Abstract

This research investigates three methods for estimating the parameters in differential equation models. These three methods are applied to a model of metabolic dynamics in order to estimate the insulin sensitivity from data obtained from an intravenous glucose tolerance test. The Levenberg-Marquardt algorithm is shown to find reasonable estimates of the parameters of the model. The parameters obtained by this method are in agreement with published parameters. The estimates produced by the other two methods were not physiologically reasonable.

Contents

1 Diabetes

Diabetes is a great threat to the well being of individuals throughout the world. In recent years the number of people diagnosed with diabetes is increasing and is expected to double between the years 2000 and 2030 [1]. Although the fundamental cause of the disease is not fully known, factors suggested to cause an increased susceptibility to the disease include, an increase in the prevalence of obesity, poor diet and lifestyle. The increase in diabetes can be a contributing factor in early death and other complications, which will subsequently lead to excessive loading on health-care systems.

Being able to quantify an individuals susceptibility to diabetes and early diagnosis of the onset of the disease would allow lifestyle changes to be made that will significantly reduce the risk of developing diabetes, or allow early treatment offsetting the need for a patient to become fully insulin dependent.

There are two main forms of diabetes, these are:

- **Type I Diabetes**

  Type I diabetes is characterised by a deficiency of insulin as a result of the destruction of insulin producing $\beta-$cells in the pancreas. This is caused by an auto-immune disorder of the body, which has a high tendency to be hereditary. The onset of type I diabetes is usually rapid, instigating the need for the patient to become insulin dependant within only a few weeks of the early signs of the disease. Type I diabetics make up approximately $5-10\%$ of the diabetic population in North America [2].

- **Type II Diabetes**

  Type II diabetes is characterised by a reduction in insulin sensitivity and a reduction in insulin secretion. Unlike type I diabetes the insulin resistance is caused not by a loss of insulin producing ability but a reduction in insulin mediated clearance of glucose in the cells of the body known as insulin resistance (IR). The risk of developing type II diabetes can be caused by a genetic disposition to the disease, however a more likely cause is increased weight, obesity and lack of exercise.

1.1 Parameter identification of diagnostic tests

The diagnosis of diabetes is achieved by quantifying insulin sensitivity (IR^{-1}). If this sensitivity is low this suggests a decrease in the ability of insulin mediated clearance of glucose in the body.

Three of the main diagnosis tools available to clinicians are:

- **Oral glucose tolerance test.**

  In this test a 75g glucose drink is given to the patient, with both plasma insulin and glucose concentrations measured four times over a two hour period.

- **Euglycaemic insulin clamp.**

  In this test a constant infusion of insulin is given intravenously, with a second infusion of glucose given and adjusted to keep the patient at a stable target blood glucose level. After the one hour of stability, the insulin and glucose concentrations are measured over the second hour of the test.

- **Intravenous glucose tolerance test.**

  In this test an intravenous dose of glucose is given, followed 20 minutes later by a dose of insulin. Both glucose and insulin concentrations are measured at $2-5$ minute intervals over a period of two hours.

These diagnostic tools allow clinicians to get a quantitative measure of the insulin sensitivity of a patient. Once the clinician has this measure, he or she can then advise the patient of the treatments available to them.

For these tests to be effective at determining insulin sensitivity of a patient, robust parameter identification techniques are needed. They must be able to efficiently and repeatably determine the parameters of the model in the face of uncertainty of measurement and modeling error.

The diagnosis of type I diabetes can generally be achieved through the oral glucose tolerance test, as type I diabetic patients exhibit much higher glucose concentrations than type II and non-diabetic patients. However for the case of type II diabetics, normal glucose concentrations are usual and it is necessary to determine the dynamics of the insulin glucose interactions to be able to get a measure of the patients insulin sensitivity.

2 Model of the insulin-glucose system

2.1 Bergman's minimal model

There are a number of models that capture the dynamics of the insulin-glucose system, the first of which was developed by R.N. Bergman in [3]. This was and still is considered the simplest valid model describing the insulin-glucose system. It is a two compartment model incorporating the plasma glucose concentration and a remote insulin compartment.

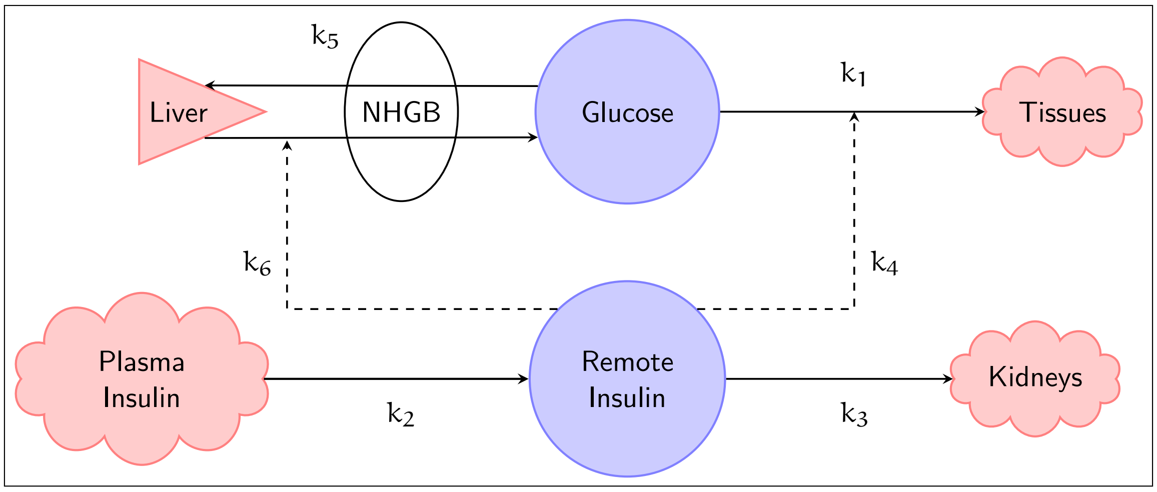


Figure 1: Schematic of Bergman's minimal model

NHGB is the net hepatic glucose balance. This describes the balance of clearance and production of glucose in the liver in response to a variation of glucose concentration from basal levels [4]. Hepatic uptake of glucose is the storage of glucose in the liver as glucagon, which is then available for release when plasma glucose concentration is low. The dynamics of the liver are suggested to be solely driven by plasma glucose concentration.

The remote insulin compartment I' represents the ability of insulin to increase the mobility of glucose across the cell membrane.

The replacement of parameters $p_1 = k_1 + k_5$, $p_2 = k_3$, $p_3 = k_2(k_4 + k_6)$ and defining a new insulin action compartment $X = (k_4 + k_6)I'$, we obtain the following formulation of the insulin glucose system, as first proposed in [3],

$$\frac{dG}{dt} = -\left(p_1 + X\right)G + p_1 G_b, \quad G(0) = G_0 \tag{1}$$
$$\frac{dX}{dt} = -p_2 X + p_3\left(I(t) - I_b\right), \quad X(0) = 0 \tag{2}$$

$G(t)$ and $I(t)$ are the plasma glucose and plasma insulin concentration respectively. $X(t)$ is the action of remote insulin on glucose disappearance. The parameter p_1 represents the effective glucose disappearance at basal insulin levels and the ability to inhibit endogenous glucose production. The parameters p_2 and p_3 combined in the parameter insulin sensitivity $S_I = \frac{p_3}{p_2}$ represent the ability of insulin to enhance glucose disappearance and inhibit endogenous glucose production [5].

From this we can see that a low p_3 value or a high p_2 value resulting in a low S_I value could be seen as a large clearance of insulin through the kidneys or liver or a resistance of insulin to bind to cell receptors. This low insulin sensitivity (or high insulin resistance) can be seen as a marker of diabetes.

The diagnostic tools for quantifying insulin sensitivity all measure only the glucose $G(t)$ and insulin $I(t)$ concentrations. Thus any parameter estimation technique must deal with these compartments, which is the justification for reformulating Bergman's model so that only glucose and insulin terms are present.

We first replace $\frac{dG}{dt}$ by some function $F = F(t)$ in (1) then solve for $X(t)$

$$X(t) = \frac{p_1 G_b - F}{G(t)} - p_1 \tag{3}$$

substituting this into (2)

$$-\frac{1}{G}\frac{dF}{dt} - \frac{F}{G^2}\left(F + p_1 Gb\right) = p_2\left(\frac{p_1 G_b - F}{G(t)} - p_1\right) + p_3\left(I(t) - Ib\right) \tag{4}$$

and solving for $\frac{dF}{dt}$ we obtain the system of coupled differential equations,

$$\frac{dG}{dt} = F, \quad G(0) = G_0 \tag{5}$$
$$\frac{dF}{dt} = \frac{F}{G}\left(F - p_1 G_b\right) - p_1 p_2\left(G - G_b\right) - p_3 G\left(I(t) - I_b\right), \quad F(0) = p_1\left(G_b - G_0\right) \tag{6}$$

2.2 Sensitivity equations

To gain an understanding of how the parameters G_0, p_1, p_2 and p_3 effect the solution to (5) we form the so called sensitivity equations of the model. These are differential equations describing how the solution changes with respect to the parameters. These sensitivity equations are formed by taking the partial derivatives of (5) and (6);

$$\frac{\partial}{\partial p_i}\left(\frac{dG}{dt}\right) = \frac{\partial F}{\partial p_i}$$
$$\frac{\partial}{\partial p_i}\left(\frac{dF}{dt}\right) = \frac{\partial}{\partial p_i}M(G_0, p_1, p_2, p_3)$$

with

$$M = \frac{F}{G}\left(F - p_1 G_b\right) - p_1 p_2\left(G - G_b\right) - p_3 G\left(I(t) - I_b\right).$$

