Analysing glucose regulation *in silico*: Modeling insulin secretion via PID control

Jean Janssen
Department of Advanced Computing Sciences
Maastricht University

Abstract

Blood glucose levels in the human body are maintained through a network of various hormones and peptides that regulate complex mechanisms including the secretion of insulin and the disposal of glucose to muscle tissue.
In this thesis, we explore PID control to model insulin secretion in a physiology-based computational model in the Julia Programming Language. Furthermore, we explore the possibility of estimating digital twins (personalized computational models) of an individual's glucose insulin-regulation using the least squares parameter estimation technique in the Julia Programming language.

Index Terms: Ordinary differential equations, PID control, least squares parameter estimation

1 Introduction

In order to ensure normal body function, the human body is dependent on a tight control of its blood glucose levels (Röder, Wu, Liu, & Han, 2016). The body accomplishes this through a network of various hormones and peptides released mainly from the brain, liver, intestine as well as adipose and muscle tissue. The pancreas is a key player within this network as it secretes the blood sugar-lowering hormone insulin. However, disturbances in the interplay of the hormones and peptides may lead to metabolic disorders such as type 2 diabetes mellitus (T2DM). T2DM is a serious condition that can be life-threatening to patients. The human body is highly dependant on glycemic control and disturbance of this control can lead to T2DM. The condition is also very costly, both in medical costs and in lost work hours (A. H. Maas et al., 2015). Research shows that the incidence of diabetes can be reduced and is directly associated with changes in lifestyle (Tuomilehto et al., 2001). Therefore, it is important to create an understanding in the glycemic control of patients to develop new therapeutic approaches.

When a meal containing carbohydrates is consumed, ingestion will cause glood glucose levels to rise. In a healthy person, insulin in secreted in response to these elevated glucose levels which allows the uptake of glucose from the plasma into, for example, muscle tissue where it is used as energy. Over the year, due to a combination of ageing and lifestyle, insulin resistance develops. This causes a decreased ability of insulin production and circulation in the body. Factors such as increasing insulin resistance, and possibly later, a decline in insulin secretion can lead to the development of T2DM (Erdos et al., 2021). The dynamic and inter-related nature of the insulin mediated glucose regulation necessitates a systems approach to comprehensively characterize.

ODE-models such as the Meal Simulation Model of the Glucose-Insulin System (Man, Rizza, & Cobelli, 2007) and the Eindhoven-Diabetes Education Simulator (E-DES) (Erdos et al., 2021) are able to simulate the glucose and insulin interaction before and after a meal. The latter has previously been successfully personalized on real data from a large number of individuals (Erdos et al., 2021). This model can simulate the glycemic control from a non-invasive Oral Glucose Tolerance Test (OGTT), a standard test that measures the body's ability to metabolize glucose. A person's ability to uptake glucose from the gut into the plasma is correlated with the secretion of insulin in the pancreas. This secretion is modelled as a Proportional Integral Derivative (PID) controller in the E-DES model. A PID controller is a controller that includes elements with those 3 functions. A PID controller calculates an error value $e = r - y$ continuously and its purpose of control is to make the process variable y follow the set-point variable r and a correction is applied using the P, I and D elements (Araki, 2009). This way, the 3 elements of the PID controller produce outputs in the following way:

1. P element: proportional to the error at the instant t, which is the "present" error.
2. I element: proportional to the integral of the error up to the instant t, which can be interpreted as the accumulation of the "past" error.
3. D element: proportional to the derivative of the error at the instant t, which can be interpreted as the the prediction of the "future" element.

Therefore, the PID controller can be understood as a controller that takes the present, the past and the future of the error into consideration (Araki, 2009). As mentioned previously, the E-DES, is a glucose-insulin model that uses PID control for the secretion of insulin from the pancreas. The downside of this model, however, is that it was implemented in the licensed software MATLAB which limits the user base to only those who have access to licensed software. In previous studies, using the Julia programming language, the model was implemented in Julia: Julia is a MATLAB alternative that is open source and offer similar functionality as MATLAB. This is a huge step forward as this allows for open source contribution to the model, hopefully leading to improvements in the software that allow new insights and a better understanding towards the interaction of glucose and insulin in the human body. The implementation in Julia, however, differs from the original model as it only utilizes a Proportional Derivative (PD) controller instead of a Proportional Integral Derivative Controller (PID) due to a lack of functionality. Therefore, we will develop and implement this missing functionality. In order for the Julia model to be a viable and complete, open source alternative to MATLAB, having the integral term is a must. Furthermore, the integral term is proportional to the magnitude and the duration of the error and therefore allows further reduction of the error, thus improving the model. With use of the E-DES in Julia, parameter estimation can be performed to personalize the model on glucose and insulin time-series data from an OGTT using Least Squares estimation. With the personalized model parameter estimates, further conclusion can be drawn about the impact of dietary interventions in real patient data.

