Armin Schlattau

Modelbased measuring data reconstruction for a fluoroscopic glucose sensor

Diplom/Master thesis

Institute of Medical Engineering
Graz University of Technology
Kronesgasse 5, A - 8010 Graz

Head: Univ.-Prof.Dipl.-Ing.Dr.techn. Rudolf Stollberger

Supervisor: Hans Köhler

Evaluator: Hermann Scharfetter

Armin Schlattau

Modelbasierte Messdaten Rekonstruktion für einen fluoroskopischen Glukose Sensor

Diplomarbeit/Masterarbeit

Institute of Medical Engineering
Graz University of Technology
Kronesgasse 5, A - 8010 Graz

Head: Univ.-Prof.Dipl.-Ing.Dr.techn. Rudolf Stollberger

Betreuer: Hans Köhler

Begutachter: Hermann Scharfetter

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, am .....

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Graz, ...

1 Table of Content

1 Table of Content....1
2 Abstract....3
3 General Introduction....4
3.1 Diabetes Mellitus....4
3.2 Hyperglycemia in intensive care patients....5
3.3 Practical problems of frequent measurements....5
3.4 Sensor from B Braun....6
3.5 Measurement setup....9
3.5.1 Flush solution....9
3.5.2 Flush solution equilibrium....10
3.5.3 Test solution....11
3.5.4 Test solution in equilibrium....12
3.6 FEM Simulator....13
3.7 Fitting....16
4 Oxygen only simulation....18
4.1 Introduction....18
4.2 Method....19
4.3 Results....20
4.4 Discussion....21
5 Quasi constant temperature simulation....22
5.1 Introduction....22
5.2 Method....24
5.3 Results....25
5.4 Discussion....26
6 Multiple quasi constant temperature simulation....29
6.1 Introduction....29
6.2 Method....30
6.3 Results....31
6.4 Discussion....36
7 Extension of the FEM simulator to account for temperature variation....37
7.1 Introduction....37
7.2 Method....38
7.3 Results....39
7.4 Discussion....42
8 Glucose Simulation....43
8.1 Introduction....43
8.2 Method....44
8.3 Result....45
8.4 Discussion....48
9 Mathematical model: 3 coefficient model for the FEM simulator....49
9.1 Introduction....49
9.2 Method....50
9.3 Results....56
9.4 Discussion....62

10 3-coefficient model for the real sensor....65
10.1 Introduction....65
10.2 Method....66
10.3 Results....69
10.4 Discussion....76
11 Superposition based model....78
11.1 Introduction....78
11.2 Method....80
11.3 Results....86
11.4 Discussion....96
12 Last adjustment....97
13 Appendix....100
13.1 Installation of the the FEM model....100
13.1.1 Performance of a simple simulation....101
13.2 PET....105
14 References....106

2 Abstract

Also non-diabetic patients suffer from hyperglycemia. The treatment of this non-diabetic hyperglycemia decreased the morbidity and mortality significantly. Parameter fitting for the glucose sensor from B. Braun is performed. The model is a linear diffusion FEM simulator that also models the reaction between glucose and oxygen. A fast mathematical inversion of the simulator is carried out. Then this inversion is the basis for an estimator of the real sensor. Then the temperature dependence of the sensor is modeled by superposition. A new estimator is constructed by the pseudo-inverse of the superpositions. Then some last adjustments are discussed.

3 General Introduction

The purpose of this chapter is to give a general introduction to the history that led to the development of the sensor. Also a short introduction into Python, PET, and the FEM simulator is given. The chapter ends with a brief description of the physical sensor, and how the measurements were taken.

3.1 Diabetes Mellitus

Diabetes Mellitus is a disease of the glucose metabolism. The culprit, however, is not glucose but the hormone insulin. This hormone is responsible for getting the glucose inside the cell. In typ 1 diabetes the production of insulin is impaired, while in typ 2 diabetes the production of insulin is working but the transport into the cells is impaired. Both conditions lead to hyperglycemia, which simply means elevated blood glucose concentration. The reaction of the body to hyperglycemia is to produce ketones out of fatty acids in order to compensate for the unavailable glucose. This, however, leads to a ketoacidosis, which itself can lead to coma and if untreated to death.

The diagnosis of diabetes is done via a simple glucose measurement from blood or urine. The blood based measurement is performed by withdrawing a drop of blood from the finger. This is the standard procedure that is slightly painful and would wake the patient up during sleep.

The glucose-concentration increases after eating a meal. Insulin is released in order to get back to normal levels. Therefore, the diagnosis of pathologic hyperglycemia cannot be done after a meal.

After about 6 hours, of the last meal, the blood sugar should have normalized in non diabetic patients. Otherwise, elevated glucose-concentrations are a sign of diabetes.

Diabetes typ 1 is treated by the administration of insulin directly into the blood stream. This administered insulin compensates for the missing insulin.

For the master thesis it is important to outline two important points:

Measurement of the glucose-concentration

→leads to→

Application of insulin

The amount of required insulin can be calculated based on the measured glucose-concentration. The administration of insulin leads then to a normalization of the glucose-concentration.

3.2 Hyperglycemia in intensive care patients

The goal of the master thesis is to find an estimator for the output of a glucose sensitive sensor in order to treat hyperglycemia in non diabetic patients. The distinction between the treatment for diabetic hyperglycemia and non-diabetic hyperglycemia arises from the following considerations.

Hyperglycemia is common during critical illness even in patients without a history of diabetes [1]. Stress hyperglycemia is believed to be the result of the release of many hormones and cytokines following the stressful event. This response may be important when no medical care is available as the individual would then have an adequate supply of energy for inflammatory mediators and immune reactants [2].

The artificial feeding of the patient interferes with this evolutionary hyperglycemia program. So artificial feeding leads to increased mortality and morbidity.

The medical treatment of this non-diabetic hyperglycemia has been to measure the blood glucose-concentration each hour, and to treat the elevated glucose to overcome the negative effect of stress hormones on the glucose-concentration. This has been done by the appropriate application of insulin. The desired range of blood glucose-concentration was held within the range of 4.4 to 6.1 mM. While the conventional treatment only starts administering insulin if a value of 11.9 mM is exceeded and tries to maintain glucose-concentration in the range of 10mM to 11.1 mM.

This consideration is not only a qualitative one with low practical quantitative consequences. Appropriate treatment led to a 43% decrease in ICU mortality, 34% decrease in hospital mortality, decreased secondary complications, and reduced duration of stay in the hospital.

The effects of the treatment are so convincing that one could speak of a major breakthrough in medicine.

3.3 Practical problems of frequent measurements

While the facts described in the previous section are very convincing and motivating, a practical problem occurs. The frequent requirement for measurements and insulin application led to the problem of work overload for the staff. Even in an ICU, where the staff to patient ratio is much higher than elsewhere, the task of permanent measurement was considered to be exhausting.

Therefore, the goal is to automate the measurement and application of insulin to the patient. Thereby, the staff is released of the burden of frequent measurements, while the patient benefits from the treatment of his non-diabetic hyperglycemia.

3.4 Sensor from B Braun

B Braun developed a sensor that measures the concentration of glucose indirectly via the depletion of oxygen, by a reaction between oxygen and glucose, which is catalyzed by Glucoseoxidase. The abbreviation for Glucoseoxidase is GOX or GOD.

$$O_2 + H_2O + \text{glucose} \xrightarrow[\text{oxidase}]{\text{glucose}} \text{gluconic acid} + H_2O_2 \qquad (I)$$

Illustration 1: Glucoseoxidase

This enzyme is immobilized inside the Polyurethan Layer of the Sensor.

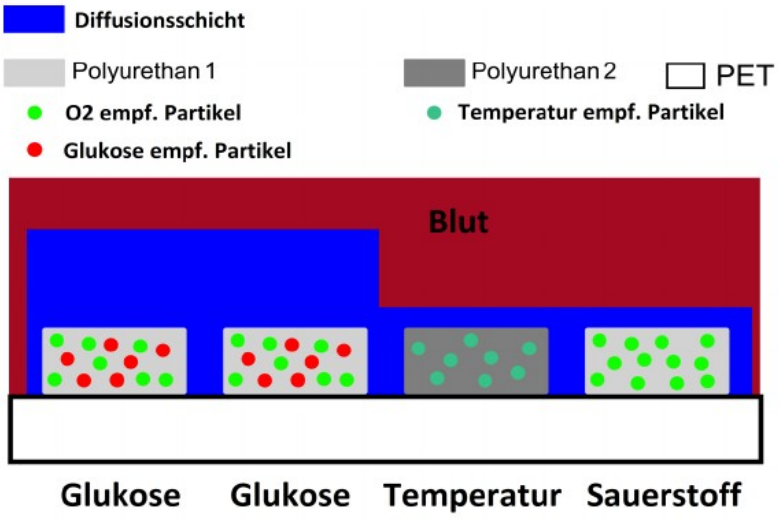

Illustration 2: Schematic of the sensor

This sensor consists of 4 measuring spots. These will be called:

glucose1-sensor or glucose-sensor

glucose2-sensor

temperature sensor

pure-oxygen sensor

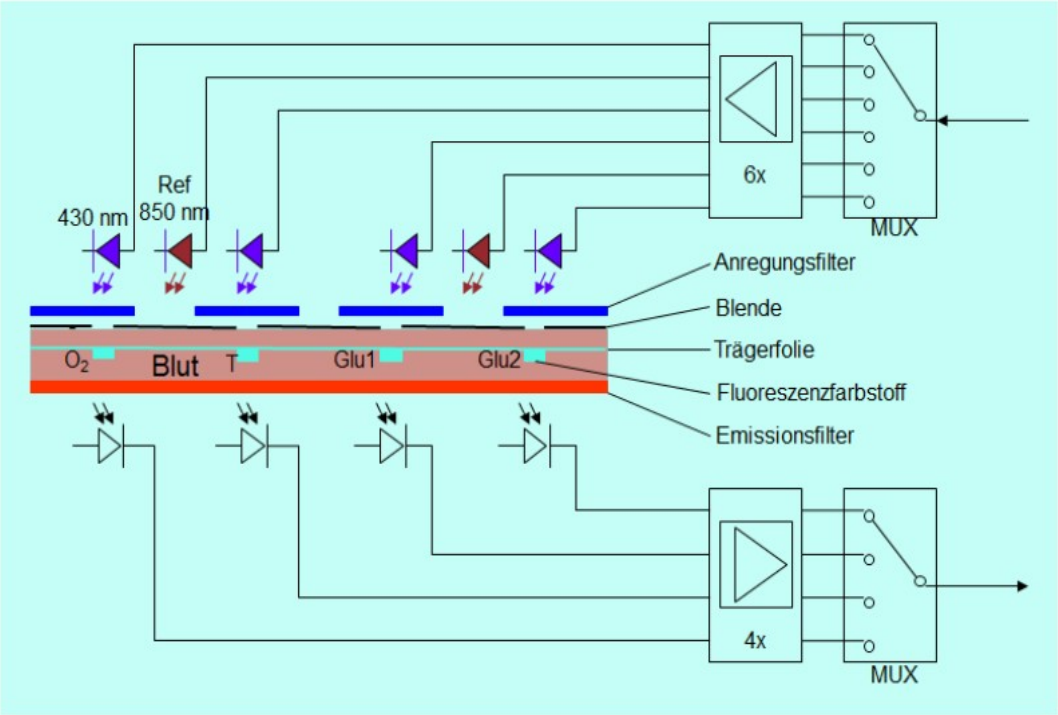

Illustration 3: Measurement of the phase difference Φ

The difference between the glucose1-sensor and the glucose2-sensor is only the geometry of the diffusion layer. The pure-oxygen sensor is a sensor for pure oxygen measurement that is much faster than the glucose1-sensor and glucose2-sensor. The temperature sensor measures the temperature at a separate spot. This temperature is not the temperature inside any sensor but rather the temperature of the blood.

The excitation of the glucose spot with a modulated light source:

$$I(t)=I_0(1+m\cos(\omega t)) \tag{1}$$

Leads to the emission a similar signal with an additional phase of Φ. The oxygen concentration is then calculated from this phase difference.

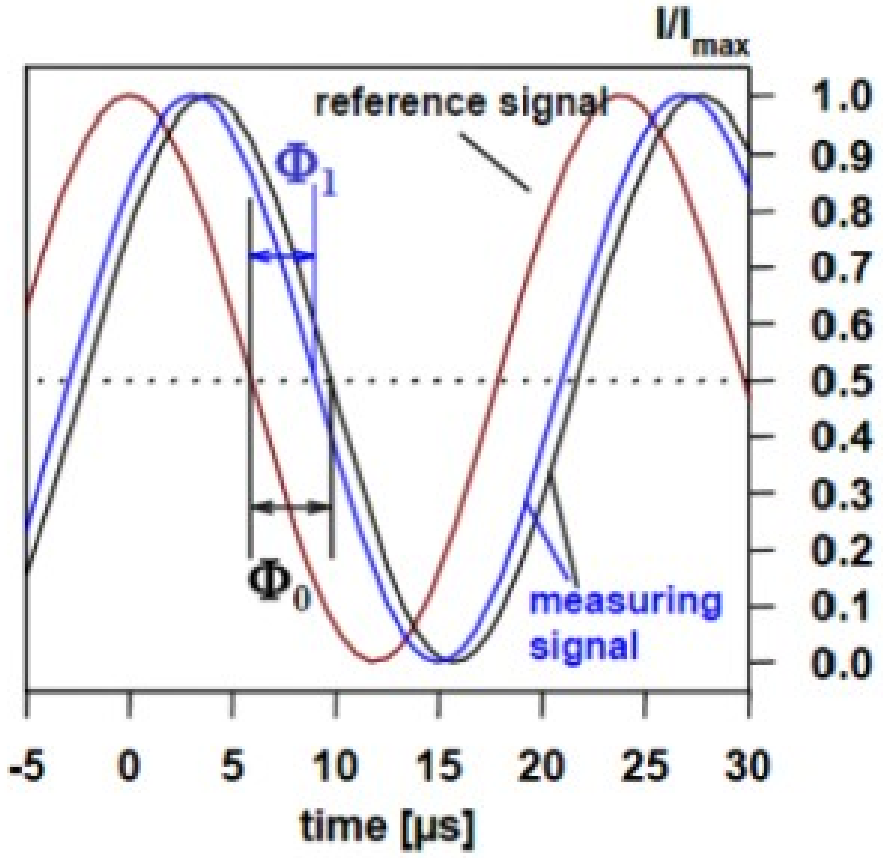

Illustration 4: Excitation and response to excitation

In the specification, the estimate for the glucose concentration should be within a 10% interval of the true value.

3.5 Measurement setup

3.5.1 Flush solution

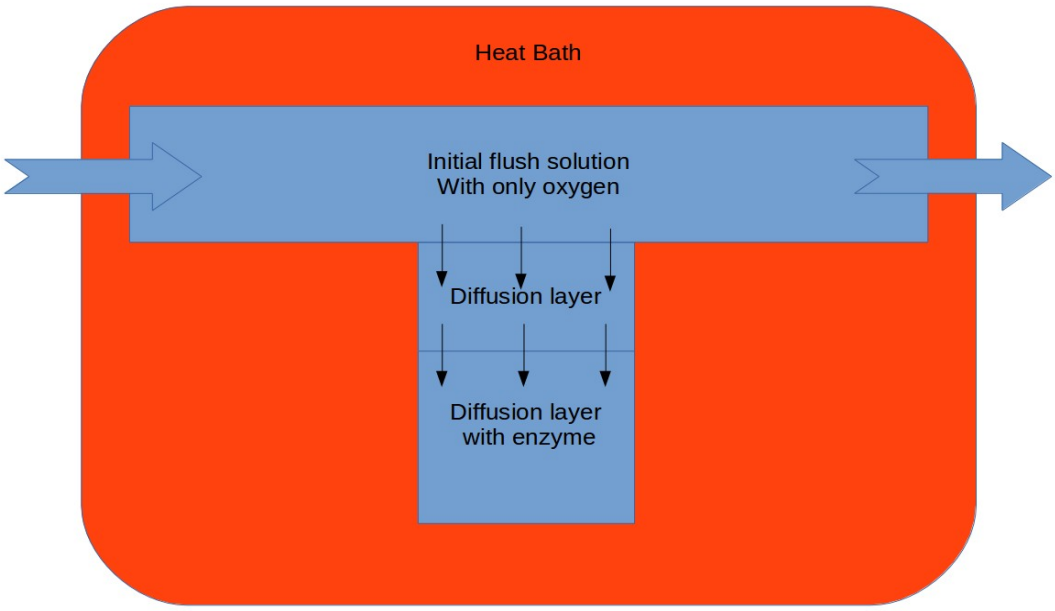

Illustration 5: Application of the flush solution

Before a new measurement the sensor still contains some unknown concentrations of oxygen and glucose from the previous measurements. In order to get rid of these residual concentrations, a flush solution with only oxygen but no glucose is applied in illustration 5. Any oxygen in a region of greater oxygen concentration than the flush solution oxygen-concentration will start to diffuse out of the sensor. A lower concentration inside the sensor will lead to a diffusion of oxygen into the sensor.

So the oxygen-concentration inside the sensor will start to reach equilibrium with the flush solution.

Any residual glucose inside the sensor will diffuse out of the sensor. Therefore, the glucose concentration inside the sensor will begin to decrease.

Normally, the flush solution has room temperature (~22°Celsius) before entering the sensor. The sensor is immersed in a heath bath that has approximately 32° Celsius. So the flush solution will first start to cool the sensor down. Then both the sensor and the flush solution will start to heat up to 32°Celsius.

3.5.2 Flush solution equilibrium

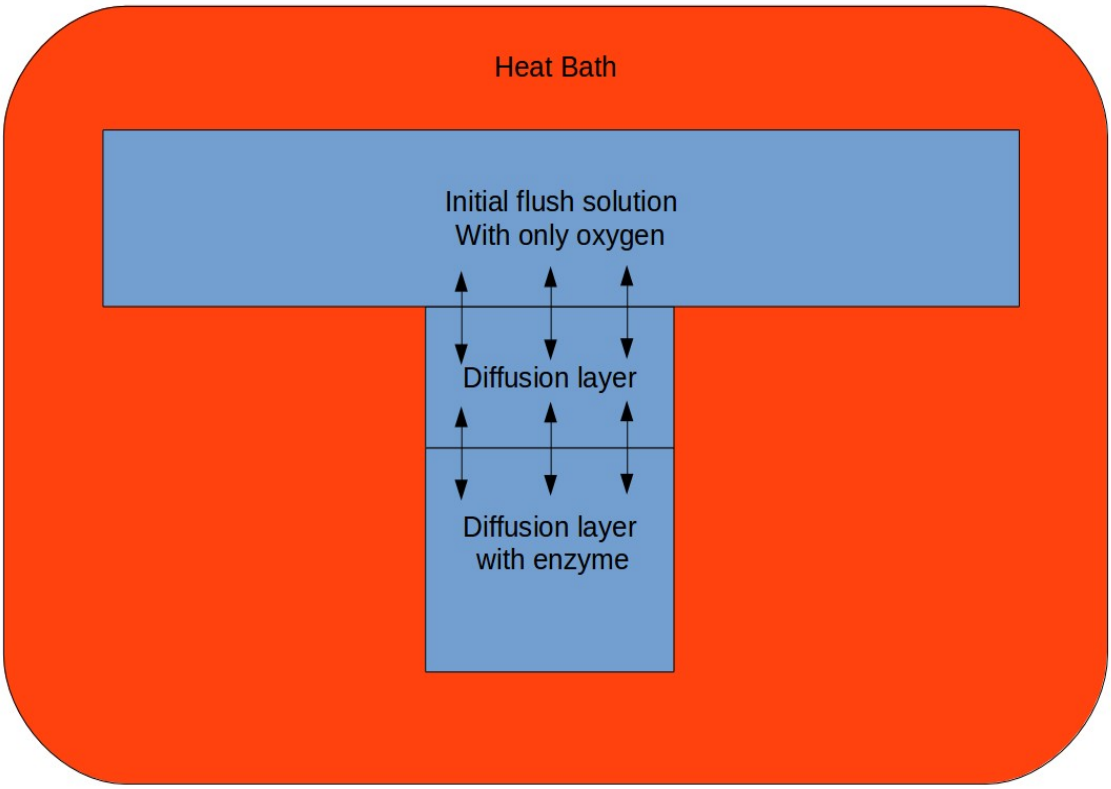

Illustration 6: Flush solution in equilibrium with sensor

In illustration 6 the sensor has reached homogeneous equilibrium with the flush solution. Both are at a temperature of 32°Celsius. This initial condition of uniform concentration inside the sensor is essential to begin a measurement for the following reason.

An inhomogeneous concentration is indistinguishable from a homogeneous concentration inside the sensor as long as both provide the same signal. The calculation of the glucose concentration, however, will be based on the assumption that the initial concentration inside the sensor is homogeneous. So an undetectable inhomogeneous concentration that has the same signal as the homogeneous solution would make the calculations incorrect.

$$Signal = const \cdot \int [Oxygen] dV \tag{2}$$

Therefore, the flushing of the sensor is essential!

3.5.3 Test solution

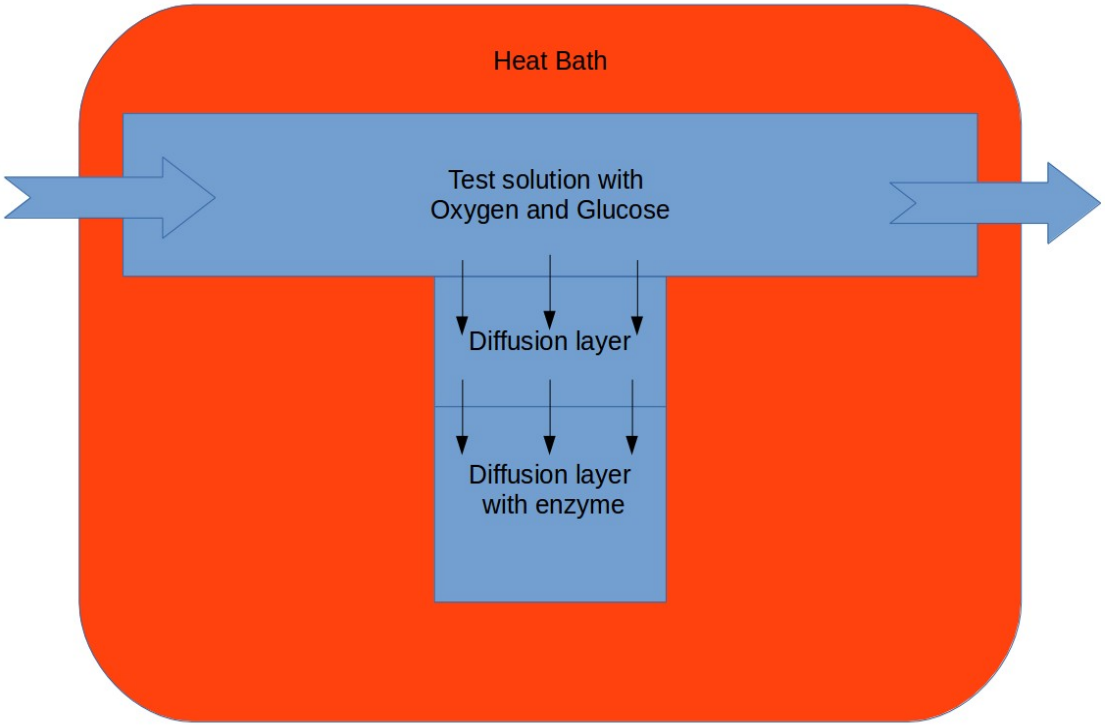

Illustration 7: The test solution with oxygen and glucose enters the sensor

In illustration 7 the test solution enters the sensor. This test solution contains an unknown concentration of oxygen and glucose. The temperature of the test solution is room temperature (~22°Celsius). This is, however, only a test solution. Real blood has a temperature of 37°Celsius or more.

The diffusion of glucose into the sensor is slower that the diffusion of oxygen into the sensor. This can be seen in the sensor signal in the following way. First the oxygen concentration reaches the sensor and the concentration starts to change. Glucose reaches the sensor after a delay time and a second delayed change of the sensor signal is added to the already changing signal. The difference between the first delayed response and the second delayed response is important for the 3-coefficient model in a later chapter.

3.5.4 Test solution in equilibrium

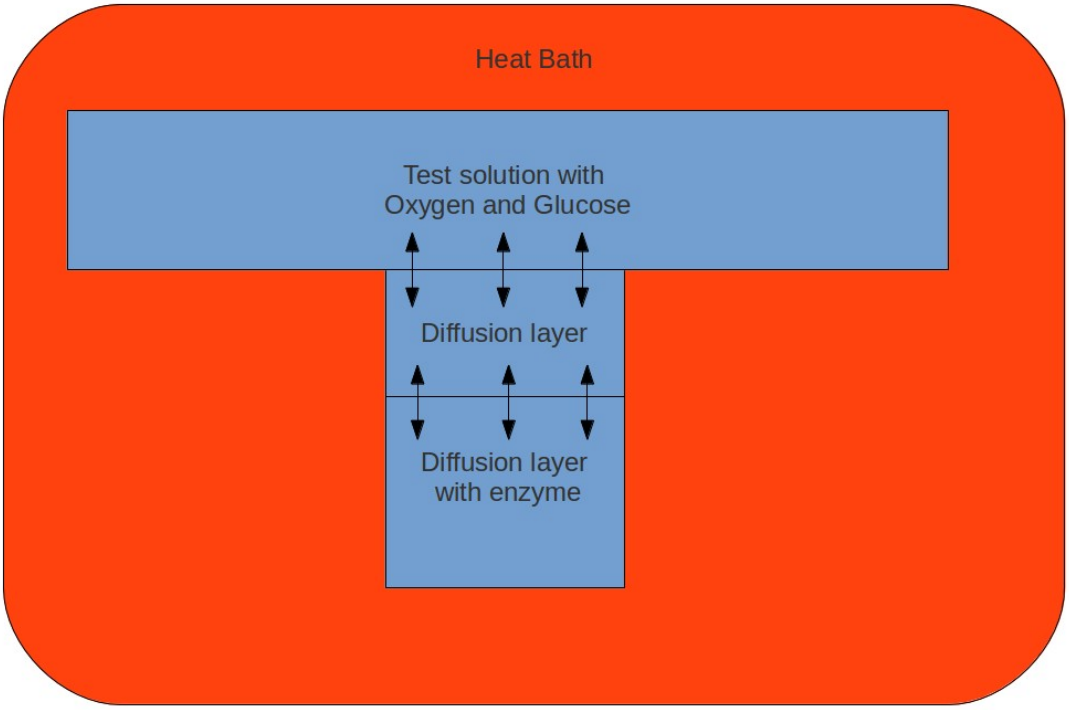

Illustration 8: The test solution with oxygen and glucose enters the sensor

In illustration 8 the test solution has reached equilibrium with the sensor. This means that the inflow of glucose equals the inflow of oxygen. The equilibrium means, for practical reasons, that the sensor signal stays constant. It would be tempting to use this constant signal and the signal from the oxygen-only sensor to calculate from it the glucose concentration. This approach, however, has the disadvantage that the lowest possible oxygen concentration inside the sensor is zero. Any glucose concentration that is, in theory, capable to drag the oxygen concentration below zero would, therefore, be excluded from the calculation.

So a measurement of the glucose-concentration based on the steady-state can only be done up to a certain glucose-concentration.

Another problem is that the equilibrium defined by equal inflow of glucose and oxygen is not a steady state equilibrium, because the test solution does not contain an infinite amount of glucose and oxygen. So after a sufficiently long time, the concentrations inside the test solution get depleted. The constant inflow of either one can, therefore, not be maintained. This problem is, however, only theoretical.

3.6 FEM Simulator

The FEM (Finte Element Model) was provided by Gernot Plank from the Institute of Biophysics. The only task in the master thesis was to install the FEM simulator and to do updates for additional features, as soon as they were required.
In order to verify the validity of the model the following graph was provided by Gernot Plank.

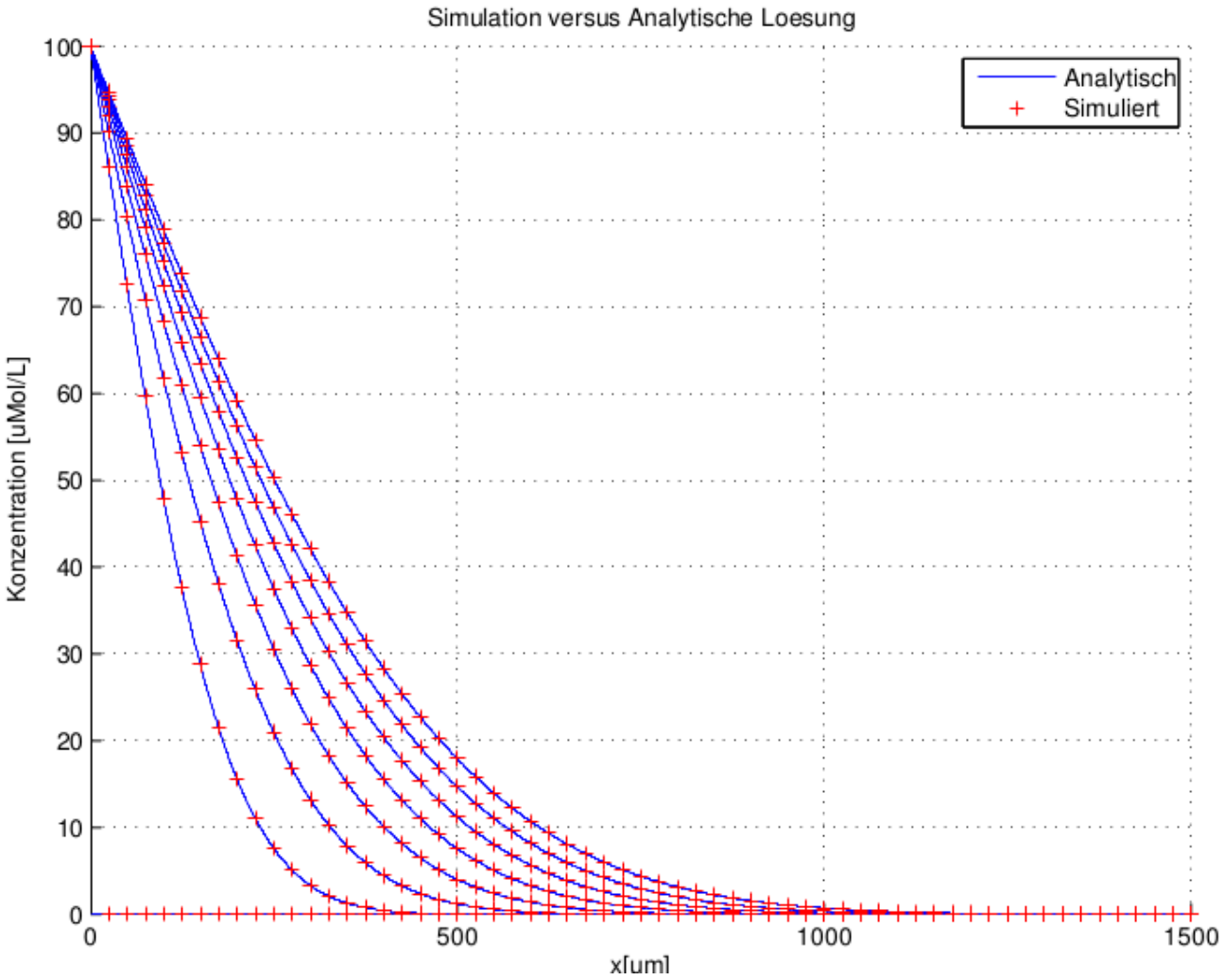

Illustration 9: Comparison of the analytical solution to the FEM simulator. The discrete solutions of the FEM simulator are the red crosses. The blue curves represents the analytical solution.

Illustration 5 shows the analytic solution and also the FEM simulator solution of the oxygen diffusion problem. It can, therefore, be concluded that at least the oxygen diffusion is simulated correctly. It should, however, be stressed that both the analytical and FEM simulator solve the problem where the marginal value of oxygen is assumed to be constant, and therefore independent of depletion by diffusion.
This distinction will become important in a later chapter!
For details see [4].