Under the assumption that the parameters G_0, p_1, p_2 and p_3 do not vary with time, we are able to swap the order of differentiation and using the chain rule to differentiate $M(G_0, p_1, p_2, p_3)$. Due to the dependence of $G(t)$ and $F(t)$ on the parameters G_0, p_1, p_2 and p_3, we obtain the system of differential equations

$$\frac{d}{dt}\left(\frac{\partial G}{\partial G_0}\right) = \frac{\partial F}{\partial G_0}, \quad \left.\frac{\partial G}{\partial G_0}\right|_{t=0} = 1 \tag{7}$$

$$\frac{d}{dt}\left(\frac{\partial G}{\partial p_1}\right) = \frac{\partial F}{\partial p_1}, \quad \left.\frac{\partial G}{\partial p_1}\right|_{t=0} = 0 \tag{8}$$

$$\frac{d}{dt}\left(\frac{\partial G}{\partial p_2}\right) = \frac{\partial F}{\partial p_2}, \quad \left.\frac{\partial G}{\partial p_2}\right|_{t=0} = 0 \tag{9}$$

$$\frac{d}{dt}\left(\frac{\partial G}{\partial p_3}\right) = \frac{\partial F}{\partial p_3}, \quad \left.\frac{\partial G}{\partial p_3}\right|_{t=0} = 0 \tag{10}$$

$$\frac{d}{dt}\left(\frac{\partial F}{\partial G_0}\right) = \frac{\partial F}{\partial G}\frac{\partial G}{\partial G_0} + \frac{\partial F}{\partial F}\frac{\partial F}{\partial G_0}, \quad \left.\frac{\partial F}{\partial G_0}\right|_{t=0} = -p_1 \tag{11}$$

$$\frac{d}{dt}\left(\frac{\partial F}{\partial p_1}\right) = \frac{\partial M}{\partial G}\frac{\partial G}{\partial p_1} + \frac{\partial M}{\partial F}\frac{\partial F}{\partial p_1} - p_2\left(G - G_b\right) - G_b\frac{F}{G}, \quad \left.\frac{\partial F}{\partial p_1}\right|_{t=0} = G_b - G_0 \tag{12}$$

$$\frac{d}{dt}\left(\frac{\partial F}{\partial p_2}\right) = \frac{\partial M}{\partial G}\frac{\partial G}{\partial p_2} + \frac{\partial M}{\partial F}\frac{\partial F}{\partial p_2} - p_1\left(G - G_b\right) - F, \quad \left.\frac{\partial F}{\partial p_2}\right|_{t=0} = 0 \tag{13}$$

$$\frac{d}{dt}\left(\frac{\partial F}{\partial p_3}\right) = \frac{\partial M}{\partial G}\frac{\partial G}{\partial p_3} + \frac{\partial M}{\partial F}\frac{\partial F}{\partial p_3} - G\left(I(t) - I_b\right), \quad \left.\frac{\partial F}{\partial p_3}\right|_{t=0} = 0 \tag{14}$$

with

$$\frac{\partial M}{\partial G} = \left(\frac{-F\left(F - p_1 G_b\right)}{G^2} - p_1 p_2 - p_3\left(I(t) - I_b\right)\right) \tag{15}$$

$$\frac{\partial M}{\partial F} = \left(\frac{2F - p_1 G_b}{G} - p_2\right) \tag{16}$$

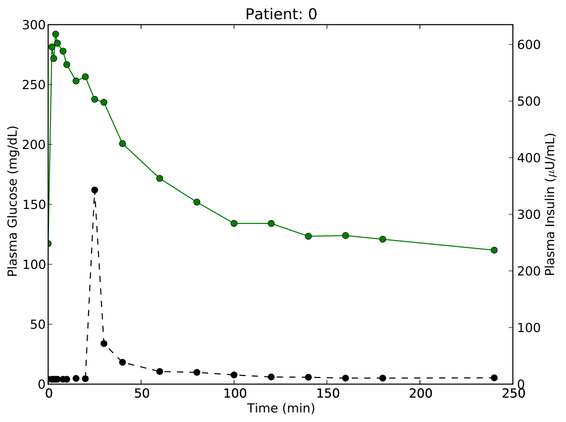
2.3 IVGTT patient data

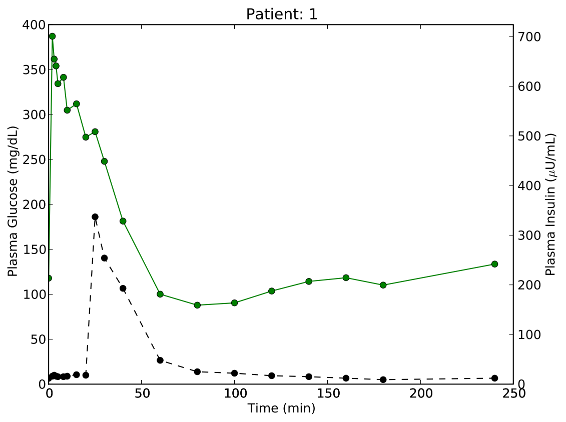
Of the diagnosis tools available to clinicians, the cost of using the Euglycaemic clamp test is the highest and is not widely used as a pure diagnosis test. However if the results of the oral glucose tolerance test are inconclusive, the intravenous glucose tolerance test is used as a more accurate diagnostic tool.

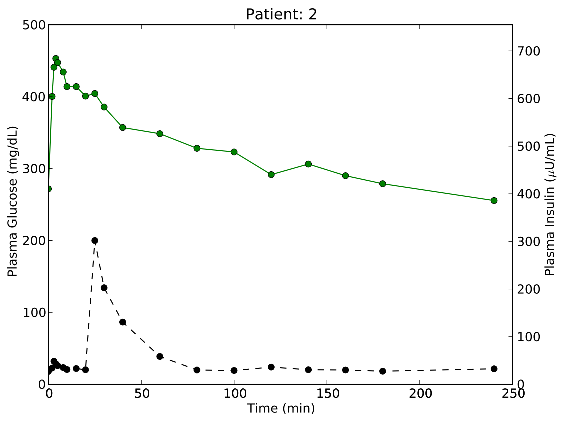
IVGTT data was taken from [6] from performing an IVGTT on 10 type II diabetic subjects, the data was extracted using Datatheif [7]. Both the insulin and glucose concentrations from these tests are shown in Figure 2.

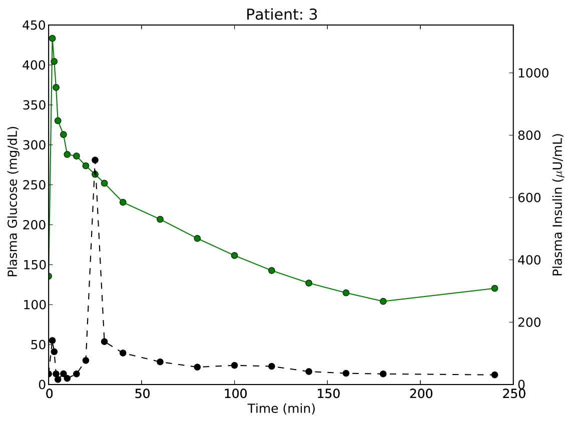
As the IVGTT is carried out after at least a 8 hour fast, what can then immediately be seen from these plots is that all patients show elevated fasting glucose concentrations. The normoglycaemic range is approximately $80 - 120$mg/dL and all patients show fasting levels above 110mg/dL with some well above this (313mg/dL). This observation may be enough to warrant further investigation. However by itself would not be enough to diagnose diabetes.

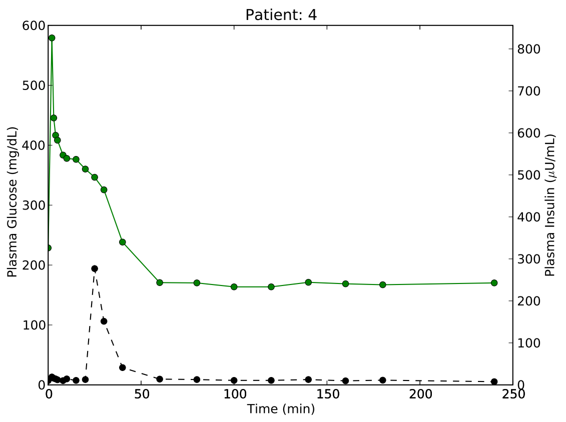
All of the patients in this study should exhibit reduced insulin sensitivity, since they have been diagnosed as type II diabetics. A further observation can therefore be made about the insulin response of these patients. All but one (patient 8) show virtually no pancreatic first-phase insulin response to the glucose stimulus at time $t = 0$. This is an increase in pancreatic insulin production in response to elevated glucose levels. Which is trait of both type I and II diabetes. In a non-diabetic subject, a response similar to patient 8 is generally observed.

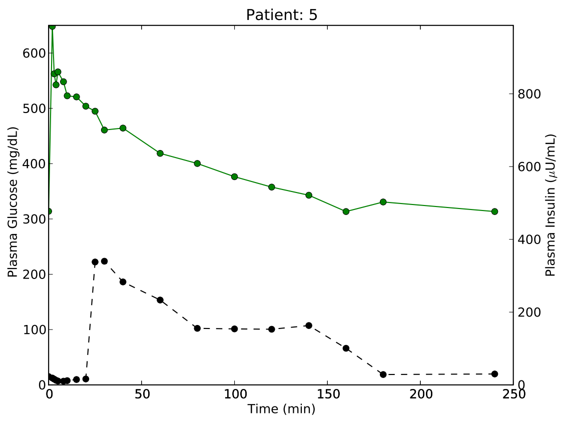

Patient: 0
Plasma Glucose (mg/dL)
Plasma Insulin (μU/mL)
Time (min)

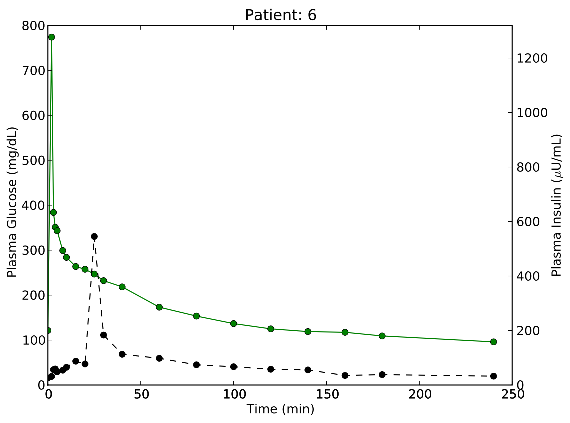

Patient: 1
Plasma Glucose (mg/dL)
Plasma Insulin (μU/mL)
Time (min)

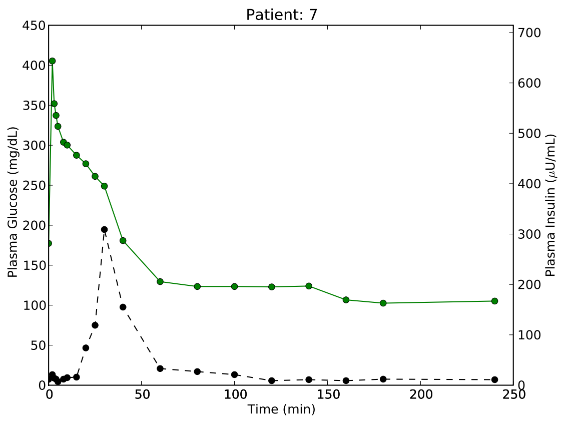

Patient: 2
Plasma Glucose (mg/dL)
Plasma Insulin (μU/mL)
Time (min)


Patient: 3
Plasma Glucose (mg/dL)
Plasma Insulin (μU/mL)
Time (min)


Patient: 4
Plasma Glucose (mg/dL)
Plasma Insulin (μU/mL)
Time (min)


Patient: 5
Plasma Glucose (mg/dL)
Plasma Insulin (μU/mL)
Time (min)


Patient: 6
Plasma Glucose (mg/dL)
Plasma Insulin (μU/mL)
Time (min)


Patient: 7
Plasma Glucose (mg/dL)
Plasma Insulin (μU/mL)
Time (min)