1.1 Research Aims

In this thesis, we aim to further extend the Eindhoven-Diabetes Education Simulator (E-DES) to provide a faster, open source model that researchers and clinicians can adopt to quantify the insulin mediated glucose control. In addition, we will attempt to use the model to create digital twins of a individuals' glucose homeostasis. We aim to achieve the following through this research:

1. Move from PD to PID control by implementing integral control into the model in Julia
2. Use the model to accurately simulate the glucose-insulin evolution for pre-specified meal
3. Determine significant changes in the model parameters due to dietary intervention through personalization of this model

2 Related Work

2.1 Eindhoven-Diabetes Education Simulator (E-DES)

E-DES is a physiology-based mathematical model that consist of 4 compartments for which the inflow and outflow of glucose and insulin are calculated using 6 nonlinear coupled differential equations and 14 parameters (Fig 1). The 4 compartments include, the gut, the plasma, the interstitial fluid and the subcutaneous tissue.

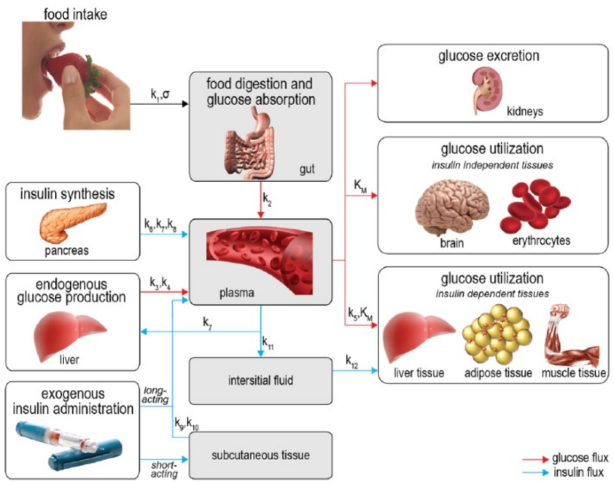

Figure 1: 4-compartment E-DES model

A later version of the Eindhoven-Diabetes Education Simulator has reduced the 4 compartments to 2. These 2 compartments are the plasma and the gut. Given a specific meal with an amount on glucose in milligrams, a set of 4 equations simulate the evolution of the glucose and insulin values. These 4 equations calculate the 4 state variables.

1. ${M_G}^{gut}(t)$: Glucose mass in the gut (mg)
2. $G^{pl}(t)$: Plasma glucose concentration (mmol/L)
3. $I^{pl}(t)$: Plasma insulin concentration (mU/L)
4. $G^{int}(t)$: Value of integrator function

Glucose mass is emptied into the gut according to an exponential decay function. This is followed by the uptake into the plasma proportionally to the amount of glucose present in the gut. Insulin and glucose fluxes are both taken into account in the plasma compartment. The amount of glucose present in the plasma will affect the insulin secretion into the blood from the pancreas. This secretion from the pancreas is modelled through the use of a proportional-integral-derivative (PID) controller (A. H. Maas et al., 2015). The goal of the PID control is to drive the glucose levels back to the basal glucose values through secreting insulin. The basal glucose value is the set-point defined as the fasted glucose concentration. Thus, the PID driving the glucose levels back to the fasting state is a negative feedback loop.

Insulin is not required for some tissues in order to take up glucose, this causes a constant take out of glucose from the plasma. Added to this, insulin is cleared by the liver proportionally to plasma insulin concentration, as well as transfer and degradation in the interstitial fluid. The model has kinetic parameters for each of these physiological processes that modify the rate of changes in glucose and insulin levels over time. Modulation of the parameters is required to simulate responses of different phenotypes or individuals *in silico*. The model may be personalized through estimating a subset of the model from data (Erdos et al., 2021). These parameters were previously identified as appropriate to personalize the model.

1. $k1$: Rate constant of glucose appearance in the gut (l/min)
2. $k5$: Rate constant of insulin-dependent glucose uptake (l/min)
3. $k6$: Rate constant of Δ G-dependant insulin production (l/min)
4. $k8$: Rate constant of dG/dt-dependant insulin production (l/min)

Estimating these parameters from data while fixing the rest to population averages allows the model to be tuned to the particular individual.

The E-DES model is implemented in MATLAB

2018b, but has previously been exported to Julia 1.6. For clarity, I will refer to the Julia implementation of the E-DES model in Julia as *"the Julia model"* and similarly, the E-DES model implemented in MATLAB will be referred to as *"the MATLAB model"*. More details about the model can be found in (Erdos et al., 2021) and the appendix.

