure 29 illustrates this scenario. As the extent of Type 1 diabetes increases, i.e. the value of insulin peak decreases, we see that the initial rise in liver glycogen levels in response to high plasma glucose concentration is less rapid. This in turn widens the time taken for blood glucose level to stabilize. Once the blood glucose level stabilizes, glycogen breakdown occurs to maintain the required plasma glucose level.

1. INSULIN RESISTANCE 0

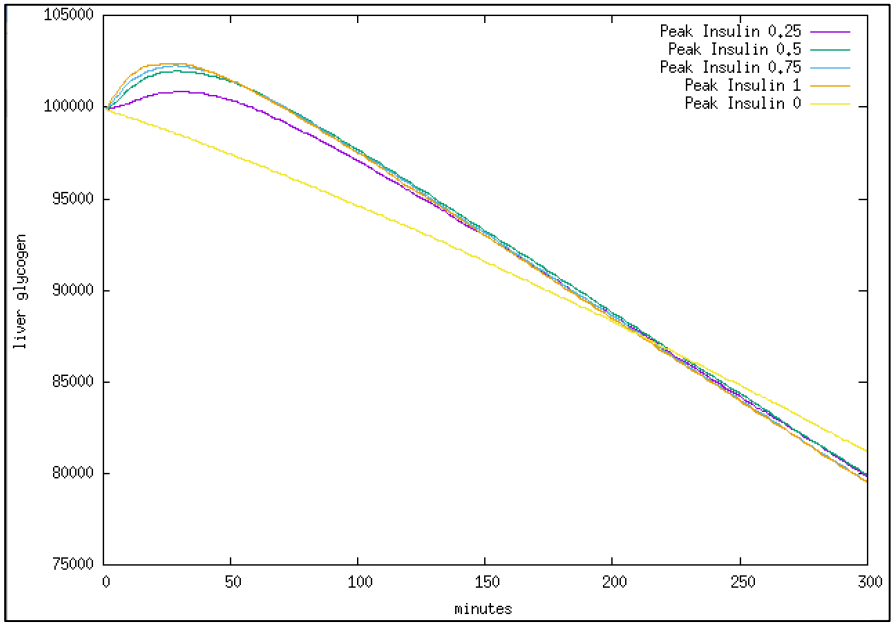

Figure 29: Liver - Insulin Resistance 0

2. INSULIN RESISTANCE 0.25

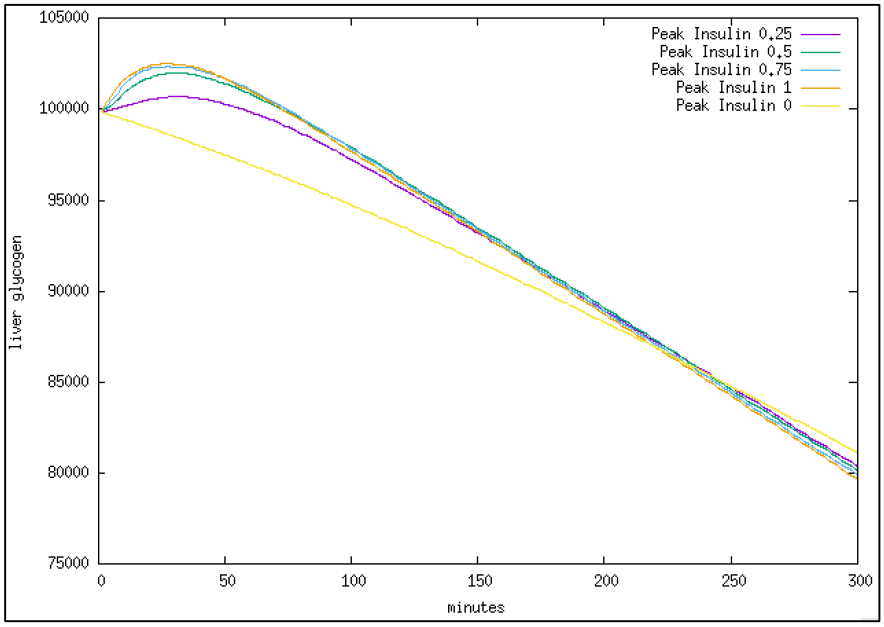

Figure 30: Liver - Insulin Resistance 0.25