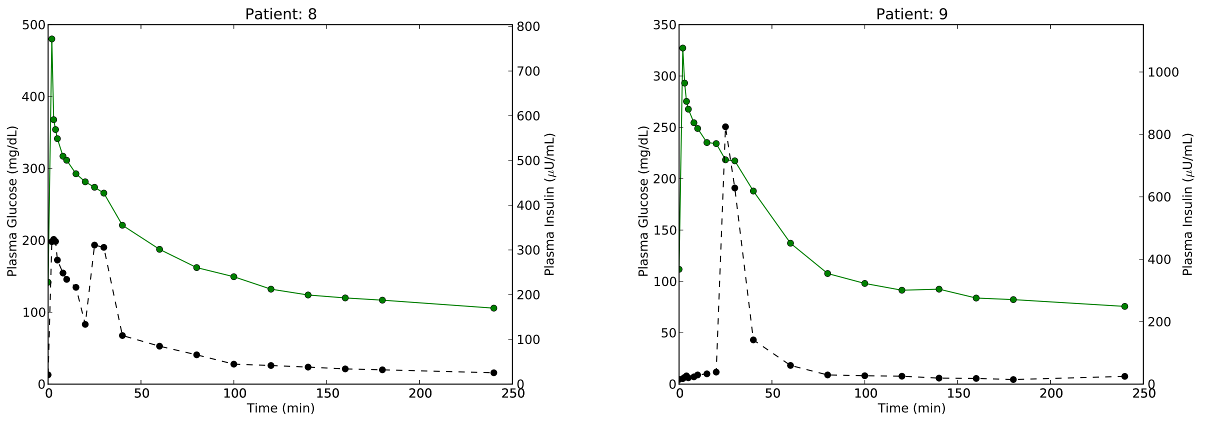


Figure 2: IVGTT data from [6]

3 Methodology

In this research we investigate three different methods of parameter identification for differential equation models. These methods are applied to models of the form

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}, t, \boldsymbol{\theta})$$

where $\mathbf{x}$ is the state of the system, t is time or some suitable independent variable and $\boldsymbol{\theta}$ is a vector of unknown parameters to be estimated. The methods investigated fall into the three different categories:

- **Differentiation of Data**

  With this method, finite differences are used to estimate the derivatives of a dynamic model, using adjacent data values. Then fitting the parameters of the model using linear least squares. The accuracy of this method is usually limited, as approximating a local differential by including non local data can lead to large errors in the computed derivatives. The speed of this method is much faster than that of the alternatives and may be useful to find an initial estimate for other methods.

- **Integration of Equations**

  This method requires that an initial estimate of the parameters be made, then a solution to the dynamic model can be evaluated and compared to the data with a suitable metric. Then derivative data is used to obtain a solution closer to the data. This method is by far the most costly to compute, however will obtain at least a local minimum. This method also has the problem of convexity. A global optimum will not always be found.

- **Integration of Data**

  This method requires the dynamic model to be transformed into an integral equations. The integrals occurring in these equations are evaluated numerically, this allows the parameter estimation problem to be transformed into a purely algebraic problem.

3.1 Differentiation of Data

In order to avoid a nonlinear optimisation (in addition to the parameter estimation) we limit this method to a small subset of models that are linear in the parameters. Thus the model will have the form

$$\frac{d\mathbf{x}}{dt} = \sum_{i=1}^{n} \boldsymbol{\theta}_i \mathbf{g}_i(\mathbf{x}, t) + \mathbf{h}(\mathbf{x}, t) \tag{17}$$

where $\mathbf{g}(\mathbf{x}, t)$ and $\mathbf{h}(\mathbf{x}, t)$ are some known functions. These functions are not required to be linear, which extends the class of models that can be handled by this method.

The essence of this method is to discretise the derivative $\frac{d\mathbf{x}}{dt}$. If we take the Taylor series expansion of a solution $\mathbf{x}(t)$ about a point t in both the forward and backward direction we obtain

$$\mathbf{x}(t_i + h) = \mathbf{x}(t_i) + h\frac{d\mathbf{x}(t_i)}{dt} + h^2\frac{d^2\mathbf{x}(t_i)}{dt^2} + \mathcal{O}(h^3)$$
$$\mathbf{x}(t_i - h) = \mathbf{x}(t_i) - h\frac{d\mathbf{x}(t_i)}{dt} + h^2\frac{d^2\mathbf{x}(t_i)}{dt^2} + \mathcal{O}(h^3)$$

subtracting the first equation from the second and solving for the first derivative yields the central difference approximation to the derivative

$$\frac{d\mathbf{x}(t_i)}{dt} = \frac{\mathbf{x}(t_{i+1}) - \mathbf{x}(t_{i-1})}{2h} + \mathcal{O}(h^2).$$

Letting $\mathbf{x}_i = \mathbf{x}(t_i)$ and replacing the derivative term (17) becomes

$$\sum_{j=1}^{n} \boldsymbol{\theta}_j \mathbf{g}_j(\mathbf{x}_i, t_i) = \frac{\mathbf{x}_{i+1} - \mathbf{x}_{i-1}}{t_{i+1} - t_{i-1}} - \mathbf{h}(\mathbf{x}_i, t_i) \tag{18}$$

giving us a system of linear equations in $\boldsymbol{\theta}_j$ which for $m \geq n$ data points we can solve using linear least squares to obtain an estimate of $\boldsymbol{\theta}_j$.

3.2 Integration of Equations

In this method we aim to find a minimum least squares error between the model curve for a given parameter set and the data. This requires the integration of the differential equations describing the system. Therefore this method is the most general as it can be applied to any equation; the down side is that the computational expense is far greater than that of other methods and it also does not guarantee that a local minimiser will be found.

The model equation for this family of methods is the sum of squares of the residual vector $\mathbf{r}$ between measured data y_i and the modeled data $f(t_i, \boldsymbol{\theta})$ where $\boldsymbol{\theta}$ is the vector of parameters of the model; that is,

$$\mathbf{r}_i = y_i - f(t_i, \boldsymbol{\theta}). \tag{19}$$

The sum of squares of the residual gives the objective function $F(\boldsymbol{\theta})$

$$F = \frac{1}{2}\mathbf{r}^T\mathbf{r}. \tag{20}$$

At a stationary point, the gradient of $F(\boldsymbol{\theta})$ will be zero. We therefore construct an iterative method to obtain this minima. We first linearise the gradient of (20) about a current parameter estimate $\boldsymbol{\theta}$,

$$\nabla F(\boldsymbol{\theta} + \delta\boldsymbol{\theta}) = \nabla F(\boldsymbol{\theta}) + \nabla^2 F(\boldsymbol{\theta})\delta\boldsymbol{\theta} + \text{higher order terms.} \tag{21}$$

If we ignore the higher order terms and set $\nabla F(\boldsymbol{\theta} + \delta\boldsymbol{\theta}) = 0$, we obtain a system of linear equations in $\delta\boldsymbol{\theta}$ that hopefully be a better approximation to the solution of $\nabla F(\boldsymbol{\theta}) = 0$,

$$\nabla^2 F(\boldsymbol{\theta})\delta\boldsymbol{\theta} = -\nabla F(\boldsymbol{\theta}). \tag{22}$$

The gradient and Hessian matrix of (20) are

$$\nabla F(\boldsymbol{\theta}) = \nabla \mathbf{r}^T \mathbf{r} = -\mathbf{J}^T \mathbf{r} \tag{23}$$

$$\nabla^2 F(\boldsymbol{\theta})_{jk} = \sum_{i=0}^{m-1} \left(\frac{\partial r_i}{\partial \theta_j} \frac{\partial r_i}{\partial \theta_k} + r_i \frac{\partial^2 r_i}{\partial \theta_j \partial \theta_k} \right) \tag{24}$$

where $\mathbf{J}$ is the Jacobian matrix of $f(t_i, \boldsymbol{\theta})$ evaluated at $\boldsymbol{\theta}$. We then make a further assumption that the second derivatives occurring in (24) are much smaller than that of the product of the first order derivatives, in order to obtain the Gauss-Newton method from (22)

$$\mathbf{J}^T \mathbf{J} \delta\boldsymbol{\theta} = \mathbf{J}^T \mathbf{r}. \tag{25}$$

Thus we are able to update our current parameter estimate by solving (25) for $\delta\boldsymbol{\theta}$ and then calculate $\boldsymbol{\theta}^{(n+1)} = \boldsymbol{\theta}^{(n)} + \delta\boldsymbol{\theta}$.

The Gauss-Newton method can be extended further by introducing a damping factor λ into (25), suitably scaled to the diagonal entries of $\mathbf{J}^T\mathbf{J}$

$$\left(\mathbf{J}^T \mathbf{J} + \lambda \operatorname{diag}\left(\mathbf{J}^T \mathbf{J} \right) \right) \delta\boldsymbol{\theta} = \mathbf{J}^T \mathbf{r}. \tag{26}$$

This method is known as the Levenberg-Marquardt method see [8, p227] for more detail, and the dampening factor λ is chosen at each iteration to ensure that a lower objective function value is obtained.

3.3 Integration of data

In the same manner as the method of differentiation of data, we choose to restrict the choice of model to that of the form of (17), then integrating both sides of this equation between 0 and t we obtain,

$$\mathbf{x}(t) - \mathbf{x}(0) = \sum_{i=1}^{n} \boldsymbol{\theta}_i \int_0^t \mathbf{g}_i (\mathbf{x}, \tau) d\tau + \int_0^t \mathbf{h} (\mathbf{x}, t) \, d\tau \tag{27}$$

If we have m discrete measurements, we can evaluate (27) at each of these time values and form a system of equations linear in θ_i and the integrals of functions of the unknown solution. From the data we are able to evaluate the integrals numerically.

4 Results

For this research the *Python* [9] language was chosen to implement each of the methods presented in the preceding section. This language was chosen as it is an interpreted language and also free software. Together with the module *Scipy* [10] the combination provides a very good extensible basis on which to carry out scientific computing. The *Scipy* module provides a wrapper of many of the *MINPACK* optimisation routines written in *FORTRAN*. *Python* also has many other interfaces to other scientific computing libraries, including the Gnu Scientific Library which also has many routines written in *C* for optimisation.

All three methods inherit the class `Patient` from the file `read_tools.py` included in Appendix D. This provides methods to read and plot the measured data for each patient. The object oriented functionality is one of the advantages to programming in *Python*.

4.1 Differentiation of data

The model described by (5) and (6) poses significant problems for this method, since the parameters p_1 and p_2 do not occur linearly in in functions of G and F. We must either set one of these parameters to a population estimate, or alternatively optimise one of the parameters separately. This however may result in being more computationally expensive than integrating the equations directly. Another significant problem with this method is the need to estimate the second derivatives $\dot{F}$, since any error in measuring G will be significant magnified in taking finite difference approximations.

Furthermore if we try to apply this method to the original formulation of the model as in (1) and (2) we do not have data for the insulin action compartment $X(t)$. Thus we are not able to form finite differences at all. This is a significant limitation to this method.

This method also does not lend itself well to approximating the parameter G_0. This can be estimated by

- Extrapolating the linear interpolant of the measured glucose samples in the range 5—10 min backward to zero as suggested in [11, p73].
- Estimating the the glucose distribution volume V_G from a patients body mass index, then calculating G_0 from the intravenous glucose dose through $G_0 = G_{dose}/V_G$.