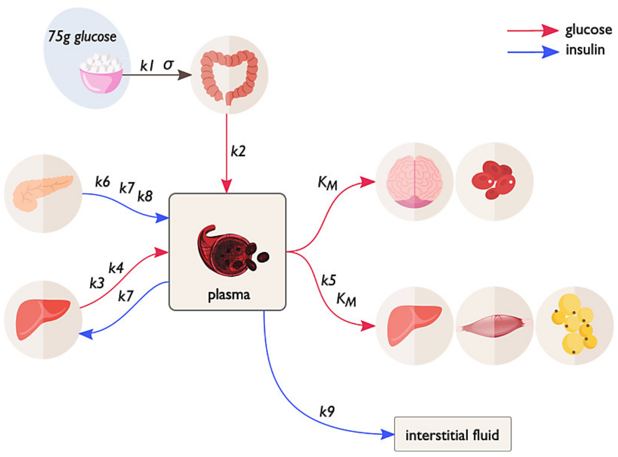

Figure 2: 2-compartment E-DES model (Erdos et al., 2021)

3 E-DES Model in Julia

The E-DES model was recently ported from MATLAB to Julia. It promises an efficient, fast and open-source alternative to the MATLAB implementation, however it is incomplete. The model only contains PD control of glucose via insulin and is missing the integral control.

3.1 Modelling the integral term in PID control

The E-DES model utilises a Proportional Integral Derivative (PID) controller to model the secretion of insulin from the pancreas, as mentioned previously. The proportional term measures the current glucose level, the integral term considers the sum of the glucose level over time. Lastly, the derivative term acts on the rate of change. It will predict the necessity of more insulin if the rate of glucose change is high. In previous studies the PID controller was substituted by a Proportional Derivative (PD) controller, within this study we will implement the integral term to improve the model. The integral term introduces a memory of the system. Together with the P and D term it acts via secreting insulin to reduce glucose. In order to implement the integral, as mentioned previously, we need to keep a memory of the system. This is achieved by storing the glucose solution at every previous time step and make it accessible to the ODE solver at the current time step. In order to achieve this we introduce *"t_saved"* and *"G^{pl}_saved"* which are both arrays where we store the previous time steps and the corresponding glucose values. Furthermore, we set a *"t_lowerbound"* as well as a *"G^{pl}_lowerbound"* because we assume that the integral control only kicks in after 30 minutes (*"t_integralwindow"*). *"t_lowerbound"* gets calculated by the equation:

$$t\_lowerbound = t - t\_integralwindow \tag{1}$$

Furthermore, we introduce *"G^{pl}_lowerbound"*. Which is the glucose value at *"t_lowerbound"*. We might not have a solution at time *"t_lowerbound"*, therefore we need to approximate this value through interpolation. The current glucose in the plasma is given by the term G^{pl} and the basal glucose value at time $t = 0$ measured during an OGTT is ${G_b}^{pl}$. Furthermore, k_6 is the rate constant of Δ G-dependant insulin production (l/min). k_7 is the rate constant of $\int$G-dependant insulin production (l/min). k_8 is the rate constant of dG/dt-dependant insulin production (l/min). The PID controller in the model is given by:

$$\beta^{-1}(k_6(G^{pl} - {G_b}^{pl}) + (\frac{k_7}{\tau_i})\int(G^{pl} - {G_b}^{pl})dt + (\frac{k_7}{\tau_i}){G_b}^{pl} + (k_8\tau_d)\frac{dG^{pl}}{dt}) \tag{2}$$

Where β is the unit conversion factor from glucose to insulin and τ_d is the derivative time constant in minutes. τ_i is the integration time constant.

3.2 Implementing PID control using callbacks and events

In Julia callbacks make use of the integrator interface and within this study it has been used to implement the integral term of the PID in the Julia model. In Julia, callbacks consists of a *condition* and an *affect* function. Once the condition is met the affect function will handle these events by modifying the parameters specified within the function itself. For this model an integral window of $t_{integral} = 30$ has been predefined. This integral window is the delay from t=0 at which we start integrating within the model. Once the condition/event $t > 30$ holds, we want this callback function to be evaluated for every time step t. The callback function has the following structure:

Algorithm 1 Periodic Callback

```
procedure Periodic Callback
    $gplsaved \leftarrow G^{pl}$
    $t\_saved \leftarrow t$
    if $t > t\_integralwindow$ then
        $t\_lowerbound = t - t\_integralwindow$
        $intpl = interpolate(G^{pl}\_saved)$
        $G^{pl}\_lowerbound = intpl$
    else $G^{pl}\_lowerbound = {G_b}^{pl}$
    end if
end procedure
```

The interpolation used in this algorithm is Spline Interpolation with cubic spline. We use interpolation to get an approximation of the *"G^{pl}_lowerbound"* at time *"t_lowerbound"* this approximation is *"intpl"*. This is necessary because, depending on the steps of the solver or the saving times of the callback function, we might not have an actual solution at time-point *"t_lowerbound"*.