The FEM simulation consists of two procedures:

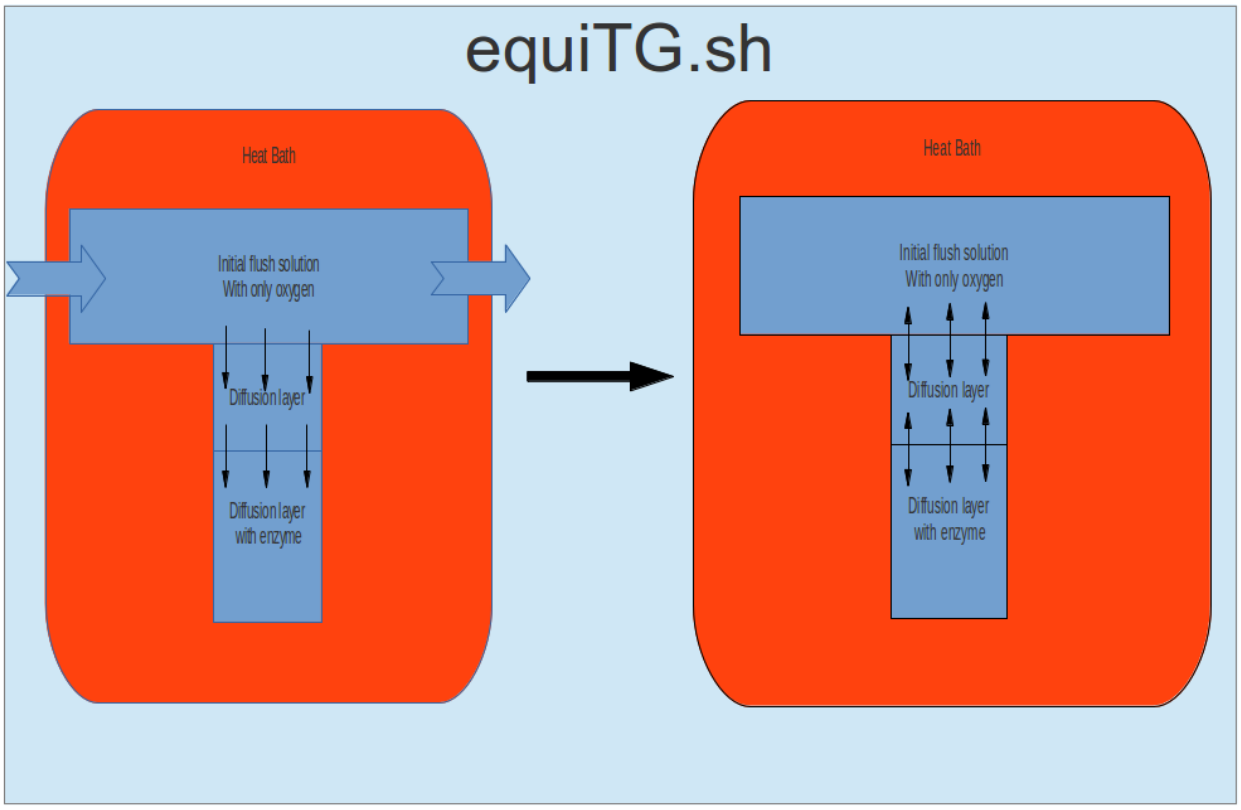

Illustration 10: Simulation done by equiTG.sh. It simulates the transition from illustration 5 to illustration 6.

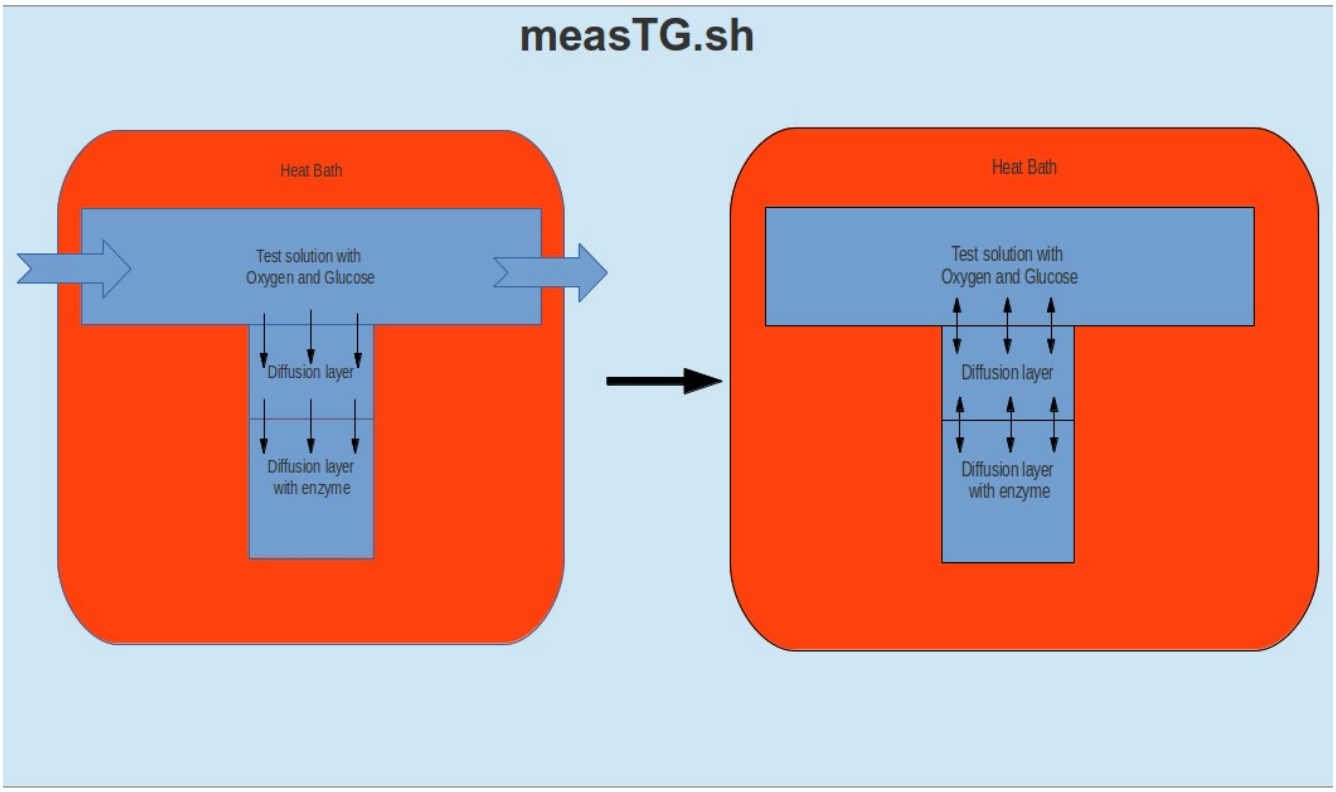

Illustration 11: Simulation done by measTG.sh.

It simulates the transition from illustration 7 to illustration 8.

3.7 Fitting

The measurements from the sensor were provided in csv-files. The parameters for the FEM-simulator like the oxygen diffusion constant and glucose diffusion constant are unknown. So these parameters have to be fitted.

This was done with the programming language Python was used. A short introduction is given, because Python is not yet as popular as other programming languages.
Python is a script language that is easy to write and learn. Python is a programming language that is freely available and normally already installed by default on many Linux systems. One great advantage of Python is that the programmer is forced to program in an easy to read fashion. Unlike in c/c++, for example, where {} are used to start or end a block, in Python the Tab is used to mark each part of the block.

This basic feature made it a lot easier and faster to navigate through the code and modify it. Python doesn't present a big challenge for those who are already familiar with Matlab. The fitting algorithm is already implemented in the Python package scipy. So it won't be necessary to start completely from scratch. The function fmin() uses the downhill simplex method. This algorithm doesn't require any knowledge of the derivatives of the objective function. The objective function will be the weighted quadratic norm of the output of the FEM simulator minus the actual measurements.

$$error = \left[\vec{weight} \cdot \left(\vec{simulation} - \vec{measurement} \right) \right]^2 \tag{3}$$

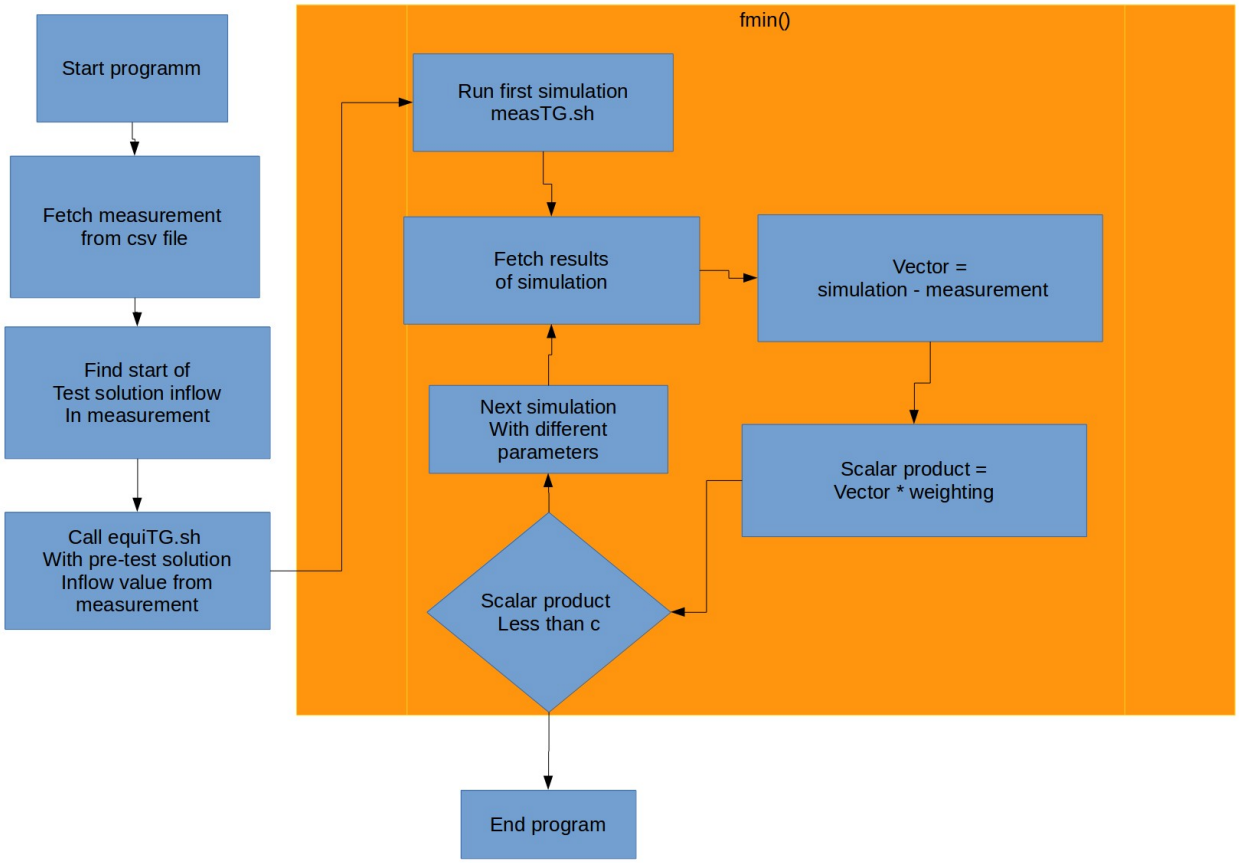

Illustration 12: Python code with fmin() and FEM simulator

4 Oxygen only simulation

4.1 Introduction

The primary goal of the fitting is to find the values of the oxygen diffusion constant and the glucose diffusion constant. It would be possible to start the fitting directly for both parameters. This approach will, however, not be taken for the following two reasons.

1. The fitting for two parameters at the same time takes longer than the fitting for one parameters at a time. (this would be important for an inversion of the problem)

2 . Once the oxygen diffusion constant is established, it can be verified easily against different inputs and their outputs without considering glucose at all.

This approach is feasible for the sensor. The test solution will simply not contain any glucose in the following chapters. Therefore, the response of the the sensor will be completely independent of the glucose diffusion constant.

At the beginning of the measurement the sensor is already in equilibrium with the flush solution as depicted in illustration 6. This means that the sensor has reached a steady constant signal. Then a test solution enters the sensor in illustration 7. This test solution contains only oxygen but no glucose. The reaction of the sensor to the test solution is not immediate, but takes a few seconds for the diffusion to reach the enzyme layer, where the signal is generated. The diffusion can be seen from the signal of the sensor, after this initial period.

The signal from the sensor is depicted in illustration 13 (in this chapter) in blue.

The only fitted parameter is the oxygen diffusion constant of the sensor.

Any dependence on temperature is ignored for the moment.

4.2 Method

The FEM simulator is controlled by two scripts equiTG.sh and measTG.sh. These two scripts reflect the way in which a measurement is taken. In equiTG.sh, the sensor reaches first equilibrium with the flush solution (see illustration 6). In measTG.sh the test solution is applied to the sensor (Illustration 7). So one simulation consist of calling

1. Calling equiTG.sh (Illustration 10) for simulating illustration 5 to illustration 6.

2. Calling measTG.sh (Illustration 11) for simulating illustration 7 to illustration 8.

The measurements are saved in csv-files. The specific structure of the csv-files is explained in the appendix. These files do not only contain the measurements of the test solution, but also of the flush solution. The transition from the flush solution to the test solution is not immediate in the sensor signal. There is a plateau phase in between the two. So the starting point of the simulation was picked by hand through trial and error.

The next task was to call the functions equiTG.sh and measTG.sh from Python with variable parameters. The strings in the files equiTG.sh and measTG.sh were copied into the python code. The options for the parameters were replaced by special marks that were then replaced through regular expressions, by variable parameters from Python. Then the strings were written back in the files equiTG.sh and measTG.sh.

The call of a script, by python, can be done via one command.

The output of the FEM simulator is then copied into the file sensorO2.dat. The units in the csv file are mg/liter. The units in sensorO2.dat were unspecified(~integral over the enzyme layer). In order to calculate a conversion factor, the following simple procedure was done:

1.Call measTG.sh (which wants units in µmol/liter) with 1000 µmol/liter.

2. Let it run for a long period

3. Do a chain multiplication to arrive at the conversion factor (one mole oxygen has 32g)

$$conversion\ factor = \frac{(32\text{mg}\,O_2/l)}{(1000\mu\text{Mol}/l)} \cdot \frac{(1000\mu\text{Mol}/liter)}{(output\ for\ 1000\mu\text{mol}/l)} \tag{4}$$

Therefore, the output of the FEM simulator in units of mg oxygen per liter is:

$$data = conversion\,factor \cdot sensorO2.dat \tag{5}$$

The error function was the quadratic difference between the data and FEM simulator.

$$error = (\vec{simulation} - \vec{measurement})^2 \tag{6}$$

4.3 Results

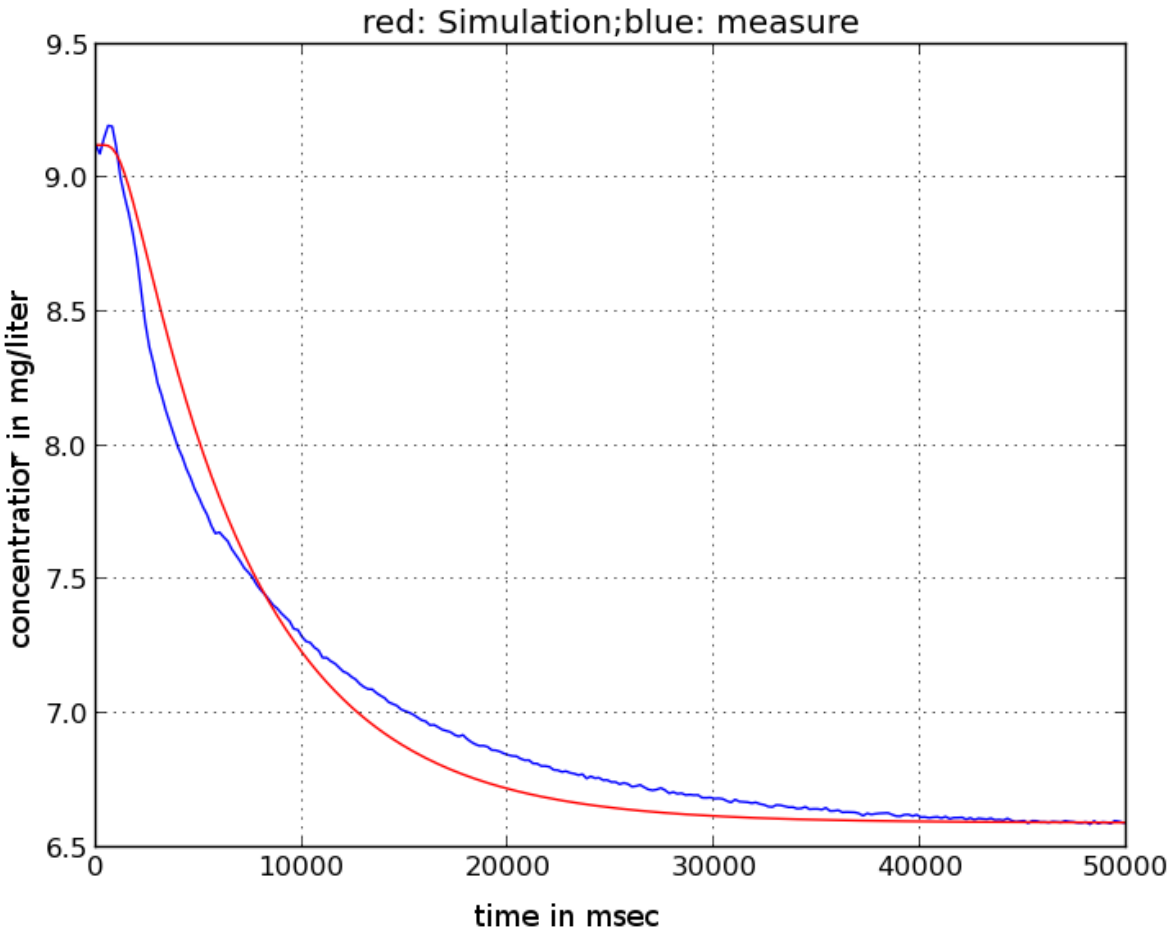

Illustration 13: Comparison between measured data and FEM simulator. The FEM simulation is in red. The measurements are in blue. Notice the plateau phase at the beginning of both. The plateau phase of the blue curve extends backwards from zero. It contains both the constant steady state value from the equilibrium solution and the delay for the signal propagation of the oxygen diffusion from the test solution. The red FEM simulation curve contains, however, only the signal propagation delay from the test solution. The blue curve should only contain the signal propagation delay from the test solution and not the steady state value from the equilibrium solution. The plateau phase precedes the decay phase.

Parameter:

The oxygen diffusion constant is 403.19921875 µm² /s

-D_O2 403.19921875 (parameter in measTG.sh)

4.4 Discussion

Illustration 13 shows the results of the parameter fitting, in comparison to the measured data. The results of the FEM simulator are in red. The measured data are drawn in blue.

The reaction of the sensor to the test solution is not immediate. Therefore, an initial plateau phase is present in illustration 13. This plateau lasts for about one second and then transitions into the decay phase. While the plateau phase of the FEM simulation is smooth, the plateau phase of the measured data has a slight oscillation. This is probably due to the temperature variation, which will be addressed in later chapters.

The choice of the starting point of the simulation was therefore not straightforward. The trivial choice of the beginning of the decay period is not appropriate. The FEM simulator would start with its plateau phase, while the measured data are already decaying.

In illustration 13 the starting point was chosen by hand, so that the plateau phases of the FEM simulator and the measured data are approximately equal.

The next phase is the decay phase of the measured data. In this phase there was a considerable deviation between simulated and measured data, although the shape of the curves is similar.

Later simulation will also contain the diffusion of glucose.
In order to start the fitting for the glucose diffusion constant, the oxygen diffusion should first be simulated to a better degree than in illustration 13. The current chapter did not consider any temperature variation. This may have led to the deviations. So better results might be achieved by taking the temperature variations into account.

5 Quasi constant temperature simulation

5.1 Introduction

The aim of this chapter is to test the hypothesis that temperature changes caused the mismatch between fitted and measured data of the previous chapter.

Illustration 14 shows the variation in temperature during the measurements. The origin of this variation is that the flush solution had 32°Celsius, while the test solution had about room temperature. The form of the curve, at about 5 seconds, originates from the way the fluids were pumped.

In the last chapter the shape of the curves was similar, but no overlap occurred. The source for this discrepancy is apparently the temperature variation of the sensor. If only the measurements after 8 seconds are considered, the effects should be reduced. This is achieved by considering (weighting with one, all else with zero) only the measurements after 8 seconds.

The FEM simulator considers the plateau phase and the decay phase. The only possible initial condition is the homogeneous concentration that is achieved by equiTG.sh. The right initial condition would reach the same values as the measurements after 8 seconds. This, however, does not imply that the starting points of the FEM simulator and the measurements are the same. So the FEM simulator can do whatever it wants, as long as it reaches the same values as the measured data after 8 seconds. This will be achieved by fitting the value of the plateau phase.

If the cause of the deviation, between measurements and FEM simulator is the temperature variation, it could be expected that the fit becomes much better for the considered range.

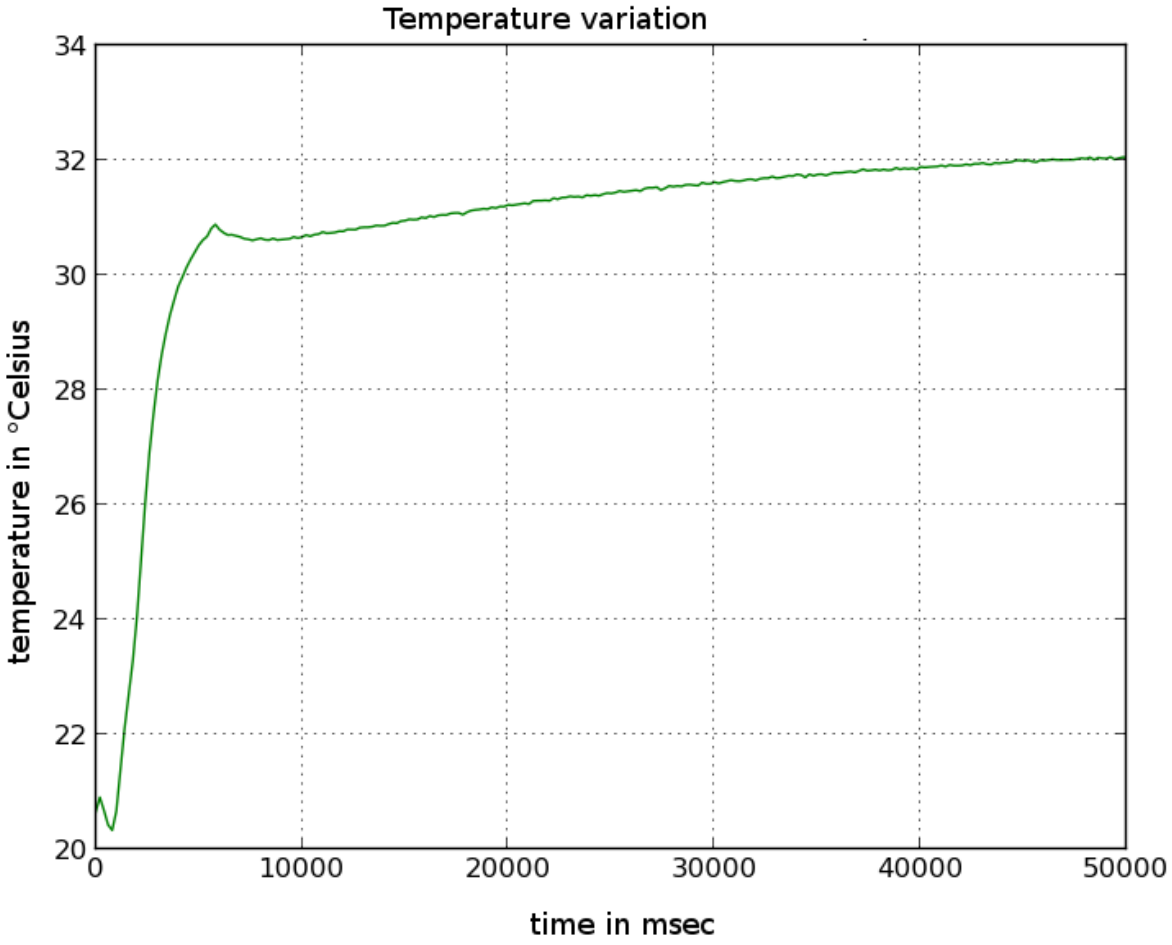

Illustration 14: Temperature variation during the measurements. Notice the influence of the pump at about 5000ms. Only the points after 8 seconds will be weighted this time

5.2 Method

The optimization works on two parameters this time:

1. The starting point of the plateau phase
2. The oxygen diffusion constant.

The essential difference in this chapter is to shrink the range over which the fitting takes place. This is done by setting the weight vector zero for the first 8 seconds.

$$error = [\vec{weight} \cdot (\vec{simulation} - \vec{measurement})]^2 \quad (7)$$

The waiting can simply be implemented by adding a vector multiplication. So the modifications of the code were very small. The fitting time, however, increases significantly for two reasons.

1. Two parameters have to be fitted.

2. Each simulation has to call equiTG.sh, which requires a long time.

5.3 Results

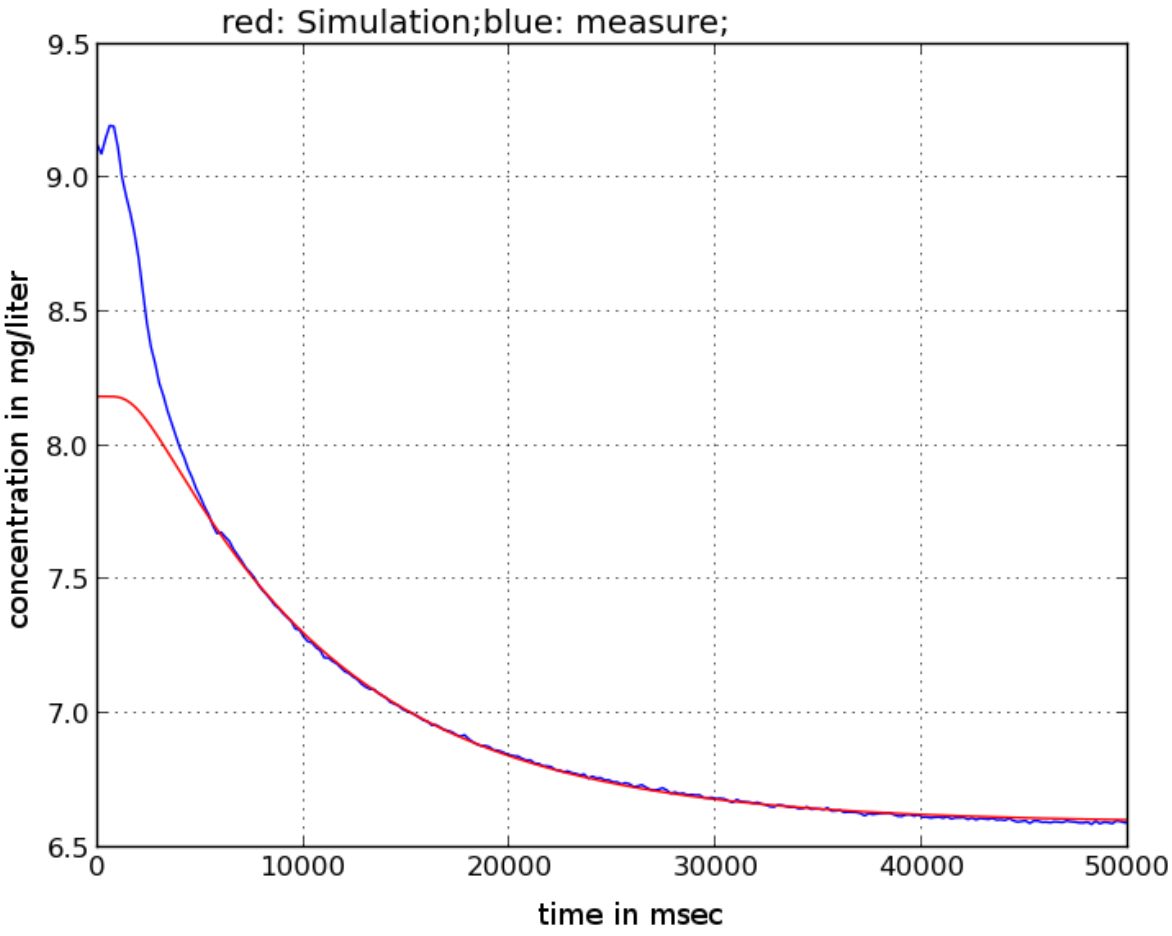

Illustration 15: Comparison of fitted and measured data when disregarding the first 8 seconds. The fit is very good in the considered(weighted with one) range and very poor in the unweighted(weighted with zero) range.

The plateau phase starts with 255.762497616 µmol/liter oxygen

The oxygen diffusion constant is 261.100375582 µm² /s

-O2_dbc[0].strength 255.762497616 (parameter in equiTG.sh)

-D_O2 261.100375582 (parameter in measTG.sh)

5.4 Discussion

Illustration 15 shows the plot of the measured data and the FEM simulation. The plot is over the entire range in order to compare the results to illustration 14. The plateau phases of the measured data and the FEM simulator begin at different values. The mismatch is huge, at the beginning, but starts to decrease quickly. At 8 seconds both curves have the same value and slope. Then for the rest of the simulation there is a good match.

The oxygen diffusion constant in the last chapter was about 403µm² /s. The oxygen diffusion constant in this chapter is about 261µm² /s. This is a difference of about 35%. But the oxygen diffusion constant of 261µm² /s is specific for the small temperature range of 31° Celsius to 32° Celsius. The oxygen diffusion constant of 403µm² /s enabled the fast drop from the plateau phase in the last chapter.

Illustration 15 shows how the sensor and the FEM simulator behave completely similar after 8 seconds, although the signal before 8 seconds is different. It could be argued that this different behavior in the first 8 seconds should also show up later. The temperature variation leads to a variation in diffusion constant that leads to a variation in the way the oxygen molecules arrive at the enzyme layer.

A short derivation is needed to show why the previous argument is invalid. The physical model of the diffusion layer is:

$$D = D(x, y, z, T(t)) \tag{8}$$

It will be assumed that there is no space dependence for the following derivation. So that the diffusion constant simplifies to:

$$D = \widetilde{D}(T(t)) \tag{9}$$

or restated:

$$D = \widetilde{\widetilde{D}}(t) \tag{10}$$

So the temperature dependence becomes a time dependence.

The uni-dimensional form of Fick's equation:

$$\frac{\partial \phi}{\partial t} = D \frac{\partial^2 \phi}{\partial x^2} \tag{11}$$

becomes, by considering the time dependence of the oxygen diffusion constant:

$$\frac{\partial \phi}{\partial t} = D(t) \frac{\partial^2 \phi}{\partial x^2} \tag{12}$$

In this equation t represent the real physical time. But for the mathematical derivation it will be assumed that the real physical time depends on a virtual time:

$$t = f(time) \tag{13}$$

The properties of f(time) will be derived in the following derivations. The first step is to take the derivation:

$$\frac{dt}{dtime} = f'(time) \tag{14}$$

Fick's equation can be multiplied:

$$\frac{\partial \phi}{\partial t}\frac{\partial t}{\partial time} = f'(time)D(t)\frac{\partial^2 \phi}{\partial x^2} \tag{15}$$

to arrive at:

$$\frac{\partial \phi}{\partial time} = f'(time)D(f(time))\frac{\partial^2 \phi}{\partial x^2} \tag{16}$$

The first product is defined as k:

$$k = f'(time)D(t) \tag{17}$$

$$k = f'(time)D(f(time)) \tag{18}$$

The function f'(time) is now defined as:

$$f'(time) = \frac{k}{D(t)} \tag{19}$$

The value of k is defined to be

$$k = D(t > 8s) \tag{20}$$

D(t>8s) is already given, but f(time) is not.

The function f(time) will not be calculated at this point. It is sufficient to show that the creation of such a function is possible.

So Fick's equation in virtual time domain becomes:

$$\frac{\partial \phi}{\partial time} = k \frac{\partial^2 \phi}{\partial x^2} \tag{21}$$

This equation can be solved in the virtual time domain.
The solution is:

$$\phi = \phi(x, time) \tag{22}$$

or expressed in the physical time:

$$\phi = \phi(x, f^{-1}(t)) \tag{23}$$

$$\phi = \tilde{\phi}(x, t) \tag{24}$$

After 8 seconds

$$f'(time) = \frac{k}{D(t)} = 1 \tag{25}$$

is constant and equals to one because D(t) is equal to k after 8 seconds by definition.
So it is possible to express f(time) after 8 seconds in the simple linear form

$$t = f(time) = a + time \tag{26}$$

This means:

$$\partial \phi = \partial time \tag{27}$$

So the system

$$\frac{\partial \phi}{\partial time} = k \frac{\partial^2 \phi}{\partial x^2} \tag{28}$$

behaves simply like

$$\frac{\partial \phi}{\partial t} = D \frac{\partial^2 \phi}{\partial x^2} \tag{29}$$

after 8 seconds.
So the previous period can be considered like a longer or shorter period of virtual time.

6 Multiple quasi constant temperature simulation

6.1 Introduction

The fit in illustration 15 was optimized for a quasi constant temperature range for one test solution. It cannot be concluded that the FEM simulator is a complete model of the oxygen diffusion of the sensor, although the FEM simulator matched in the temperature range from 31°Celsius to 32°Celsius with the data.