The parameter p_1 must be estimated through population averages since there is no other way that this parameter can be estimated using this method. The p_1 value was calculated from taking the average of the estimates obtained by [6] as $p_1 = 1.305e-2$.

We obtain the system of linear equations from (6)

$$p_2 p_1 (G_i - G_b) + p_3 G_i (I_i - I_b) = \frac{F_i}{G_i}(F_i - p_1 G_b) - \frac{dF_i}{dt} \quad i = 0 \ldots m-1 \tag{28}$$

where G_i is the glucose concentration at time t_i, F_i and $\frac{dF_i}{dt}$ are the first and second derivatives of G at time t_i which are estimated using central difference approximations from the measured glucose concentrations.

The *Python* implementation is included in Appendix D as `int_diff_method.py` which contains a class `Int_Diff_method` with methods to approximate the first and second order derivatives of G and estimate the parameters using both differentiation of data and integration of data.

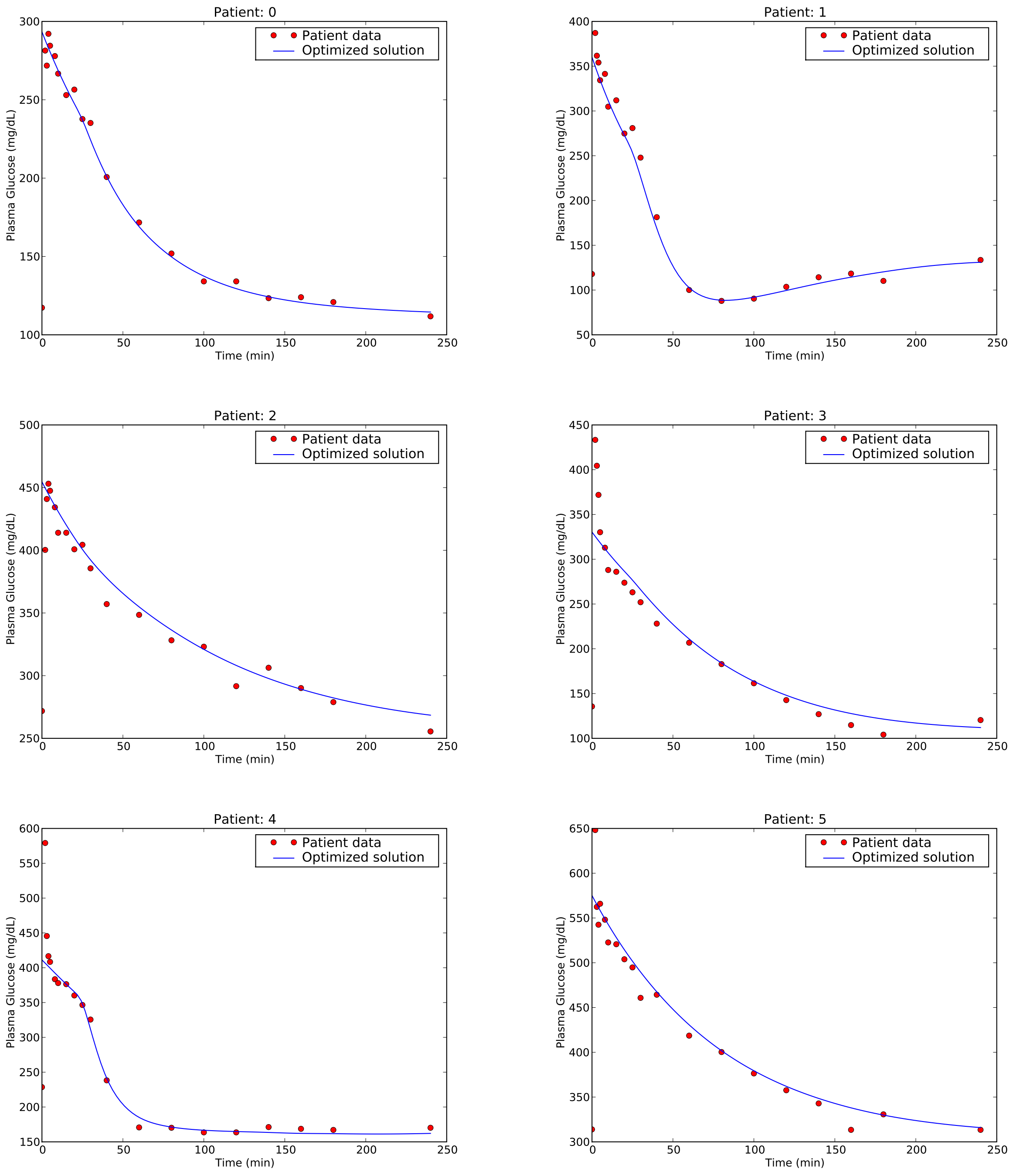
The results of this method are presented in Table 6 in Appendix I, as all except one of the estimates for the parameter S_I were found to be negative. This is not acceptable as this will not conserve the mass of the system.

The conclusion from this result should not suggest that this method is completely invalid, it does however illustrate some of the pitfalls of using this method. Namely the accuracy of measurements must be good to produce accurate derivative approximations. It should also be noted that the solution curve should be smooth enough, so that the differencing step size is small enough to not induce large amounts of error.

4.2 Integration of equations

This method is by far the most flexible of the three methods. The implementation of the Levenberg-Marquardt algorithm in the *Scipy* module requires only the vector of residuals to be computed. Finite difference approximations are used to calculate the Jacobian matrix. However we can solve the sensitivity equations given by (7) to (14) to obtain the Jacobian matrix with much greater accuracy than finite differences.

The fitting of parameters was carried out using the methods found in `model.py` and `optim_tools.py` in Appendix D both of these files provide methods to solve the system of differential equations and optimise the parameters G_0, p_1, p_2 and p_3. Figure 3 below shows the glucose concentrations obtained from the optimised parameters and the patient data. All show a good correspondence between the data and the optimised solution.



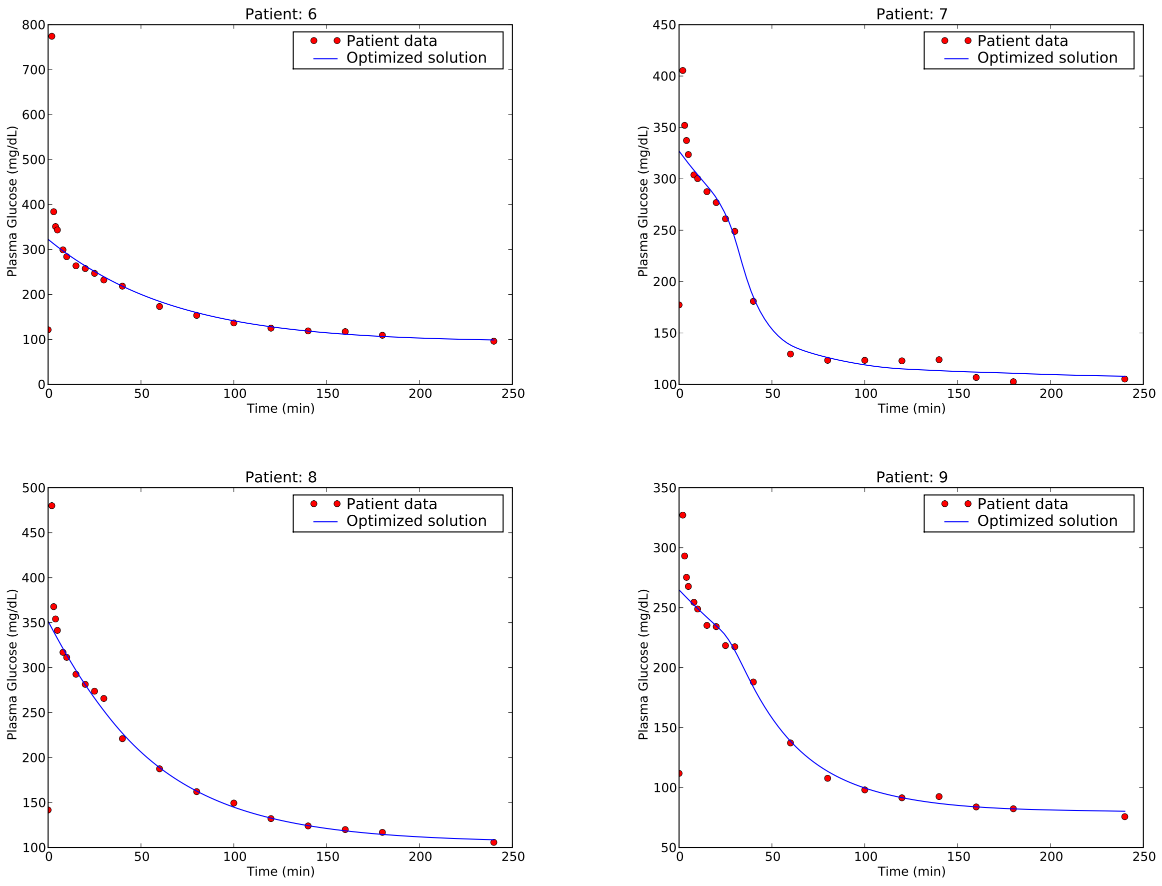


Figure 3: Optimised solutions and patient data

Table 1 shows the optimised parameter estimates obtained and the norm of the residual at the optimum. All estimates, except for patient 2, have positive values for the parameters p_1, p_2 and p_3. As this particular implementation of the Levenberg-Marquardt algorithm performs unconstrained minimisation, these results are very promising for this method.

Table 1: Optimised parameter estimates

Patient	G_0	p_1	p_2	p_3	S_i	Residual
0	293.3326	0.01463	0.04316	$2.877e-6$	$6.666e-5$	16.8
1	360.1065	0.02387	0.03953	$8.374e-6$	$2.119e-4$	43.2
2	454.5988	0.01267	0.01533	$-2.274e-7$	$-1.483e-5$	35.1
3	330.3164	0.01155	0.004632	$4.282e-7$	$9.246e-5$	39.4
4	411.1837	0.009455	0.08611	$1.798e-5$	$2.088e-4$	29.9
5	575.0986	0.01317	0.002355	$2.222e-8$	$9.436e-6$	38.4
6	322.5899	0.01508	0.007609	$1.203e-7$	$1.581e-5$	48.2
7	326.9480	0.01072	0.256	$2.556e-5$	$9.985e-5$	24.7
8	351.6261	0.01611	0.026	$2.991e-7$	$1.15e-5$	20.7
9	264.8381	0.008629	0.02673	$1.572e-6$	$5.881e-5$	19.7

The estimation of G_b used in the preceding optimisation was found by taking the glucose measurement at $t = 240$min. This assumes that the glucose concentration will return to basal levels by this time. However we further investigate the role that this parameter plays by optimising G_b also. The sensitivity equations for G_b are

$$\frac{d}{dt}\left(\frac{\partial G}{\partial G_b}\right) = \frac{\partial F}{\partial G_b} \tag{29}$$

$$\frac{d}{dt}\left(\frac{\partial F}{\partial G_b}\right) = \frac{\partial M}{\partial G}\frac{\partial G}{\partial G_b} + \frac{\partial M}{\partial F}\frac{\partial F}{\partial G_b} - p_1\frac{F}{G} + p_1 p_2 \tag{30}$$

where $\frac{\partial M}{\partial G}$ and $\frac{\partial M}{\partial F}$ are defined in (15) and (16).