3. INSULIN RESISTANCE 0.5

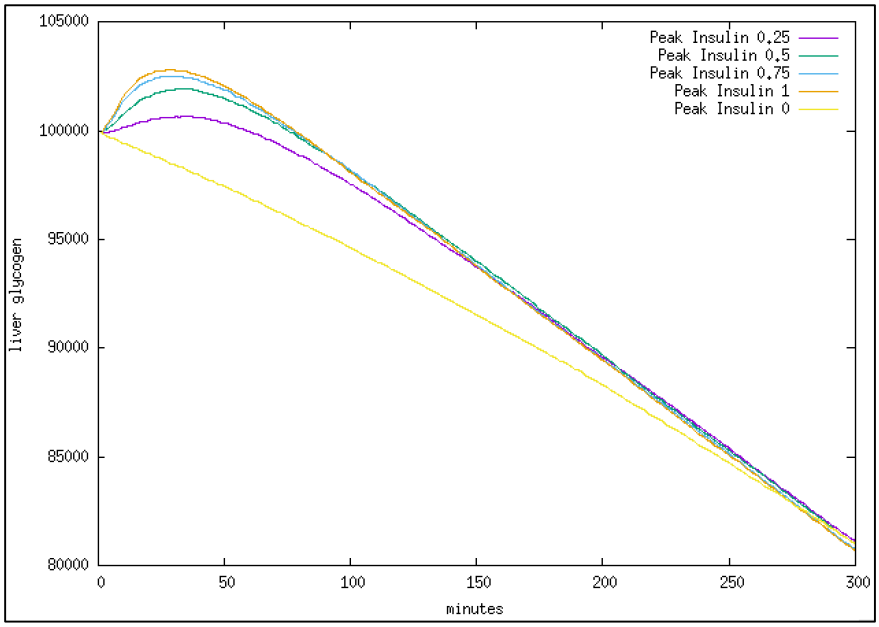

Figure 31: Liver - Insulin Resistance 0.5

4. INSULIN RESISTANCE 0.75

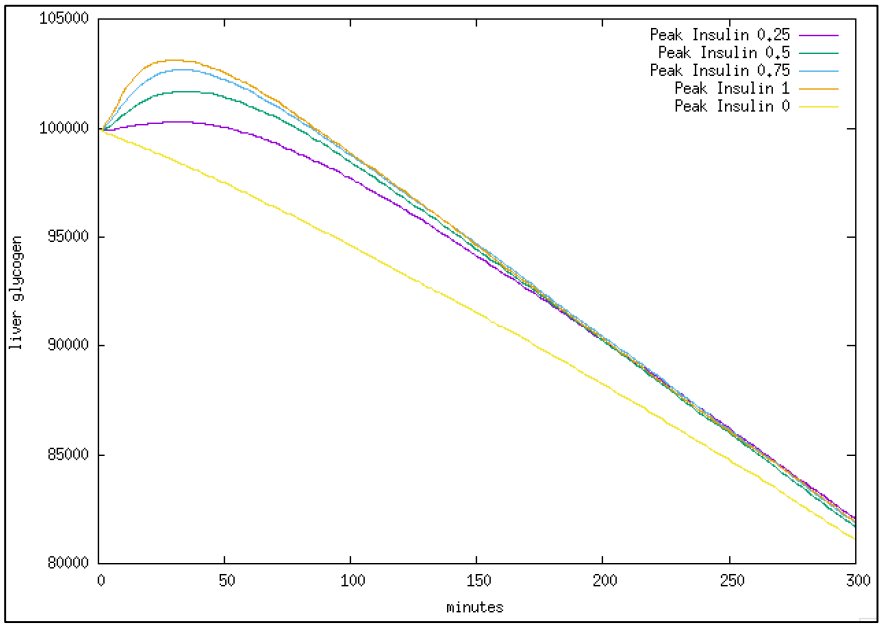

Figure 32: Liver - Insulin Resistance 0.75

5. INSULIN RESISTANCE 1

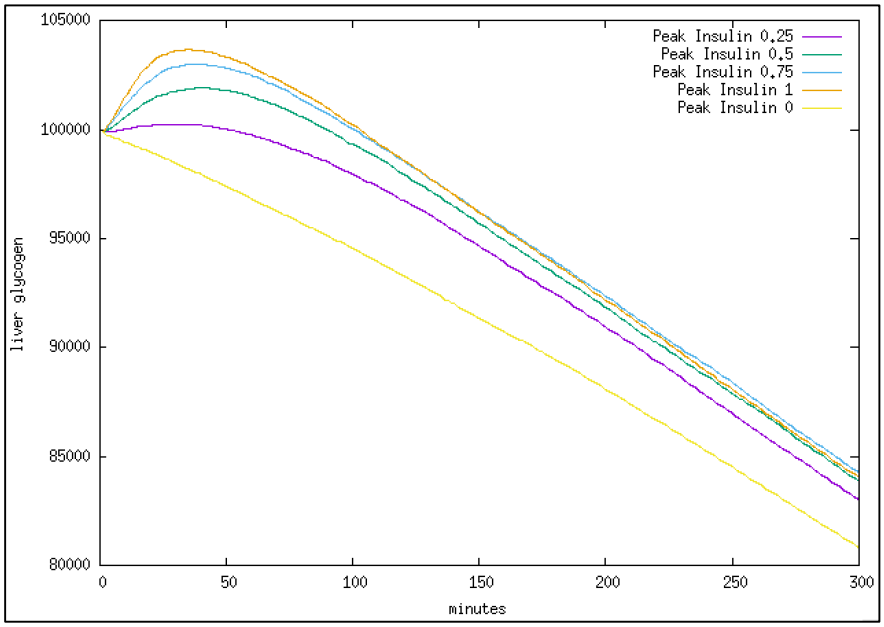

Figure 33: Liver - Insulin Resistance 1

Simulations - Muscle Glycogen

Figures 34 though 38 illustrate the change in muscle glycogen levels in response to high plasma glucose concentration in individuals with varying extents of Type 1 diabetes. As explained above, when a person suffers from Type 1 diabetes insulin sensitive processes in various organs are affected. In the muscle, glucose absorption - which typically increases in response to increased insulin levels in the bloodstream when plasma glucose concentration is high - is affected. The GS simulator models this insulin dependency as expected. For each level of insulin resistance, we see that the amount of glucose absorbed from the bloodstream (and stored as glycogen in the muscle) in response to high glucose concentration reduces as the extent of Type 1 diabetes increases. This in turn widens the time taken for blood glucose level to stabilize. Once the blood glucose level stabilizes, glycogen levels decrease as a result of glycolysis.