The fitted oxygen diffusion constant of 261µm² /s has to be tested against other test solutions with different oxygen concentrations.

A further point is that the parameter fitting in the last section was only so good because it was done in a quasi constant temperature range. In order to solve the whole problem systematically, the temperature should be held constant, while different test solutions are fitted.

Therefore, the test solutions in this chapter have different oxygen concentrations, but all the same temperature variation. The temperature variation in illustration 14 was between 31°Celsius and 32°Celsius. The temperature variation in this chapter will be between 30°Celsius and 31°Celsius, as seen in illustration 16.

The standard procedure would be to take the oxygen diffusion constant from the last chapter. Then make a simulation for each specific test solution. Then calculate the residuals and evaluate them. This approach, however, doesn't consider the temperature drop of 1°Celsius. The residuals would get influenced by the lower oxygen diffusion constant. Therefore, a different approach is taken.

The oxygen diffusion parameter will be fitted for each test solution!

The expected result of the fitting should be that the oxygen diffusion constant turns out to be a little bit lower than 261 µm² /s, due to the slightly lower temperature.

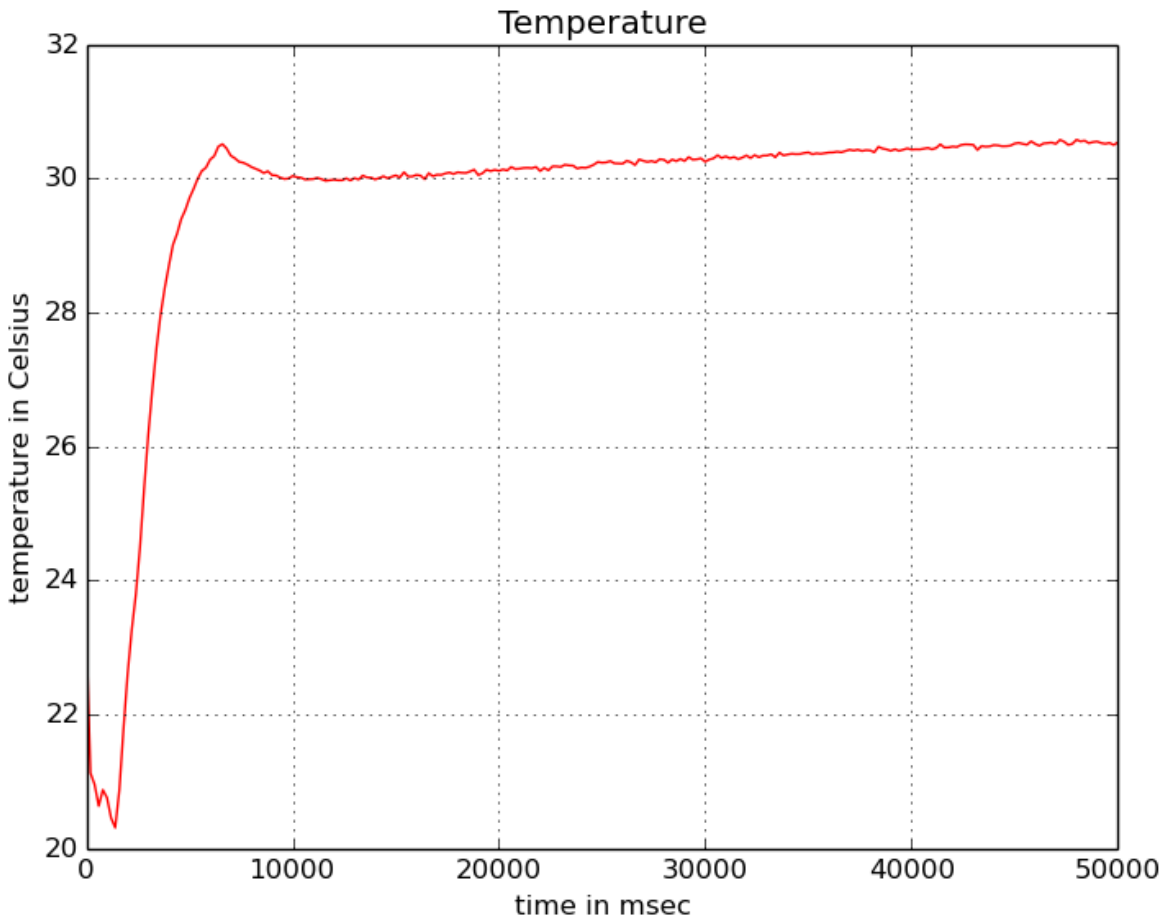

Illustration 16: The common temperature variation of the oxygen-only test solutions in this chapter. Notice that the plot is similar to illustration 14, but that unlike illustration 14 the temperature doesn't rise to 32°Celsius. Therefore, the temperature variation is smaller after 8seconds compared to illustration 14.

6.2 Method

The method is the same as it was in the last chapter. Only the quasi constant temperature range is considered(weighted with one), while the simulation is also simulating the previous range.

The optimization is done for all files independently. The fits will therefore be optimal for each individual test solution.

The FEM simulator is, therefore, only valid if all optimizations arrive at about the same oxygen diffusion constant.

In the next runs it was tested, if this was the case for different test solutions.

6.3 Results

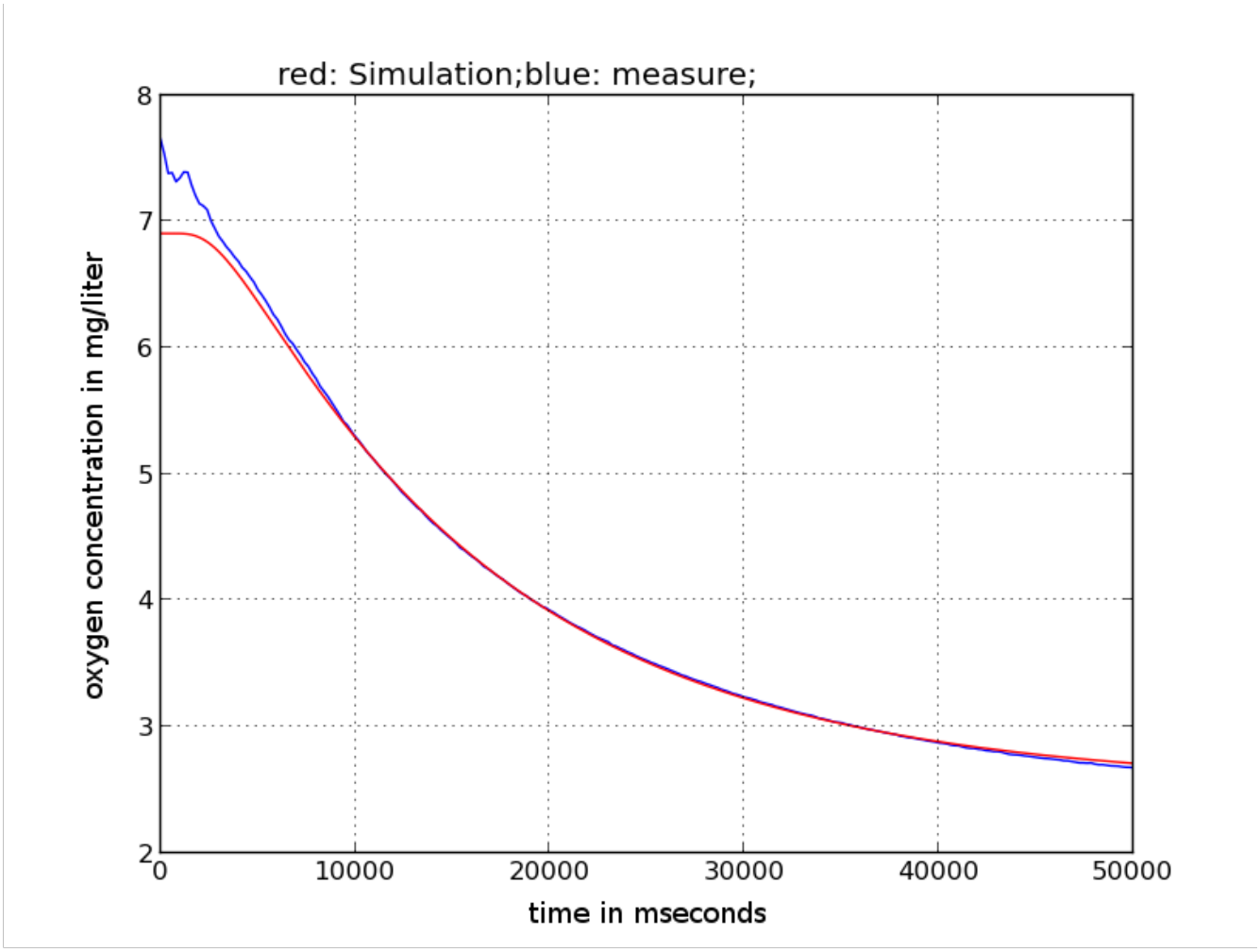

Illustration 17: Test solution with the lowest oxygen concentration.

The oxygen diffusion constant is 86.8583793166µm² /s

-D_O2 86.8583793166 (parameter for measTG.sh)

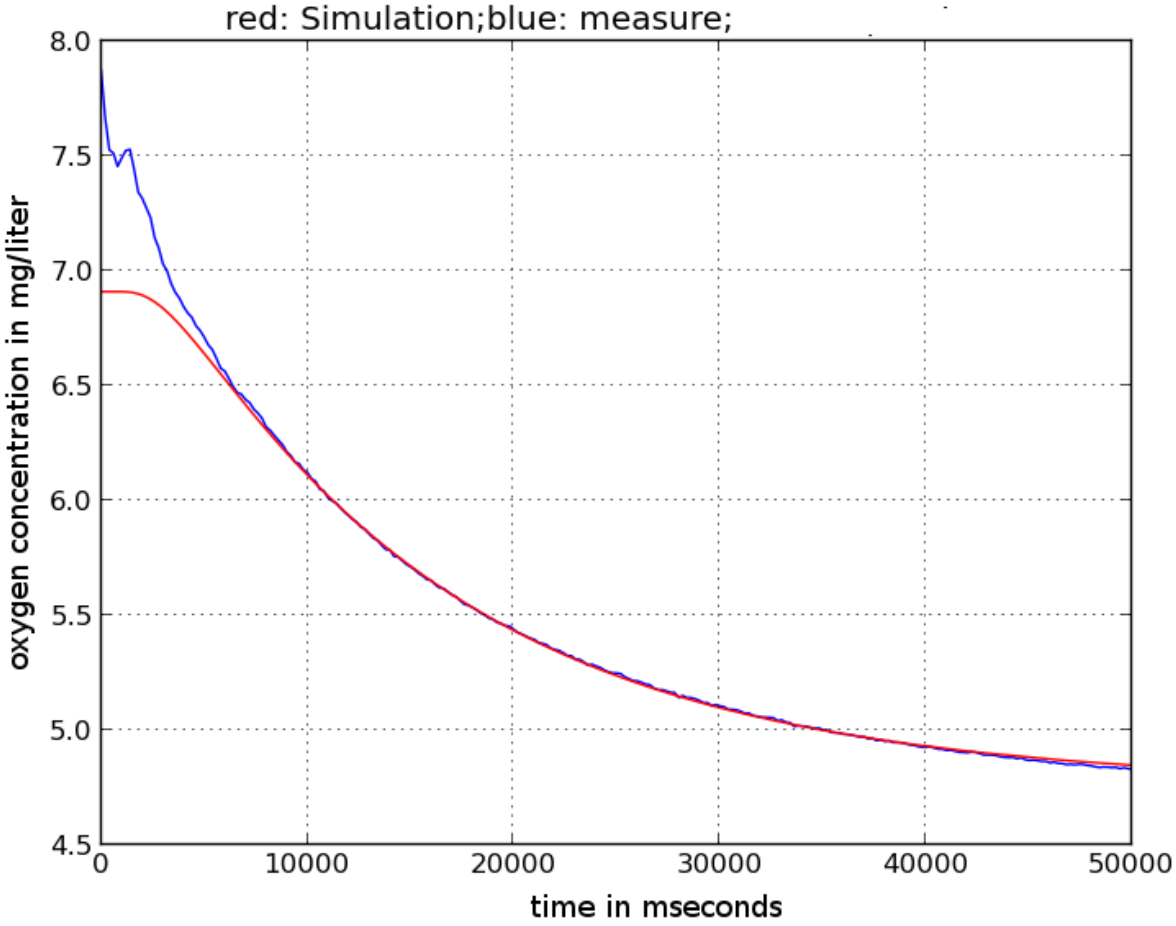

Illustration 18: Test solution with the second lowest oxygen-concentration.

The oxygen diffusion constant is 87.4238907041µm² /s

-D_O2 87.4238907041(parameter for measTG.sh)

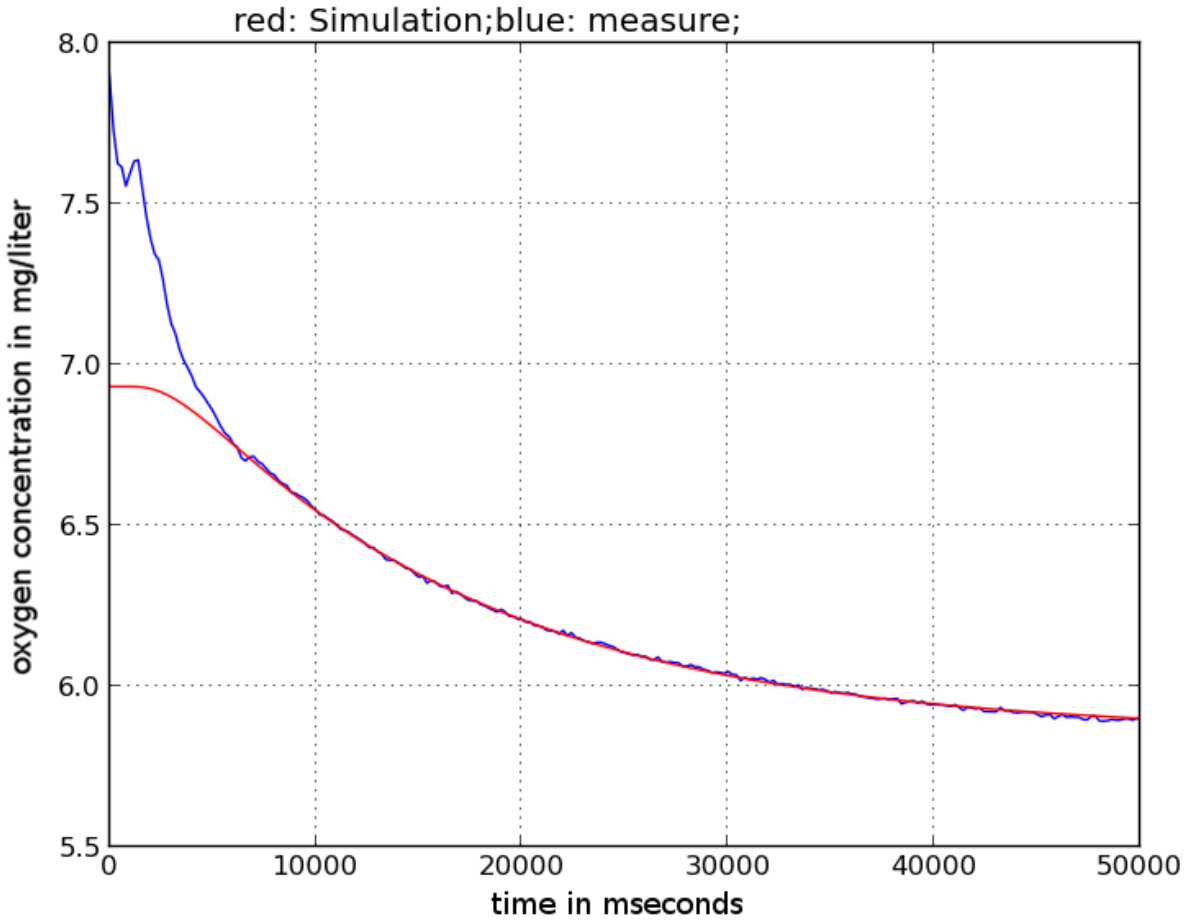

Illustration 19: Test solution with median oxygen-concentration.

The oxygen diffusion constant is 84.3424451385µm² /s

-D_O2 84.3424451385(parameter for measTG.sh)

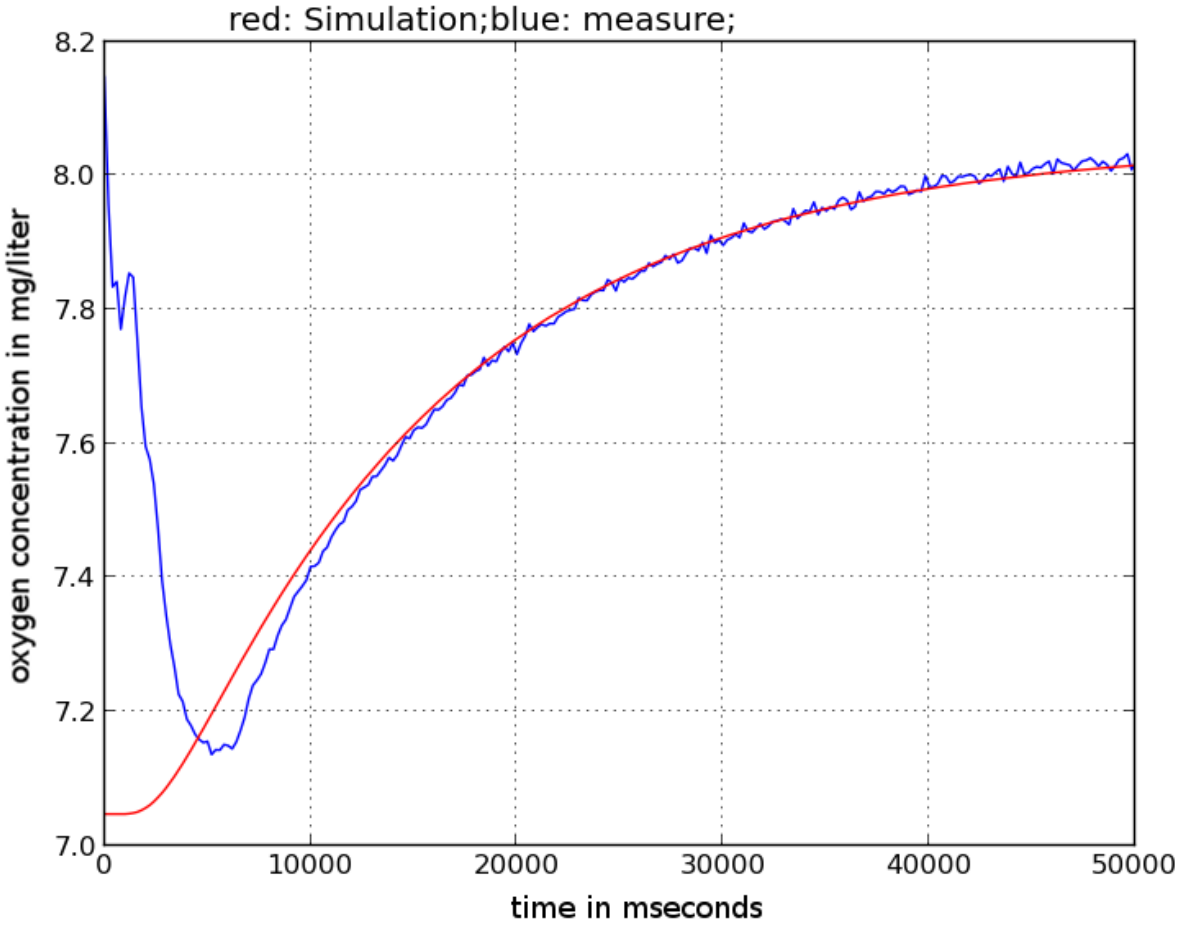

Illustration 20: Test solution with the second highest oxygen-concentration

The oxygen diffusion constant is 91.8364013672µm² /s

-D_O2 91.8364013672 (parameter for measTG.sh)

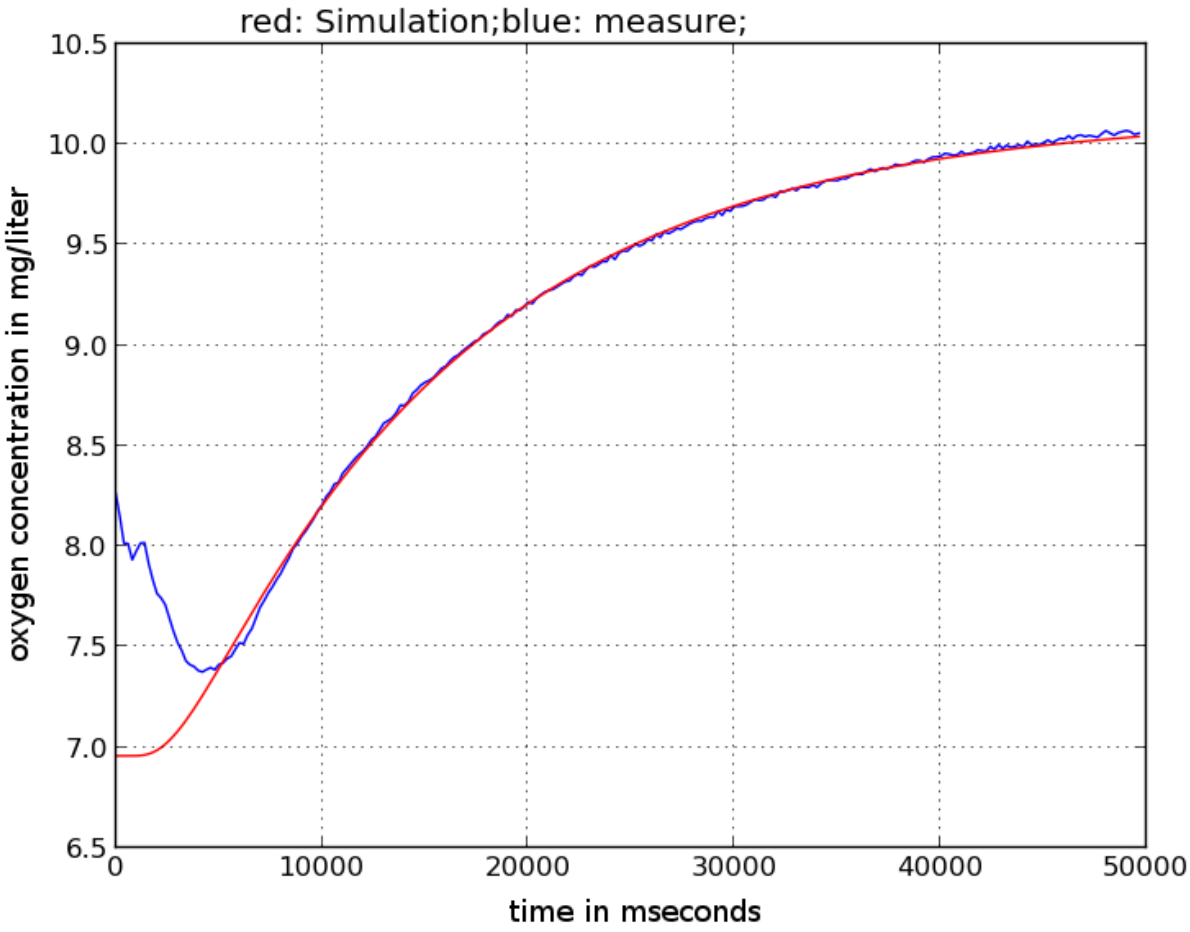

Illustration 21: Test solution with the highest oxygen-concentration

The oxygen diffusion constant is 91.0498144531µm² /s

-D_O2 91.0498144531(parameter for measTG.sh)

6.4 Discussion

The fits are good in all illustrations. This was predicable because each illustration presents a specific fit. The fitted parameters, however, range from 84μm^2 /s in illustration 19 to 91μm^2 /s in illustration 21. In the last chapter the diffusion constant was 261μm^2 /s. So a temperature decrease of 1°Celsius has decreased the oxygen diffusion by about 177μm^2 /s.

A decrease of 177μm^2 /s seems very unlikely. But a closer look at the first 8 seconds in Illustration 20, where the temperature varies between 20°Celsius and 30°Celsius, shows that the influence of the temperature variation is very strong. The expected common sense behavior would be that the temperature variation would only influence the oxygen diffusion constant. This is, however, not the case as can be seen in illustration 21. The temperature variation seems also to have an immediate influence on the signal. The direct effects on signal behavior variation should, however, be small in the considered range. So this does not explain the variation in oxygen diffusion constant.

The reason for this variation could still be due to the small, but still present, temperature variation in the considered range. The mathematical derivation of the last chapter only says that the part of the curve, where the oxygen diffusion constant stays the same, matches with a FEM simulator that doesn't consider temperature variation at all.

7 Extension of the FEM simulator to account for temperature variation

7.1 Introduction

A closer look at illustration 17 and illustration 18 shows how the FEM simulation first assumes a slightly lower value at 8 seconds, than the measured data, but ends up with a slightly higher value at 50 seconds. The inverse is true for illustration 20 and illustration 21, while illustration 19 is a match.

The oxygen diffusion of the sensor slows down or speeds up depending on its temperature. Therefore, the task in this chapter is to implement the temperature dependence of the sensor into the FEM simulator.

The oxygen diffusion constant depends on the coordinates and temperature.

$$D = D(x, y, z, T(t)) \tag{30}$$

The temperature is, however, not only depended on time, as (30) might suggest, but also on the coordinates.

$$T = T(x, y, z, t) \tag{31}$$

The behavior of temperature obeys the heat equation, which is similar to the diffusion equation.

The implementation of the heat equation into the simulator would be very time consuming during a simulation, therefore, as a first approximation, the temperature will only be dependent on time in the FEM simulation.

$$T = T(t) \tag{32}$$

Illustration 2 shows the sensor. The glucose sensor, where all measurements take place, is on a different spot than the temperature sensor. So the next approximation, after the homogeneous temperature, is that the temperature in the glucose sensor equals the temperature in the temperature sensor.

This assumption may be invalid, because the temperature is measured on a completely different spot than the oxygen concentration. It might be possible that the temperature sensor takes on immediately the temperature of the test solution, while the glucose sensor has a delay.

7.2 Method

The temperature dependence was implemented into the internal code of the FEM simulator. The model of the temperature dependence is that the oxygen diffusion constant depends linearly on temperature.

$$D(T) = D_{ref} \cdot (1 + 0.01 k \cdot (T - T_{ref})) \quad (33)$$

$$D_{ref} \ldots oxygen\, diffusion\, coefficient\, at\, reference\, temperature \quad (34)$$

$$k \ldots temperature\, coefficient\, in\, percent\ per\ Kelvin \quad (35)$$

$$T_{ref} = 20\,°C \quad (36)$$

A temperature increase of 1° Kelvin leads to an increase of k% of the oxygen diffusion constant. The oxygen diffusion constant is calculated for each time step or measured point. So a vector with the temperatures at each time step has to be provided to the FEM simulator.

The values of T were provided over a simple text file that contained the temperature value for each time step separated by new lines. This temperature file was created by Python, and its location was added in the command line options of measTG.sh. So only the location of the file that contained the temperature variation had to be added to the command line call of the FEM simulator.

The temperatures were read from the csv files.

The following two parameters will be fitted in this chapter:

1. Oxygen diffusion constants

2. Linear temperature coefficient of the oxygen diffusion constant

7.3 Results

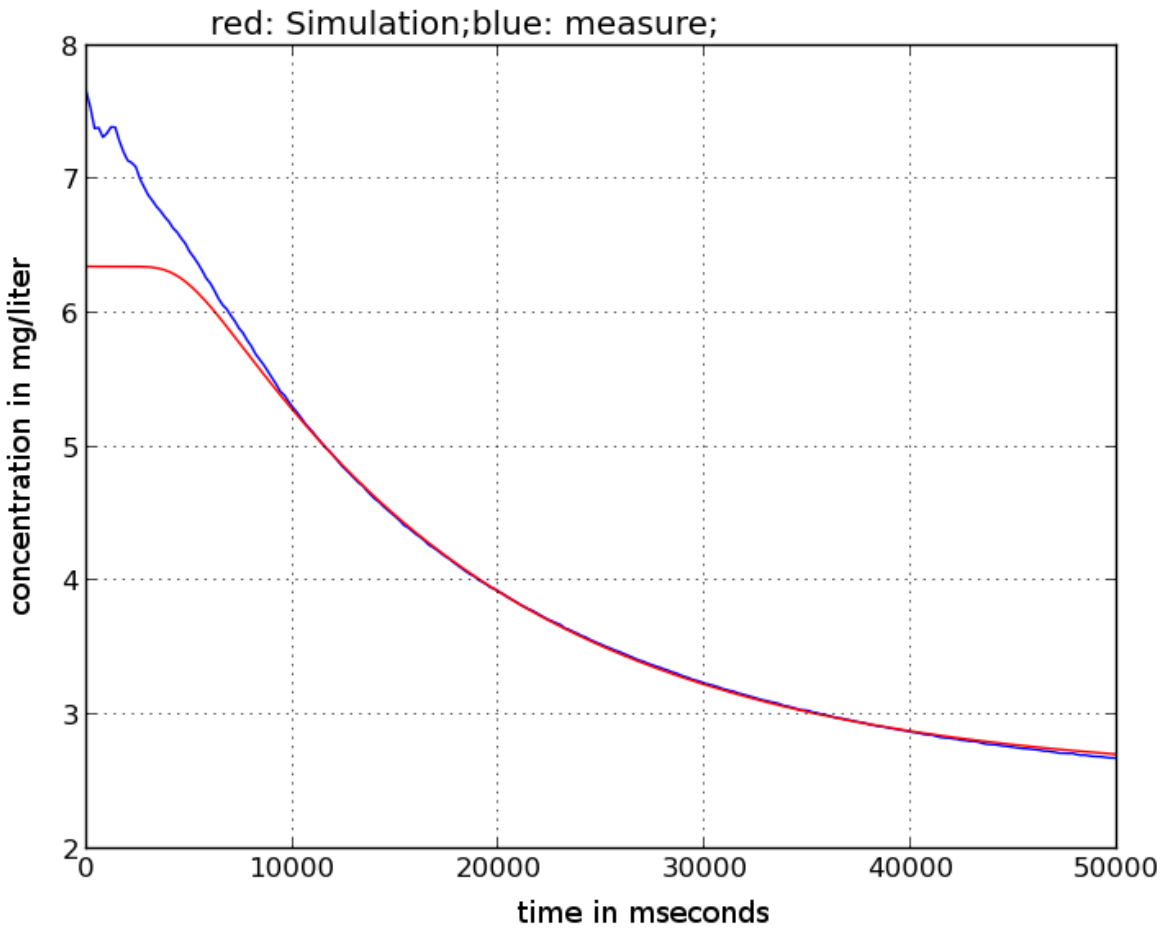

Illustration 22: Fitted vs measured data for the lowest oxygen concentration

oxygen diffusion constant =79.46060177µm² /s

k of oxygen diffusion constant = 1.17326102%/K

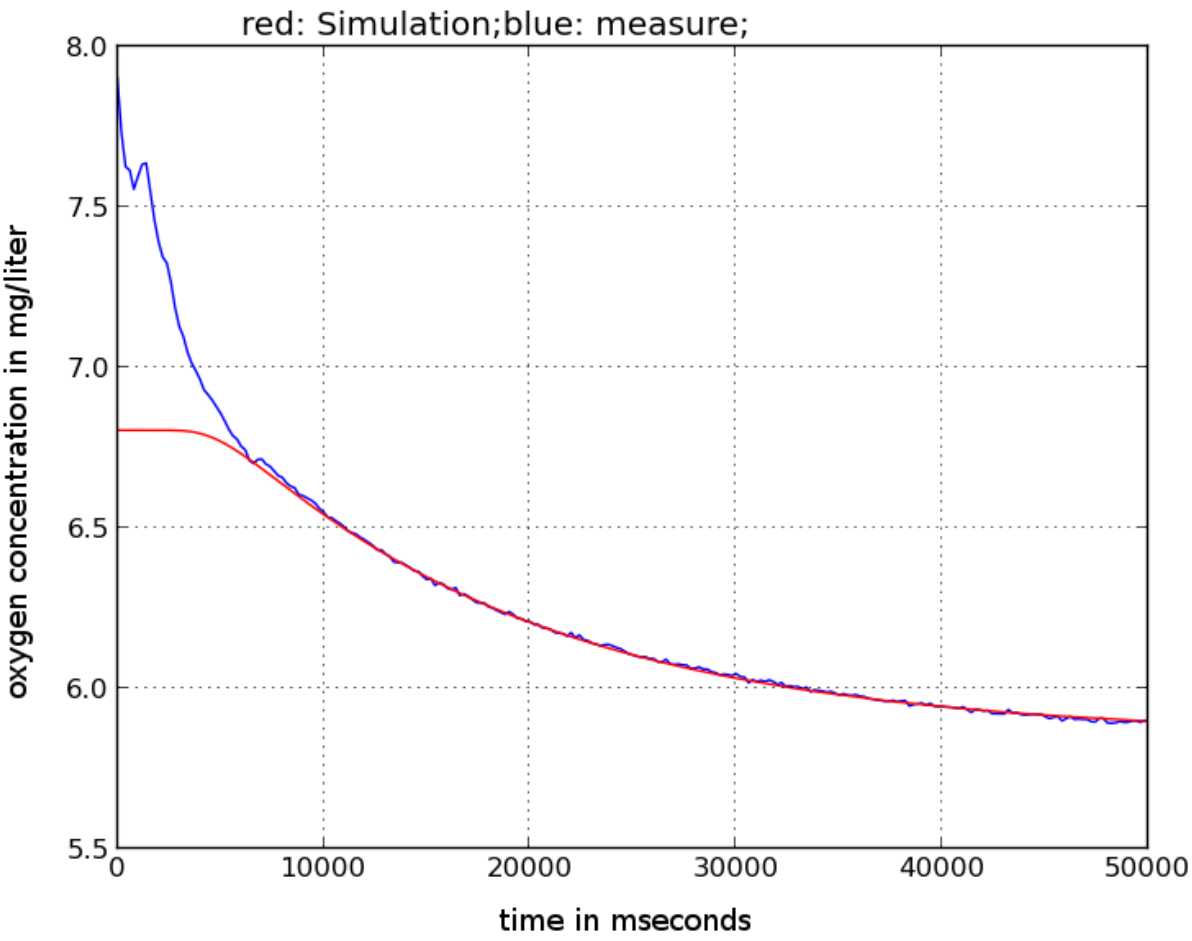

Illustration 23: Fitted vs measured data for the median oxygen-concentration

oxygen diffusion constant = 79.29589125µm² /s

k of oxygen diffusion constant = 1.10805016%/K

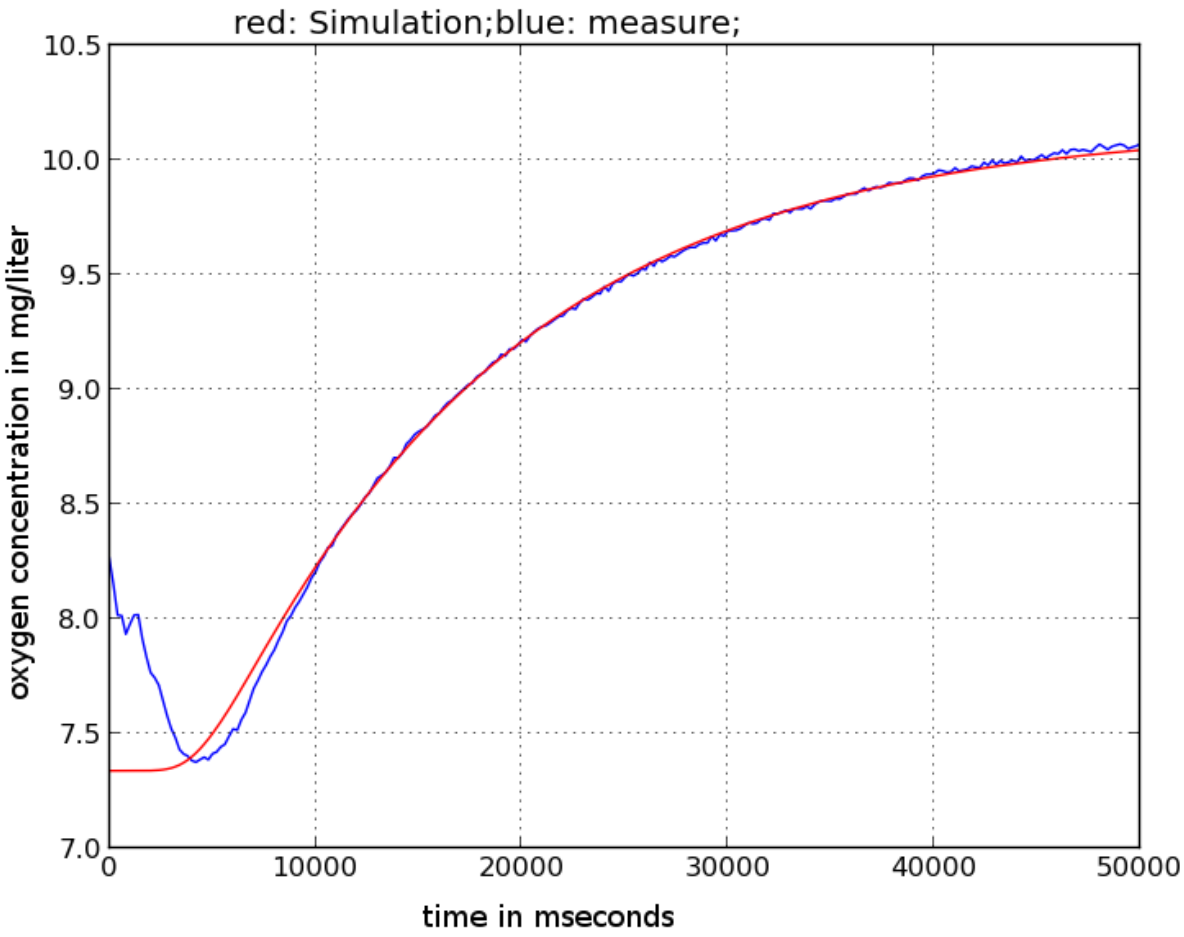

Illustration 24: Fitted vs measured data for the highest oxygen-concentration

oxygen diffusion constant = 84.76206378µm² /s

k of oxygen diffusion constant = 0.98586379%/K

7.4 Discussion

Illustration 22 to illustration 24 show the parameter fitting results that were obtained with the implementation of a linear temperature coefficient. Each illustration is the result of one specific fit. Therefore, it is to be expected that the curves match, for each specific optimization.

A closer look at the fitted parameters shows that each fit arrived at different values. The prerequisite for considering the temperature dependence is that the oxygen diffusion constant coefficient value is approximately equal for each fit. This, however, is not the case. The variation in the offset oxygen diffusion constant makes the linear dependence of the model automatically obsolete.

The temperature that is used in the FEM simulator is not the temperature on the glucose spot, but rather the temperature on the temperature sensor. One of the basic assumption was that this temperature equals the temperature inside the glucose sensor. So one explanation for the failure of the FEM simulator might be that the heat transfer in the probe led to a different temperature distribution inside the glucose sensor than the temperature sensor.

The first 3 seconds of illustration 23 show a strong initial oscillation in the measured oxygen concentration. In the time range between 3 seconds and 8 seconds, the measured data approach the FEM simulator. A comparison between illustration 16 and illustration 23 shows that the oscillation in the first 3 seconds are similar. In the time range between 3 seconds and 8 seconds, the FEM simulator approaches the measured data, while the temperature becomes almost constant.

A simple equation of the form:

$$measured\ oxygen\ concentration = (simulated\ oxygen\ concentration) \cdot f(T)$$

would be a model for these oscillations.
The assumption would be that the temperature at the temperature sensor equals the “uniform” temperature inside the glucose spot.

This temperature dependence was implemented in Python, with different polynomials for f(T). The results, however, did also not satisfy(see the master project for details).

8 Glucose Simulation

8.1 Introduction

The results of the previous chapters weren't able to provide matches, between the FEM simulator and the measured data, for one specific oxygen diffusion constant. There were specific parameters for each specific test solution. These parameters remained, however, in a certain interval.

One explanation for the variation could be that the oxygen inside the test solution gets depleted. An analog problem, in electrical engineering, would be a charged capacitor instead of a voltage source. The FEM simulator assumes a constant oxygen concentration inside the test solution that cannot be depleted. So an implementation of a finite source of oxygen into the FEM simulator would be the next step.

This implementation will, however, not be done for two reasons:

1. The impact of glucose inside the test solution is so strong, compared to oxygen, that an error in the oxygen estimation has little influence on the glucose concentration estimation.

The final goal is an estimator for the glucose concentration with a low SNR. The signal will be the true glucose concentration, while the noise is the real noise plus everything that is not modeled. The SNR of the estimator will therefore be proportional to the glucose-concentration.
This means that the systematic error from oxygen depletion is assumed to be so weak that it has little effect on the glucose estimation.

2. The test solution in clinical applications will be blood.

There is a significant difference between the oxygen that is dissolved in water and the oxygen that is dissolved in blood. A decrease of the oxygen concentration in blood is counterbalanced by the buffering of hemoglobin, while the oxygen concentration inside water decreases by diffusion into the glucose sensor without buffering. Therefore, hemoglobin will provide an almost infinite source of oxygen.

So, although, the results of the preceding chapters had some variation, this chapter continues to the fitting of the glucose diffusion constant.

8.2 Method

The fitting of the oxygen diffusion constant would be as it was in the last chapters, but a constant oxygen diffusion constant will be chosen for each fitting. The oxygen-only fitting in chapter 4 produced an oxygen diffusion constant of 403µm² /s. This oxygen diffusion constant will be chosen for this chapter. Thereby, the fitting will consume less time and the influence of the “wrongest” oxygen diffusion constant can be seen! The problem with the plateau phase remains unchanged.

A look at illustration 27 shows a new problem. The oxygen concentration of the test solution in the last chapters, see illustration 23 for example, was simply the steady state point after 50 seconds. This, however, is not the case in illustration 27. The glucose concentration inside the test solution, as intended, drags the signal further down than the pure oxygen concentration did in the last chapters. This means that the systematic oxygen error is small compared to the signal generated by glucose.

The oxygen concentration of the test solution is unknown, and not the constant steady state signal at the end of the measurements (as it was in the last chapters). But there is also a pure-oxygen sensor in illustration 2. So the oxygen concentration of the FEM simulator will simply be the signal from the pure-oxygen sensor plus a small offset.

This small offset is +0.298mg/liter, and was the empirical difference between the steady state point of the oxygen concentration in the test solution, and the pure-oxygen sensor for the last chapters. For details see the csv file “zero-gluc.csv” at about 90 seconds or illustration 34:

m-no;sample-no;time;G1 [mg/L];G2 [mg/L];O [mg/L];T [degC];refGluc [mM]; refGas [% a.s.]
5;450;89.7969;**6.5915**;6.5595;**6.2167**;32.0157;0.0;0
5;451;89.9971;**6.5913**;6.5580;**6.2165**;32.0282;0.0;0
5;452;90.1963;**6.5942**;6.5607;**6.2189**;32.0181;0.0;0
5;453;90.3965;**6.5858**;6.5516;**6.2172**;32.0431;0.0;0

The source of this offset is unknown.
But the value of 0.298mg/liter provided the the best results!

There will be a specific fit for each test solution. Then the results of each specific fit will be compared!

8.3 Result

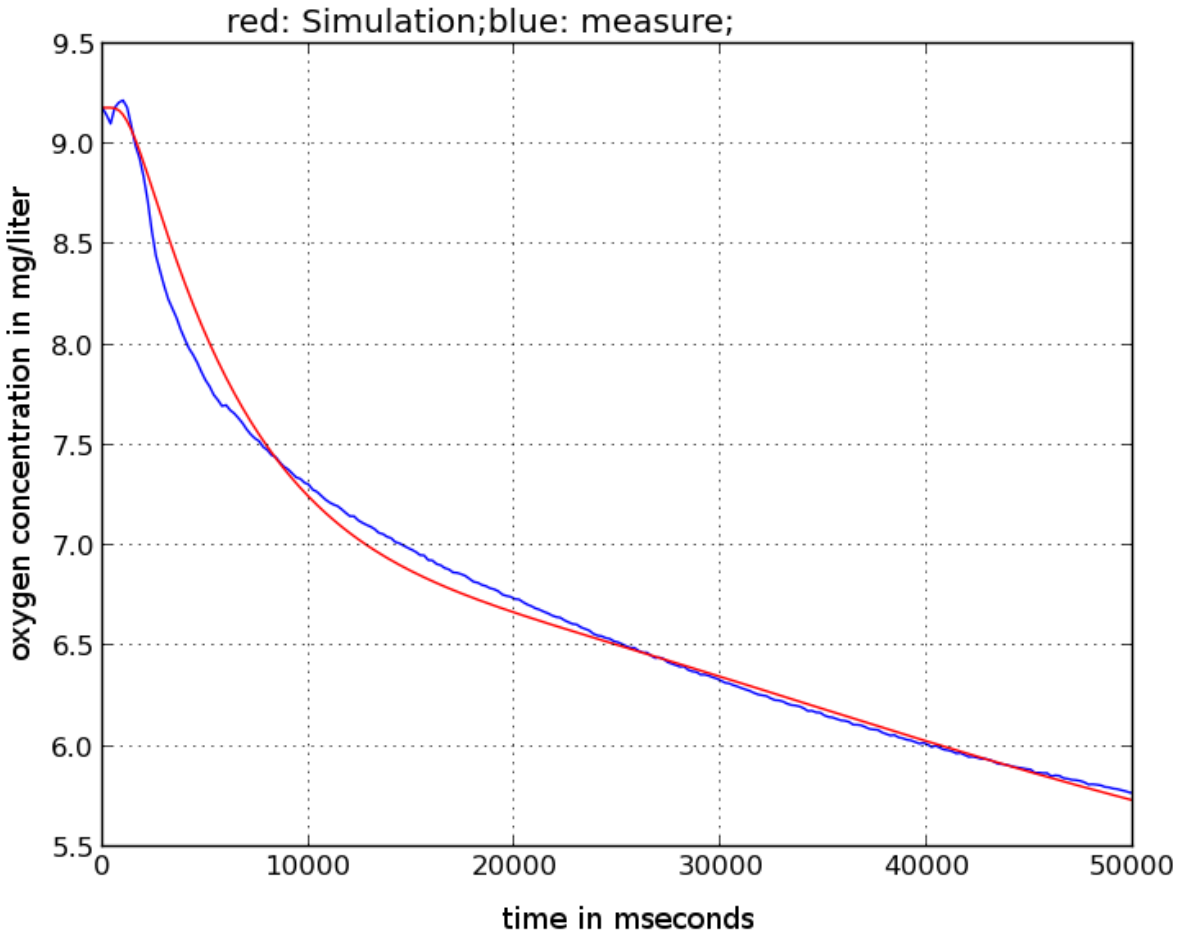

Illustration 25: Test solution with 2.5 mMol/liter glucose vs fitted data

glucose diffusion constant = 12.4609298668µm²/s

oxygen diffusion constant = 403µm² /s

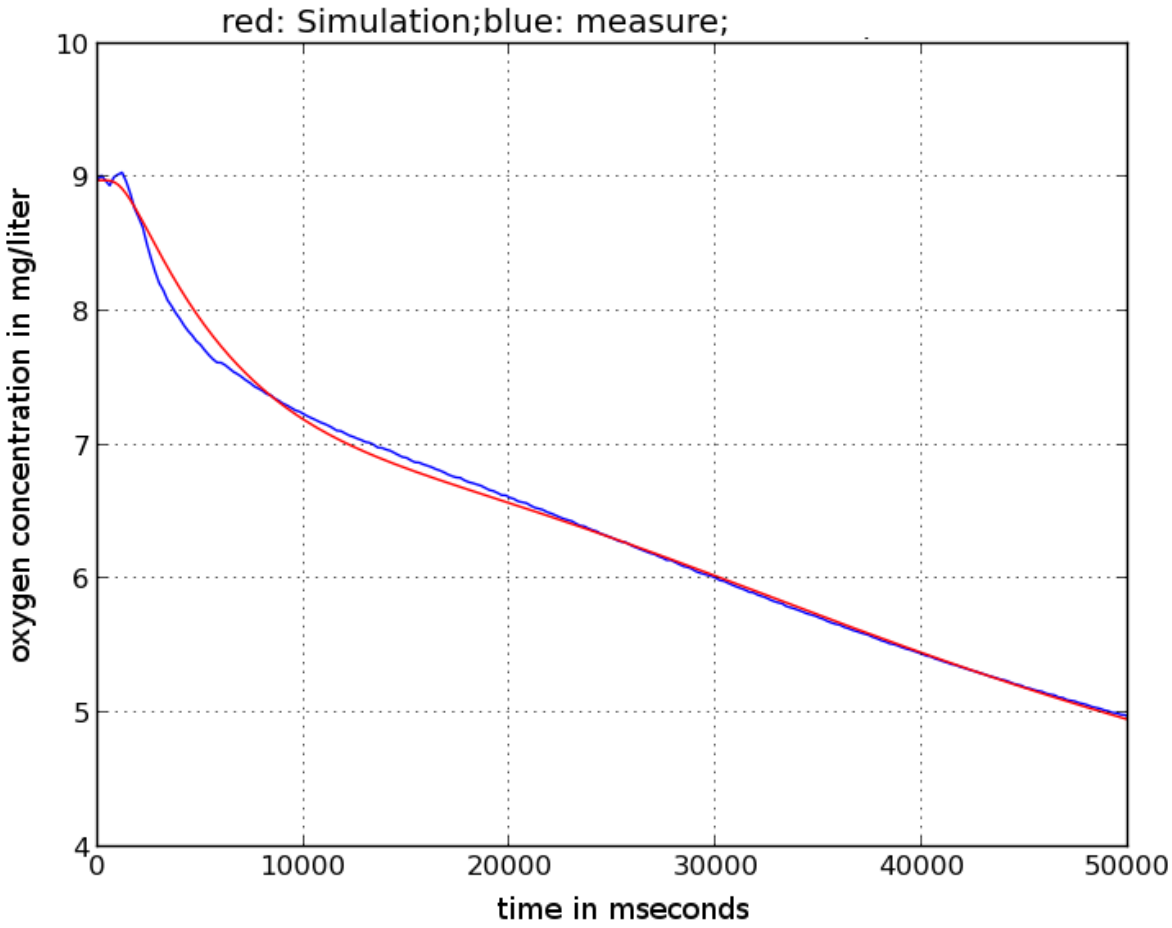

Illustration 26: Test solution with 5 mMol/liter glucose vs fitted data

glucose diffusion constant = 15.9894682898 µm²/s

oxygen diffusion constant = 403µm² /s

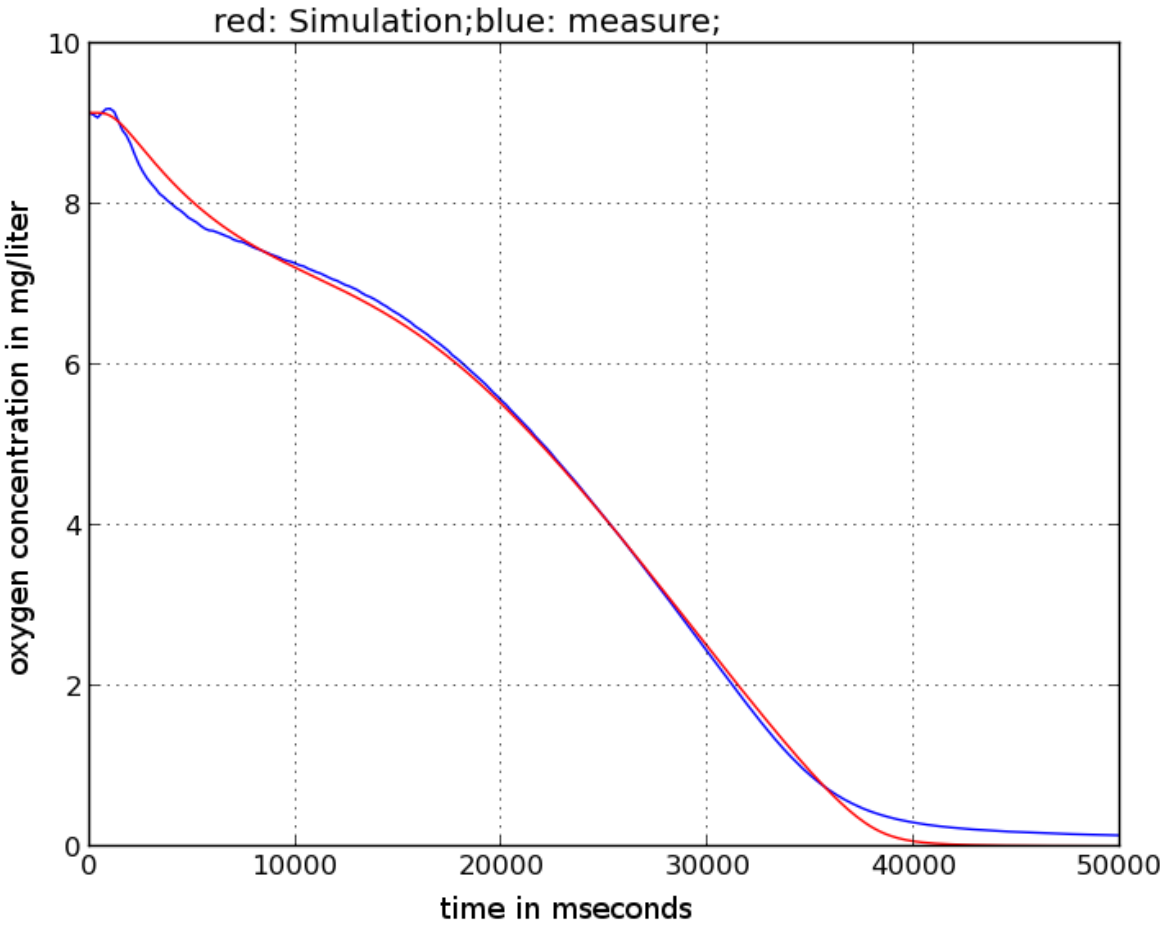

Illustration 27: Test solution with 30 mMol/liter glucose

glucose diffusion constant = 12.4609298668µm²/s

oxygen diffusion constant = 403µm² /s

8.4 Discussion

The fitted curves don't completely match with the measured data and the parameter glucose diffusion varies significantly between the different fits. These problems don't have to be severe in their implication. If an estimation is performed at 25sec for example, where all fits are approximately equal to their corresponding data, the result should be adequate. The variation in glucose diffusion for other points could be ignored and its effects considered as noise.

A similar estimator was already implemented by B. Braun. It was an estimator based on two time points and empirical coefficients. This empirical estimator provided good results

The viewpoint would be the following:

The estimator can only function as good as the FEM simulator is able to simulate the data. If a certain set of parameters is chosen, the estimator will produce a result that has a certain SNR. The better the FEM simulator approximates the measured data at one point, the better the estimator estimates the right glucose concentrations.

In the specification, the estimate for the glucose concentration should be within a 10% interval of the true value.

Once an estimator is derived and implemented, one can make a statistical analysis to assess the interval of confidence.

9 Mathematical model: 3 coefficient model for the FEM simulator

9.1 Introduction

An estimator of the sensor will be derived in this chapter. First, the available possibilities for the implementation of the estimator are listed and discussed. Then one of them is chosen and developed.

The possible approaches for the implementation of an estimator are:

1. By external backward optimizations
One approach, for the estimator, would be to perform a backward fit.
This time, the parameters glucose diffusion constant and oxygen diffusion constant are given, but the test solution oxygen-concentration and test solution glucose-concentration are searched for. The already existing Python code would have to be modified slightly. So the implementation is very easy.

The big disadvantage of this approach is the required time for the calculation of the estimator. An estimation would consume a few hours with this approach.

2. By building a FEM simulator that works backwards(from output to input)
Another approach would be to do an inversion of the internal FEM simulator matrix. This problem would be ill-posed, so it would basically be an internal backward optimization. The FEM simulator would then be given the measured data, together with the glucose diffusion constant and oxygen diffusion constant, to calculate the oxygen-concentration and glucose-concentration.

This estimator would be much faster than the internal fitting, due to some additional internal optimizations. One disadvantage is the cost of the required hardware compared to the third approach.

3. By a new model that is an approximation of the FEM simulator
This new model will be called the 3-coefficient model, unlike the other approaches it is easy to invert. The 3-coefficient model is a very simple mathematical model. One advantage is that the hardware requirements are very low. Another advantage is that the estimator can be calculated instantaneously. The only disadvantage is that the 3 coefficient model is only applicable after the sensor entered into a steady-flow.

Therefore the 3-coefficient model is chosen.

9.2 Method

The sensor is first analyzed from a theoretical perspective, which is going to provide deeper insight into the way the sensor reacts to its environment.

The signal that is provided by the sensor is proportional to the total amount of oxygen in the sensor:

$$Signal = const \cdot \int [Oxygen]\, dV \tag{37}$$

The interest, however, lies in glucose and not in oxygen. But glucose and oxygen are linked together by the chemical reaction:

Chemical reaction:

$$Oxygen + Glucose \rightarrow product \tag{38}$$

This equation assumes implicitly that the reaction is irreversible. The problem is to find a rule that relates the amount of oxygen in the sensor to the amount of glucose in the sensor.

The following equations hold true under all circumstances:

$$[Glucose] + [Oxygen] + 2[product] = constant \tag{39}$$

$$\Delta[Glucose] = -\Delta[product] \tag{40}$$

By a little derivation:

$$\Delta[Glucose] + \Delta[Oxygen] + 2\Delta[product] = 0 \tag{41}$$

$$\Delta[Glucose] + \Delta[Oxygen] - 2\Delta[Glucose] = 0 \tag{42}$$

So the consumption of oxygen is coupled to the consumption of glucose by:

$$\Delta[Oxygen] = \Delta[Glucose] \tag{43}$$

Which is also very intuitive, because one oxygen molecule reacts with one glucose molecule.

From chemical kinetics it is known that:

$$-k[Oxygen][Glucose]=\frac{d[Glucose]}{dt}=\frac{d[Oxygen]}{dt} \tag{44}$$

The oxygen-concentration is assumed to be much higher that the glucose-concentration. The constant k is very big. So the reaction is instantaneous on the regarded timescale.

$$[Oxygen]\gg[Glucose] \tag{45}$$

Although, (44) is a nonlinear differential equation, it is easy to see that as soon as the glucose concentration decreases to zero, the left term becomes zero. Therefore, the consumption of glucose stops.

The glucose-concentration moves toward zero, by the consumption of the equivalent amount of oxygen. From one measurement to the next one the oxygen-concentration evolves as follows:

$$[Oxygen]_{next}=[Oxygen]_{previous}-[Glucose] \tag{46}$$

The meaning of (46) is the following: Whatever amount of glucose is present in the sensor, it immediately reacts with oxygen and disappears. So the oxygen-concentration decreases by the total glucose-concentration.

Equation (46) considers only the volume of the sensor in which the enzyme for the reaction is present, without diffusion. In the following sub-section the diffusion layer and the blood above are is included.

Diffusion:

The following equations hold true for an already established steady-flow:

$$Glucose_{inflow}=P_{gluc}([Glucose_{outside}]-[Glucose_{sensor}]) \tag{47}$$

$$Oxygen_{inflow}=P_{oxy}([Oxygen_{outside}]-[Oxygen_{sensor}]) \tag{48}$$

P …. permeability surface area product

From the previous sub-section:

$$Glucose_{sensor} = 0 \tag{49}$$

$$\Delta Glucose = \Delta Oxygen \tag{50}$$

$$[Oxygen]_{next} = [Oxygen]_{previous} - [Glucose] \tag{51}$$

The change of the oxygen inside the sensor consists of two terms.

$$dOxygen_{sensor} = Oxygen_{inflow}\, dt - Glucose_{inflow}\, dt \tag{52}$$

The oxygen-concentration inside the sensor is increased by the inflow of oxygen, and decrease by the reaction with inflowing glucose.

Putting all equations together:

$$\frac{dOxygen_{sensor}}{dt} = P_{oxy}([Oxygen_{outside}] - [Oxygen_{sensor}]) - P_{gluc}([Glucose_{outside}]) \tag{53}$$

This equation is very simple and has two inputs:

$[Oxygen_{outside}]$

$[Glucose_{outside}]$

The only process, that is neglected in (53), is the time delay for establishing a steady flow.

While this equation is already very useful, it still is a continuous equation. A transformation of this continues equation into a discrete equation would be very useful. Therefore, an approach from system theory is used to arrive at an exact discrete form of this equation (53).

By performing the Laplace transformation with the following abbreviations, (53) becomes (58)

$$Oxygen_{sensor} = y(t) \tag{54}$$

$$Oxygen_{outside} = o(t) \tag{55}$$

$$Glucose_{outside} = g(t) \tag{56}$$

$$sY - y(t=0) = P_{oxy}(O - Y) - P_{gluc} G \tag{57}$$

Or after rearranging the terms:

$$Y(s)=y(0)\frac{1}{s+P_{oxy}}+O(s)\frac{P_{oxy}}{s+P_{oxy}}+G(s)\frac{-P_{gluc}}{s+P_{oxy}} \tag{58}$$

The only purpose of the Laplace transformation was to arrive at the convolution terms

$$y(t)=y(0)e^{-t*P_{oxy}}+o(t)*f_1+g(t)*f_2 \tag{59}$$

For a given t=τ, the first term will be expressed as the constant a, in order to calculate y(τ),:

$$a=e^{-\tau*P_{oxy}} \tag{60}$$

So the first term becomes a multiplication by a coefficient:

$$y(t=0)\,\mathrm{a} \tag{61}$$

The convolution terms can also be expressed as a constant coefficient under the assumptions:

In the time interval t∈[0, τ]:

1. o(t) is a constant number called o

2. g(t) is a constant number called g

So the function o(t) and g(t) can simply be written as oσ(t) and gσ(t) in the time interval [0, τ].
The constants o and g can be drawn out of the convolution. The convolution of the Heaviside jump with a function is the integral of that function.

$$y(\tau)=y(0)a+o\left[\int_0^{\tau} f_1\,dt\right]+g\left[\int_0^{\tau} f_2\,dt\right] \tag{62}$$

The integrals could then be solved and evaluated, but the result will certainly be a real number for both integrals. These numbers will be called:

$$b=\int_0^{\tau} f_1\,dt \tag{63}$$

$$c=\int_0^{\tau} f_2\,dt \tag{64}$$

So with (62),(63) and (63), the result is a discrete equation

$$y(\tau)=y(t=0)a+ob+cg \tag{65}$$

With (65) it is possible to proceed from y(t=0) to y(t= τ), by a simple linear discrete equation.

By the same logic, it is possible to proceed from y(τ) to y(2 τ). This approach can then be generalized to proceeding from y(nτ) to y((n+1)τ).

Therefore, the sensor signal obeys the discrete series:

$$y((n+1)\tau)=y(n\tau)a+ob+cg \tag{66}$$

Written in more readable form:

$$oxygen_{n+1}=a\cdot oxygen_{n}+b\cdot oxygen_{outside}+c\cdot glucose \tag{67}$$

Where · means simple multiplication!

One further simplification can be made. Let's consider the case, in which no outside glucose is present, the outside oxygen-concentration is [k], and the initial oxygen concentration is given.
In this case the equation reduces to:

$$oxygen_{n+1}=a\cdot oxygen_{n}+b\cdot k \tag{68}$$

Equilibrium will be reached after a long period of time. In this case the following equation holds true

$$oxygen_{n+1}=oxygen_{outside}=k \tag{69}$$

or using the notation with k:

$$k=k\cdot a+k\cdot b \tag{70}$$

$$k=k\cdot(a+b) \tag{71}$$

or

$$a+b=1 \tag{72}$$

or

$$b=1-a \tag{73}$$

The sensor can, therefore, be described by a discrete series with only two coefficients.

In order to make the notation more readable, the coefficient b will be written as b and not as (1-a).

The fundamental equation:

$$oxygen_{n+1} = a \cdot oxygen_n + b \cdot oxygen_{outside} + c \cdot glucose \tag{74}$$

is very easy to solve.

In case of zero glucose, the outside oxygen-concentration is:

$$oxygen_{outside} = \frac{oxygen_{n+1} - oxygen_n \cdot a}{b} \tag{75}$$

In case of glucose:

$$glucose = \frac{oxygen_{n+1} - oxygen_n \cdot a - oxygen_{outside} \cdot b}{c} \tag{76}$$

(76) requires knowledge of the outside oxygen-concentration.

This is no problem, because the sensor has an independent pure oxygen sensor. So the outside oxygen concentration is known.

The only possible problem in this equation may arise from noise. This model will, however, be first evaluated with artificial data from FEM simulations. Then some statistical noise considerations will be added, in order to progress to the measured data.

Obtaining the coefficients

The results in this chapter were obtained by running the FEM simulator with known concentrations and diffusion parameters.