Table 2 below show the parameter estimates obtained from fitting the model parameters G_0, p_1, p_2, p_3 and G_b. This extra parameter has drastically changed the estimates of p_2 and p_3 unfortunately being negative for some of the patients. This suggests that there is a trade off in fitting the parameters from such a small number of observations $m = 20$. For patients 0, 1, 4, 7 and 9, where the estimates are physiologically reasonable, there is a reduction in the parameter p_1 with a corresponding increase in in G_b. From (1) the cross term $p_1 G_b$ implies that there is likely to be some tradeoff in fitting these parameters together.

Table 2: Optimised parameter estimates fitting G_b

Patient	G_0	p_1	p_2	p_3	G_b^{est}	G_b^{opt}	S_i	Residual
0	292.9451	0.01306	0.0454	$3.586e-6$	111.8	110.7	$7.898e-5$	15.8
1	353.8095	0.01682	0.0547	$1.083e-5$	133.6	138.2	$1.981e-4$	32.4
2	455.2201	0.01042	0.0009064	$-6.378e-7$	255.5	174.3	$-7.037e-4$	29.9
3	329.0296	0.01072	−0.01629	$-3.345e-8$	120.4	55.37	$2.054e-6$	27.4
4	407.9166	0.009819	0.06887	$1.677e-5$	170.2	189	$2.435e-4$	24.3
5	608.8128	0.07574	−0.001648	$1.297e-6$	313.4	467.3	$-7.868e-4$	32.8
6	362.2127	0.03	0.2658	$-1.304e-5$	96.01	104	$-4.906e-5$	29.6
7	328.1521	0.01185	0.2395	$2.316e-5$	105.2	106.1	$9.672e-5$	24.3
8	349.7680	0.02105	−0.003085	$7.162e-7$	105.7	181.4	$-2.321e-4$	19.9
9	271.5651	0.01085	0.0261	$1.29e-6$	75.69	75.25	$4.941e-5$	17.3

Figure 7 in appendix B shows the optimised solutions given by the parameters in Table 2. They show good correspondence to the data. However for patients 0, 1, 4, 7 and 9 the extra parameter does not significantly effect the dynamics of the optimised solution.

The motivation behind optimising G_b was to ensure that the basal glucose concentration is correctly estimated. Another approach would be to take a further measurement much after $t = 240$min. This would ensure that a patient had returned to a basal level. However this would increase the time that a clinician must supervise the patient which may not be acceptable.

4.3 Integration of data

As with the method of differentiation of data, the integration of data also has many similar problems in estimating the parameters of this model. We must also estimate the parameter p_1 some other way, due to the nonlinear term $p_1 p_2$ in (6).

This method does not need to estimate the second derivatives from the data. Only the first derivatives need to be estimated. Also the first derivatives occur inside an integral term. Thus one could hope that some of the error will be taken up in integrating the estimated derivatives.

The system of linear equations are formed by integrating (6) between 0 and t_i for $i = 0..m-1$ and substituting the initial condition $F(0) = p_1 (G_b - G_0)$ from (6)

$$G_0 + p_2 \int_0^{t_i} p_1 (G(t) - G_b)\, dt + p_3 \int_0^{t_i} G(t) (I(t) - I_b)\, dt = \int_0^{t_i} \frac{F(t)}{G(t)} (F(t) - p_1 G_b)\, dt - F(t_i) + p_1 G_b \quad (31)$$

This allows the parameter G_0 to be estimated directly. Thus one could hope that better estimates are obtained than through indirect estimation.

The least squares solution to this system is then found giving the parameters G_0, p_2 and p_3, for all except patient 0 the estimates for S_I are negative. This does not represent physiologically reasonable conditions.

The approximation of derivative data needed for this method is likely to be one of the main causes of the the failure. One way to try to get around this problem might be to use the original model described by (1) and (2) also fitting the insulin action compartment $X(t)$ as a piecewise constant parameter between measurements, however this was not carried out in this research.

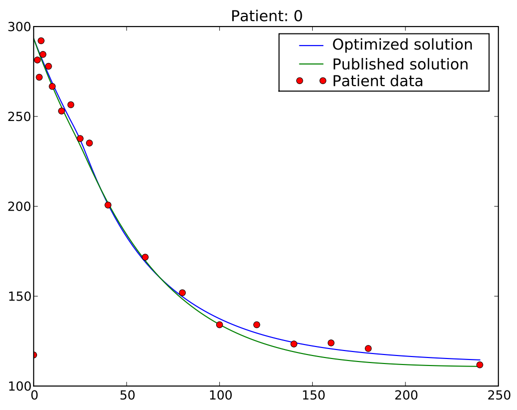
This method does have a little more flexibility than differentiating the data, since the integrals are defined over a range of points. In contrast, differentiating data requires the inclusion of non local data into the derivative estimate.

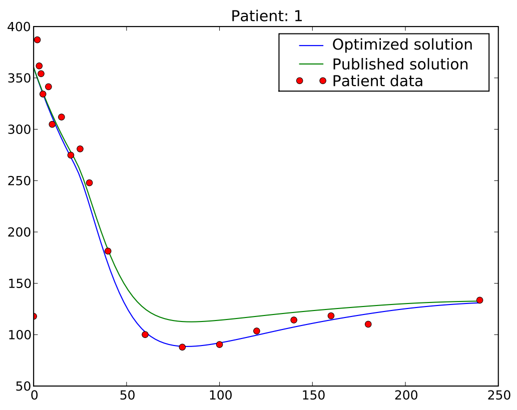
Again the integration of data method should not be completely disregarded on the evidence presented here. We have just illustrated some of the problems associated with using this method. For other models of the metabolic system this method produces credible parameter estimates (see [11, p72]).

5 Published parameter estimates

In this section we present a comparison between the parameters published in [6] and the parameters we have estimated using the Levenberg-Marquardt algorithm presented earlier. Figure 4 shows the differences between the solutions. This shows half of the solutions correlate very well. However the published parameters for patient 1 does not fit the data well at all.

The software used in [6] was *MINMOD* [12]. This software uses a gradient based algorithm to estimate the parameters. It also uses the sensitivity equations to evaluate the Jacobian matrix. The algorithm used for the estimation also has the ability to perform constrained optimisation.





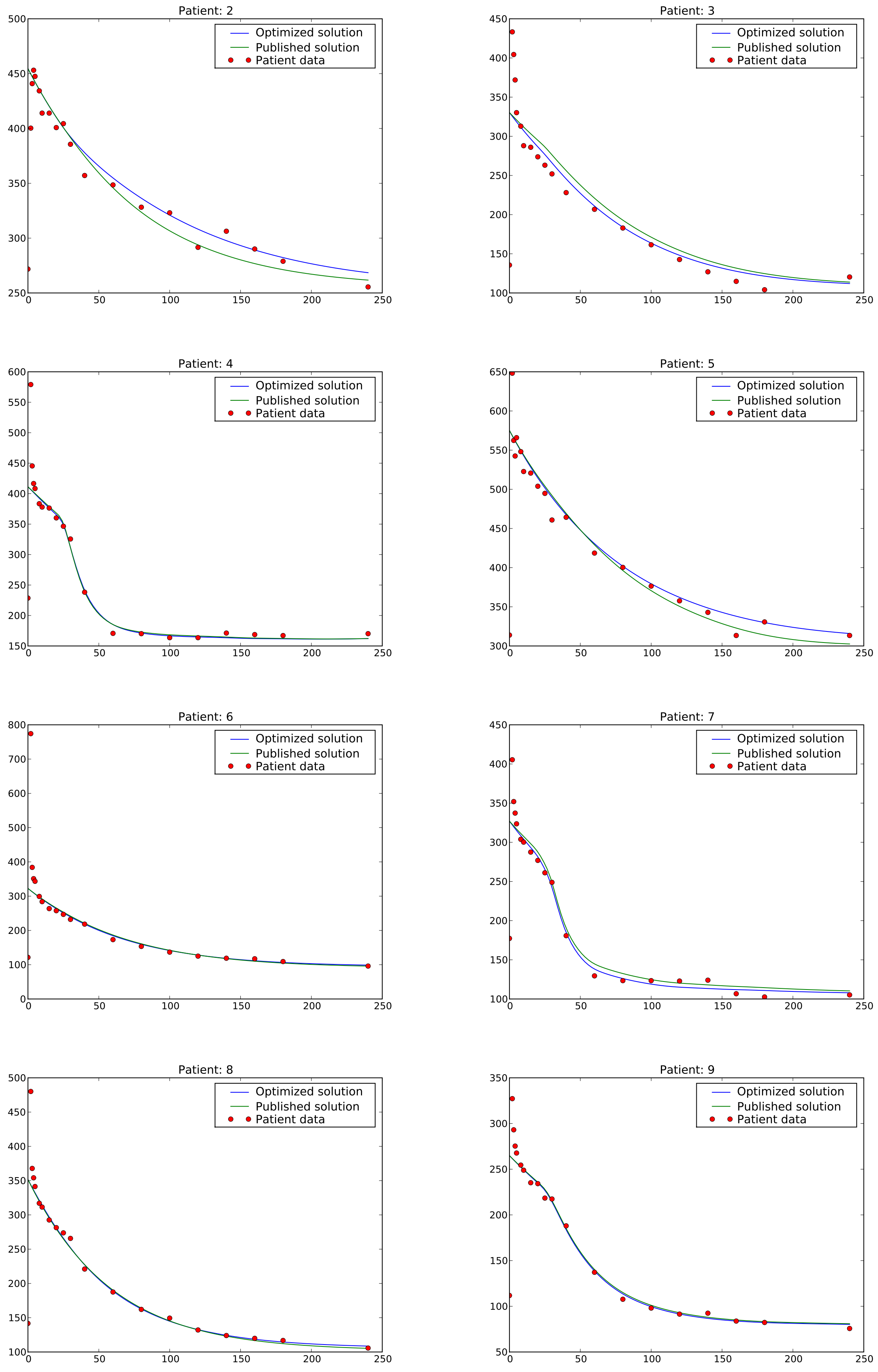


Figure 4: Optimised solutions, patient data and published solutions

Table 3 shows the parameter estimates from [6]. As well as the norm of the residual for the solution obtained from these parameters.