1. INSULIN RESISTANCE 0

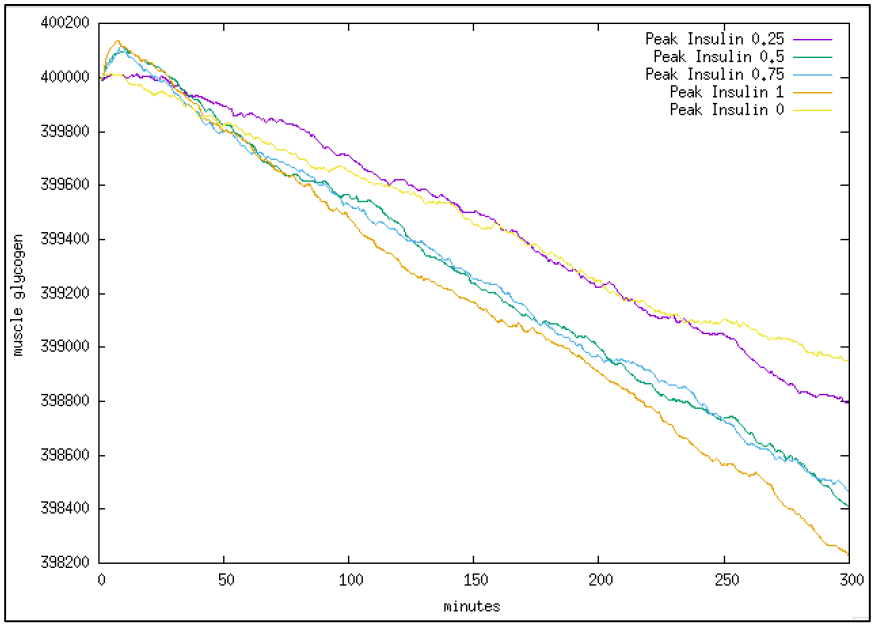

Figure 34: Muscle - Insulin Resistance 0

2. INSULIN RESISTANCE 0.25

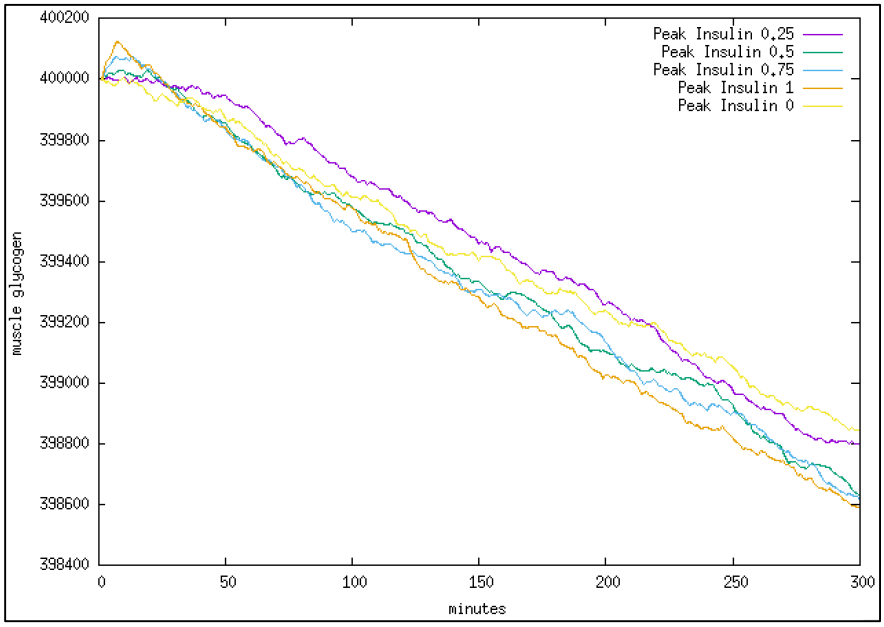

Figure 35: Muscle - Insulin Resistance 0.25

3. INSULIN RESISTANCE 0.5

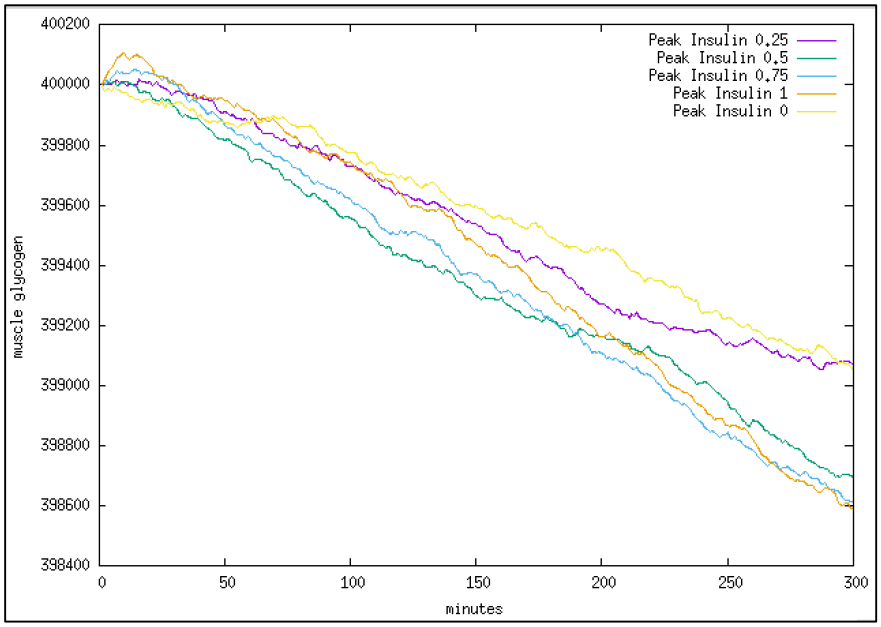

Figure 36: Muscle - Insulin Resistance 0.5

4. INSULIN RESISTANCE 0.75

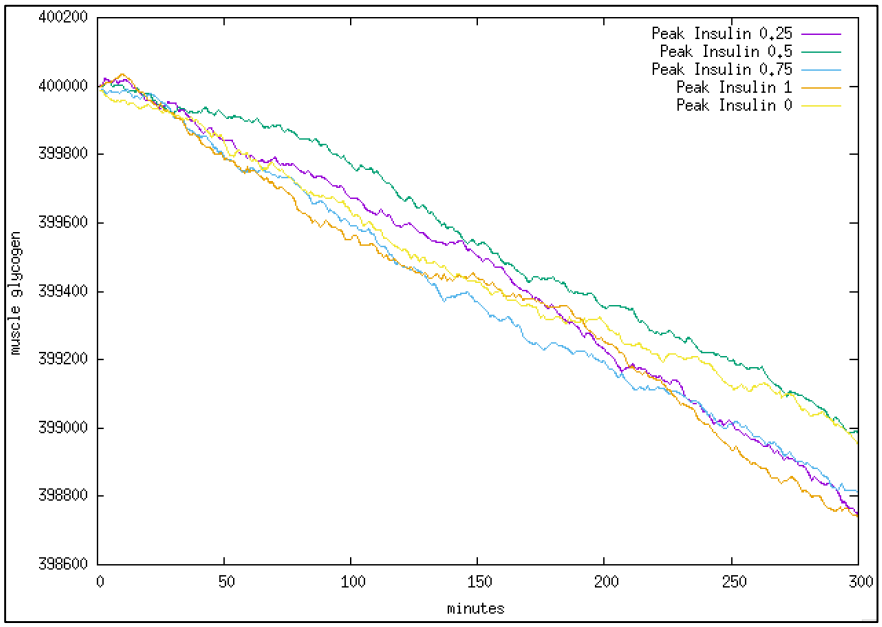

Figure 37: Muscle - Insulin Resistance 0.75

5. INSULIN RESISTANCE 1

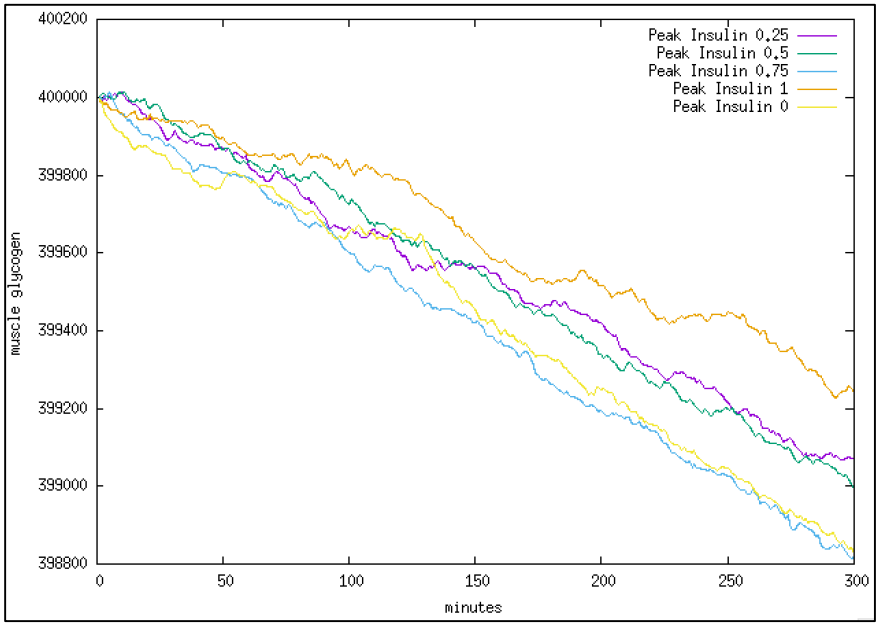

Figure 38: Muscle - Insulin Resistance 1

CHAPTER 4

CONCLUSION

This research is introducing and implementing a discrete event simulator for glucose metabolism in human beings. The simulator was designed and implemented using an object oriented approach, with different classes representing the human body and its organs. The simulator models actual biochemical reactions related to glucose metabolism occurring in the human body, and can be configured to handle different metabolic states. Since the GS simulator is modeled to simulate the variation in plasma glucose level as the human body goes through a sequence of diet and exercise activities, this simulator can be used to implement a diabetes self-management tool that will allow the user to achieve good control over the plasma glucose level.

The proposed simulator was verified for basic metabolic pathways by simulating the response to variation in plasma glucose concentrations in healthy and diabetic individuals. In healthy individuals, the GS simulator accurately demonstrated how the normal glucose homeostasis is maintained, irrespective of initial blood glucose concentrations. The relationship between plasma glucose and insulin was also simulated and verified for healthy individuals and individuals suffering from different degrees of Type 1 and Type 2 diabetes.