One simulation with zero glucose and known parameters was performed. The coefficient a is obtained from (77):

$$oxygen_{n+1} = oxygen_n \cdot a + oxygen_{outside} \cdot b \tag{77}$$

By a simple rearrangement:

$$a = \frac{oxygen_{n+1} - oxygen_{outside}}{(oxygen_n - oxygen_{outside})} \tag{78}$$

So the parameter a was determined from the output of the FEM simulator. Then the parameter b was determined by the simple equation.
b = 1-a

Another simulation, this time with glucose, but otherwise with the same parameters, was performed. The coefficient c was calculated from:

$$c = \frac{oxygen_{n+1} - oxygen_n \cdot a - oxygen_{outside} \cdot b}{glucose} \tag{79}$$

It is possible to determine either the glucose-concentration or the oxygen-concentration from the output of the FEM-simulator given either the oxygen- or the glucose-concentration!

9.3 Results

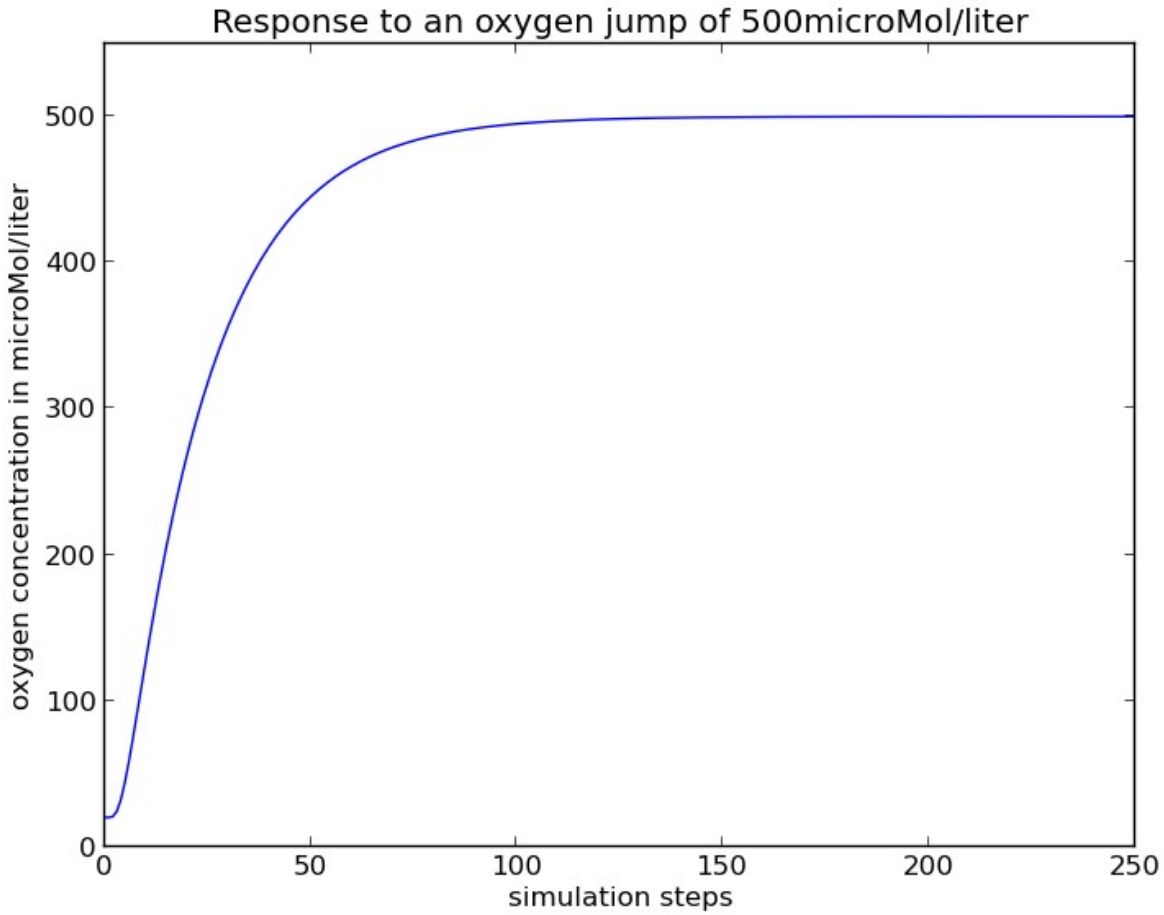

Illustration 28: Response of the FEM simulator to a test solution with an oxygen-concentration of 500µmol/liter. The equilibrium solution had an oxygen-concentration of 20µmol/liter. The oxygen diffusion constant was set to 300µm² /s. Notice the first few steps of the graph where the concentration remains at 20µmol/liter and has no reaction to the 500µmol/liter. After about 150 steps a quasi steady state is reached.

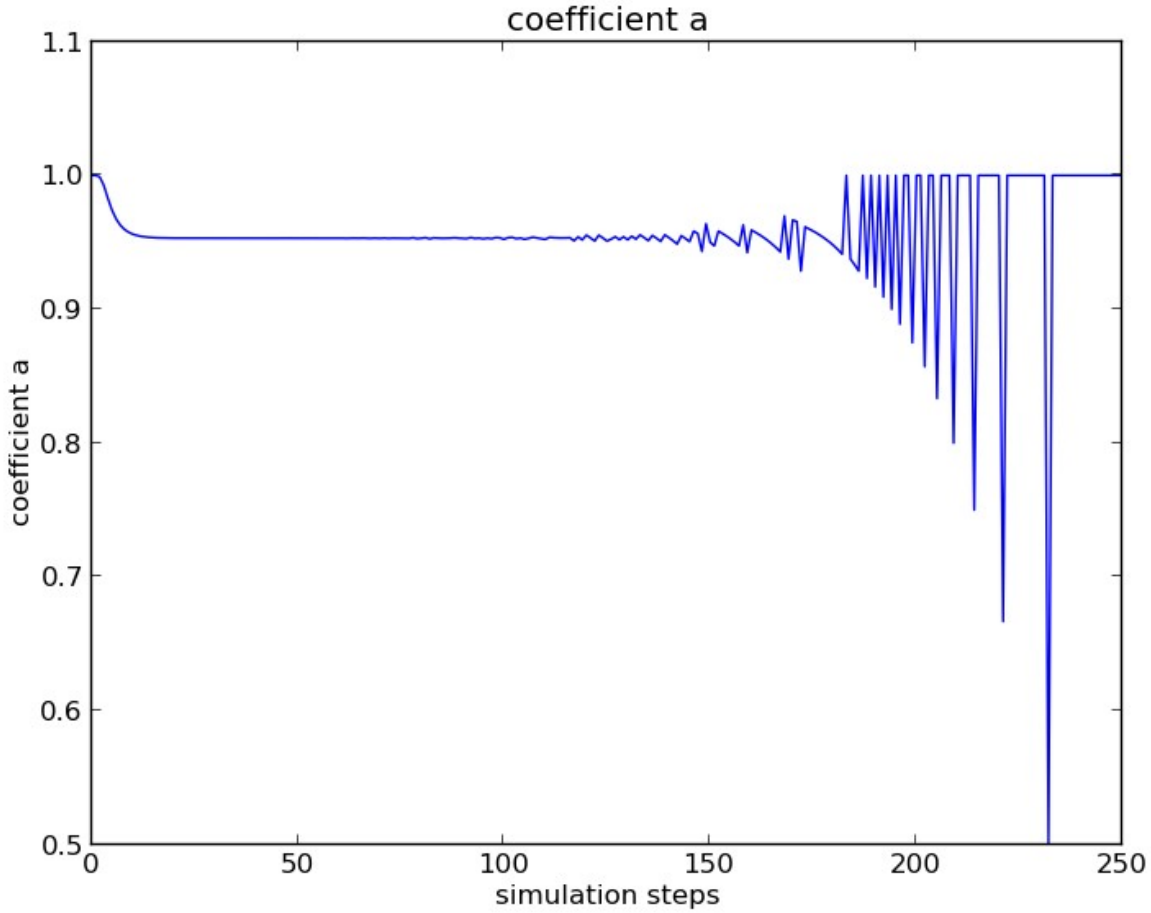

Illustration 29: Coefficient a from previous plot

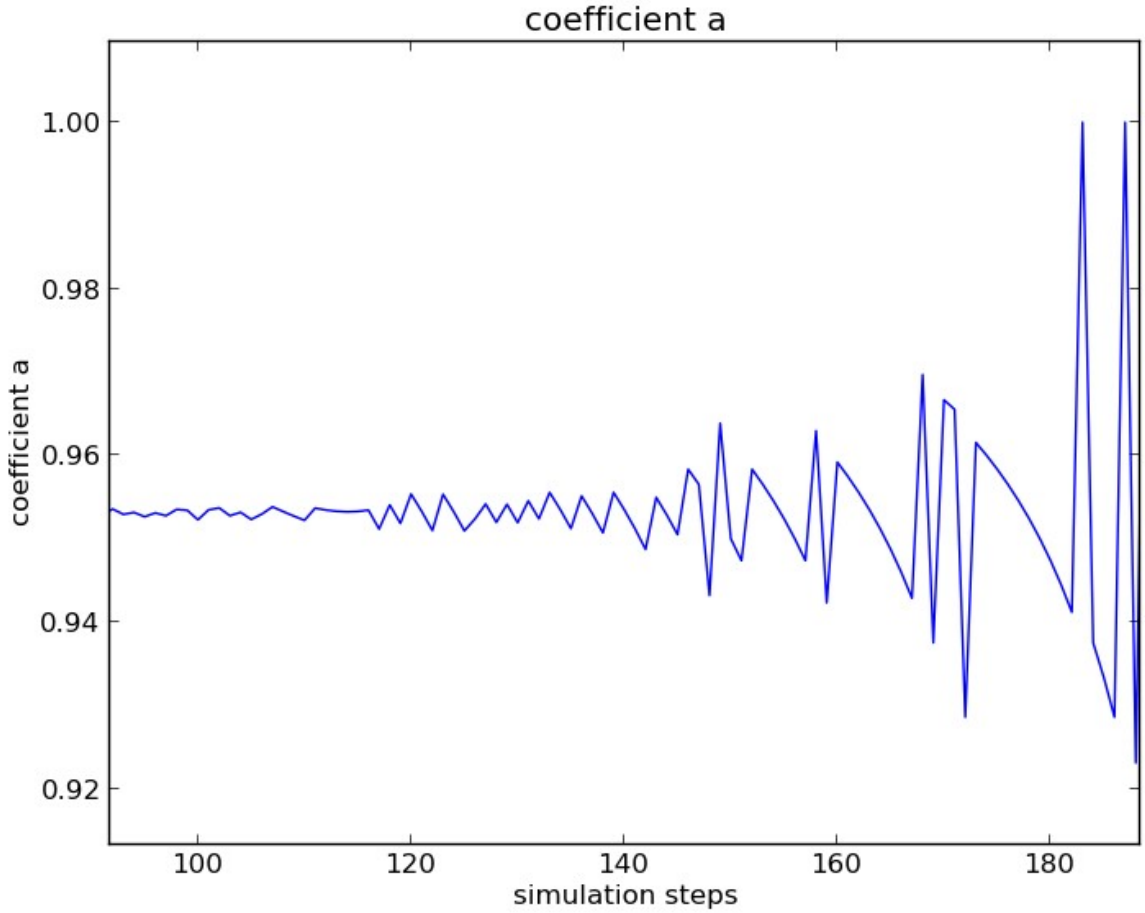

Illustration 30: Augmented version of previous illustration

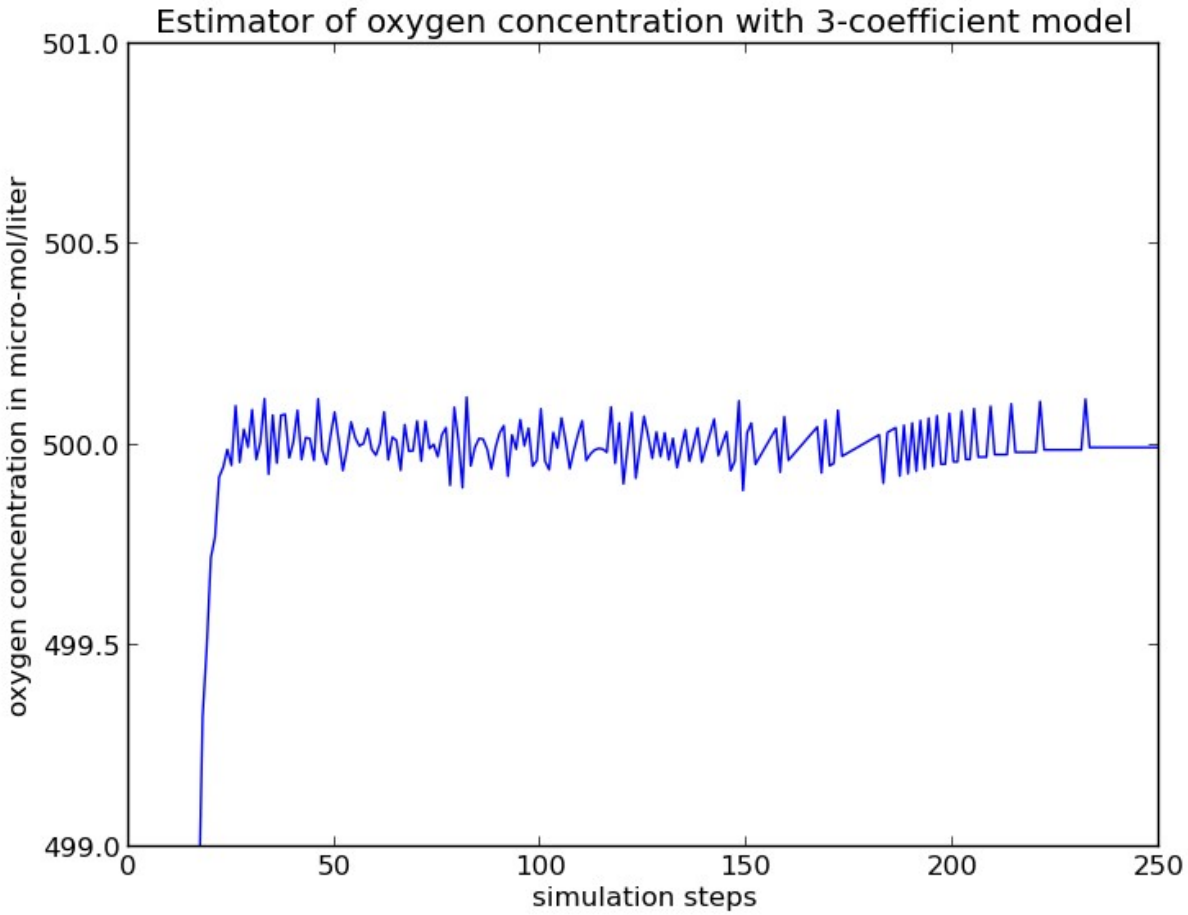

Illustration 31: Application of the 3-coefficient estimator for pure oxygen estimation. Based on illustration 28(last Illustration). The parameter a was chosen to be 0.9530842439196348. The oxygen diffusion constant in the FEM simulator was 300μm² /s. Notice that the model fails for the first steps. This is due to the fact that the 3-coefficient model assumes steady-flow, which takes time to be established.

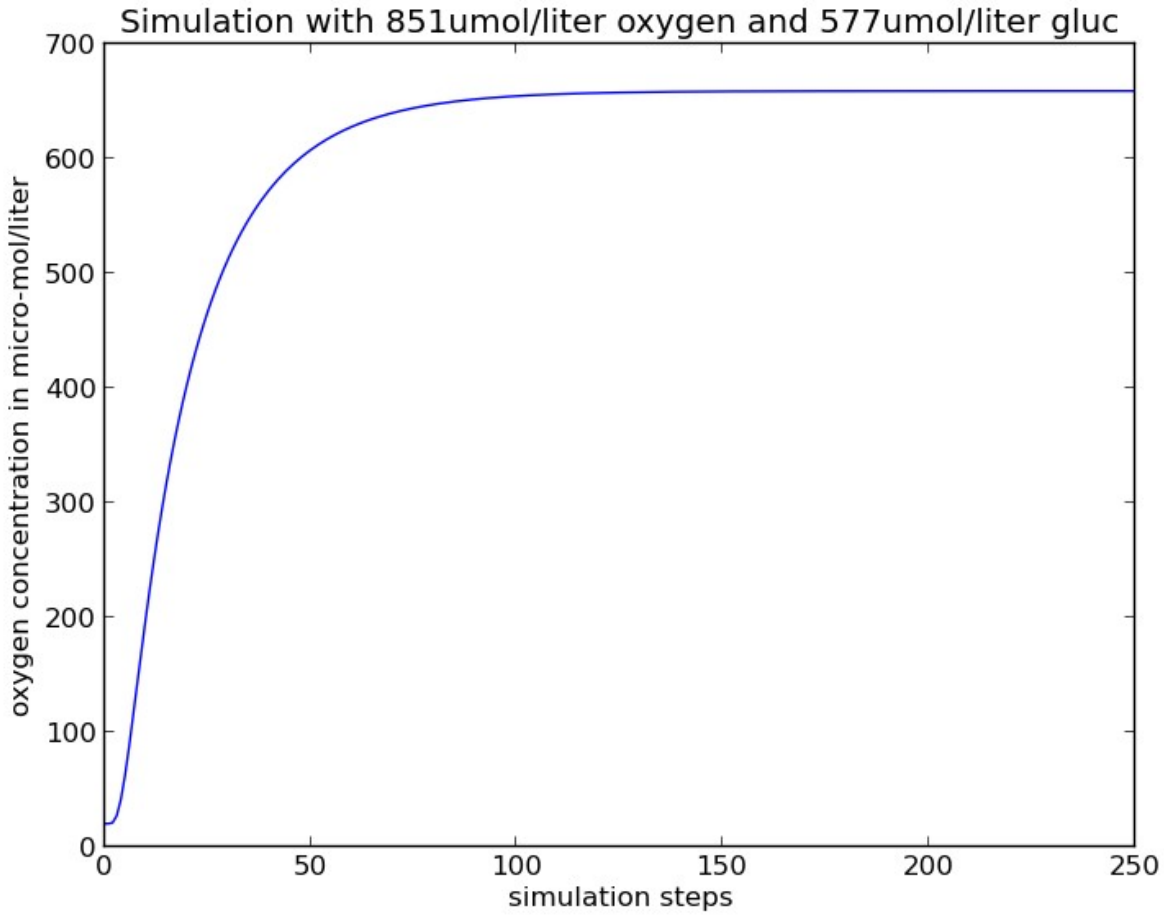

Illustration 32: Response of the FEM simulator to a test solution with an oxygen-concentration of 877µmol/liter and a glucose-concentration of 577µmol/liter. The equilibrium solution had an oxygen-concentration of 20µmol/liter. The oxygen diffusion constant was set to 300µm² /s and the glucose diffusion constant was 100µm² /s. Notice that the steady state concentration is not the oxygen-concentration of the test solution.

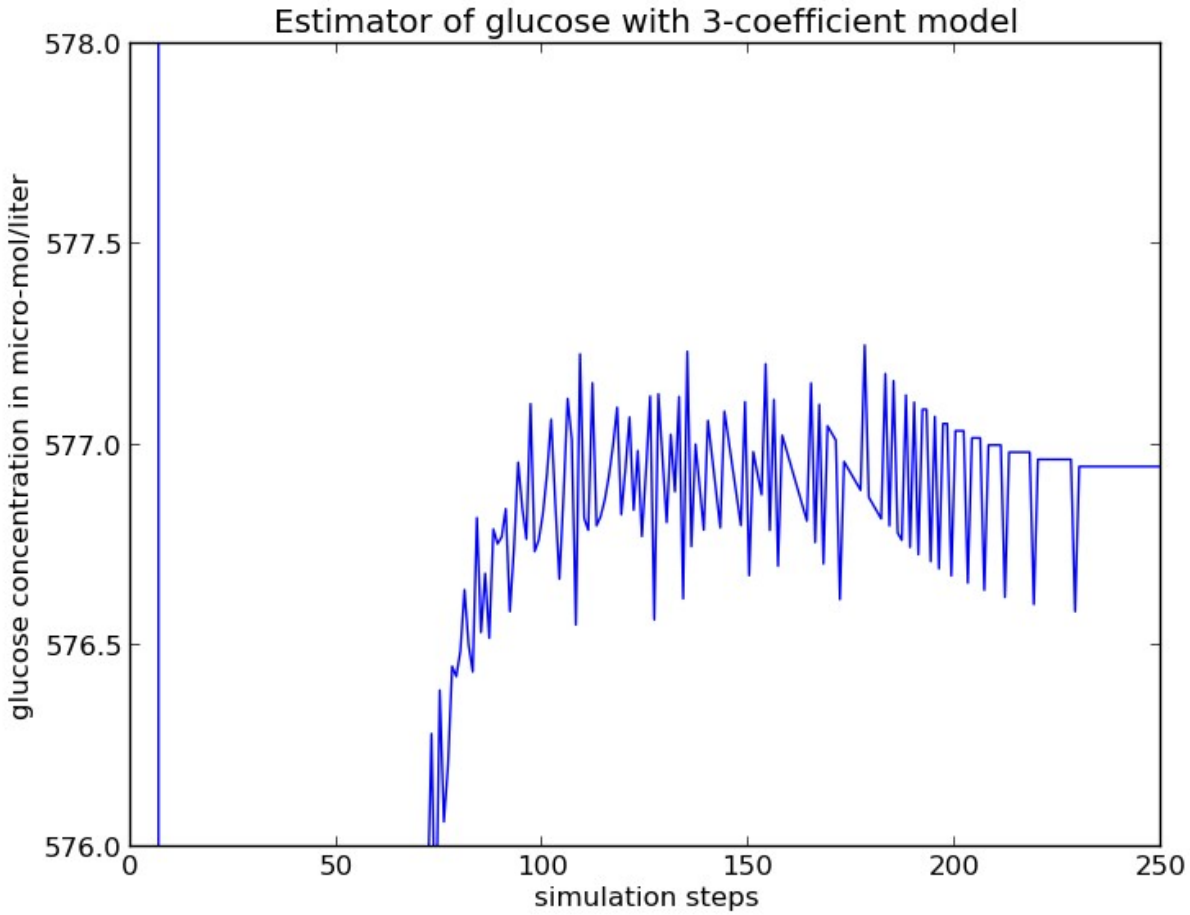

Illustration 33: Application of the 3-coefficient estimator for determining the glucose concentration. The parameter c was chosen to be -0.0156405127036. The oxygen diffusion constant in the FEM simulator was 300µm² /s and the glucose diffusion constant was 100µm² /s. Notice that the model fails for the first steps, like in the pure oxygen estimation, but the failing time is longer. This is due to the fact that glucose diffuses slower than oxygen. So the steady flow is reached much later than for pure oxygen diffusion.

9.4 Discussion

Oxygen estimation:

The response of the FEM simulator to a test solution with an oxygen-concentration of 500µmol/liter is plotted in illustration 28. It is important to notice that the response of the FEM simulator does not start immediately but rather with a short delay. This delay has to exist, because of the finite signal propagation. The 3-coefficient estimator is equivalent to the FEM simulator, except for this signal propagation delay. It should also be mentioned that the definition of the steady-state depends on the quantization of the FEM simulator. If the FEM simulator returned real numbers, which have an infinite precision, then the steady-state would never be reached. The FEM simulator, however, returns only quantized numbers with finite precision. Therefore, the following definition can be made.

Definition:

The steady-state is reached as soon as the consecutive quantized numbers become indistinguishable.

The steady-state occurs in the last 20 seconds in illustration 31, where there is no more oscillation in the estimator. The 3-coefficient model provides correct results for the steady-state. The proof of this statement can be found by a simple examination of the equation:

$$oxygen_{outside} = \frac{oxygen_{n+1} - oxygen_n \cdot a}{b} \tag{80}$$

With the steady-state condition:

$$k = oxygen_{outside} = oxygen_{n+1} = oxygen_n \tag{81}$$

It is easy to arrive at (82)

$$k = \frac{k - k \cdot a}{b} = \frac{k \cdot (1 - a)}{(1 - a)} = k \tag{82}$$

which must always be true.

So the limiting common sense case of equilibrium between the test solution and the sensor is included in the oxygen estimator.

Definition:

The FEM simulator is in steady-flow as soon as the 3-coefficient estimator starts to jitter around the right solution.

This oscillation would not occur for real numbers. The cause for the occurrence is the jitter of the output of the FEM simulator.

Glucose estimation:

The response of the FEM simulator to a test solution with an oxygen-concentration of 877µmol/liter, and a glucose-concentration of 577µmol/liter, is plotted in illustration 32. As in illustration 28 the FEM simulator has an initial delay time, until it reacts to the oxygen-concentration of the test solution. This initial delay time can be seen by visual inspection of the plot.

There is, however, a seconds initial phase that can not be seen by visual inspection.

This second initial phase can be seen in illustration 33, where the glucose estimator starts to provide the right estimation after about 75 time steps.

The diffusion of glucose is slower than the diffusion of oxygen. Oxygen will therefore reach its steady-flow first, and glucose second.

The steady state in illustration 32 is not an equilibrium as in illustration 28. The inflow of glucose will be constant as soon as it reaches steady-flow. The glucose-concentration inside the sensor will remain zero, due to the fact that oxygen reacts with glucose. The steady state occurs because the consumption of glucose is equal to the inflow of oxygen.

In the last sub-section it was easy to determine the oxygen-concentration in the steady state. It is possible to calculate the glucose-concentration in the steady state by a simple subtraction and multiplication.

Without any glucose the steady-state should be:

$$k = oxygen_{outside} = oxygen_{n+1} = oxygen_n \tag{83}$$

This, however, is not the case, because there is glucose in the test solution, but a steady state value **s** occurs anyway for the oxygen concentration inside the sensor:

$$oxygen_{n+1} = oxygen_n = s \neq oxygen_{outside} \tag{84}$$

By using equation (84) in the 3-coefficient estimator:

$$glucose = \frac{oxygen_{n+1} - oxygen_n \cdot a - oxygen_{outside} \cdot b}{c} \tag{85}$$

yields:

$$glucose = \frac{s - s \cdot a - oxygen_{outside} \cdot (1 - a)}{c} \tag{86}$$

or

$$glucose = \frac{s \cdot (1 - a) - oxygen_{outside} \cdot (1 - a)}{c} \tag{87}$$

or

$$glucose = \frac{(s - oxygen_{outside}) \cdot (b)}{c} \tag{88}$$

Or with the definition:

$$d = \frac{b}{c} \tag{89}$$

The final formula:

$$glucose = (s - oxygen_{outside}) \cdot d \tag{90}$$

or in a more readable form:

$$glucose = (oxygen_{steadystate} - oxygen_{outside}) \cdot d \tag{91}$$

So the difference of the steady-state oxygen concentration and the oxygen-concentration of the test solution, times a constant, equals the glucose concentration of the test solution.

The jitter is due to the quantization of the FEM simulator as in the last sub-section. If the outputs were real number, there would be no jitter.

The estimator correctly predicts the concentration in the steady flow phase for synthetic, noise free data from the FEM simulator. Its application to the real sensor will be the topic of the next chapter.

10 3-coefficient model for the real sensor

10.1 Introduction

In the last chapter the 3-coefficient model was applied to the FEM simulator. The direct application of this model to the measured data, however, leads to problems that are due to noise.
The 3-coefficient equation

$$glucose = \frac{oxygen_{n+1} - oxygen_n \cdot a - oxygen_{outside} \cdot b}{c} \tag{92}$$

does not make a distinction between the real oxygen concentration and the measured oxygen concentration. So the following model for the measured oxygen concentration is adopted:

$$measuredoxygen_n = oxygen_n + noise_n \tag{93}$$

$$measuredoxygen_{outside} = oxygen_{outside} + noise_{outside} \tag{94}$$

By replacing the real oxygen concentrations with the measured oxygen concentrations:

$$glucose = \frac{oxygen_{n+1} - oxygen_n \cdot a - oxygen_{outside} \cdot b}{c} + noise_{n+1}\frac{1}{c} + noise_n\frac{a}{c} + noise_{outside}\frac{b}{c} \tag{95}$$

Or all noise terms put together:

$$glucose = \frac{oxygen_{n+1} - oxygen_n \cdot a - oxygen_{outside} \cdot b}{c} + totalnoise_n \tag{96}$$

The source of this noise is quantization noise and also measurement noise.
This noise problem is so severe that it worsens the estimation considerably, because (78) recovers the parameter a from two numbers, which are very close to each other. Thus the ratio in (78) will be very sensitive to noise. The assumption of an ergodic signal will be tested. If this assumption is true, then the application of the central limit theorem would lessen the noise significantly, due to the huge number of measurements that are taken in the steady-flow region.

This approach does, however, not consider the temperature dependence of the sensor.
The next problem is that this model is a complete description of the FEM simulator in the steady flow range. The FEM simulator itself, however, even with a linear temperature dependence of the oxygen diffusion constant is not a 100%-valid description of the real sensor. From simple logic, it follows that the 3-coefficient estimator cannot be better than the FEM simulator.

The effects of noise are shown, and an untested noise reduction that was not implemented, due to the fact that the mass production of the sensor strip was already done.

10.2 Method

This section is indented to describe the problems from a theoretical perspective to analyze the noise problem. It is therefore an explanation of the results and not an improvement of them.

Noise problem:

In the last section the total noise originated from three terms:

$$totalnoise = noise_{n+1}\frac{1}{c} + noise_n\frac{a}{c} + noise_{outside}\frac{b}{c} \tag{97}$$

It is assumed that all noise terms have an expected value of zero. Therefore, the total noise has also an expected value of zero.

$$E[totalnoise] = E[noise_{n+1}]\frac{1}{c} + E[noise_n]\frac{a}{c} + E[noise_{outside}]\frac{b}{c} = 0 \tag{98}$$

The next question is the variation of the total-noise. The assumption is made that the noise at n+1 and n is independent. Therefore the covariance is zero:

$$cov(noise_{n+1}, noise_n) = 0 \tag{99}$$

The variance remains constant:

$$var(noise_{n+1}) = var(noise_n) = var(noise) \tag{100}$$

Therefore the total variance is

$$var(totalnoise) = \sigma^2 = var(noise)\left(\frac{1}{c}^2 + a\frac{2}{c}^2\right) + var(noise_{outside})b\frac{2}{c}^2 \tag{101}$$

The SNR of the one estimation is therefore:

$$SNR = \frac{glucose}{\sigma} \tag{102}$$

With the formula for glucose:

$$SNR = \frac{oxygen_{n+1} - oxygen_n \cdot a - oxygen_{outside} \cdot b}{c \cdot \sigma} \tag{103}$$

Chebyshev's inequality from probability theory states:

$$P[|(X - E[x])| \geq k\,\sigma] \leq \frac{1}{k^2} \tag{104}$$

$$\text{where } \sigma^2 \text{ is the variance} \tag{105}$$

This formula is true independent of the distribution of X!

k	Min % within k standard deviations of mean	Max % beyond k standard deviations from mean
1	0%	100%
$\sqrt{2}$	50%	50%
1.5	55.56%	44.44%
2	75%	25%
3	88.8889%	11.1111%
4	93.75%	6.25%
5	96%	4%
6	97.2222%	2.7778%
7	97.9592%	2.0408%
8	98.4375%	1.5625%
9	98.7654%	1.2346%
10	99%	1%

Table 1: Precalculated values for Chebyshev's inequality

The probability that the true glucose value lies within an interval kσ of the estimator can be calculated from (104).

Noise reduction by matrix inversion

The sensor strips were already created by mass production. Therefore the following sub-section is only a suggestion that couldn't be implemented. The first sensor, as already mentioned, is governed by the 3-coefficient equation:

$$oxygen_{n+1} = oxygen_n \cdot a + oxygen_{outside} \cdot b + glucose \cdot c \tag{106}$$

A second sensor that would be equivalent to the first, except for the missing GOX, would be governed by the following equation.

$$oxygen_{n+1} = oxygen_n \cdot a + oxygen_{outside} \cdot b \tag{107}$$

The difference between these two sensors would be:

$$differencesignal = senor\,1 - senor\,2 = glucose \cdot c \tag{108}$$

So the the glucose concentration, in steady flow, could simply be determined by dividing the difference-signal by c:

$$glucose = \frac{differencesensor}{c} \tag{109}$$

The temperature dependence of the oxygen diffusion constant should be equal for both sensors. Therefore, the temperature dependence of the oxygen diffusion is eliminated by subtraction. The only temperature dependence would be the temperature dependence of the glucose diffusion constant. So the estimator is only dependent on temperature variations in c.