Table 3: Published parameters

Patient	G_0	p_1	p_2	p_3	S_i	Residual
0	297.9	0.0154	0.015	$1.38e-6$	$9.2e-5$	25.19
1	381.6	0.021	0.072	$8.28e-6$	$1.15e-4$	61.44
2	444.13	0.0133	0.00495	$1.089e-7$	$2.2e-5$	43.31
3	315.6	0.0097	0.007	$6.02e-7$	$8.6e-5$	69.46
4	409.0	0.0087	0.098	$2.058e-5$	$2.1e-4$	31.66
5	563.0	0.0133	0.0074	$1.036e-7$	$1.4e-5$	50.95
6	315.0	0.015	0.0016	$1.568e-7$	$9.8e-5$	46.64
7	323.8	0.009	0.3	$2.97e-5$	$9.9e-5$	34.83
8	353.9	0.017	0.0011	$5.72e-8$	$5.2e-5$	22.57
9	263.0	0.0084	0.027	$1.566e-6$	$5.8e-5$	20.06

6 Discussion

In this research we have described three methods for estimating parameters of differential equation models. Applying each of these methods to estimate the parameters of Bergman's minimal model from IVGTT data.

Both the integration and differentiation of data methods produced estimates that were not physiologically reasonable. We suspect this is mainly due to the error introduced from approximating the derivatives using finite differences. For this particular estimation problem, both methods are forced to make assumptions about some of the parameters. Which has been shown to significantly limit the method. However for other models of the metabolic system these limitations do not arise see [11, p72].

The Levenberg-Marquardt algorithm performed well in estimating the parameters the model. However there was an obvious limitation in using unconstrained optimisation. This caused some of the parameters to become negative which is not physiologically possible. The Levenberg-Marquardt algorithm implementation from *MINPACK* does allow for constrained optimisation. However the wrapper in *Scipy* for this algorithm used does not provide this functionality. There is however an interface to the *Gnu Scientific Library* which does provide this functionality. This implementation warrants further investigation.

The optimisation of the parameter G_b in addition to the other parameters did not give physiologically reasonable parameter estimates. However where the estimates were reasonable, the basal glucose concentration was above the concentration estimated using the last measurement from the IVGTT. This shows that this parameter may be worth optimising, provided that the algorithm can be constrained to give positive parameter estimates.

The robustness of the parameter estimates produced by the Levenberg-Marquardt algorithm was investigated. Table 8 in appendix C shows the parameter estimates obtained for patient 1. This uses random initial estimates of the parameters within 70% of the published estimates. All estimates are equal up to 3 significant figures. This shows that for this patient the local minimum is well defined. However this may not be the case for other patients. In cases where the local minima is not so well defined, it may be necessary to peform a grid search to obtain the best local minima within a reasonable range of the parameters.

The comparison to the published data shows good correspondence between the solution curves of the model. There are cases where the published results showed significantly different solutions. This is illustrated better in figure 5 and 6. The solution for patient 1 shows significant discrepencies between the data and published soultion. This is due to the weights put on the residuals. The weights used in [6] are the reciprocal of the variance of the measurement error. This is suggested to be an optimal weighting scheme for optimizing the parameters. However this weighting has significantly reduced the goodness of fit. Table 4 shows significant differences in the parameters estimates for patient 1.

For patient 4 the situation is far better. Both parameter estimates produce solutions that fit the data very well. However Table 5 shows that there are significant differences in the parameters p_2 and p_3 yet both produce solutions that are within 2% of each other. The S_I parameter provides the answer to this discrepency as the difference between the estimated S_I values is much smaller. This shows that it is the combination of p_2 and p_3 rather than the parameters themselves that is most important to estimating optimal parameters. Reformulating the model to include S_I rather than just p_2 and p_3 warrants fruther investigation.

The model was not reformulated to include S_I since the some of the parameter terms become nonlinear in both p_2 and S_I. To use differentiation and integration of data methods on the reformulated model this would require assumptions about the parameter p_2. Since the parameter p_2 contains information about the insulin sensitivity any assumption about this parameter would bias the estimation of S_I. However the Levenberg-Marquardt algorithm can be applied to the reformulated model without such issues.

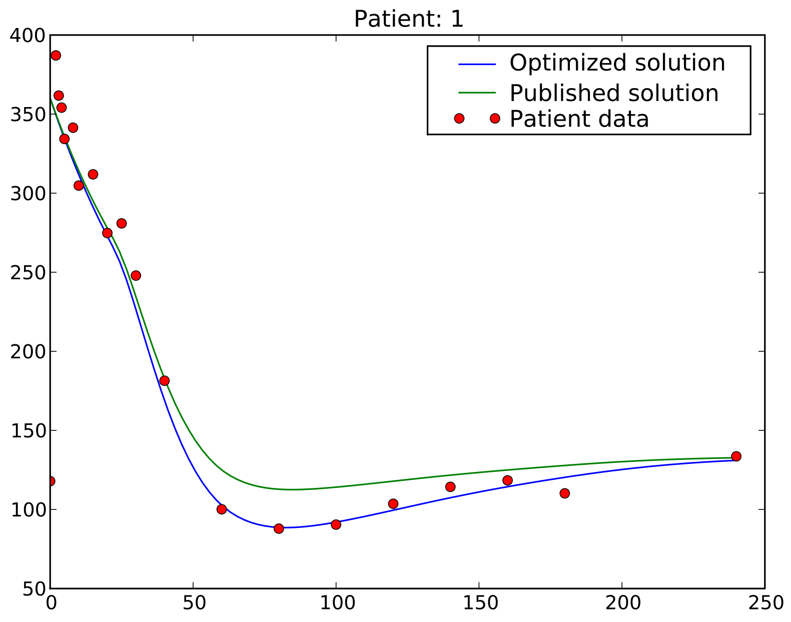


Figure 5: Comparison of solutions for patient 1

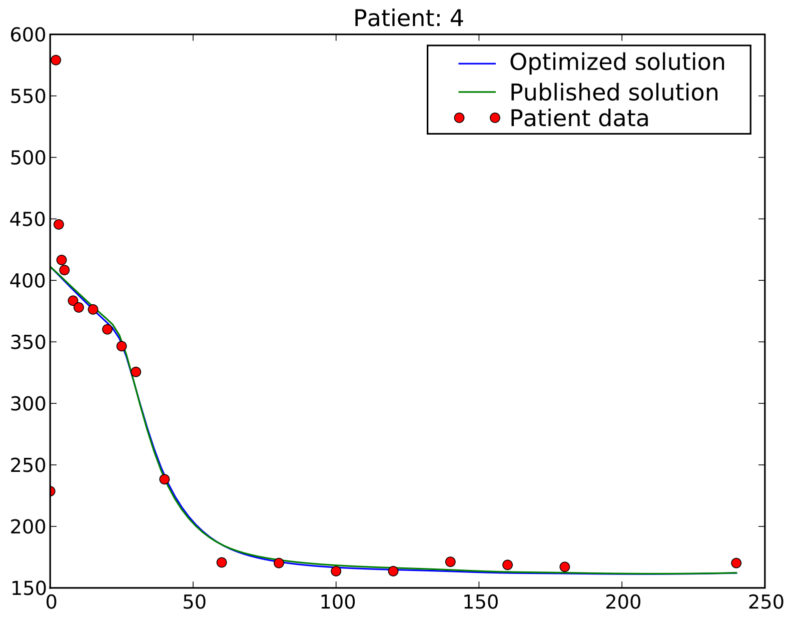


Figure 6: Comparison of solutions for patient 4

Table 4: Comparison of parameter estimates for patient 1

	G_0	p_1	p_2	p_3	S_i	Residual
Optimised parameters	360.1065	0.0239	0.0395	$8.374e-06$	0.0002119	43.24
Published parameters	381.6000	0.0210	0.0720	$8.28e-06$	0.000115	61.44
Percentage difference	-5.9686	12.0142	-82.1550	1.1209	45.7171	-42.08

Table 5: Comparison of parameter estimates for patient 4

	G_0	p_1	p_2	p_3	S_i	Residual
Optimised parameters	411.1837	0.0095	0.0861	$1.798e-5$	$2.0878e-4$	29.92
Published parameters	409.0000	0.0087	0.0980	$2.058e-5$	$2.1e-4$	31.65
Percentage difference	0.5311	7.9866	-13.8143	-14.4818	-0.5866	-5.81

Appendices

A Differentiation and Integration method parameter estimates

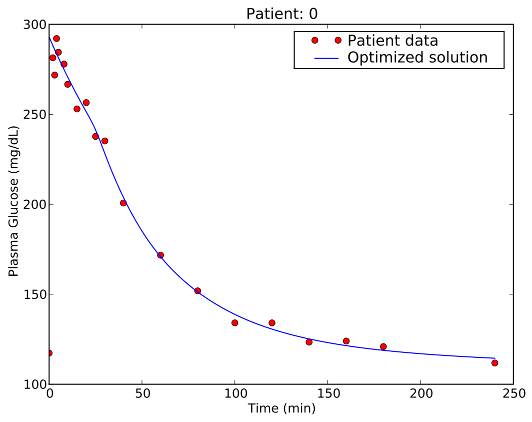
Patient	G_0	p_2	p_3	S_I
0	302.3	-0.009292	$-6.269e-06$	0.0006747
1	363.8	1.86	$-3.333e-05$	$-1.792e-05$
2	481	-2.664	$9.184e-05$	$-3.447e-05$
3	372.4	3.533	$-1.28e-05$	$-3.623e-06$
4	438.8	8.141	-0.0002387	$-2.932e-05$
5	609.1	2.908	$-3.627e-05$	$-1.247e-05$
6	402.8	28.56	-0.00105	$-3.678e-05$
7	347	3.375	-0.0001024	$-3.033e-05$
8	371.4	25.55	-0.0008403	$-3.289e-05$
9	286.5	2.462	$-3.615e-05$	$-1.468e-05$

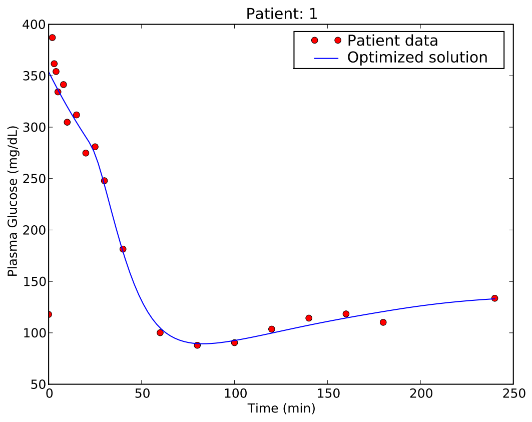
Table 6: Estimated parameters using differentiation of data

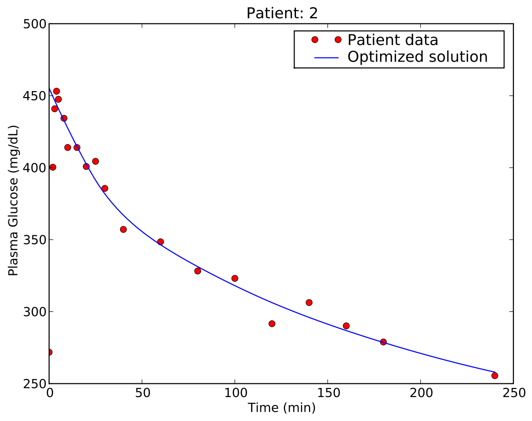
Patient	G_0	p_2	p_3	S_I
0	327.6	0.003213	$1.604e-06$	0.0004991
1	735.6	-0.004928	$1.116e-06$	-0.0002265
2	690.3	0.02379	$-4.488e-07$	$-1.886e-05$
3	1657.6	-0.09166	$3.482e-06$	$-3.799e-05$
4	3769.2	-0.3228	$1.943e-05$	$-6.018e-05$
5	2003.3	-0.1995	$3.528e-06$	$-1.769e-05$
6	15748.1	0.1703	$-1.725e-05$	-0.0001013
7	1469.8	-0.1498	$8.939e-06$	$-5.967e-05$
8	2624.0	0.1245	$-1.028e-05$	$-8.256e-05$
9	963.7	-0.08294	$2.919e-06$	$-3.52e-05$