4 Parameter Estimation

"Parameter estimation is a disciple that provides tools for the efficient use of data in the estimation of constants appearing in mathematical models and for aiding modelling in phenomena." (Beck & Arnold, 1977). Parameter estimation can be seen as a study of inverse problems. In the solution of differential equations one classically seeks a solution in a domain knowing the boundary and initial conditions and any constants. In the inverse problems not all these constants would be known (Beck & Arnold, 1977). In this study, a mathematical model of a dynamic process involving ordinary differential equations is used. "A canonical estimation problem is the fitting of a function, defined be a collection of parameters, to a data set." (Aster, Borchers, & Thurber, 2018). In this case, the fitting procedure cannot be done as a linear inverse problem. However, a nonlinear inverse problem can be solved by means of Least Squares Method or Bayesian Inference. Within this study, the Least Squares method will be used to personalize the E-DES by fitting the model to human data.

4.1 Parameter estimation in E-DES

Using the E-DES, personalization can be performed in order to provide insight to the glucose and insulin regulation within a person's body. One of the most common practices to estimate a subset of parameters in a mathematical model is by means of the Least Squares Method. In this case, we estimate the k-parameters introduced in section 2.1 and we keep the rest of the parameters fied to population average values found in the literature (A. Maas, 2017). The MATLAB model makes use of the package "lsqnonlin", a non-linear least squares solver. In the Julia model, we use the "DiffEqFlux" package to perform the optimization (Rackauckas et al., 2019). Optimization using "DiffEqFlux" is done through the method "sciml_train" which trains the model through use of optimization algorithms such as ADAM, BFGS or Newton's method. For the parameter estimation within the E-DES model, we need to make sure a postive domain is maintained. The model needs to ensure a positive domain, non-negativity of the variables at time t_0 has to be ensured such that non-negativity is ensured at all times $t \geq t_0$ for which the solution is defined. Therefore we need to set an upper and lower bound to the k-parameters that will be optimized. Because of the requirement for box-constraint we can only use optimization algorithms that allow for these constraint. BFGS is a suited candidate for this reason. Furthermore, the "DiffEqFlux" package, unlike "lsqnonlin", does not allow a subset of the parameters defined in the model to be optimized. This required us to remove all generalizability to be removed from the model. This means that functions outside of the model, such as the callback function, have no access to the parameters outside of the k-parameters. Thus we cannot include the integral function in the optimization of the parameters. Further parameter estimation will be done through use of the E-DES model with the PD controller.

We need to calculate the loss between observed and computed values (Abdi et al., 2007). This is done through the sum of squared residuals (SSR) of the simulation at the 5 time-points $t = (0, 30, 60, 90, 120)$ where we have the corresponding measurements. The measurements of the glucose and insulin are measured using different units (mmol/L and mU/l, respectively). In order to account for this scaling was performed where for each person their maximum glucose measurement and insulin measurement respectively were used for the weight factors (Johnson & Faunt, 1992).

$$SSR = \sum_{j=1}^{m} \sum_{i=1}^{N} \left(\frac{y_{i,j}(\theta) - d_{i,j})}{\max(d_i)}\right)^2 \tag{3}$$

Where m is the number of metabolites (in this case glucose and insulin), N is the number of time-points for which measured data is available. $d_{i,j}$ is the measured metabolite concentration of metabolite i at time-point j. $y_{i,j}(\theta)$ denotes the model prediction for metabolite i at time-point j given the parameter vector θ. To account for the difference in scales due to the units of the metabolites the error is weighted by the maximum of the observed value per metabolite.

We optimize from a physiologically based initial guess of ($k_1 = 1.35e^{-2}, k_5 = 3.80e^{-3}, k_6 = 5.82e^{-1}, k_8 = 4.71$). Since our initial guess corresponds to the average response of the population of individuals of the data we use (Erdos et al., 2021), in most cases we can proceed with the BFGS optimization algorithm. The algorithm is run for a maximum of 15 seconds. This criterion was set to allow for a feasible run-time of the total population. Based on our experiments, further optimization does not improve the fit significantly enough to justify tweaking the optimizer settings further. Based on physiological measure we can set constraint for each of the 4 state parameters that we want to optimize. These bounds are:

1. $k1 = (1.35e^{-5}, 1.35e^{-1})$
2. $k5 = (3.80e^{-4}, 3.80e^{-2})$
3. $k6 = (5.82e^{-4}, 5.82)$
4. $k8 = (4.70e^{-3}, 47.00)$