10.3 Results

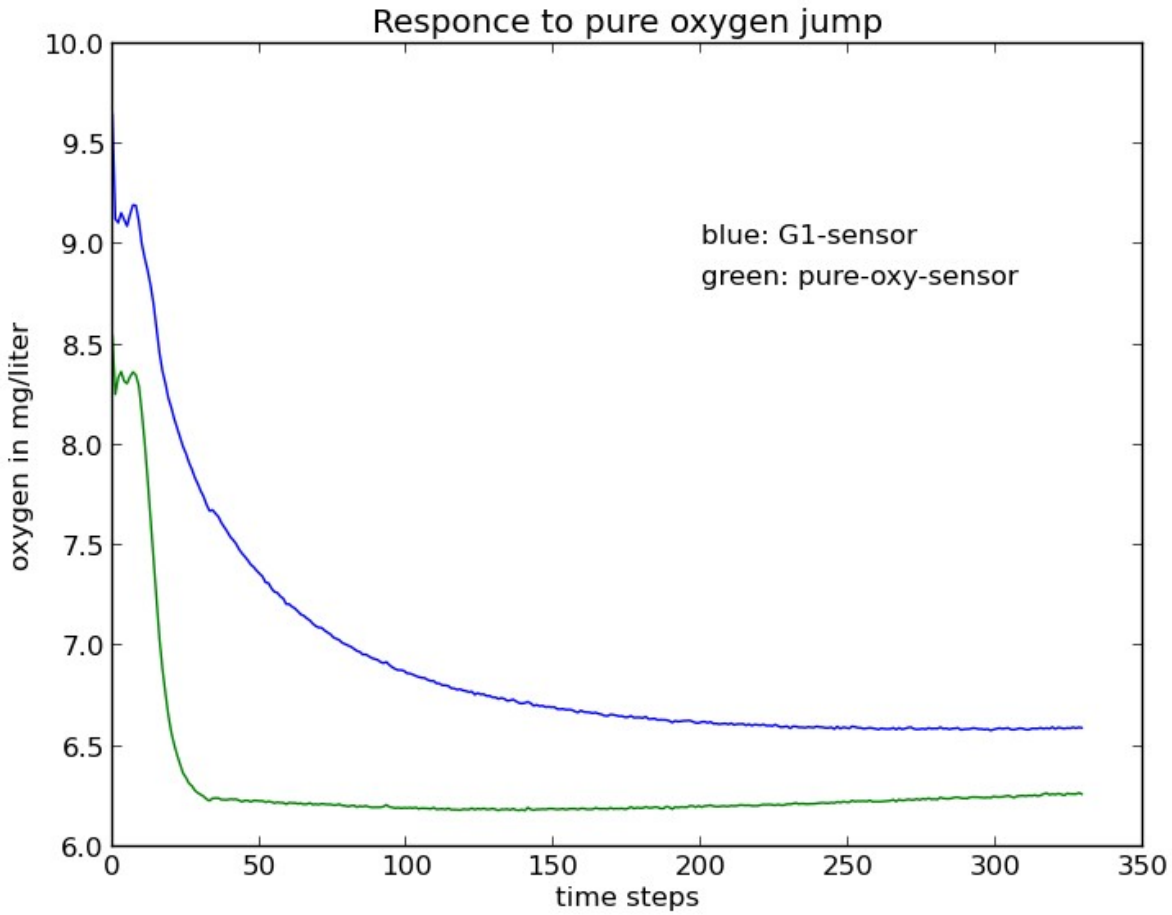

Illustration 34: Response of the glucose1-sensor and pure-oxy-sensor to a pure change in oxygen concentration. Notice that the pure oxygen sensor is much faster and has an offset of about ~0.32. The steady state of the pure-oxy-sensor is not completely flat! (The source for this discrepancy is unknown)

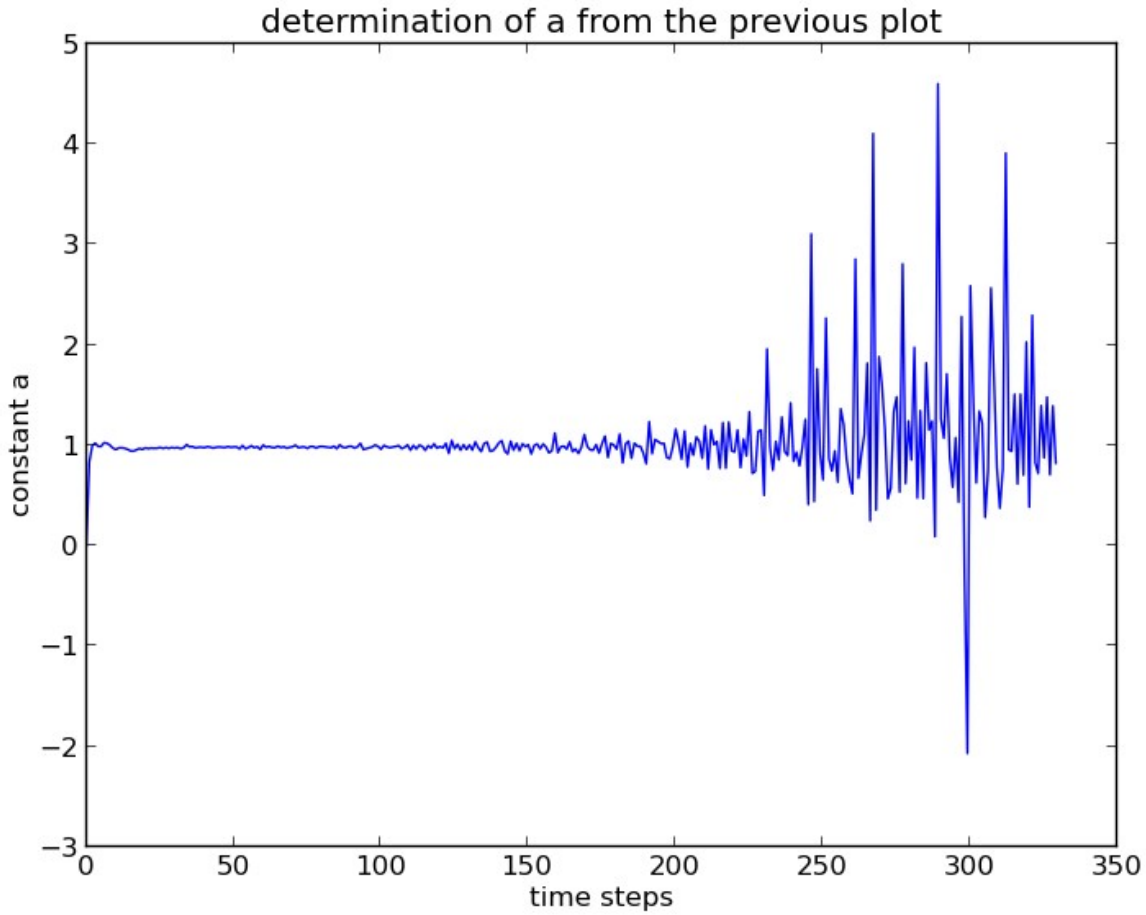

Illustration 35: Determination of the 3-coefficient constant a from the previous plot with (78). Notice that the constant has almost no variation at first and then starts to oscillate very strongly in the last interval. The difference between sample n and n+1 gets very low and noise dominates, when the blue oxygen curve in illustration (34) flattens out.

So it is better to choose the 3-coefficient constant a from a large slope in illustration (34)

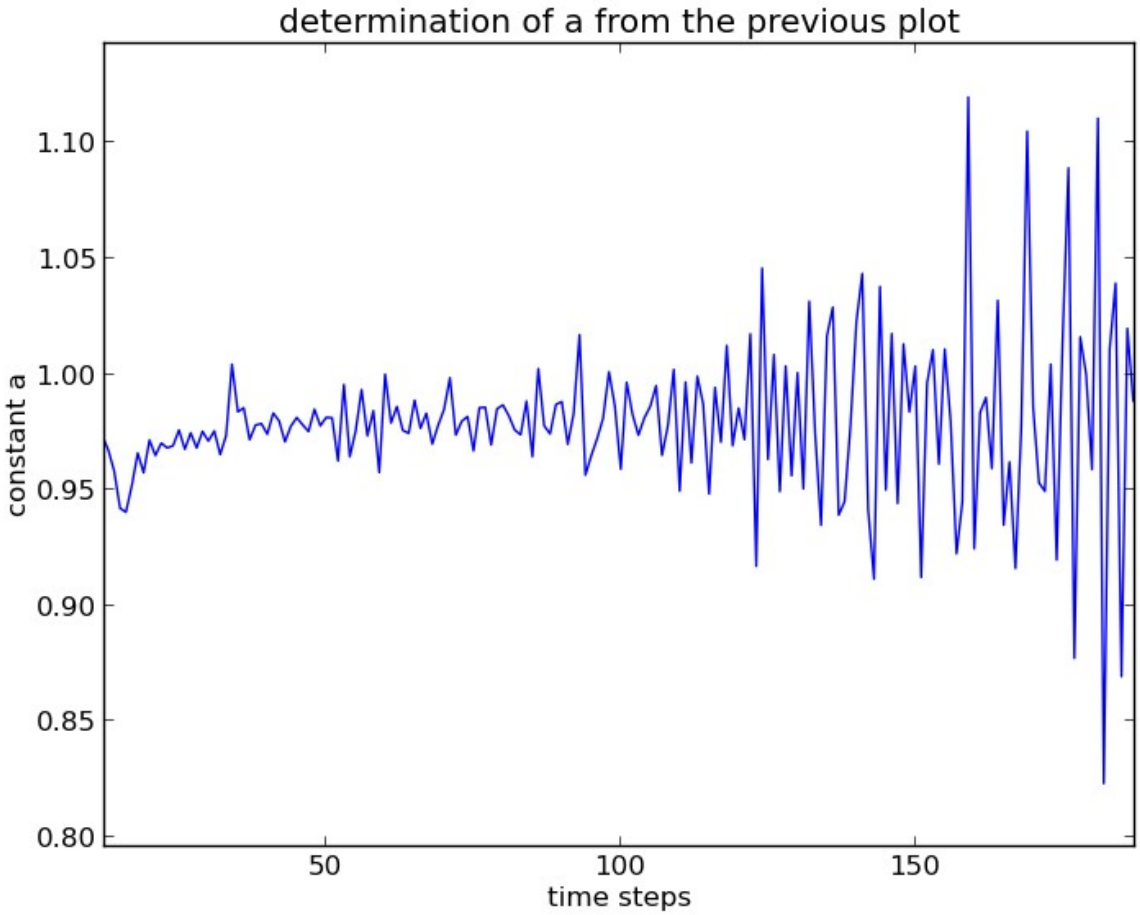

Illustration 36: Same plot as before, but now magnified. The constant a was chosen to be 0.985. Averaging had no effect on the jitter.

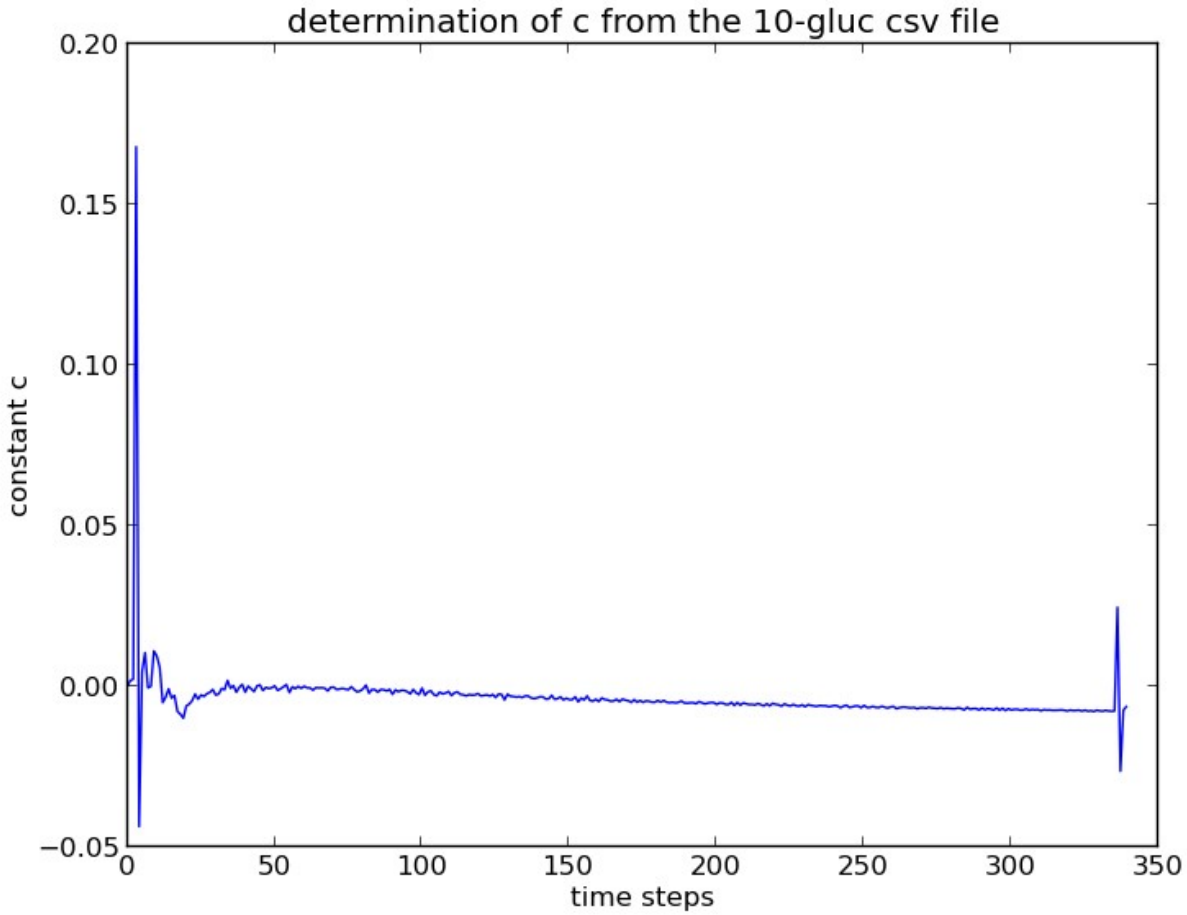

Illustration 37: Determination of the 3-coefficient constant c from the 10-gluc.csv file

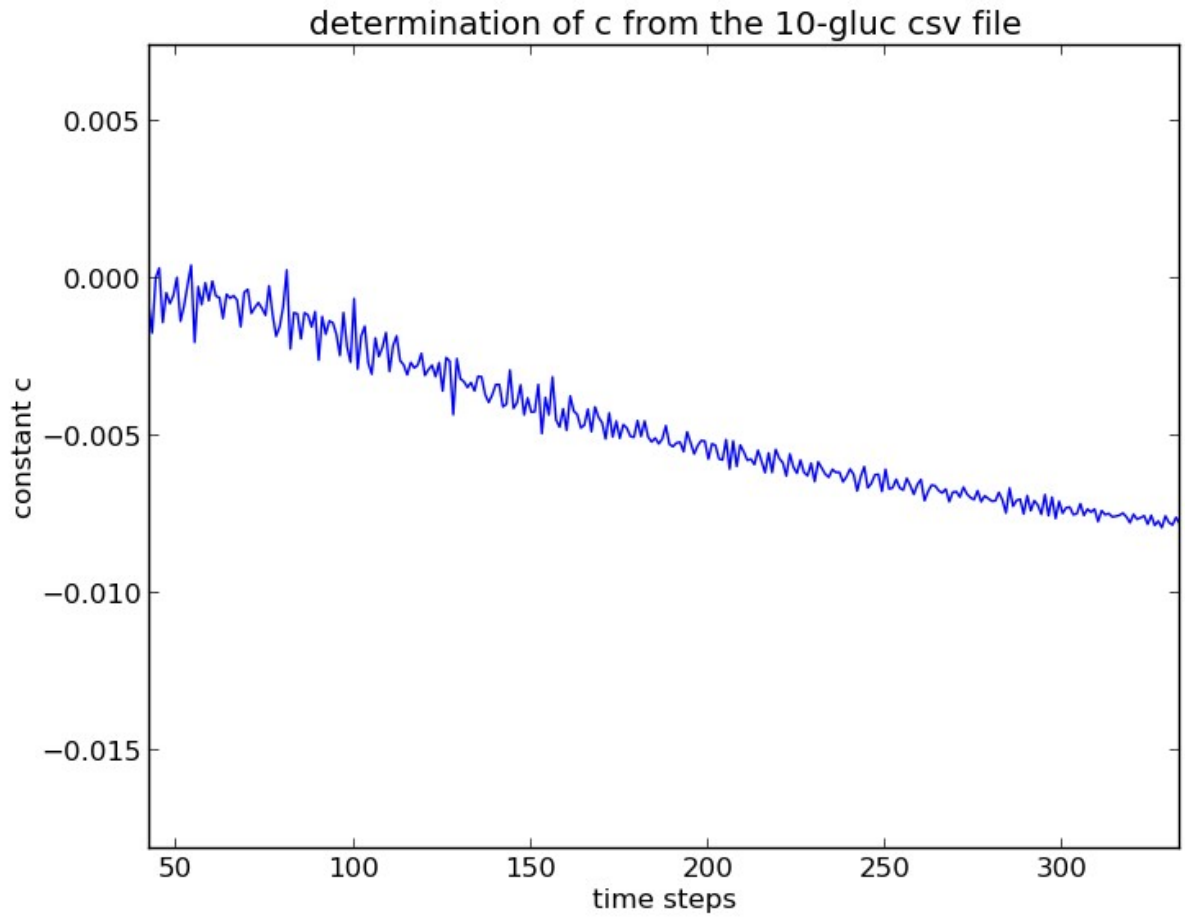

Illustration 38: Determination of 3-coefficient constant c from the 10-gluc.csv file, but this time magnified. The constant c was chosen to be -0.0078. The last value was chosen because in the previous chapter the later values for c were also more trustworthy, due to established steady flow.

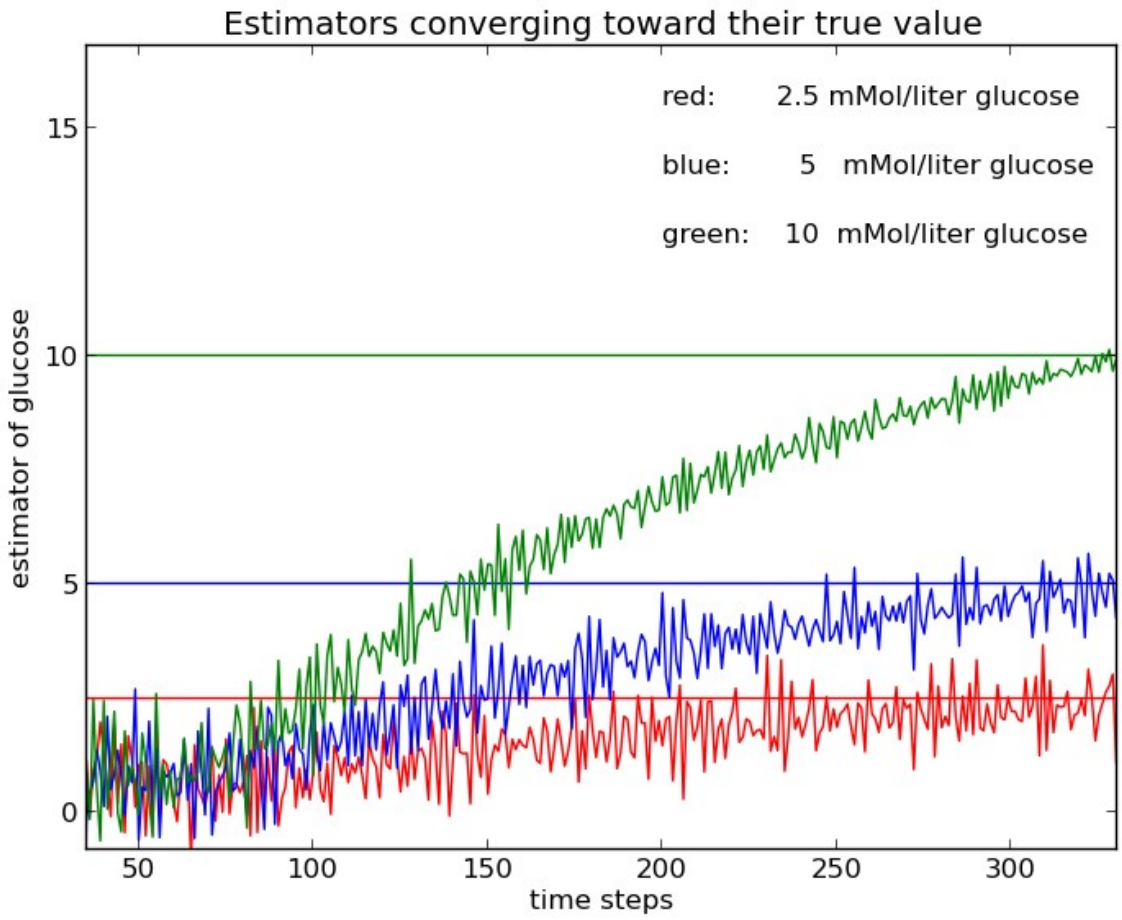

Illustration 39: The estimator has the following parameters

a = 0.985

b = 1-a = 0.015

c = -0.0078

The oxygen value was not take from the pure-oxy-sensor but assumed to be

oxy= 6.58

The estimator has not to provide an estimation for all time steps. It is sufficient if it provides one good estimator for one known time step

The 3-coefficient model works very well in the range between 0 and 10 mMol glucose per liter.

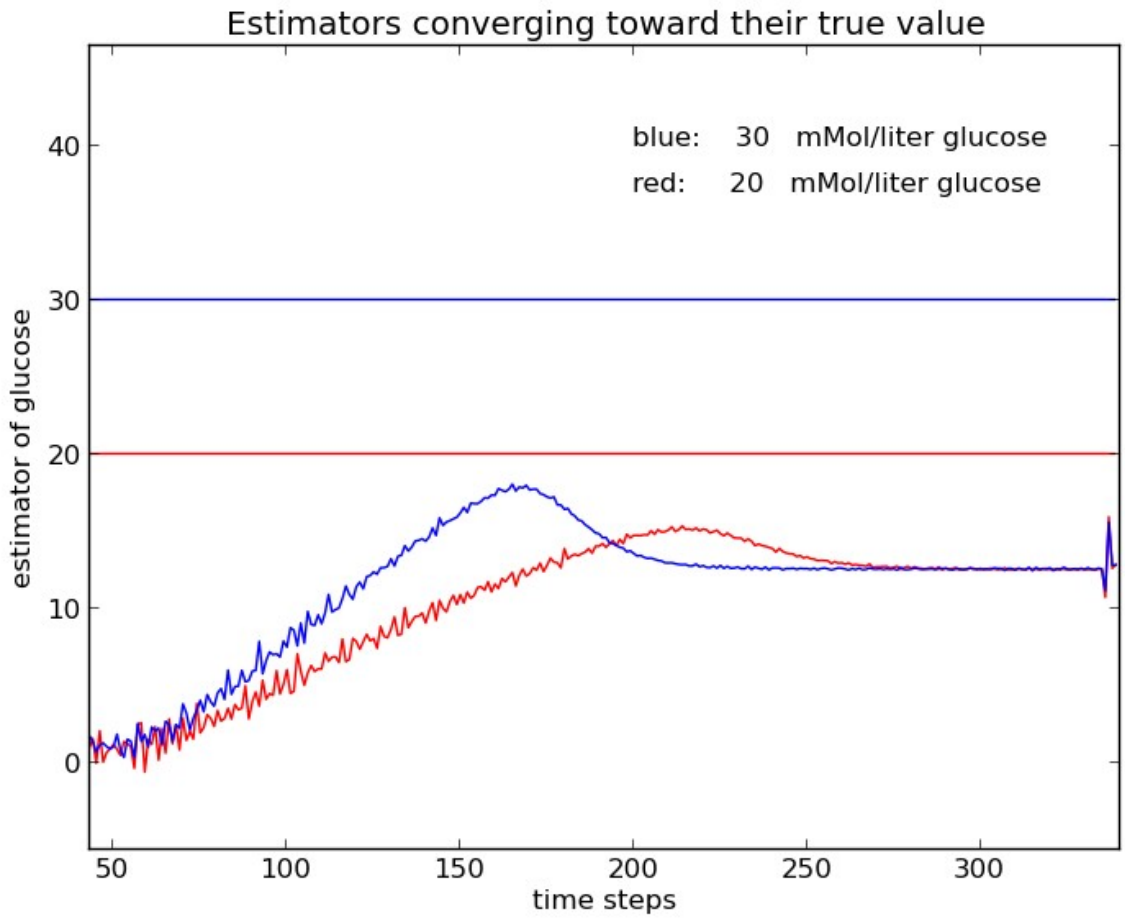

Illustration 40: The estimator has the following parameters

a = 0.985

b = 1-a = 0.015

c = -0.0078

The oxygen value was not take from the pure-oxy-sensor but assumed to be

oxy= 6.58

The 3-coefficient model doesn't works very well in the range between 20 and 30 mMol glucose per liter.

10.4 Discussion

Illustration 34 shows the response of the glucose sensor and pure-oxygen sensor to a test solution with only oxygen. The pure-oxygen sensor is much faster, due to its specific design. The steady-state equilibrium concentration is, however, not the same, although it should be.
The source for this discrepancy is unknown.

This offset error is about 0.32.

The discrepancy between the pure-oxygen sensor and the glucose sensor is very problematic for the 3-coefficient estimator. The estimator is not capable of providing a good estimation of the glucose concentration, without an accurate knowledge of the outside oxygen concentration.

A possibility would be to use the glucose2-sensor, which obeys the following equation

$$oxygen_{n+1} = oxygen_n \cdot a_2 + oxygen_{outside} \cdot b_2 + glucose \cdot c_2 \tag{110}$$

while the glucose1-sensor obeys:

$$oxygen_{n+1} = oxygen_n \cdot a_1 + oxygen_{outside} \cdot b_1 + glucose \cdot c_1 \tag{111}$$

a rearrangement the two equations into one matrix equation:

$$\begin{bmatrix} (oxygen_{n+1} - oxygen_n \cdot a_1) \\ (oxygen_{n+1} - oxygen_n \cdot a_2) \end{bmatrix} = \begin{bmatrix} (b_1)(c_1) \\ (b_2)(c_2) \end{bmatrix} . \begin{bmatrix} oxygen_{outside} \\ glucose \end{bmatrix} \tag{112}$$

Or by inversion

$$\begin{bmatrix} oxygen_{outside} \\ glucose \end{bmatrix} = \begin{bmatrix} (b_1)(c_1) \\ (b_2)(c_2) \end{bmatrix}^{-1} . \begin{bmatrix} (oxygen_{n+1} - oxygen_n \cdot a_1) \\ (oxygen_{n+1} - oxygen_n \cdot a_2) \end{bmatrix} \tag{113}$$

The inversion of the matrix was, however, not possible, due to the similar design of both sensor. This made the problem ill posed.
The best design of the second sensor would be a complete independence on glucose(no enzyme), as already mentioned in this chapter.

The 3 coefficient constant a was chosen from the middle part of illustration 36. The quantization caused wild oscillations in the later part. The 3 coefficient constant c was chosen from illustration 38. The later value was chosen for steady-flow reasons.

The estimation of the glucose-concentration, by the 3 coefficient estimator, is shown in illustration 39. The real glucose-concentrations are the straight lines of the corresponding colors. The estimators come closer and closer to the true value. The slow convergence toward the real value is probably caused by several factors.

1. The non steady-flow period before steady flow of glucose

2. The temperature independence of the 3 coefficient model

3. The inaccurate knowledge of the outside oxygen-concentration

The same was done for higher glucose-concentration in illustration 40. The estimator failed, this time, to provide good estimations. A look at illustration 27 shows that the sensor goes into saturation in the later part of the plot. The sensor would have to provide a signal below zero in order for the 3-coefficient model to work. So the failure of the 3-coefficient model in illustration 40 is due to the unmodeled saturation of the sensor above the zero signal.

The normal glucose levels in non diabetics is 1g per liter, which is about 5mMol per liter. Therefore the sensor would work very well in the normal glucose range. The strange form of the glucose-concentration estimator curves at higher concentrations would be a sufficient condition for making the diagnosis of hyperglycemia.

11 Superposition based model

11.1 Introduction

The previous chapter tried to model the sensor by very simple assumptions without any considerations of temperature variation effects. This approach was partially capable of modeling the sensor but had a lot of disadvantages. The first disadvantage was that the outside oxygen concentration had to be known. This seems at first like something easy, given that there is a fast pure-oxygen-sensor nearby. However, this was practically not so simple. The pure-oxygen-sensor doesn't consume any oxygen, while the glucose sensor starts to consume oxygen in the presence of glucose. This makes the estimation of the glucose concentration difficult, because any good estimation has to be based on the oxygen-concentration present at the consuming sensor.

An analog problem from electrical engineering would be

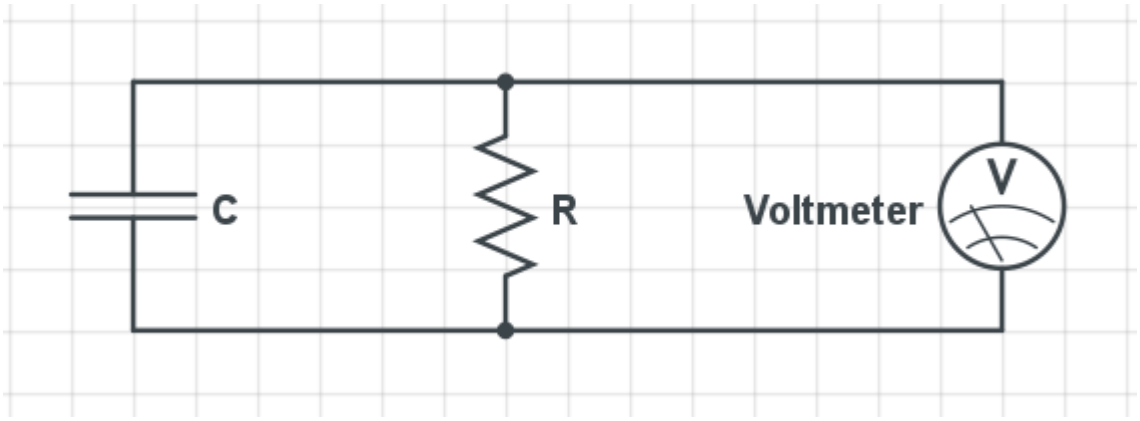

Illustration 41: Analog model of the glucose sensor. The measured voltage doesn't stay the same due to the power consumption of the resistance.

Illustration 42: Analog model of pure oxygen sensor. The voltage stays the same.

The problem is that one can not take the voltage from illustration 41 and assume that it stays equal to the voltage in illustration 42

The next problem arose from the exact determination of the 3-coefficient parameters. Finally, the inversion of the model suffers from noise, quantization error, and the inaccurate c-parameter, which is a very small denominator(yields strong noise amplification).

The goal of this chapter is, therefore, to find an estimator that minimizes or discards all of the aforementioned problems.
So the new estimator should have the following properties:

The outside oxygen concentration is derived from the sensor itself, and not from another sensor.

There should be nothing to determine that involved an ambiguous choice of coefficients (see illustration 36), like in the 3-coefficient model.

The inversion should be less sensitive to noise and quantization errors.

The problem of temperature variation should at least be partially solved.

Although, this estimator seems at first very difficult to derive, it turns out to be easier than the 3-coefficient model.

This estimator will show the nature of the sensor in a very visual way, and make further improvements very intuitive.

11.2 Method

So far, the basic assumption of the models was linearity in all variables. The basic problem lies apparently in the variation of temperature. If the influence of temperature variation is not linear, it is still possible to use superposition, given that the temperature variation is always the same!

There are five csv files with different oxygen concentrations and no glucose.

There are measurements with the following glucose concentrations.

zero glucose
2.5 mMol per liter glucose
5 mMol per liter glucose
10 mMol per liter glucose
20 mMol per liter glucose
30 mMol per liter glucose
all with the same oxygen concentration

A very general statement can be made about all these measurements:

At the start there is an oxygen concentration inside the sensor, which is known, and equals the oxygen concentration in the equilibrium solution at the outside.
In the next moment another oxygen concentration and glucose concentration is applied to the outside, by the test solution.

The oxygen-concentration of the test solution is only known for pure oxygen test solutions.