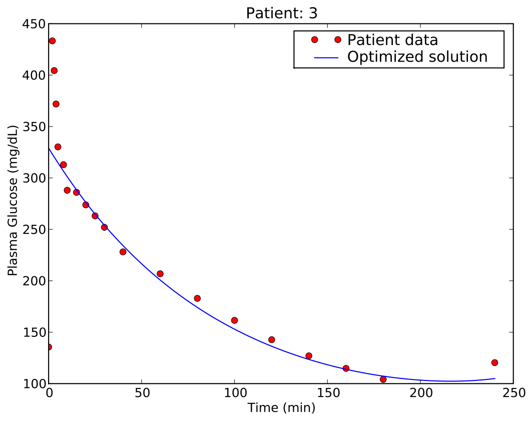
Table 7: Estimated parameters using integration of data

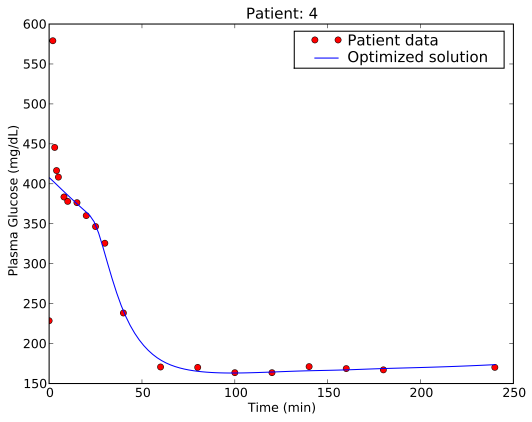
B Parameter estimates fitting G_b

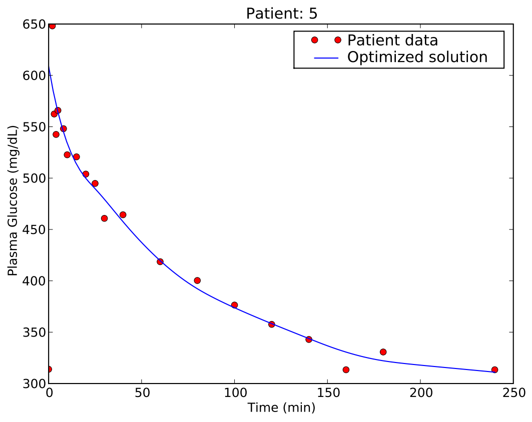


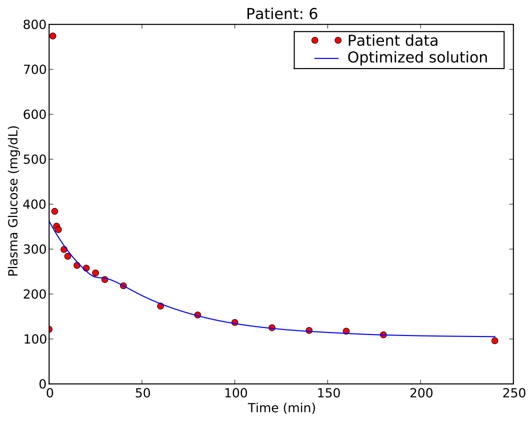


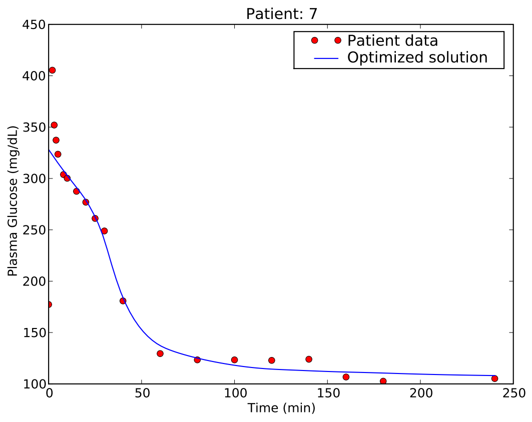


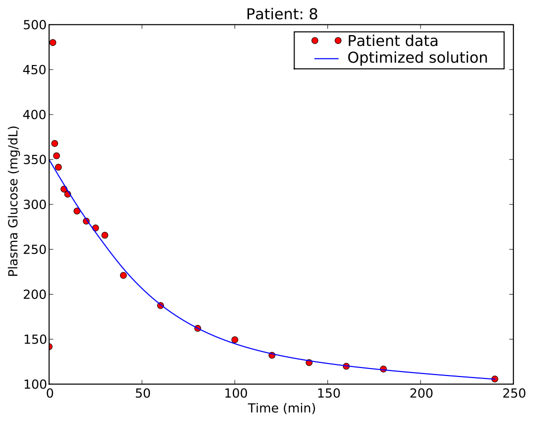












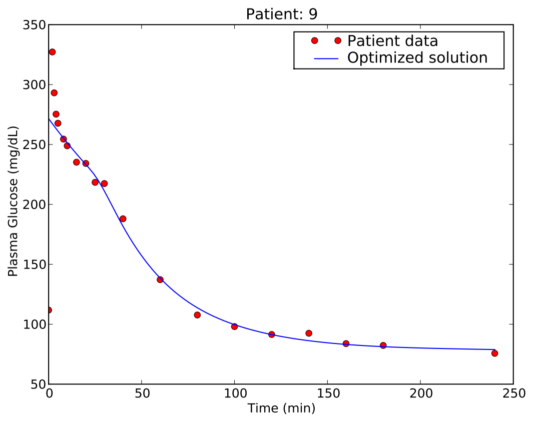


Figure 7: Optimised solutions and patient data fitting parameter G_b

C Robustness of parameter estimates

G_0	p_1	$p2$	p_3	residual
355.1	0.01683	0.05959	$1.104e-5$	32.85
355.1	0.01683	0.05958	$1.104e-5$	32.85
355.1	0.01683	0.05959	$1.104e-5$	32.85
355.1	0.01683	0.05958	$1.103e-5$	32.85
355.1	0.01683	0.05952	$1.103e-5$	32.85
355.1	0.01683	0.05958	$1.104e-5$	32.85
355.1	0.01683	0.05958	$1.104e-5$	32.85
355.1	0.01683	0.05958	$1.104e-5$	32.85
355.1	0.01683	0.05959	$1.104e-5$	32.85
355.1	0.01683	0.05958	$1.104e-5$	32.85
355.1	0.01685	0.05952	$1.103e-5$	32.85
355.1	0.01683	0.05958	$1.104e-5$	32.85
355.1	0.01683	0.05959	$1.104e-5$	32.85
355.1	0.01685	0.05952	$1.103e-5$	32.85
355.1	0.01683	0.05959	$1.104e-5$	32.85
355.1	0.01683	0.05958	$1.104e-5$	32.85
355.1	0.01684	0.05953	$1.103e-5$	32.85
355.1	0.01683	0.05959	$1.104e-5$	32.85
355.1	0.01684	0.05956	$1.103e-5$	32.85
355.1	0.01683	0.05959	$1.104e-5$	32.85