FUTURE ENHANCEMENTS

Exercise affects normal (at rest) glucose metabolism in a variety of ways. The GS simulator is currently designed to model only aerobic exercise. In order to increase its applicability, subsequent versions of the simulator can be designed to handle anaerobic exercise events as well. Additionally, further research can be conducted in order to model more metabolic pathways including fat and protein metabolism.

REFERENCES

[1] K. N. Frayn, Metabolic regulation: a human perspective, John Wiley & Sons, 2009.

[2] M. Z. Shrayyef and J. E. Gerich, "Normal glucose homeostasis," in *Principles of diabetes mellitus*, Springer, 2010, pp. 19-35.

[3] C. Meyer, J. M. Dostou, S. L. Welle and J. E. Gerich, "Role of human liver, kidney, and skeletal muscle in postprandial glucose homeostasis," *American Journal of Physiology-Endocrinology And Metabolism,* vol. 282, no. 2, pp. E419-E427, 2002.

[4] H. J. Woerle, C. Meyer, J. M. Dostou, N. R. Gosmanov, N. Islam, E. Popa, S. D. Wittlin, S. L. Welle and J. E. Gerich, "Pathways for glucose disposal after meal ingestion in humans," *American Journal of Physiology-Endocrinology and Metabolism,* vol. 284, no. 4, pp. E716-E725, 2003.

[5] D. Kelley, A. Mitrakou, H. Marsh, F. Schwenk, J. Benn, G. Sonnenberg, M. Arcangeli, T. Aoki, J. Sorensen and M. Berger, "Skeletal muscle glycolysis, oxidation, and storage of an oral glucose load," *Journal of Clinical Investigation,* vol. 81, no. 5, p. 1563, 1988.

[6] B. Capaldo, A. Gastaldelli, S. Antoniello, M. Auletta, F. Pardo, D. Ciociaro, R. Guida, E. Ferrannini and L. Sacca, "Splanchnic and leg substrate exchange after ingestion of a natural mixed meal in humans.," *Diabetes,* vol. 48, no. 5, pp. 958-966, 1999.

[7] P. J. Hanson and D. S. Parsons, "The interrelationship between glutamine and alanine in the intestine," *Biochemical Society Transactions,* vol. 8, no. 5, pp. 506-509, 1980.

[8] J. Gerich, "Role of the kidney in normal glucose homeostasis and in the hyperglycaemia of diabetes mellitus: therapeutic implications," *Diabetic Medicine,* vol. 27, no. 2, pp. 136-142, 2010.

[9] M. Stumvoll, G. Perriello, C. Meyer and J. Gerich, "Role of glutamine in human carbohydrate metabolism in kidney and other tissues," *Kidney international,* vol. 55, no. 3, pp. 778-792, 1999.

[10] R. Bergman, G. Toffolo, C. Bowden and C. Cobelli, "Minimal modeling, partition analysis, and identification of glucose disposal in animals and man," *IEEE Trans Biom Eng,* vol. 12935, 1980.

[11] A. Roy and R. S. Parker, "Dynamic modeling of exercise effects on plasma glucose and insulin levels," *Journal of Diabetes Science and Technology,* vol. 1, no. 3, pp. 338-347, 2007.

[12] A. Bock, G. François and D. Gillet, "A minimal exercise extension for models of the glucoregulatory system," in *21st European Symposium on Computer Aided Process Engineering-ESCAPE 21*, Elsevier, 2011, pp. 1520-1524.

[13] X.-Q. Xia and M. J. Wise, "DiMSim: A discrete-event simulator of metabolic networks," *Journal of chemical information and computer sciences,* vol. 43, no. 3, pp. 1011-1019, 2003.

[14] P. A. Meric and M. J. Wise, "Quantitative, scalable discrete–event simulation of metabolic pathways," in *ISMB*, 1999, pp. 187-194.

[15] G. Brooks, T. Fahey and K. Baldwin, "Glycogenolysis and glycolysis in muscle: the cellular

degradation of sugar and carbohydrate to pyruvate and lactate," *Exercicse Physiology: Human Bioenergetics and Its Applications,* pp. 59-65, 2005.

[16] T. E. Jensen and E. A. Richter, "Regulation of glucose and glycogen metabolism during and after exercise," *The Journal of physiology,* vol. 590, no. 5, pp. 1069-1076, 2012.