4.2 BFGS

Let us go into a bit more detail as to how BFGS works.

$$p_k = -H_k \nabla f(x_k) \tag{4}$$

and

$$H_{k+1} = (I - \rho_k s_k (y_k)^T) H_k (I - \rho_k y_k (s_k)^T) + \rho_k s_k (s_k)^T H_{k+1} \tag{5}$$

$$s_k = x_{k+1} - x_k \tag{6}$$

$$y_k = \nabla f(x_{k+1}) - \nabla f(x_k) \tag{7}$$

$$\rho_k = \frac{1}{y_k^T s_k} \tag{8}$$

The function are expressed in iterative style where k is the iteration. p_k is the searching direction, s_k and y_k are the curvature conditions. We initialize the matrix H^1 with $H^1 = I$. BFGS does this because it is a quasi-Newton method. Therefore, H_k is the approximation of the inverse of the Hessian matrix H^{-1} which approximates the Hessian Matrix H. $\nabla f(x_k)$ is the gradient of the function at x_k (Wu, Wang, Zheng, & Chang, 2015). The complexity of BFGS is $O(n^2)$ (Fletcher, 2013).

In this thesis, we are going to estimate the responses of individuals to the OGTT of 7 hand selected examples that have been used in the MATLAB

Time-points (minutes)	t=0	t= 30	t=60	t=90	t=120
Glucose (mmol/L)	5.00	7.50	7.90	6.50	5.80
Insulin (mU/L)	8.58	55.70	71.20	58.10	47.70

Table 1: Sample Data

model, using the Julia model. We label these 7 individuals (a), (b), (c), (d), (e), (f) and (g) respectively. With these estimations we can compare the fits of the Julia and MATLAB model.
Then, we will estimate the individuals measured responses from the data described in section 5.

5 Data

Within this thesis, data of 3 sets of oral glucose tolerance tests (OGTT) was used from 738 individuals. The first set of OGTTs were performed before dietary intervention. The second set of OGTTs was taken after strict dietary intervention and, finally, the third set of OGTTs was taken after another more relaxed dietary intervention. In an OGTT the body's ability to metabolize sugar and clear it from the body is tested. With the use of a blood sample of the person, the baseline blood glucose and insulin levels are measured. Then, participants consume a concentrated glucose beverage which contains $75g$ of glucose. Throughout the span of the following 2 hours, blood samples are taken at time $t = [30, 60, 90, 120]$ where all the values represent minutes. Prior to optimizing, the data was reduced by removing those individuals that had missing values for the glucose or insulin measurement at $t = 0$. Furthermore, the data of an individual was removed if it had 2 or more measurement values missing from the time series $t = (0, 30, 60, 90, 120)$.

For each of the tests completed we have a 2 by 5 matrix with the numeric values of the glucose values in $mmol/L$ and the insulin values at mU/L with one row for the glucose and one for the insulin. The 5 columns represent the time points as mentioned previously. A sample data is shown in Table 1.

6 Results and Discussion

6.1 Verification of PID controller in the model

To verify a correct implementation of the PID controller (section 3.2) within the model a comparison between the calculation of the state variables was performed (Fig 3).

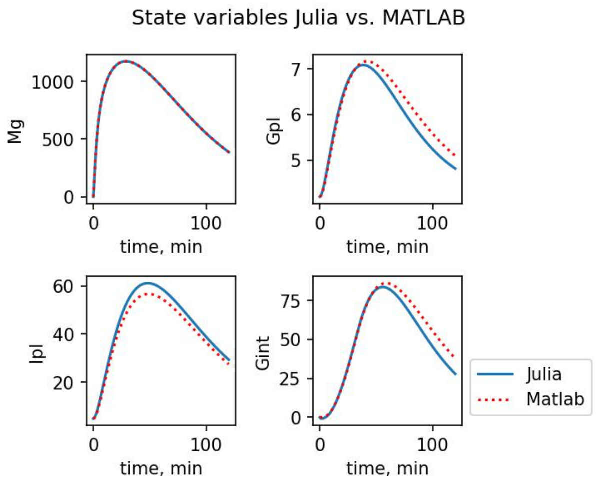

Figure 3: Simulated state variables in Julia vs. Matlab. Model states simulated after an oral bolus of 75g glucose.

We can see that the Matlab and Julia model provide the same curve behavior (Fig 3), the difference here is caused by the difference in implementation. MATLAB is utilizing an integrator function that saves at variable time-steps whereas the integrator function in Julia is periodic with 1 minute periods because we have set our *"t_integralwindow"* to exactly 30 minutes. Having an increment of 1 minute we can exactly determine when the integral function will start performing calculations.
For both models the ODE solver uses variable steps.