Figure 8: Parameter estimates with randomly selected initial conditions

D *Python* Code

```
#!/usr/bin/env python

from pylab import plot,show,figure,twinx,xlabel,ylabel,ylim,savefig,legend,title
from glob import glob
from scipy.io.netcdf import netcdf_file
from scipy import array

class Patient:
    def __init__(self,filename):
        self.nc_file=filename
        self.data={}
        self.read_nc()

    def read_nc(self):
        fp=netcdf_file(self.nc_file,'r')

        for key in fp.variables.iterkeys():
            self.data[key]=array(fp.variables[key][:])
        fp.close()

    def plot_paper_data(self,out=None):
        imax=max(self.data['glucose'])+max(self.data['insulin'])
        ax1=figure()
        plot(self.data['t_glucose'],self.data['glucose'],'go-')
        ylim(0)
        xlabel('Time (min)')
        ylabel('Plasma Glucose (mg/dL)')

        ax2=twinx()
        plot(self.data['t_insulin'],self.data['insulin'],'ko--')
        ylabel('Plasma Insulin ($\mu$U/mL)')
        ylim([0,imax])
        ax2.yaxis.tick_right()
        title('Patient: '+str(self.nc_file[-4]))
        if(out==None):
            pass
            #show()
        else:
            savefig(out)

#!/usr/bin/env python

from glob import glob
from scipy.optimize import leastsq
from read_tools import Patient
from scipy import array,zeros,linspace

from scipy import interp,zeros,sqrt,diag
from scipy.integrate import odeint
from scipy import rand,ones,concatenate
from pylab import plot,show,draw,figure,close,savefig,xlabel,ylabel,clf,legend,title



class B_optim(Patient):
    """B_optim provides optimization tools for Bergman's\n
    minimal model, it provides methods to cook data and\n
    run the optimization"""

    def __init__(self,filename):
        #Read in data and parameters from file
        Patient.__init__(self,filename)

        #Set parameter array
        self.theta_t=concatenate([self.data['G0'],\
                                  self.data['p1'],\
                                  self.data['p2'],\
                                  self.data['p3']])


        #Initialize parameters
        self.p={'Gb':self.data['glucose'][-1],\
                'Ib':self.data['insulin'][-1],\
                'G0':self.data['G0'],\
                'p1':self.data['p1'],\
                'p2':self.data['p2'],\
                        'p3':self.data['p3']}

        #Set weights
        self.weights=1.0e-0*ones(20)
        self.weights[0:4]=0.0
#        self.weights[-6:]=1.0

        #Initialize time vector to plot
        self.t=linspace(0,self.data['t_glucose'][-1],100)


    def cook_data(self):
        self.yc=self.optimize_fdf(self.theta_t)
        self.cooked_error=(self.yc[:,0]-self.data['glucose'])*self.weights

    def plot_cooked(self):
        plot(self.t,self.yc[:,0],label='Published soultion')

    def interp(self,t):
        return interp(t,self.data['t_insulin'],self.data['insulin'])

    def plot_fitted(self):
        plot(self.data['t_glucose'],self.y[:,0],'g-')

    def plot_data(self):
        plot(self.data['t_glucose'],self.data['glucose'],'ro',label='Patient data')

    def calc_si(self):
        self.p['Si']=self.theta[3]/self.theta[2]

    def calc_var(self):
        self.var=sqrt(diag(self.cov_x))

    def write_data(self):
        self.calc_si()
        fp=open(self.outparam,'w')
        for name in self.p.keys():
            fp.write(name+":"+str(self.p[name])+"\n")
        fp.close()

        self.y.tofile(self.outsol)


    def simulate_model(self,theta):
        self.p['p1']=theta[1]
        self.p['p2']=theta[2]
        self.p['p3']=theta[3]

        y0=array([self.theta[0],self.p['p1']*(self.p['Gb']-theta[0])\
                  ,1,0,0,0,-self.p['p1'],self.p['Gb']-theta[0],0,0])

        if(len(theta)>4):
            self.p['Gb']=theta[4]
            y0=array([self.theta[0],self.p['p1']*(self.p['Gb']-theta[0])\
                      ,1,0,0,0,-self.p['p1'],self.p['Gb']-theta[0],0,0,0,self.p['p1']])
        if(len(theta)>5):
            self.p['Ib']=theta[5]
            y0=array([self.theta[0],self.p['p1']*(self.p['Gb']-theta[0])\
                      ,1,0,0,0,-self.p['p1'],self.p['Gb']-theta[0],0,0,0,self.p['p1'],0,0])

        return odeint(self.minimal_model,y0,self.t)

    def plot_simulated(self):
        plot(self.t,self.ys[:,0],label='Optimized solution')


    def plot_legends(self):
        xlabel('Time (min)')
        ylabel('Plasma Glucose (mg/dL)')
        title('Patient: '+str(self.nc_file[-4]))
        legend()

#!/usr/bin/env python

from glob import glob
from scipy.optimize import leastsq
from read_tools import Patient
from scipy import array,zeros,linspace,rand,ones,concatenate,interp
from scipy.integrate import odeint
from pylab import plot,show,savefig,xlabel,ylabel,clf,legend,title
from optim_tools import B_optim

class Model(B_optim):
    def __init__(self,filename):
        B_optim.__init__(self,filename)
        self.theta=concatenate([self.data['G0'],\
                                self.data['p1'],\
                                self.data['p2'],\
                                self.data['p3']])

        self.outparam='../dat/model/dat_param'+self.nc_file[-4]+'.dat'
        self.outsol='../dat/model/dat_sol'+self.nc_file[-4]+'.dat'

    def minimal_model(self,x,t):
        f=zeros(10)
        gg=(-x[1]/x[0]**2*(x[1]-self.p['p1']*self.p['Gb'])\
            -self.p['p1']*self.p['p2']\
            -self.p['p3']*(self.interp(t)-self.p['Ib']))

        ff=(2*x[1]-self.p['p1']*self.p['Gb'])/x[0]-self.p['p2']

        f[0]=x[1]
        f[1]=(x[1]**2-self.p['p1']*self.p['Gb']*x[1])/x[0]\
              -self.p['p2']*x[1]\
              -self.p['p1']*self.p['p2']*(x[0]-self.p['Gb'])\
              -self.p['p3']*(self.interp(t)-self.p['Ib'])*x[0]
        f[2]=x[6]
        f[3]=x[7]
        f[4]=x[8]
        f[5]=x[9]
        f[6]=gg*x[2]+ff*x[6]
        f[7]=gg*x[3]+ff*x[7]-self.p['p2']*(x[0]-self.p['Gb'])\
              -x[1]/x[0]*self.p['Gb']
        f[8]=gg*x[4]+ff*x[8]-self.p['p1']*(x[0]-self.p['Gb'])-x[1]
        f[9]=gg*x[5]+ff*x[9]-x[0]*(self.interp(t)-self.p['Ib'])
        return f

    def optimize_fdf(self,theta):
        self.p['p1']=theta[1]
        self.p['p2']=theta[2]
        self.p['p3']=theta[3]
        y0=array([self.theta[0],self.p['p1']*(self.p['Gb']-theta[0])\
                  ,1,0,0,0,-self.p['p1'],self.p['Gb']-theta[0],0,0])
        self.y=odeint(self.minimal_model,y0,self.data['t_glucose'])
        return self.y

    def optimize_f(self,theta):
        y=self.optimize_fdf(theta)
        return (y[:,0]-self.data['glucose'])*self.weights

    def optimize_df(self,theta):
        y=self.optimize_fdf(theta)
        return y[:,2:6]

    def run_optimize(self):
        print "Optimizing patient: "+self.nc_file[-4]
        self.theta,self.cov_x,self.infodict, self.mesg,self.iter=leastsq(\
            self.optimize_f,self.theta,Dfun=self.optimize_df,full_output=1)


if(__name__=='__main__'):

    path=glob('../IVGTT_data/netcdfdata/patient*.nc')
    path.sort()
    cohort=[]
    for line in path:
        cohort.append(Model(line))

    for patient in cohort:
        patient.run_optimize()
        clf()
        patient.plot_data()
        patient.ys=patient.simulate_model(patient.theta)
        patient.plot_simulated()

        patient.plot_legends()
        title('Patient: '+str(patient.nc_file[-4]))
        savefig('../figures/model/patient'+patient.nc_file[-4]+'.pdf')
        patient.calc_var()
        patient.write_data()

#!/usr/bin/env python

from glob import glob
from scipy.optimize import leastsq
from read_tools import Patient
from scipy import array,zeros,linspace,interp,rand,ones,concatenate
from scipy.integrate import odeint
from pylab import plot,show,savefig,xlabel,ylabel,clf,legend,title
from optim_tools import B_optim

class New_minmod(B_optim):
    def __init__(self,filename):
        B_optim.__init__(self,filename)
        self.outparam='../dat/new_model/dat_param'+self.nc_file[-4]+'.dat'
        self.outsol='../dat/new_model/dat_sol'+self.nc_file[-4]+'.dat'

        self.theta=concatenate([self.data['G0'],\
                                self.data['p1'],\
                                self.data['p2'],\
                                self.data['p3'],\
                                array([self.data['glucose'][-1]])])


    def minimal_model(self,x,t):
        f=zeros(12)
        gg=(-x[1]/x[0]**2*(x[1]-self.p['p1']*self.p['Gb'])\
            -self.p['p1']*self.p['p2']\
            -self.p['p3']*(self.interp(t)-self.p['Ib']))

        ff=(2*x[1]-self.p['p1']*self.p['Gb'])/x[0]-self.p['p2']

        f[0]=x[1]
        f[1]=(x[1]**2-self.p['p1']*self.p['Gb']*x[1])/x[0]\
              -self.p['p2']*x[1]\
              -self.p['p1']*self.p['p2']*(x[0]-self.p['Gb'])\
              -self.p['p3']*(self.interp(t)-self.p['Ib'])*x[0]
        f[2]=x[6]
        f[3]=x[7]
        f[4]=x[8]
        f[5]=x[9]
        f[6]=gg*x[2]+ff*x[6]
        f[7]=gg*x[3]+ff*x[7]-self.p['p2']*(x[0]-self.p['Gb'])\
              -x[1]/x[0]*self.p['Gb']
        f[8]=gg*x[4]+ff*x[8]-self.p['p1']*(x[0]-self.p['Gb'])-x[1]
        f[9]=gg*x[5]+ff*x[9]-x[0]*(self.interp(t)-self.p['Ib'])
        f[10]=x[11]
        f[11]=gg*x[10]+ff*x[11]-self.p['p1']*x[1]/x[0]+self.p['p1']*self.p['p2']
        return f

    def optimize_f(self,theta):
        y=self.optimize_fdf(theta)
        return (y[:,0]-self.data['glucose'])*self.weights

    def optimize_df(self,theta):
        y=self.optimize_fdf(theta)
        return y[:,[2,3,4,5,10]]

    def optimize_fdf(self,theta):
        self.p['p1']=theta[1]
        self.p['p2']=theta[2]
        self.p['p3']=theta[3]
        self.p['Gb']=theta[4]
        y0=array([self.theta[0],self.p['p1']*(self.p['Gb']-theta[0])\
                  ,1,0,0,0,-self.p['p1'],self.p['Gb']-theta[0],0,0,0,self.p['p1']])
        self.y=odeint(self.minimal_model,y0,self.data['t_glucose'])
        return self.y
    def run_optimize(self):
        print "Optimizing patient: "+self.nc_file[-4]
        self.theta,self.cov_x,self.infodict, self.mesg,self.iter=leastsq(\
            self.optimize_f,self.theta,Dfun=self.optimize_df,full_output=1)




if(__name__=='__main__'):

    path=glob('../IVGTT_data/netcdfdata/patient*.nc')
    path.sort()
    cohort=[]
    for line in path:
        cohort.append(New_minmod(line))

    for patient in cohort:

        patient.run_optimize()
        clf()
        patient.plot_data()
        patient.ys=patient.simulate_model(patient.theta)
        patient.plot_simulated()
        patient.plot_legends()

        title('Patient: '+str(patient.nc_file[-4]))
        savefig('../figures/new_model/patient'+patient.nc_file[-4]+'.pdf')
        patient.write_data()
        patient.calc_var()
```

References

[1] S Wild, G Roglic, A Green, R Sicree, and H King. Global prevalence of diabetes - Estimates for the year 2000 and projections for 2030. *DIABETES CARE*, 27(5):1047–1053, MAY 2004.

[2] American Diabetes Association . Diagnosis and Classification of Diabetes Mellitus. *Diabetes Care*, 29(S1):S43–48, 2006.

[3] RN Bergman, YZ Ider, CR Bowden, and C Cobelli. Quantitative estimation of insulin sensitivity. *Am J Physiol Endocrinol Metab*, 236(6):E667–677, 1979.

[4] Bergman, R. N., and Bucolo, R. J. Nonlinear metabolic dynamics of the pancreas and liver. *Journal of Dynamic Systems Measurement and Control*, 95:296–300, 1973.

[5] Andrea Caumo, Paolo Vicini, Jeffrey J. Zachwieja, Angelo Avog aro, Kevin Yarasheski, Dennis M. Bier, and Claudio Cobelli. Undermodeling affects minimal model indexes: insights from a two-c ompartment model. *Am J Physiol Endocrinol Metab*, 276(6):E1171–1193, 1999.

[6] Gianluigi Pillonetto, Giovanni Sparacino, Paolo Magni, Riccardo Bellazzi, and Claudio Cobelli. Minimal model SI=0 problem in NIDDM subjects: nonzero Bayesian estimates with credible confidence intervals. *Am J Physiol Endocrinol Metab*, 282(3):E564–573, 2002.

[7] Tummers, B. DataThief III. http://datathief.org/, 2006.

[8] Jr. J. E. Dennis and Robert B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations (Classics in Applied Mathematics, 16)*. Soc for Industrial & Applied Math, 1996.

[9] Python Software Foundation. Homepage for Python. http://www.python.org/.

[10] Enthought Inc. Homepage for Scipy.org. http://www.scipy.org/.

[11] Thomas Lotz. *High resolution clinical model-based assessment of insulin sensitivity*. PhD in in Mechanical Engineering, University of Canterbury, 2007.

[12] G. Pacini and R. N. Bergman. MINMOD: a computer program to calculate insulin sensitivity and pancreatic responsivity from the frequently sampled intravenous glucose tolerance test. *Comput Methods Programs Biomed*, 23:113–22, 1986.