A general graph of the input functions would look like:

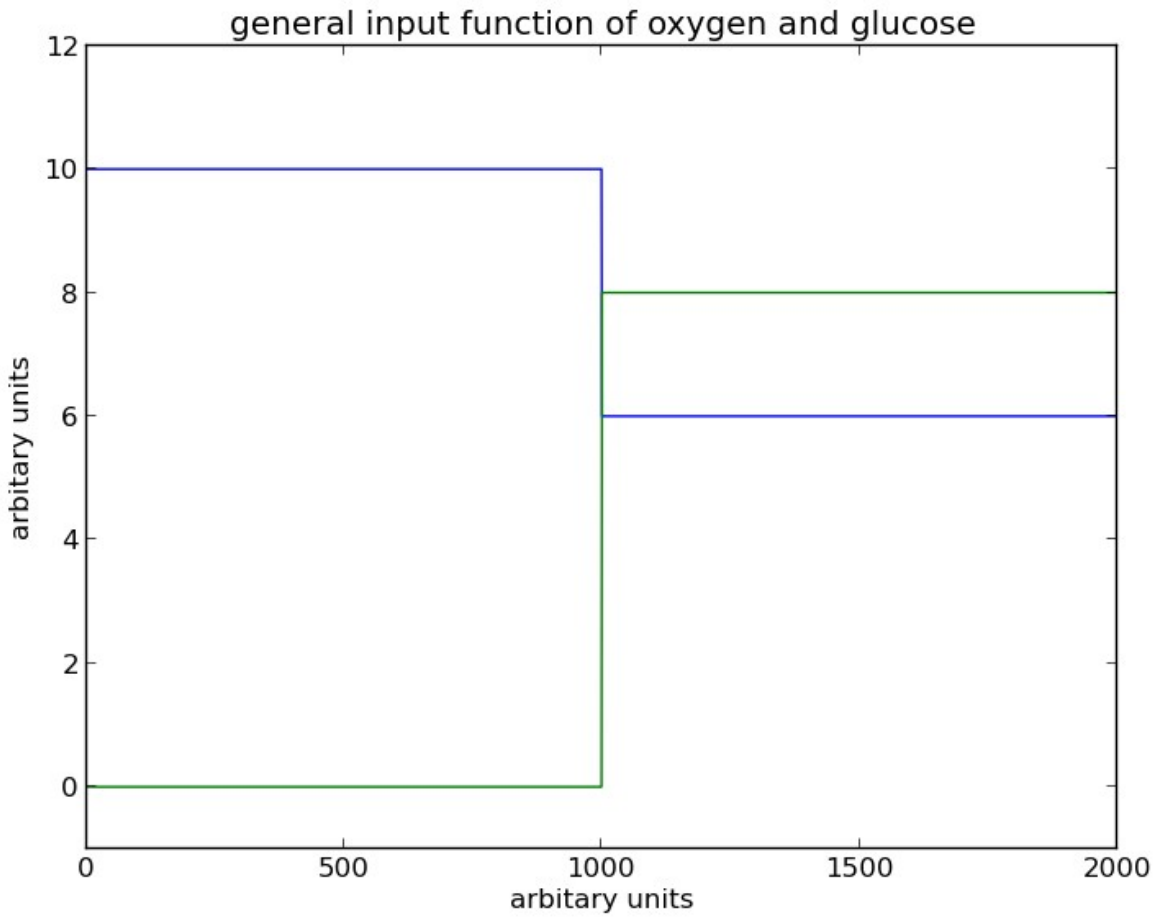

Illustration 43: The general input function of oxygen(blue) and glucose(green), that is applied to the outside of the sensor

The general input function is drawn in illustration 43. At the beginning there is a certain constant oxygen-concentration at the outside of the sensor and no glucose. After 1000 arbitrary units the oxygen-concentration changes its value to another constant value and the glucose-concentration at the outside rises to a constant value.

The general input of illustration 43 can be subdivided into three inputs that occur at the same time.

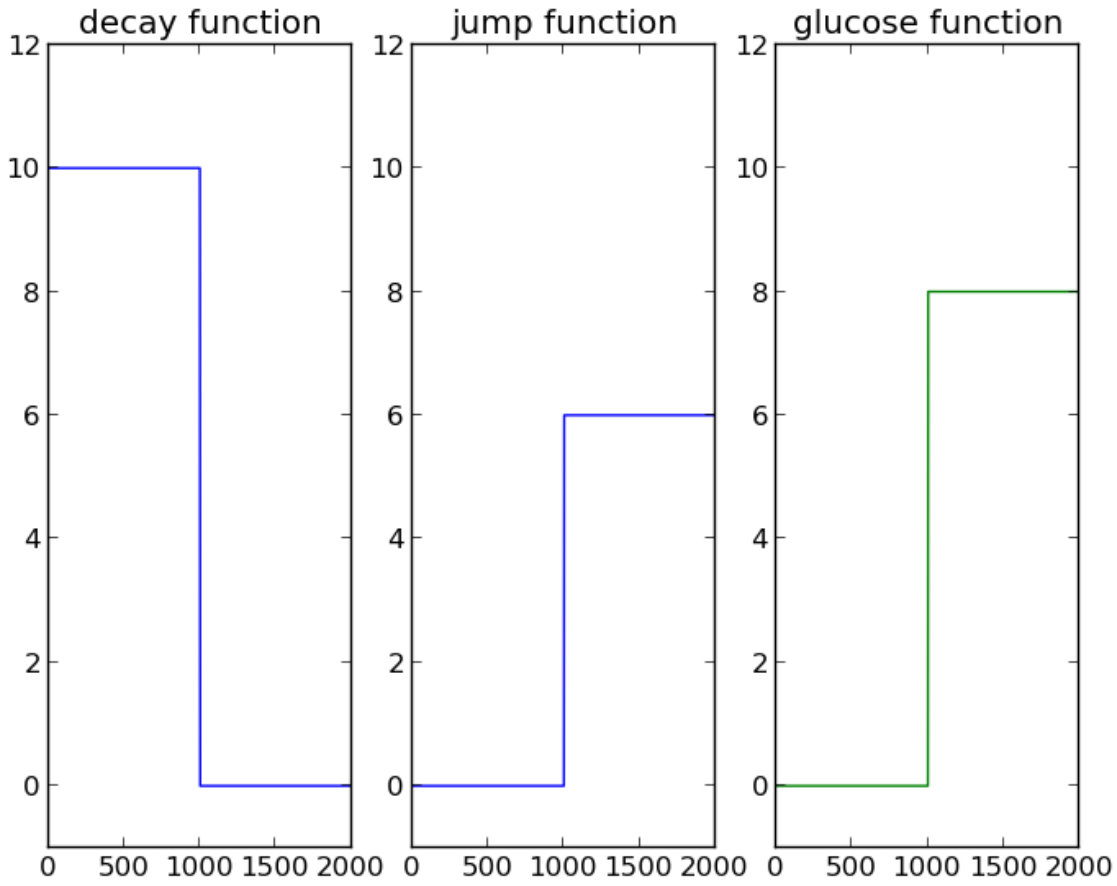

Illustration 44: The split of the general input function into the three input functions:

The constant oxygen-concentration of the equilibrium solution will be called the oxygen decay input.

The constant oxygen-concentration of the test solution will be called the oxygen jump input.

The constant glucose-concentration of the test solution will be called glucose jump input

The general input curve is a linear superposition of theses three input curves.

The mathematical description of this superposition is:

$$\vec{input} = \begin{Bmatrix} oxy-decay \\ oxy-jump \\ glucose-jump \end{Bmatrix} = \begin{Bmatrix} oxy-decay \\ 0 \\ 0 \end{Bmatrix} + \begin{Bmatrix} 0 \\ oxy-jump \\ 0 \end{Bmatrix} + \begin{Bmatrix} 0 \\ 0 \\ glucose-jump \end{Bmatrix} \tag{114}$$

Or written in unit vectors:

$$\vec{input} = \begin{Bmatrix} oxy-decay \\ oxy-jump \\ glucose-jump \end{Bmatrix} = oxy-decay \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + oxy-jump \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + glucose-jump \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{115}$$

This expression for the input is general. It is capable of representing all inputs, by varying the coefficients of the unit vectors.

The sensor appears to be dependent on the input, but also on the temperature variation. The problem is that the dependence on temperature variation is unknown, while the dependence on the input is linear.

$$output = SYSTEM \begin{Bmatrix} . \\ a\,input1 + b\,input2 \\ . \end{Bmatrix} = a\,SYSTEM \begin{Bmatrix} . \\ input1 \\ . \end{Bmatrix} + b\,SYSTEM \begin{Bmatrix} . \\ input2 \\ . \end{Bmatrix} \tag{116}$$

$$output = SYSTEM \begin{Bmatrix} . \\ a\,temp1 + b\,temp2 \\ . \end{Bmatrix} \neq a\,SYSTEM \begin{Bmatrix} . \\ temp1 \\ . \end{Bmatrix} + b\,SYSTEM \begin{Bmatrix} . \\ temp2 \\ . \end{Bmatrix} \tag{117}$$

This non-linearity in temperature variation does, however, not affect the linearity of the input:

$$output = SYSTEM \begin{Bmatrix} a\,input1 + b\,input2 \\ temp \end{Bmatrix} = a\,SYSTEM \begin{Bmatrix} input1 \\ temp \end{Bmatrix} + b\,SYSTEM \begin{Bmatrix} input2 \\ temp \end{Bmatrix} \tag{118}$$

SYSTEM symbolizes a general system with the mentioned properties. For the purpose of the master it means, however, the system of the sensor.

The idea is to find the 3 isolated unit responses from the given csv-files.

In order to proceed to the responses of the unit inputs, it is first necessary to calculate the unit input functions from the measurements. This can be done by linear algebra, or by a more intuitive approach.

The files 100.csv and 50.csv contain a different oxygen jump input and no glucose. The initial value is in equilibrium with the according oxygen decay input. The steady state end value must be in equilibrium with the according oxygen jump. So the corresponding oxygen decay input and oxygen jump input are known for both files. So the base vectors can be constructed by superpositions from these files.

The oxy-decay input function represents an equilibrium solution with a normalized(unity) oxygen-concentration, followed by a test solution without glucose and without oxygen. This virtual input can be constructed by:

$$\vec{oxydecay} = \begin{Bmatrix} 1 \\ 0 \\ 0 \end{Bmatrix} = \frac{100.csv - \frac{(endvalue\ of\ 100.csv)}{(endvalue\ of\ 50.csv)} \cdot 50.csv}{inital\ value\ of\ numerator} \tag{119}$$

The oxy-jump input function represents an equilibrium solution with no oxygen, followed by a test solution with a normalized oxygen-concentration and no glucose. The virtual input can be constructed by:

$$\vec{oxyjump} = \begin{Bmatrix} 0 \\ 1 \\ 0 \end{Bmatrix} = \frac{100.csv - (initalvalue\ of\ 100.csv\ file) \cdot oxydecay}{end\ value\ of\ numerator} \tag{120}$$

The file 5-gluc.csv is at the beginning in equilibrium with its oxygen decay input. The oxygen jump input is obtained from the value of the pure oxygen sensor. If these known inputs are subtracted, then there is only the glucose jump input left. The 5-gluc.csv file contains 5mMol glucose per liter. So the normalization is done by dividing by 5.

The glucose-jump function represents an equilibrium solution with no oxygen, followed by a test solution with no oxygen and a normalized glucose-concentration. The virtual input can be constructed by:

$$\vec{glucjump} = \begin{Bmatrix} 0 \\ 0 \\ 1 \end{Bmatrix} = \frac{5 - gluc.csv - (initial\ value\ of\ 5 - gluc.csv) \cdot oxydecay - (pureoxy\ sensor)\ oxyjump}{5} \tag{121}$$

Whatever superposition is true for the unit inputs, must also be true for the unit outputs.

$$SYSTEM \begin{Bmatrix} 1 \\ 0 \\ 0 \end{Bmatrix} = \frac{100.csv - \frac{(endvalue\ of\ 100.csv)}{(endvalue\ of\ 50.csv)} \cdot 50.csv}{inital\ value\ of\ numerator} \tag{122}$$

$$SYSTEM \begin{Bmatrix} 0 \\ 1 \\ 0 \end{Bmatrix} = \frac{100.csv - (initalvalue\ of\ 100.csv\ file) \cdot oxydecay}{end\ value\ of\ numerator} \tag{123}$$

$$SYSTEM \begin{Bmatrix} 0 \\ 0 \\ 1 \end{Bmatrix} = \frac{5 - gluc.csv - (initial\ value\ of\ 5 - gluc.csv) \cdot decay - (pureoxy\ sensor)\ oxyjump}{5} \tag{124}$$

The inversion of this model is a simple problem from linear algebra. Suppose the sensor provides the output in form of the vector:

$$\vec{output}$$

The question is now:
What input caused this output?
The input was a superposition of the normalized input functions.

$$\vec{input} = x_1 \vec{oxydecay} + x_2 \vec{oxyjump} + x_3 \vec{glucjump} \tag{125}$$

So the output response can be expressed as:

$$\vec{ouput} = x_1 SYSTEM \begin{Bmatrix} 1 \\ 0 \\ 0 \end{Bmatrix} + x_2 SYSTEM \begin{Bmatrix} 0 \\ 1 \\ 0 \end{Bmatrix} + x_3 SYSTEM \begin{Bmatrix} 0 \\ 0 \\ 1 \end{Bmatrix} \tag{126}$$

So the problem reduces to:

$$\vec{ouput} = A\vec{x} \tag{127}$$

Where the matrix A is:

$$A = colum \left\{ SYSTEM \begin{Bmatrix} 1 \\ 0 \\ 0 \end{Bmatrix}, SYSTEM \begin{Bmatrix} 0 \\ 1 \\ 0 \end{Bmatrix}, SYSTEM \begin{Bmatrix} 0 \\ 0 \\ 1 \end{Bmatrix} \right\} \tag{128}$$

And where x are the coefficients

$$x = \begin{Bmatrix} oxy-decay \\ oxy-jump \\ glucose-jump \end{Bmatrix} \tag{129}$$

By adding noise to the output and remembering that A is not quadratic.

$$\vec{ouput} = A\vec{x} + \vec{noise} \tag{130}$$

The solution can simply be found by:

$$\vec{x} = pinv(A)\vec{output} \tag{131}$$

where pinv(A) means the pseudo-inverse of A.
The third entry of x(glucose-jump) is the predictor or estimator of the glucose-concentration

11.3 Results

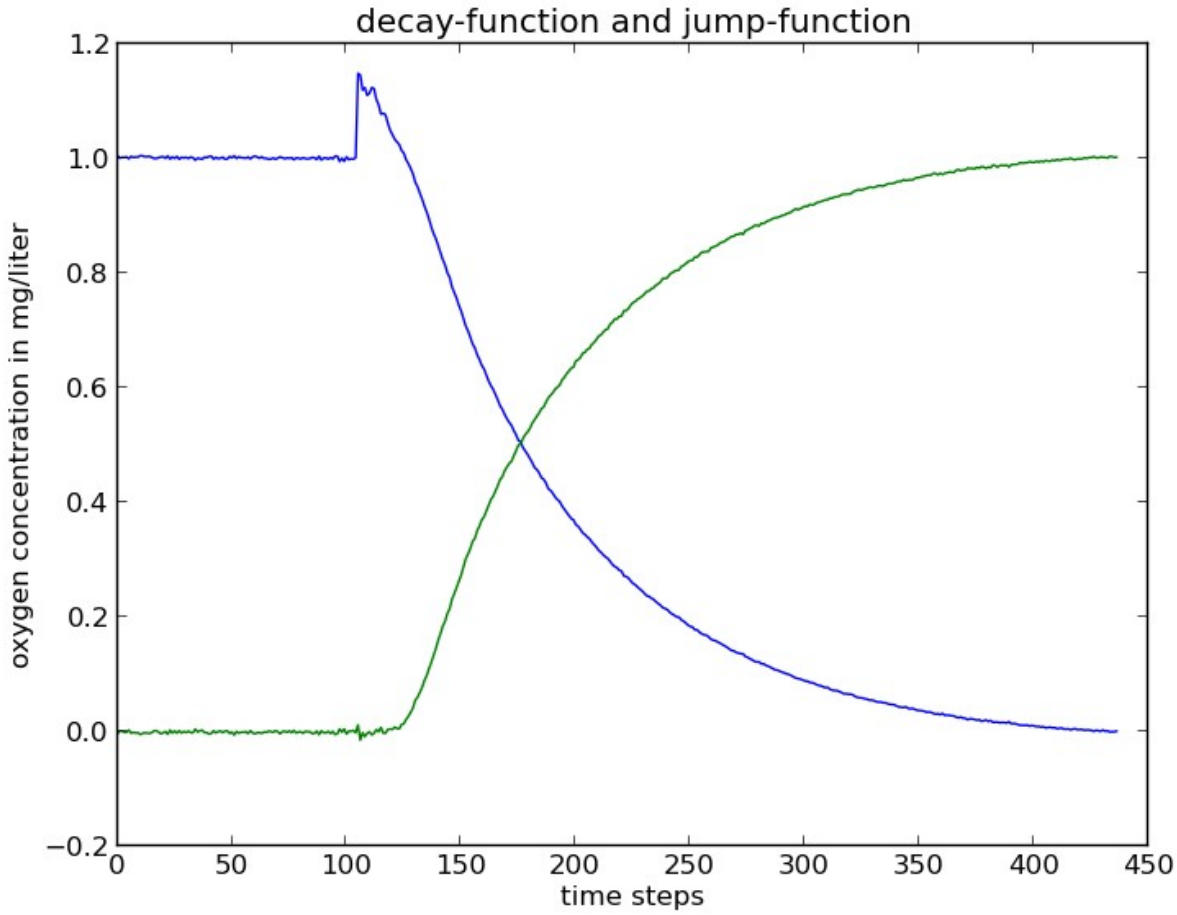

Illustration 45: The theoretical functions

oxygen-decay (blue)

and

oxygen-jump (green)

These functions were obtained from 100.csv and 50.csv.

These function will be used as unit output vectors

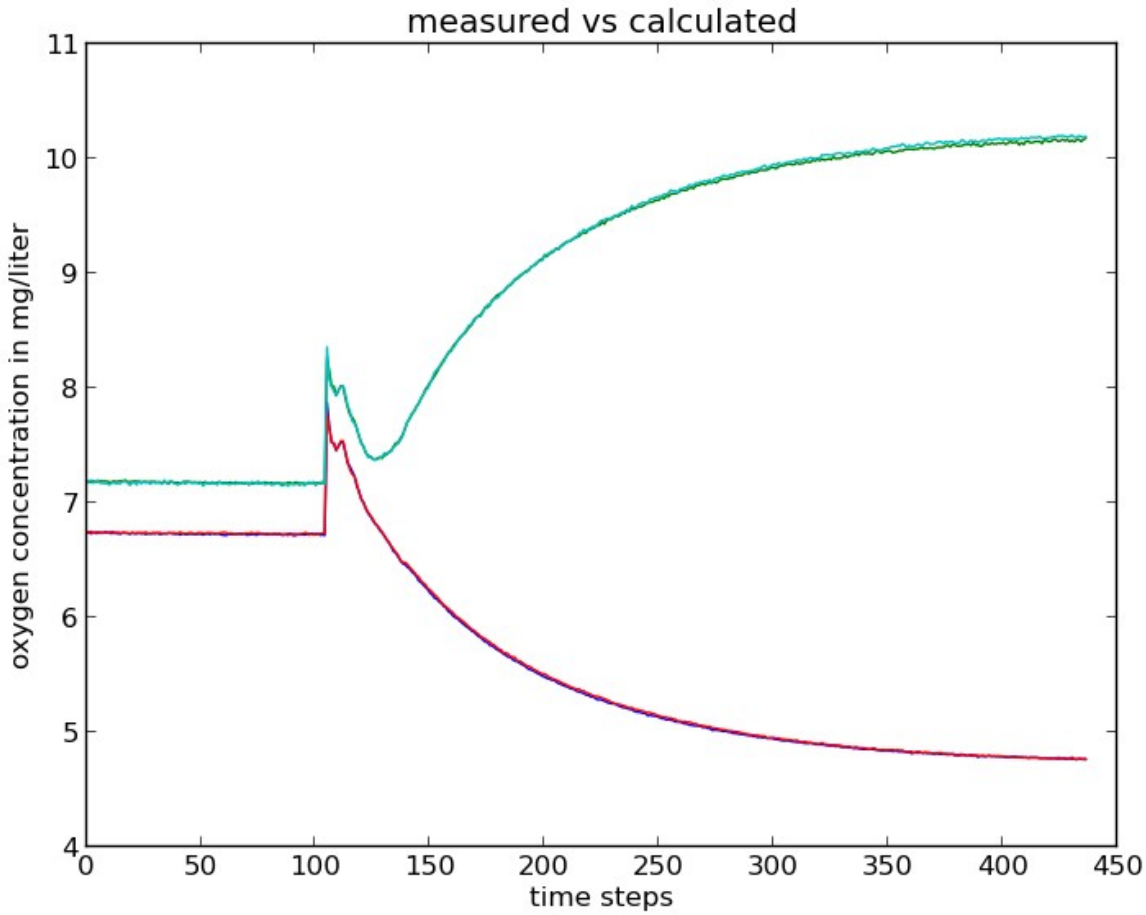

Illustration 46: Calculated and measured curves

Lower curve: 25.csv is shown in blue and the hand-calculated 25.csv in red.

Higher curve: 200.csv is shown in blue and the hand-calculated 200.csv in light-blue.

The overlap is so huge that the measured curves are almost indistinguishable from the calculated curves

hand calculated means

25.csv = initialvalue(25.csv)·oxygen-decay + endvalue(25)·oxygen-jump

200.csv = initialvalue(200.csv)·oxygen-decay + endvalue(200)·oxygen-jump

oxygen-decay and oxygen-jump from the previous illustration.

So the files 25.csv and 200.csv are a superposition of the files 100.csv and 50.csv

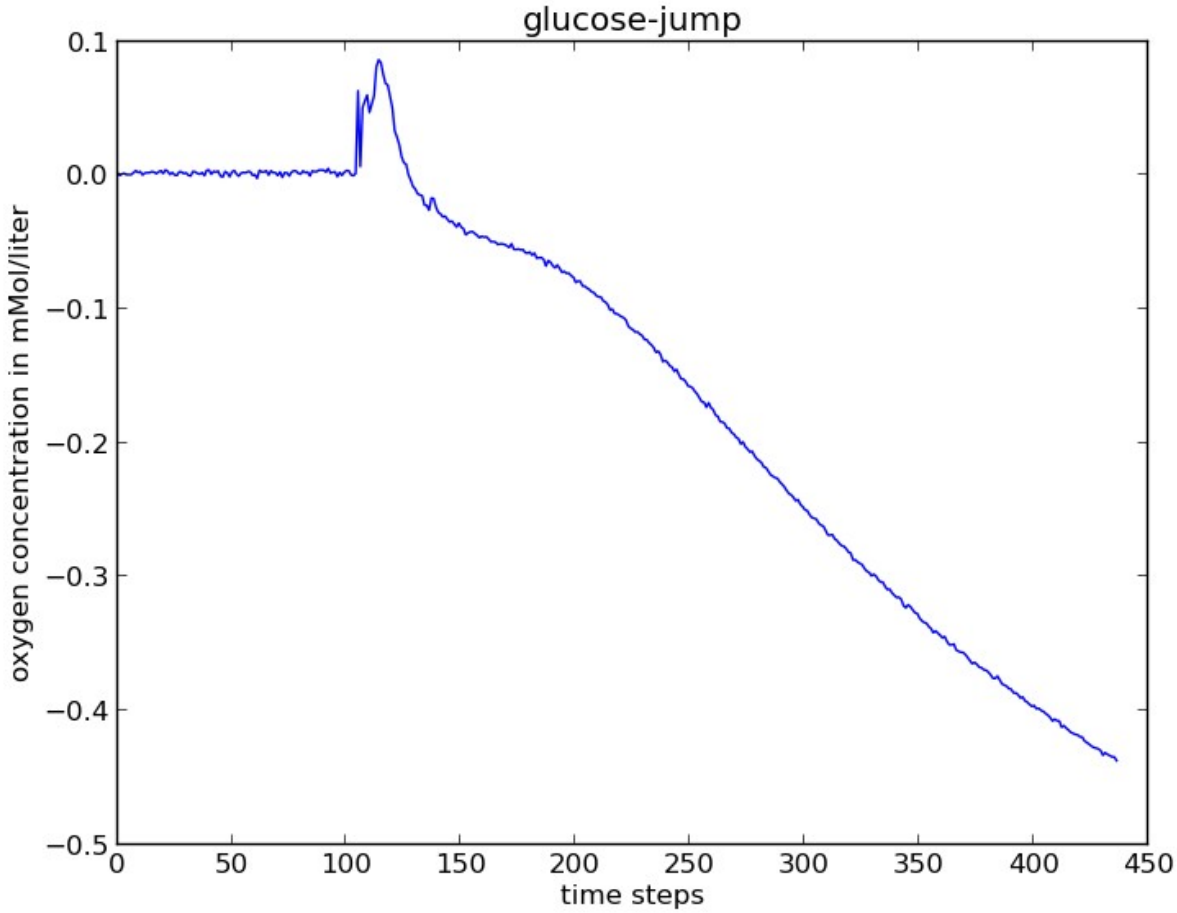

Illustration 47: Calculated response to unity glucose-jump input from (124)

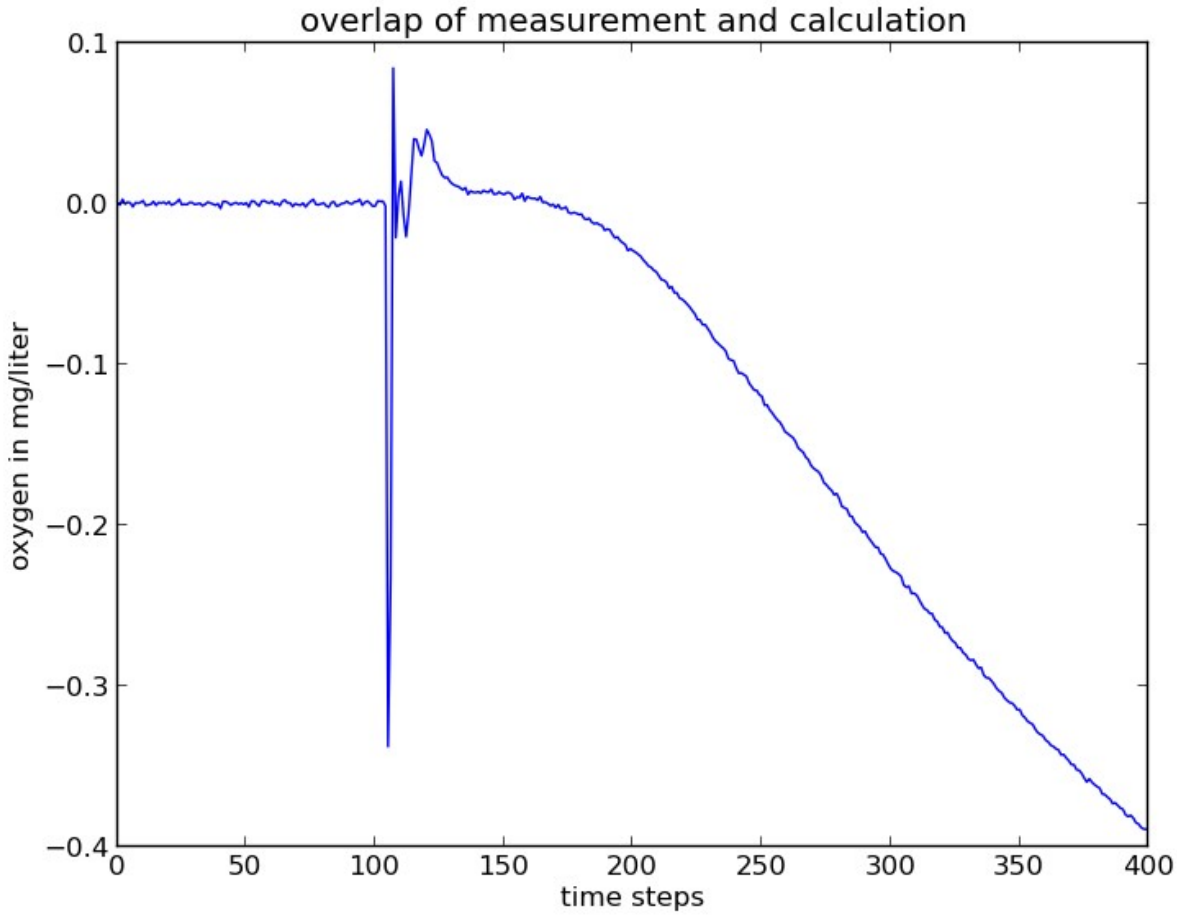

Illustration 48: Response of unity glucose jump. This time from a different csv files with a different temperature variation than the previous illustration.

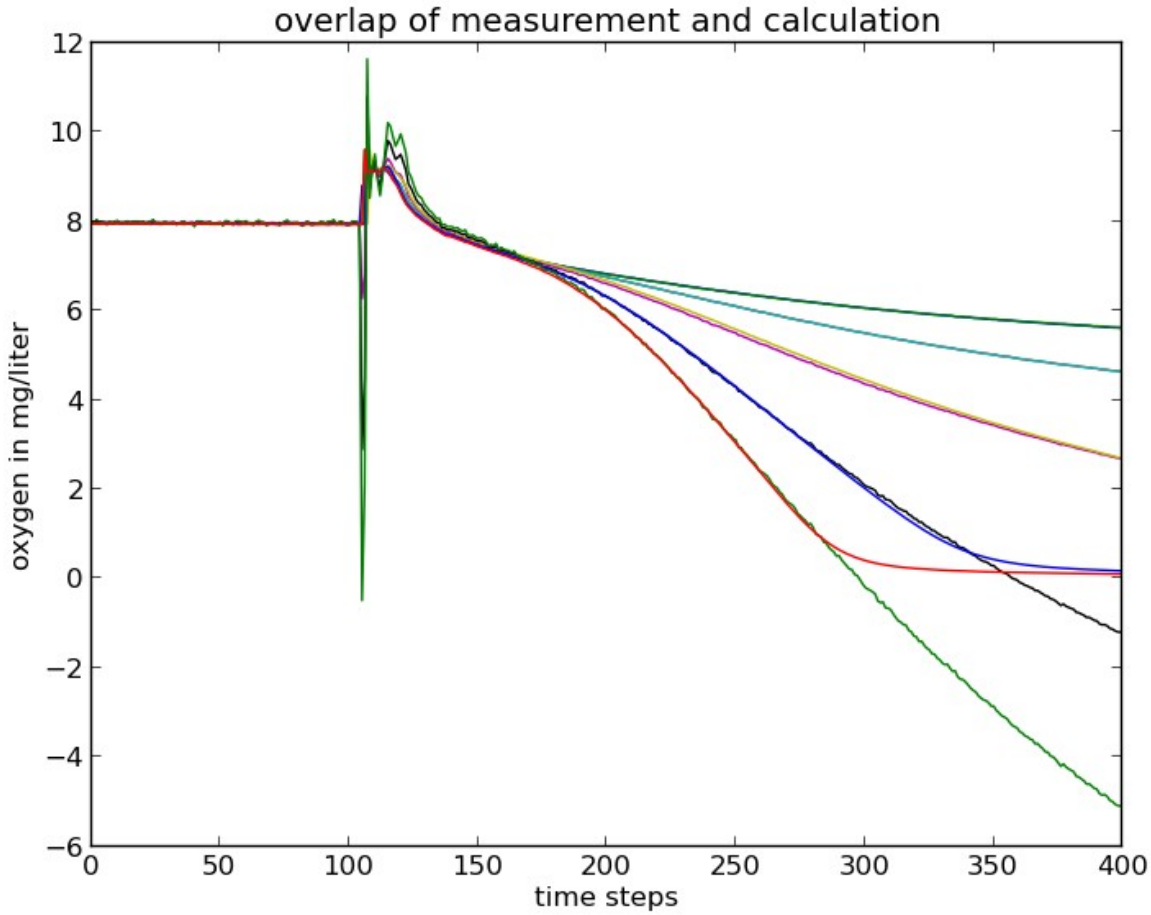

Illustration 49:Shown are the measured data from 2.5mMol/liter, 5mMol/liter, 10mMol/liter, 20mMol/liter and 30 mMol/liter glucose. The higher the glucose concentration the further the signal gets dragged down.

The measured glucose responses and the calculated glucose responses. There is a perfect match between measured and calculated data, except for negative values.