6.2 Comparison between PID and PD controller

The implementation of the integral function allows the model to secrete more insulin in time if the system does not reach the set-point. Within figure 4, we can see more insulin secretion due to the implemented integral term which, in turn, leads to lower plasma glucose concentrations. The difference here is

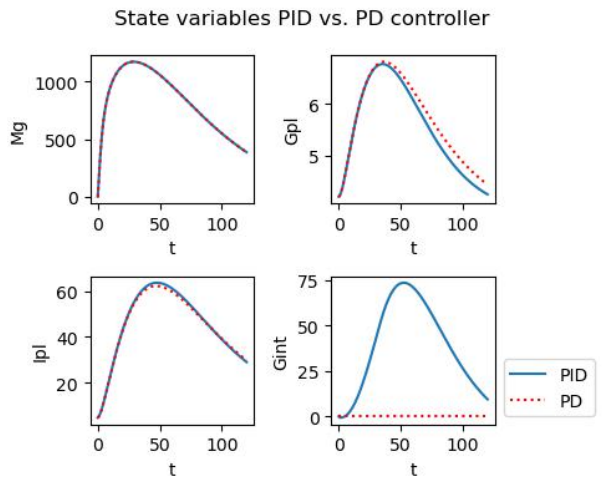

Figure 4: State variables PID vs. PD. Model states simulated after an oral bolus of 75g glucose.

bigger when comparing figure 3 than when comparing 4. The figures suggest that the difference between Julia and MATLAB has a larger effect on the state variables as compared to having either PD or PID control. However, the implementation of the PID controller allows for more flexibility in the model. In particular we hypothesize, that having the integrator function allows for different responses to be fitted. The simulations in figure 3 and 4 are for a theoretical average healthy individual where, in this case, the integral term makes little difference.

6.3 Least squares parameter estimation comparison

When we look at the model fits after parameter estimation in figure 5, we can see that for figure 5(a), (b), (c), (d), (e) and (g) we get a good fit although for scenarios (c), (d) and (e) there is a worse fit based on the parameters of the Julia model. Due to BFGS being highly sensitive to the initial condition, we hypothesize this to be the case. This may be remedied by introducing multi-start optimization randomly sampled around the initial condition, or first using a global search and then a local search algorithm to refine the solution Hypothetically, this causes a worse fit in some scenarios. Nonetheless, these fits provide a good baseline for estimating the parameters on real patient data.

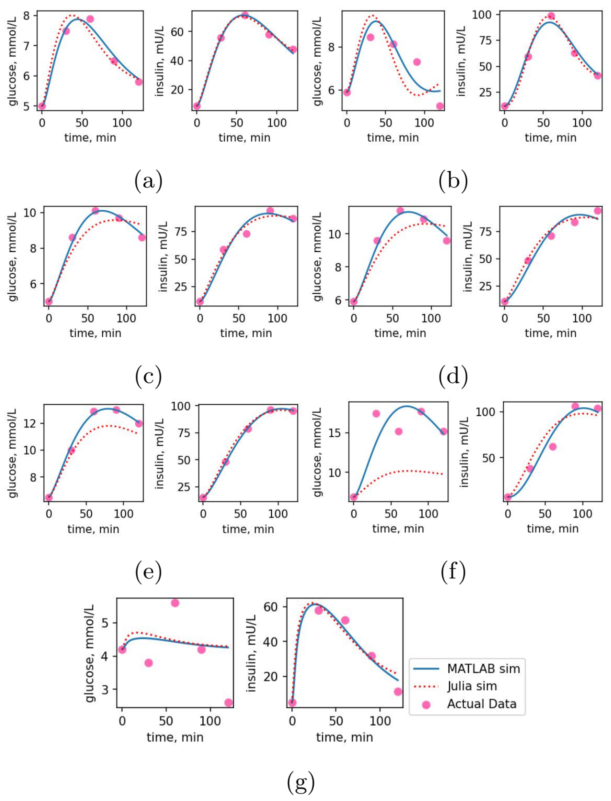

Figure 5: Comparison of model simulations using parameters estimated in Matlab and Julia for 7 hand selected examples. Continuous blue line and dotted red line corresponds to the Matlab and Julia simulations, respectively. Red dots represent the measured data.