[17] J. Romijn, E. Coyle, L. Sidossis, A. Gastaldelli, J. Horowitz, E. Endert and R. Wolfe, "Regulation of endogenous fat and carbohydrate metabolism in relation to exercise intensity and duration," *American Journal of Physiology-Endocrinology And Metabolism,* vol. 265, no. 3, pp. E380-E391, 1993.

[18] G. Collier, A. McLean and K. O'dea, "Effect of co-ingestion of fat on the metabolic responses to slowly and rapidly absorbed carbohydrates," *Diabetologia,* vol. 26, no. 1, pp. 50-54, 1984.

[19] E. Moghaddam, J. A. Vogt and T. M. Wolever, "The effects of fat and protein on glycemic responses in nondiabetic humans vary with waist circumference, fasting plasma insulin, and dietary fiber intake," *The Journal of nutrition,* vol. 136, no. 10, pp. 2506-2511, 2006.

[20] K. N. Englyst, H. N. Englyst, G. J. Hudson, T. J. Cole and J. H. Cummings, "Rapidly available glucose in foods: an in vitro measurement that reflects the glycemic response," *The American journal of clinical nutrition,* vol. 69, no. 3, pp. 448-454, 1999.

[21] B. Owen and T. M. Wolever, "Effect of fat on glycaemic responses in normal subjects: a dose-response study," *Nutrition research,* vol. 23, no. 10, pp. 1341-1347, 2003.

[22] J. T. Gonzalez, C. J. Fuchs, J. A. Betts and L. J. van Loon, "Liver glycogen metabolism during and after prolonged endurance-type exercise," *American Journal of Physiology-Endocrinology and Metabolism,* vol. 311, no. 3, pp. E543-E553, 2016.

[23] F. K, Metabolic Regulation: A human perspective.

[24] Brooks, "Glycogenolysis and Glycolysis in Muscle. The Cellular Degradation of Sugar and Carbohydrate to Pyruvate and Lactate".

[25] T. E. Jensen and E. A. Richter, "Regulation of glucose and glycogen metabolism during and after exercise," *The Journal of Physiology.*

[26] J. A. ROMIJN, E. F. COYLE, L. S. SIDOSSIS, A. GASTALDELLI, J. F. HOROWITZ, E. ENDERT and R. R. WOLFE, "Regulation of endogenous fat and carbohydrate metabolism in relation to exercise intensity and duration".

[27] R. Bergman, G. Toffolo, C. Bowden and C. Cobelli, "Minimal modeling, partition analysis, and identification of glucose disposal in animals and man," *IEEE Trans Biom Eng,* 1980.

[28] A. Roy and R. S. Parker, "Dynamic Modeling of Exercise Effects on Plasma Glucose and Insulin Levels," *Journal of Diabetes Science and Technology,* 2007.

[29] A. Bocka, G. Françoisa, T. Prud'hommeb and D. Gilleta, "A Minimal Exercise Extension for Models of the Glucoregulatory System," *21st European Symposium on Computer Aided Process Engineering-ESCAPE 21.*

[30] X.-Q. Xia and M. J. Wise, "DiMSim: A Discrete-Event Simulator of Metabolic Networks," *Journal of chemical information and computer sciences.*

[31] P. A. Meric and M. J. Wise, "Quantitative, scalable discrete–event simulation of metabolic

pathways," *ISMB,* 1999.

[32] K. N. Englyst, H. N. Englyst, G. J. Hudson, T. J. Cole and J. H. Cummings, "Rapidly available glucose in foods: an in vitro measurement that reflects the glycemic response," *The American Journal of Clinical Nutrition,* 1999.

[33] G. Collier, A. McLean and K. O'dea, "Effect of co-ingestion of fat on the metabolic responses to SAG and RAG," *Diabetologia,* 1984.

[34] E. Moghaddam, J. Vogt and T. Wolever, "The effects of fat and protein on glycemic responses in nondiabetic humans vary with waist circumference, fasting plasma insulin, and dietary fiber intake.," *The Journal of nutrition ,* 2006.

[35] B. Owen and T. M. Wolever, "Effect of fat on glycaemic responses in normal subjects: a dose-response study.," *Nutrition research,* 2003.

[36] Gerich.

[37] J. T. Gonzalez, C. J. Fuchs, J. A. Betts and L. J. van Loon, "Liver glycogen metabolism during and after prolonged endurance-type exercise," *American Journal of Physiology-Endocrinology and Metabolism,* vol. 311, 2016.