The calculation is based on (125):

$$\vec{input} = x_1 \vec{oxydecay} + x_2 \vec{oxyjump} + x_3 \vec{glucjump}$$

where

x1 is the initial value of the curve

x2 is the value from the pure oxygen sensor

x3 is the known glucose concentration of the test solution

The vector x is then used in (127)

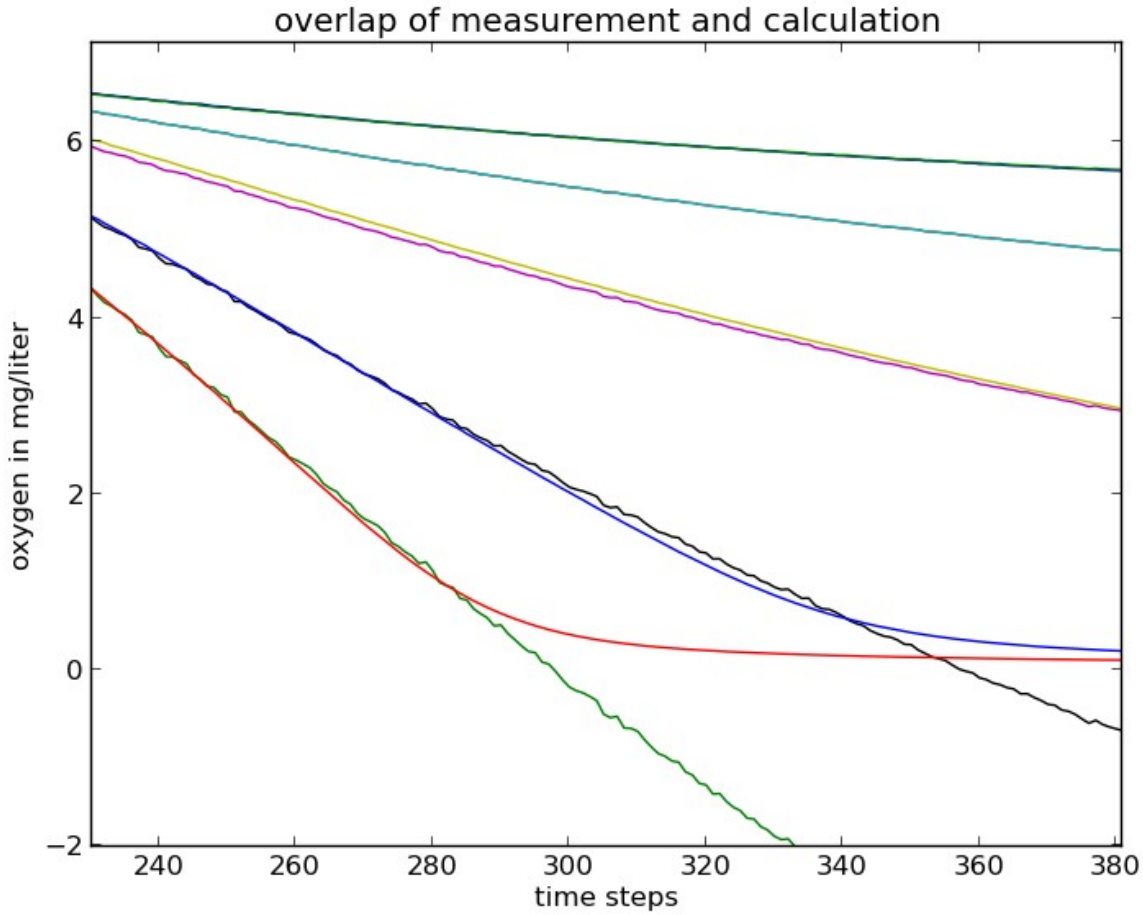

Illustration 50:The same graphic as before but this time magnified. Shown are the measured data from 2.5mMol/liter, 5mMol/liter, 10mMol/liter, 20mMol/liter and 30 mMol/liter glucose. The higher the glucose concentration the further the signal gets dragged down.

The second curve counted from above is itself by definition.

This means that the base vectors were calculated from 5mMol/liter glucose file. So that the reconstruction of the 5mMol/liter must be identical to the 5mMol/liter glucose file that it was based on.

So it is a mathematical identity that is true by definition and therefore not very surprising.

The interesting thing is that the other measured curves could also be reconstructed from the 5mMol/liter glucose measurement curve.

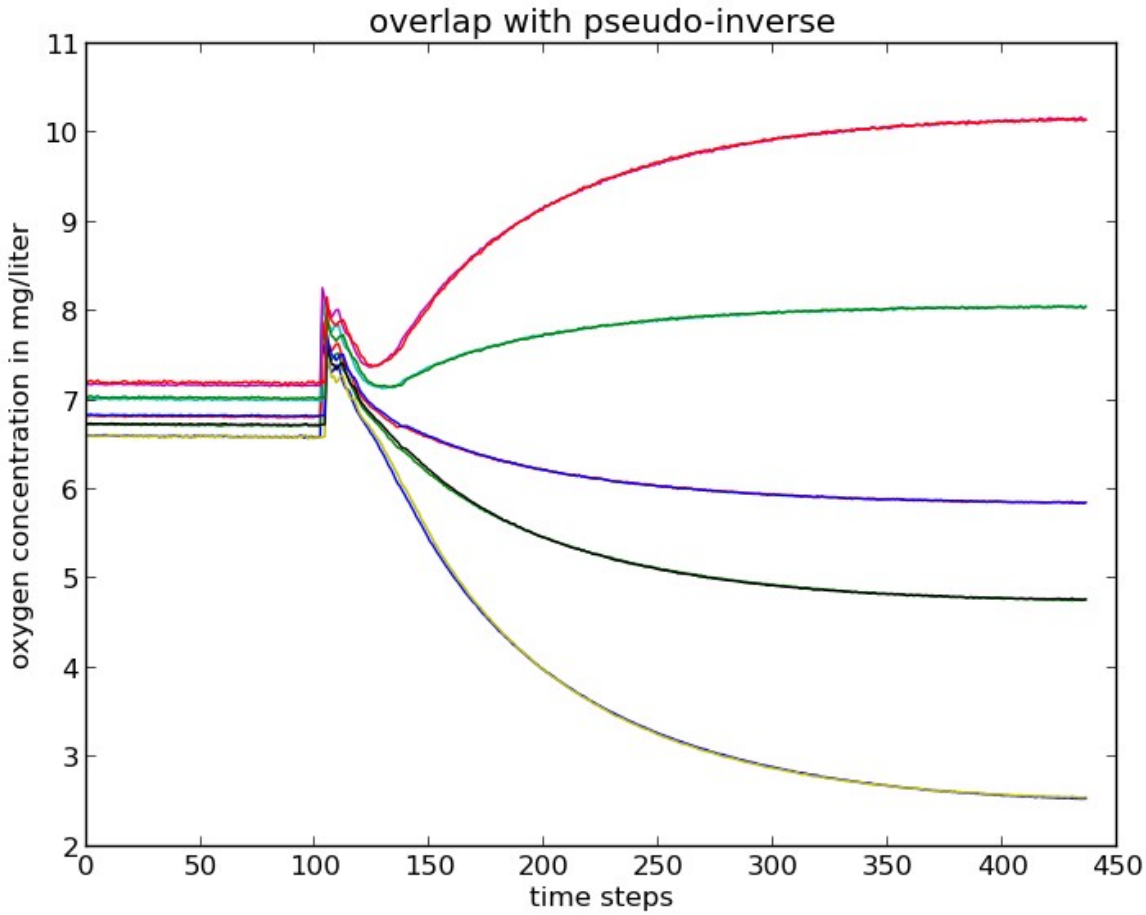

Illustration 51: This Illustration shows the overlap that was achieved by the pseudo inverse calculation

	Oxy25	Oxy75	Oxy100	Oxy150	Oxy200
Offset value	0.947	0.995	1.090	1.148	1.309
oxydecay	5.649	5.741	5.748	5.894	5.906
oxyjump	1.509	3.7033	4.739	6.934	8.905
glucjump	-0.21	-0.157	-0.047	0.106	0.194

Table 2: Estimator (131) for the measurements with only oxygen.

The offset value was added to make the estimator independent of an offset to the entire measurement. It is simply another column with all entry equal to 1 in (131).

The values from the table are only listed for completion. The value of glucjump should be zero.

There is no use, in the practical application of the sensor, for the other values of the predictor.

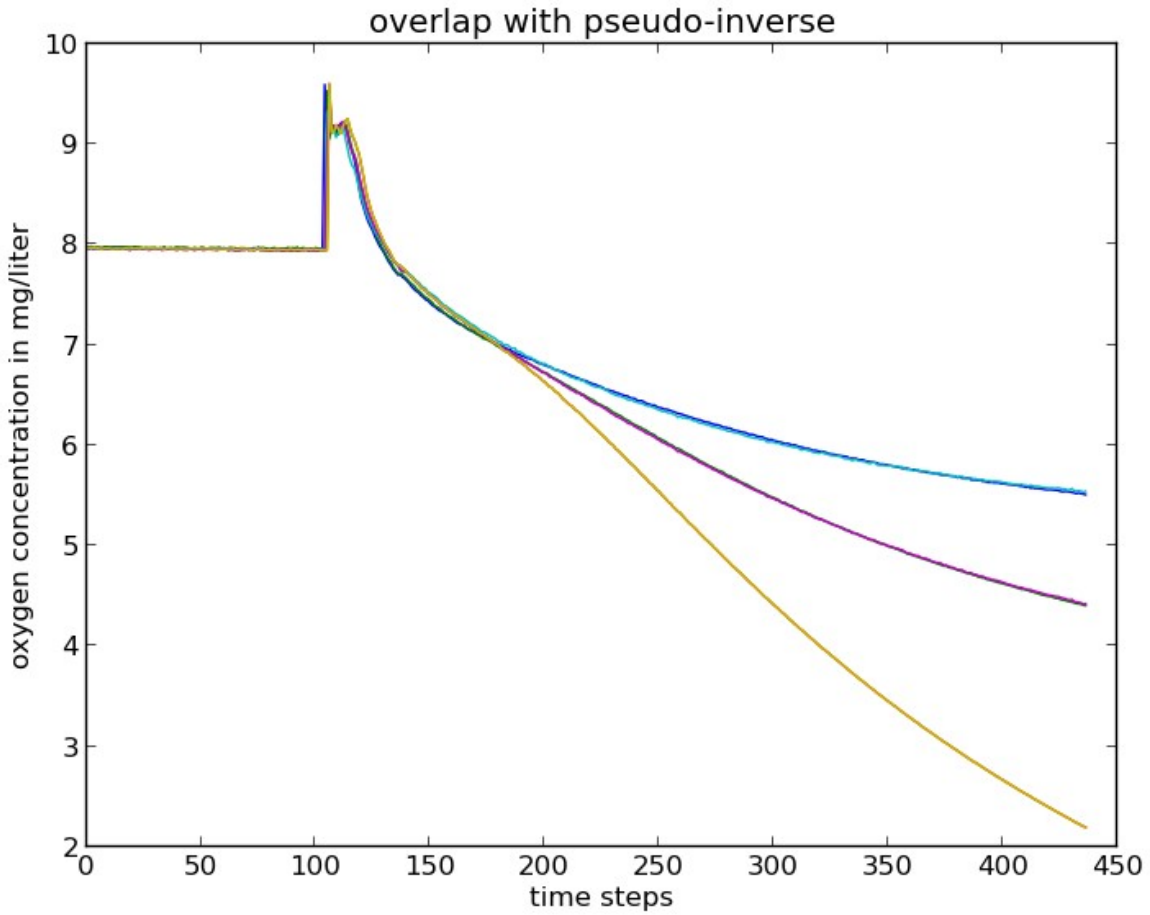

Illustration 52: This Illustration shows the match between measured data and the estimator from (131)

	Gluc2.5	Gluc5	Gluc10
Offset	-1.891	-1.787	4.839
oxydecay	9.852	9.746	7.962
oxyjump	8.092	8.0574	6.578
glucjump	**1.529**	**4.234**	**10**
truegluc	**2.5**	**5**	**10**

Table 3: Estimator applied to glucose curves

This table shows the results of the pseudo-inverse. Only the last two rows are of interest. The possible sources for the mismatch will be discussed in the next section.

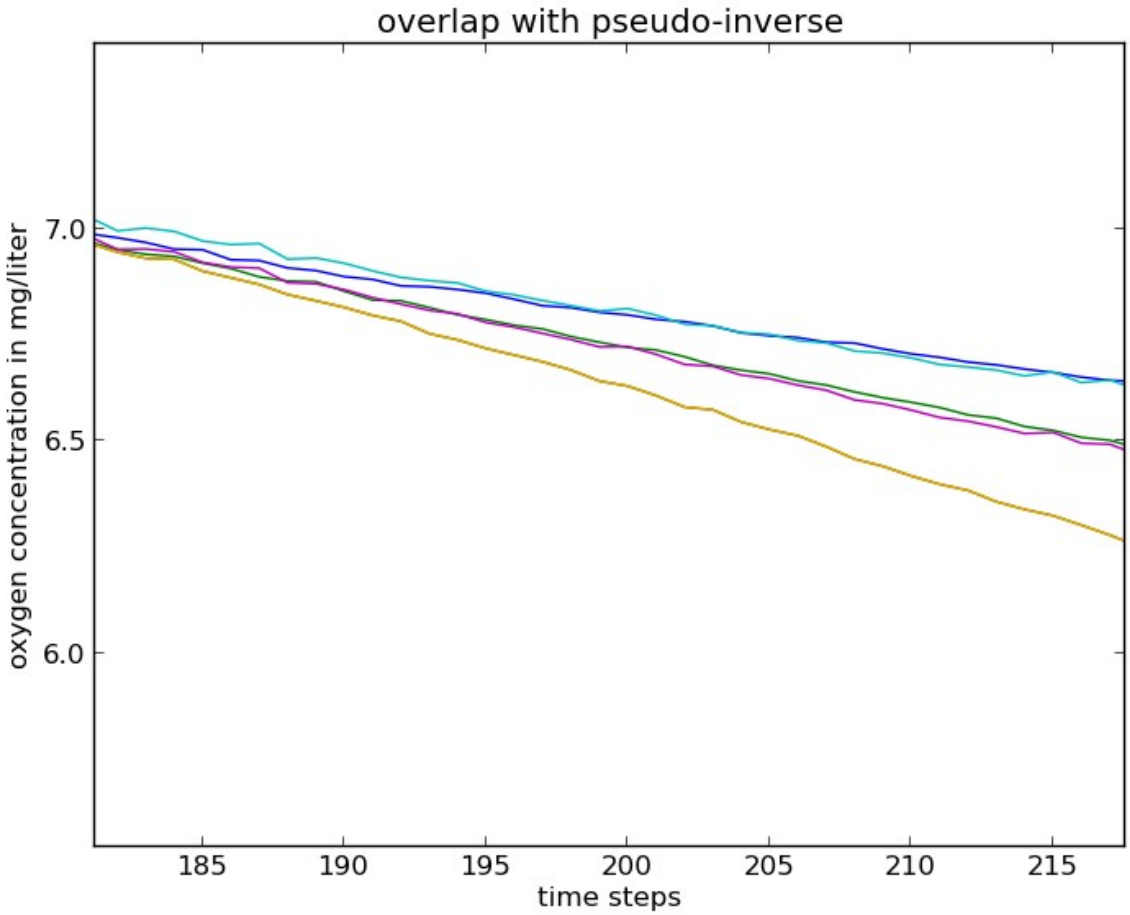

Illustration 53: Same Illustration as before but this time magnified. The lowest curve is itself by definition. This means that it is recreated from base vectors that are derived from it.

	Gluc2.5	Gluc5	Gluc10
Offset	-1.891	-1.787	4.839
oxydecay	9.852	9.746	7.962
oxyjump	8.092	8.0574	6.578
glucjump	**1.529**	**4.234**	**10**
truegluc	**2.5**	**5**	**10**

Table 4: Estimator applied to glucose curves. Same table as on the last page.

This table shows the results of the pseudo-inverse. Only the last two rows are of interest. The possible sources for the mismatch will be discussed in the next section.

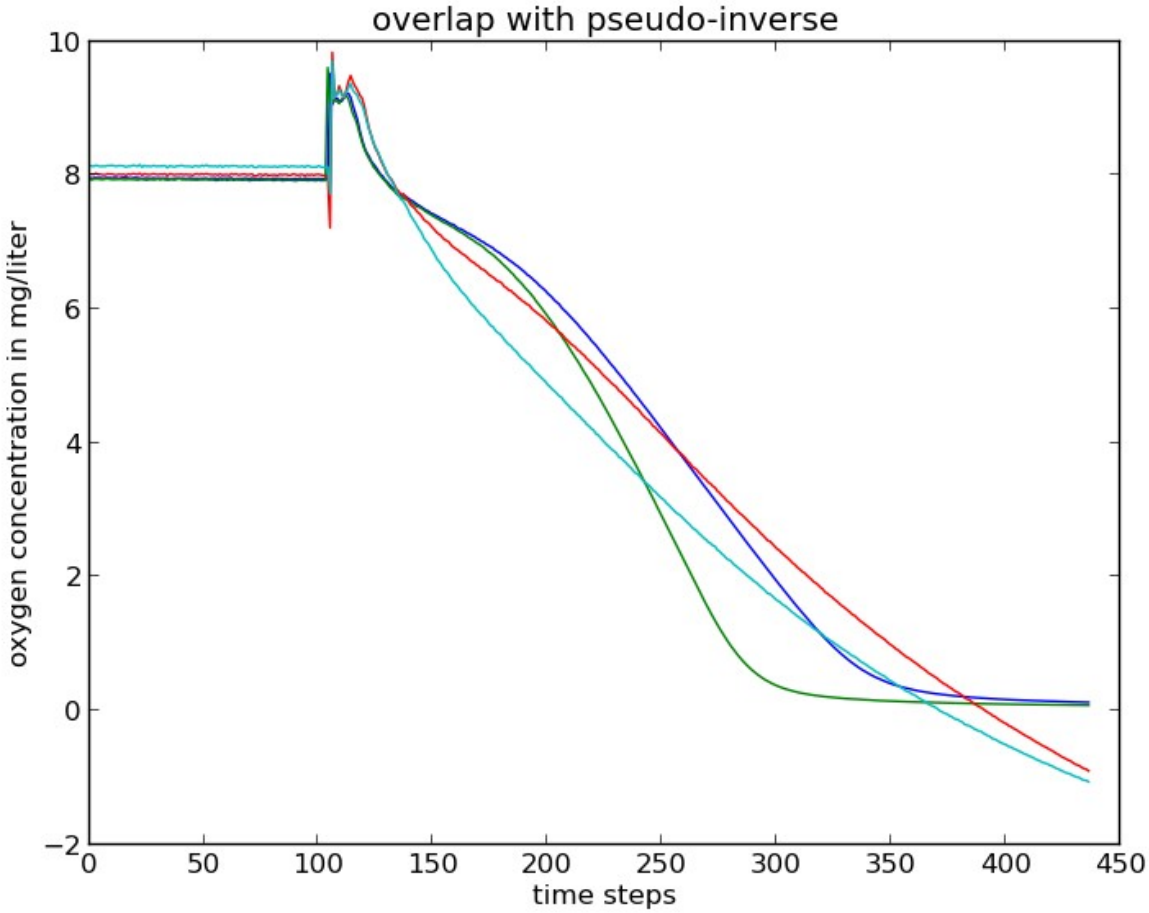

Illustration 54: Failure of the pseudo-inverse to achieve an overlap for the 20gluc and 30gluc curves. The value of the pseudo-inverse estimator was:

14.909mMol/liter for 20gluc(20mMol/liter)

and

11.201mMol/liter for 30gluc(30mMol/liter)

A solution to this problem is suggested in the next chapter.

11.4 Discussion

Illustration 45 shows the calculated(by superposition) unit responses.
The blue curve represents the oxygen-decay. The decay starts at time step 100. The curve does, however, not start to decay immediately but rather makes a slight upward oscillation, due to the temperature variation. **Therefore, it can be concluded that the temperature variation has a direct effect on the signal, and not only on the diffusion constants!**
The green curve represents the oxygen-jump. The input oxygen-jump takes place at time step 100. The response starts after a short delay and is as expected.

Illustration 46 shows the measured oxygen-only curves together with their superposition based curves. The match is perfect!

Illustration 47 and illustration 48 show the calculated unit responses of a glucose jump. The two curves are derived from different csv files. These curves have negative values. The lowest possible value of the sensor is zero. There cannot be less oxygen inside the sensor than no oxygen. This can be a problem, because the superposition model does not consider saturation. So it should be emphasized that the calculated curves are virtual responses, which means that not all superpositions are real responses, but sometimes negative “virtual” responses.

Notice that illustration 47 and illustration 48 have a region of linear decline, while the oxygen-decay and oxygen-jump curves have no region of linear decline. The origin of this linear decline is the steady flow of glucose into the sensor. The outside glucose-concentration stays almost constant, while the inside glucose-concentration stays zero.
Illustration 49 shows the measured glucose curves together with their superposition based curves. The overlap is perfect in illustration 46, except for two curves. Illustration 50 shows a zoomed version of the curves. The superposition based curves are based on a virtual glucose-jump output that makes negative values possible, while the sensor cannot go below zero. This problem could be easily corrected by replacing all negative values of a superposition based curve by zero.

Illustration 51 shows the application of the pseudo-inverse to the different oxygen curves. The matrix A was augmented by an offset column vector(all entries equal to one). So the matrix A consisted of offset, oxy-decay, oxy-jump and glucose jump. The pseudo-inverse was calculated from A and then multiplied by the measured data. Then the calculated coefficients were used to construct the corresponding prediction curves. The match was so good that the curves are indistinguishable, as can be seen in illustration 51.

Illustration 52 shows the measured glucose curves and the superposition based curves, obtain by the pseudo-inverse. The coefficients of the estimator are shown in the table 3. The match is perfect. The estimator is in the row “glucjump.” The true glucose-concentration is in the row “truegluc”. It is possible that the test solutions don't have the claimed glucose-concentration, and that the estimator is closer to the true physical glucose-concentration. This could have been caused by a depletion of glucose in prior measurements. Illustration 53 shows a magnification.
Illustration 54 shows the failure of the pseudo-inverse for higher glucose-concentrations. The problem is that the “virtual” glucjump function provides negative values instead of saturations above zero.

12 Last adjustment

The results of the last chapter showed perfect results for all pure oxygen curves and all glucose curves upto a concentration of 10mMol per liter. The glucose model started to fail as soon as the theoretical superposition model had to assume negative values to get a better fit. Negative values can, however, due to the physical design of the sensor, never be a valid measurement.

So the problem is that linearity is not universally true, because it assumes negative values for too high glucose concentrations.

The unit glucjump output could be identified from measured data with a high glucose-concentration. This glucjump curve would then have a saturation above zero. This unit glucjump output would be capable of modeling higher glucose-concentration, but it would fail for lower glucose-concentration.

So one possibility would be to have several A matrices based on different unit glucose jump outputs. The estimator from the matrix that achieved the lowest residuals would be the best estimator for the glucose-concentration.

Another possibility with only one glucose jump curve, would be to discard all data below a threshold.

$$valid\ data = \{\ measurements \mid measurements > threshold > 0\ \} \tag{132}$$

This was, however, not implemented in this master thesis.

Safety issues:

Let's assume the sensor would give an arbitrary signal, due to the denaturalization of the enzymes for example, that is very different from all the possible signals that can be constructed with low residuals from the A matrix. The pseudo-inverse would still find the quadratically closest curve and give the 4 estimations.

In the worst case scenario these 4 estimations would be in a normal or pathological range, but still be possible values for human beings under severe pathological conditions. The measurement of the glucose concentration would be generated from the denaturalized enzyme. The state of this denaturalized enzyme has, however, no relation to the blood glucose concentration. So a measurement based on a physiologically-independent(enzyme state) event would be the base for an insulin injection!

In order to avoid such problems the following measures are proposed:

The elements of the estimator

$$estimator = [\ pinv(A)] \cdot measurment \tag{133}$$

should be in a physiological or pathological range.

The difference between the pure-oxygen sensor and the glucose-sensor should be below a safety value

$$|estimator.oxyjump - pureoxygen\ sensor| < safety\ value \tag{134}$$

$$|estimator.oxydecay - pureoxygen\ sensor| < safety\ value \tag{135}$$

The error of the estimator is

$$error = norm[measurement - modeloutput] \tag{136}$$

or

$$error = norm([E - A \cdot pinv(A)] \cdot measurment) \tag{137}$$

The safety condition should be an error below a safety value

$$error < safety\ value \tag{138}$$

True blood

Hemoglobin at the outside can be considered as a constant indepletable source of oxygen. This is important, because the unit oxygen-jump output was derived from water solutions, and can therefore not be used in this case.

The procedure for obtaining unit base vectors for blood should be:

The first equilibrium solution reaches a steady-state that will be called oxygen-decay-water1. Then the first blood sample comes in with a known glucose concentration that will be called glucose-jump-blood1, and an unknown oxygen concentration that will be called oxygen-jump-blood1.

Then the same procedure is repeated, but this time with another equilibrium solution, and another sample of blood that should have a different but known glucose concentration.

The second equilibrium solution reaches a steady-state that will be called oxygen-decay-water2. Then the second blood sample comes in with a known glucose concentration that will be called glucose-jump-blood2, and an unknown oxygen concentration that will be called oxygen-jump-blood2.

Then the same procedure is repeated, with another equilibrium solution, and another sample of blood that should have a different but known glucose concentration.

The equilibrium solution reaches a steady-state that will be called oxygen-decay-water3. Then the third blood sample comes in with a known glucose concentration that will be called glucose-jump-blood3 and unknown oxygen concentration that will be called oxygen-jump-blood3.

And so on ...

Until the following condition is met.

$$0 \approx linear\ combination\ of\ measured\ output \tag{139}$$

or

$$norm\left(linear\ combination\ of\ measured\ output\right) < threshold \tag{140}$$

with a nontrivial(not all coefficients equal to zero) solution

The measured output vectors are arranged as columns of the matrix G.

The column vector OSE that has the elements:

glucose-jump1
glucose-jump2
…
glucose-jump n-1
glucose-jump n

is build

The pseudo-inverse is built:

$$pinvG = pinv\left(G\right) \tag{141}$$

an arbitrary measurement is multiplied by the pseudo-inverse

$$GLUC = pinvG \cdot measurement \tag{142}$$

The glucose concentration of this arbitrary measurement is:

$$glucoseconcentration = GLUC \cdot OSE \tag{143}$$

This method has no base vector, but gives nonetheless a glucose estimator. The derivation is based on the superposition model with some additional considerations.

This method could not be tested in the master thesis.

The best option would, however, be to build a difference sensor(as mentioned in discussion 10.4):

$$glucose\ signal = glucsensor - oxysensor \tag{144}$$

The glucose-concentration could then simply be obtained from the slope of the glucose signal.

13 Appendix

13.1 Installation of the the FEM model

In order to make the results of the master thesis reproducible, a short manual on the installation of the FEM simulator is provided:

the input will be in green
and
the relevant output will be in red

First one has to go to the installation folder:

```
cd glucO2/

pwd

/home/armin/GP-temp/glucO2

ls -a

.  ..  FEMLIB  GlucO2  libredblack-1.3  make.def  Makefile
my_switches.def  NumComp  PrM  .svn  switches.def
```

do an svn update to get the latest version

```
svn update .
```

Compile a new version:

```
cd FEMLIB/

make clean

make

make glucO2
```

Now that everything is compiled, the execution of the following commands should run without errors.

```
./equiTG.sh

./measTG.sh
```

13.1.1 Performance of a simple simulation

After the FEM has been successfully installed, an introduction into the simulation of one measurement is given.

The simulation consists of two steps:

1. Go from a zero oxygen concentration at the outside of the sensor to a non zero oxygen concentration, in order to reach the equilibrium oxygen concentration inside the sensor.

2. Do the relevant simulation with the glucose and oxygen-concentration of the test solution

Before the first step, all finite elements in the sensor have an oxygen and glucose concentration of zero. In order to get from a homogeneously distributed zero oxygen concentration to a homogeneous non-zero oxygen distribution, ./equiTG.sh is called.

This command applies an oxygen jump at the outside of the sensor that diffuses into the sensor for a sufficiently long time to reach equilibrium with the applied jump. Although this equilibrium is only reached at infinity, a simulation time of 30 seconds showed to be very close.

The file equiTG.sh is not a binary executable but rather a shell script to control the FEM simulator. Therefore, the script is edited and then called from the prompt.

The content of equiTG.sh is a follows:

```
#!/bin/bash

sz="small"
#sz="full"
simID="equi"

# equilibration phase
mpiexec -n 1 ./glucO2 +F TG.par -G_dbc[0].strength    0. \
                               -O2_dbc[0].strength 237. \
                               -num_tsav    1  \
                               -tsav[0]   30000. \
                               -tend      30001. \
                               -timedt    100. \
                               -spacedt   100. \
                               -dt         10. \
                               -D_O2     1980. \
                               -D_G       560. \
                               -moxOn       0  \
```

```
-parab_solve 1  \
-simID    ${simID}_${sz} \
-gridout     2 \
-meshname GlucoSpot1_${sz}
```

The only parameter of interest is the -O2_dbc[0].strength parameter. This is the height of the oxygen jump in µMol per liter.

After the simulation is done by the command:

```
./equiTG.sh
```

The file
`/home/armin/Templates/glucO2/FEMLIB/equi_small/sensorO2.dat`

should look the following way:

```
0.0 0.00
100.0 4.96
200.0 40.61
300.0 87.67

...
...
...

29800.0 399.94
29900.0 399.94
30000.0 399.94
```

The last lines of this file should have converged to a constant number. Unlike expected, the last line in the file has not converged to 237, but to 399.94. The reason is that the units are not in µMol/l but rather:

$$value = k \int [oxygen]\, dV \tag{145}$$

This problem can be solved, by the assumption that the concentration is almost constant as soon as the output values begin to converge. A simple multiplication, by a self explanatory constant:

$$[oxygen] = \frac{237}{399.94} k \int concentration\, dV \tag{146}$$

Leads to units of µMol per liter!

The sensor is now filled with oxygen and the simulation of interest can begin, by calling measTG.sh. Before doing so, its content is discussed.

The file measTG.sh is not a binary executable but rather a shell script to control the FEM simulator. The desired oxygen-concentration and glucose-concentration is simply filled in as an option.

The content of the file looks as follows:

```
#!/bin/bash

simID="TG"
sz="small"
#sz="full"

mpiexec -n 1 ./glucO2 +F TG.par -G_dbc[0].strength 0. \
                                -O2_dbc[0].strength 192. \
                                -tend 10000 \
                                -num_tsav 1 \
                                -tsav[0] 9999. \
                                -stateInit.G_state  ./equi_${sz}/state_G.30000.0.dat \
                                -stateInit.O2_state ./equi_${sz}/state_O2.30000.0.dat \
                                -moxOn      1 \
                                -kRate       .00001 \
                                -timedt   200. \
                                -spacedt  100. \
                                -D_G      560. \
                                -D_O2    2500. \
                                -dt       500. \
                                -parab_solve 1 \
                                -simID    ${simID}_${sz} \
                                -gridout 2 \
                                -meshname GlucoSpot1_${sz}
```

There are 4 parameters of interest:

```
-G_dbc[0].strength 0.
-O2_dbc[0].strength 192
-D_G      560.
-D_O2    2500.
```

These parameters do the following. The glucose-concentration at the outside equals 0µMol per liter. The oxygen concentration equals 192µMol per liter. The glucose diffusion constant equals 560µm² /s and the oxygen diffusion constant equals 2500µm² /s.

The file
`/home/armin/Templates/glucO2/FEMLIB/TG_small/sensorO2.dat`
should look the following way after the the execution of measTG.sh is done:

```
0.0 399.94
200.0 387.50
400.0 367.07
600.0 352.87
...
...
...
9600.0 324.00
9800.0 324.00
10000.0 324.00
```

The following three observations are of importance:

1. The simulation starts where equiTG.sh has finished.

2. The FEM simulator converges to the new steady-state in the same manner as equiTG.sh.

3. By doing the chain-multiplication from (4) or (146)

237µMol/liter produced 399.4 outputunits

$$(324\,outputunits)\cdot\frac{(237\ \mu Mol/liter)}{(399.4\ outputunits)}=192\ \mu Mol/liter \tag{147}$$

Which was the input

```
-O2_dbc[0].strength 192
```

from the last page that corresponds to

192µMol per liter

13.2 PET

While the parameter fitting was done in Python, the simulation of the sensor was done in a FEM-model, which was programmed in C, and used the PET library.

http://www.mcs.anl.gov/petsc/index.html

Because the only requirement in the master thesis was to properly install PET, only a short documentation of the installation is given.

The installation was done under Lubuntu 12.10. The installation required some effort, although the homepage talked about a simple one line installation:

```
./configure --with-cc=gcc --with-fc=gfortran --download-f-blas-lapack --download-mpich
make all test
```

Other programs like:

cmake and co.

had to be installed

All simulation were done on the following hardware:

TravelMate P253-E Notebook
Intel 2.2GHz, 2MB L3 cache
Intel HD Graphics

Although, the hardware was pretty weak, the simulations ran much faster on this notebook than on the stronger computers at B Braun. The cause for this high performance on low-tech was probably that the PET installation was done with all optimization options for the FEM simulator. Apparently a slight misconfiguration at B. Braun caused their simulation to slow down significantly.

One simulation with equiTG.sh and measTG.sh took about 3 minutes.

14 References

[1]
B. Bode, K. Gross, N. Rikalo, S. Schwarz, T. Wahl, C. Page, T. Gross, J. Mastrototaro, Alarms based on real-time sensor glucose values alert patients to hypo- and hyperglycemia: the guardian continuous glucose monitoring system, Diabetes Technol. Ther. 6 (2004) 105-13

[2]
A. Poscia, M. Mascini, D. Moscone, M. Luzzana, G. Caramenti, P. Cremonesi, F. Valgimigli, C. Bongiovanni, M. Varalli, A microdialysis technique for continuous subcutaneous glucose monitoring in diabetic patients (part 1), Biosens. Bioelec. 18 (2003) 891-898

[3]
Bozidar Todic, Walter Gschohsmann, Hans Koehler,
Kalibration und Endprüfung eines neuartigen Sensors für Blutglukose im Bereich der Intensivmedizin

[4]
Gernot Plank
Manual