6.4 Density of k-parameters over OGTTs

We take a look at the density curves of the k-parameters, that were optimized, (Fig 6) for each of the 3 OGTTs. As mentioned in section 5, the set of the first OGTTs was taken before intervention, the second after strict dietary intervention and the third set after a different dietary intervention. In figure 6(a) we can see that mean for *k1* in the set OGTT1 is the lowest. This suggest that, going from a state before to a state after a dietary intervention, there is, on average, a faster appearance rate of glucose in the gut. In figure 6(b), we can compare the densities of *k5*. What we can see here is that going from OGTT1 to OGTT2 there is, on average, a higher rate of insulin-dependent glucose uptake. However, from OGTT2 to OGTT3, there is a lesser rate of insulin-dependent glucose uptake on average. This suggests that after the strict dietary intervention from OGTT1 to OGTT2, individuals show more insulin sensitivity as there is a higher rate of glucose uptake. Now, for figure 6(c), we can see that from OGTT1 to OGTT2 there is less glucose dependent insulin production, but from OGTT1 and OGTT2 there is an average higher rate of glucose dependent insulin production in OGTT3. And also for *k8* in figure 6(d) we can see that overall from OGTT1 to OGGT2 and OGTT3 there is a higher dG/dt-dependant insulin production. These results suggest that after dietary intervention, insulin sensitivity in the human body can be increased. In addition, these results suggest that individuals produce more insulin after dietary intervention and also suggests that there is less insulin resistance for the individuals. This suggests that these developments can be treated with lifestyle and possible prevent the development of T2DM.

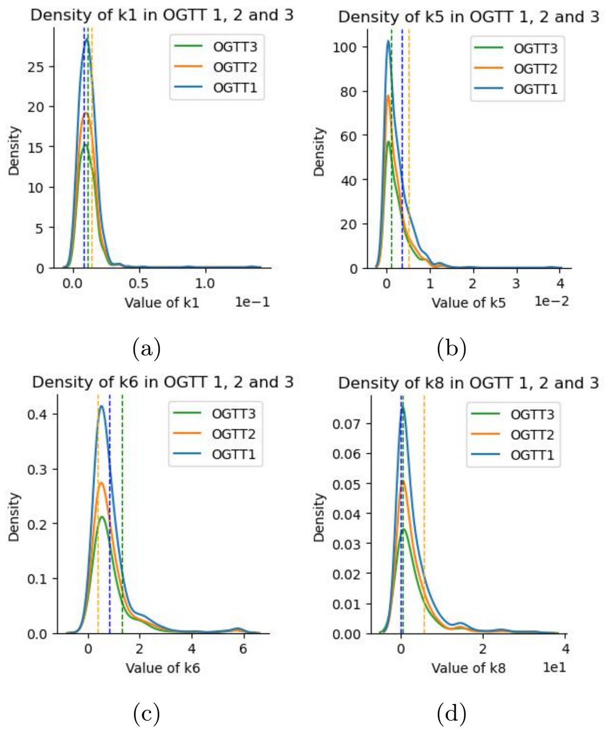

Figure 6: Comparison of the densities of the 4 k-parameters personalized on the OGGT data. The continuous blue, orange and green lines represent the density curves of the sets OGTT1, OGTT2 and OGTT3 respectively. Theblue, orange and green dashed lines represent the mean values of the sets OGTT1, OGTT2 and OGTT3 respectively.

7 Conclusion

To conclude, we have successfully extended the E-DES model with PID control in the E-DES model in Julia. The PID implementation of the model produces the appropriate behaviour with slight discrepancies in the states due to implementation differences. The implemented PID model allows for more flexibility and better fitting of the model in certain scenarios.
The Julia model still needs to be extended, since the current implementation only allows for simulation of the model and not personalizing it. Therefore, further investigation whether it is possible to include the integrator function in the process of personalizing the model through parameter estimation.

Using Least Squares parameter estimation method, we were able to produce comparable results with the MATLAB model. Through parameter estimation we generated 1689 personalized models representing individuals' glucose regulation throughout a dietary intervention study. Finally, using this model we were able to determine changes in the insulin glucose regulation of individuals after dietary intervention.
Within this thesis, we examined Least Squares using the BFGS optimization algorithm. Further research should be put into different optimization algorithms and approaches. Such as Newton's method with Trust Region and ADAM followed up by BFGS.

References

Abdi, H., et al. (2007). The method of least squares. *Encyclopedia of measurement and statistics*, *1*, 530–532.

Araki, M. (2009). Pid control. *Control Systems, Robotics and Automation: System Analysis and Control: Classical Approaches II*, 58–79.

Aster, R. C., Borchers, B., & Thurber, C. H. (2018). *Parameter estimation and inverse problems.* Elsevier.

Beck, J. V., & Arnold, K. J. (1977). *Parameter estimation in engineering and science.* James Beck.

Erdos, B., van Sloun, B., Adriaens, M. E., O'Donovan, S. D., Langin, D., Astrup, A., ... van Riel, N. A. W. (2021). Personalized computational model quantifies heterogeneity in postprandial responses to oral glucose challenge. doi: 10.1371/journal.pcbi.1008852

Fletcher, R. (2013). *Practical methods of optimization.* John Wiley & Sons.

Johnson, M. L., & Faunt, L. M. (1992). [1] parameter estimation by least-squares methods. In *Methods in enzymology* (Vol. 210, pp. 1–37). Elsevier.

Maas, A. (2017). *Playing with numbers: the development of an educational diabetes game* (Unpublished doctoral dissertation). Applied Physics. (Proefontwerp.)

Maas, A. H., Rozendaal, Y. J. W., van Pul, C., Hilbers, P. A. J., Cottaar., W. J., R.Haak, H., & van Riel, N. A. W. (2015). A physiology-based model describing heterogeneity in glucose metabolism: The core of the eindhoven diabetes education simulator (e-des). *Journal of Diabetes Science and Technology*, *9*. doi: 10.1177/1932296814562607

Man, C. D., Rizza, R., & Cobelli, C. (2007). Meal simulation model of the glucose-insulin system. *IEEE Transactions on Biomedical Engineering*, *54*, 1740-1749.

Rackauckas, C., Innes, M., Ma, Y., Bettencourt, J., White, L., & Dixit, V. (2019). Diffeqflux. jl-a julia library for neural differential equations. *arXiv preprint arXiv:1902.02376*.

Röder, P. V., Wu, B., Liu, Y., & Han, W. (2016). Pancreatic regulation of glucose homeostasis. *Experimental & molecular medicine*, *48*(3), e219–e219.

Tuomilehto, J., Lindström, J., Eriksson, J. G., Valle, T. T., Hämäläinen, H., Ilanne-Parikka, P., ... others (2001). Prevention of type 2 diabetes mellitus by changes in lifestyle among subjects with impaired glucose tolerance. *New England Journal of Medicine*, *344*(18), 1343–1350.

Wu, S., Wang, Y., Zheng, Y., & Chang, X. (2015, 06). Limited-memory BFGS based least-squares pre-stack Kirchhoff depth migration. *Geophysical Journal International*, *202*(2), 738-747. Retrieved from `https://doi.org/10.1093/gji/ggv156` doi: 10.1093/gji/ggv156

A E-DES Model

1. Glucose in the gut:

$$\frac{dM^{gut}{}_G}{dt} = m_G{}^{meal}(D^{meal}, t) - m_G{}^{pl}(M_G{}^{gut}) \tag{9}$$

$$m_G{}^{meal} = \sigma k^{\sigma} t^{\sigma-1} exp(-(k_1 t)^{\sigma}) D^{meal} \tag{10}$$

$$m_G{}^{pl} = k_2 M^{gut} \tag{11}$$

2. Glucsose in the plasma:

$$\frac{dG^{pl}}{dt} = g^{liv}(G^{pl,Ipl} + g^{gut}(M^{gut}{}_{G}) - g^{non-it}(G^{pl}) - g^{it}(G^{pl}, I^{pl}) - g^{ren}(G^{p}) \tag{12}$$

$$g^{liv} = g_b{}^{liv} - k^3(G^{pl} - G_b^{pl}) - k_4\beta(I^{pl} - I_b{}^{pl}) \tag{13}$$

$$c_2 = g_b^{liv}(\frac{K_M + G_b{}^{pl}}{G_b{}^{pl}}) - k_5\beta I_b{}^{pl} \tag{14}$$

$$g^{non-it} = c_2 \frac{G_b{}^{pl}}{K_M + G^{pl}} \tag{15}$$

$$g^{it} - k_5\beta I^{pl} \frac{G_b{}^{pl}}{K_M + G^{pl}} \tag{16}$$

$$g^{ren} = \begin{cases} \frac{e_1}{V_G M^b}(G^{pl} - G_{th}{}^{pl}), & \text{if } G^{pl} > G_{th}{}^{pl} \\ 0, & \text{if } G^{pl} \leq G_{th}{}^{pl} \end{cases} \tag{17}$$

3. Insulin in the plasma:

$$\frac{dI^{pl}}{dt} = i^{pnc}(G^{pl}) - i^{liv}(I^{pl}) - i^{if}(I^{pl}) \tag{18}$$

$$i^{pnc} = \beta^{-1}(k_6(G^{pl} - G_b{}^{pl}) + (\frac{k_7}{\tau_i}) \int (G^{pl} - G_b{}^{pl})dt + (\frac{k_7}{\tau_i})G_b{}^{pl} + (k_8\tau_d)\frac{dG^{pl}}{dt}) \tag{19}$$

$$c_3 = k_7 \frac{G_b{}^{pl}}{\beta\tau_i I_b{}^{pl}} \tag{20}$$

$$i^{liv} = c_3 I^{pl} \tag{21}$$

$$i^{if} = k_9(I^{pl} - I_b{}^{pl}) \tag{22}$$