CHAPTER 2

Diabetes Glucose Tolerance Test

2.1 Introduction

This case study has a two-fold purpose:

1. Application of the ODE numerical methods of Chapter 1 to a model for a diabetes glucose tolerance test [3].
2. Consideration of features of the model and its numerical solutions that illustrate some basic properties of the ODE models.

2.2 Mathematical Model

This chapter is based on the pioneering and highly informative mathematical model of the glucose tolerance test for diabetes presented by Randall ([3], p 69) and authors referenced therein. Although this model consists of just two ordinary ODEs in time, it provides a clear and basic introduction to the physiological response from changes in glucose level, particularly through variations in the insulin level. The two ODEs for the glucose and insulin levels, $G(t)$ and $I(t)$, respectively, are next represented in words followed by mathematical symbols.

Differential Equation Analysis in Biomedical Science and Engineering: Ordinary Differential Equation Applications with R, First Edition. William E. Schiesser.

2.2.1 Glucose Balance

A mass balance on the glucose in the extracellular fluid can be stated as follows:

$$\begin{aligned}\begin{matrix}\text{rate of change}\\ \text{of glucose}\end{matrix} &= \begin{matrix}\text{liver}\\ \text{production}\end{matrix} + \begin{matrix}\text{glucose}\\ \text{infusion}\end{matrix} - \begin{matrix}\text{insulin}\\ \text{control}\end{matrix}\\ &\quad - \begin{matrix}\text{first order}\\ \text{metabolism}\end{matrix} - \begin{matrix}\text{renal}\\ \text{removal}\end{matrix}\end{aligned} \tag{2.1a}$$

If eq. (2.1a) is applied to 100 ml of the extracellular fluid, a two-part ODE follows.

$$C_g \frac{dG}{dt} = Q + I_n - G_g I G - D_d G; \ G < G_k \tag{2.1b}$$

$$C_g \frac{dG}{dt} = Q + I_n - G_g I G - D_d G - M_u(G - G_k); \ G \geq G_k \tag{2.1c}$$

We can then make a term-by-term comparison between eq. (2.1a) and eqs. (2.1b) and (2.1c).

Eq. (2.1a)	Eqs. (2.1b) and (2.1c)	Comments
rate of change of glucose	$C_g dG/dt$	$dG/dt < 0$, $G(t)$ decreasing with t $dG/dt > 0$, $G(t)$ increasing with t
liver production	Q	prescribed positive constant
glucose infusion	I_n	positive function of t
insulin control	$-G_g IG$	nonlinear from $I(t)G(t)$
first-order metabolism	$-D_d G$	decrease in $G(t)$ with $D_d > 0$
renal removal	$-M_u(G - G_k)$	negative function for $G \geq G_k$ zero function for $G < G_k$

(2.1d)

Numerical values and units for the model parameters for eqs. (2.1a), (2.1b), and (2.1c) are as follows:

G: extracellular glucose (mg glucose/100 ml extracellular fluid)

t: time (h)

C_g: glucose capacitance $= E_x/100$ (number of 100 ml extracellular volumes)

E_x: total extracellular space (ml)

Q: liver release of glucose (mg glucose/h)

I_n: glucose infusion, 0 or Q_t

Q_t: glucose infusion rate (mg glucose/h)

I: extracellular insulin (mg insulin/100 ml extracellular fluid)

G_g: controlled glucose loss

$$\left(\frac{\text{mg glucose}}{\text{h}}\right)\left(\frac{1}{\text{mg insulin/100 ml extracellular fluid}}\right)\left(\frac{1}{\text{mg glucose/100 ml extracellular fluid}}\right)$$

D_d: first order glucose loss

$$\left(\frac{\text{mg glucose}}{\text{h}}\right)\left(\frac{1}{\text{mg glucose/100 ml extracellular fluid}}\right)$$

G_k: renal threshold (mg glucose/100 extracellular fluid)

M_u: renal loss rate

$$\left(\frac{\text{mg glucose}}{\text{h}}\right)\left(\frac{1}{\text{mg glucose/100 ml extracellular fluid}}\right)$$

We can note some interesting and important features of eqs. (2.1b) and (2.1c).

- The dependent variable $G(t)$ is the glucose concentration in units of mg glucose/100 ml of the extracellular fluid; these units are designated subsequently as mGml.
- These ODEs are nonlinear; for example, the term $G_g IG$ is a product of G with a second dependent variable, I, the insulin concentration. Therefore, an analytical solution to the ODE system is probably precluded and a numerical solution is required.
- The ODEs also have a variable coefficient in the sense that the term $M_u(G - G_k)$ is switched on to account for renal loss (kidney removal) of glucose. In addition, the glucose infusion, I_n, is an explicit function of the independent variable, t.

- Each term in eqs. (2.1b) and (2.1c) has the unit mg glucose/hr:
  Q: mg glucose/hr
  I_n: mg glucose/hr
  $G_g IG$:

$$\left(\frac{\text{mg glucose}}{\text{hr}}\right)\left(\frac{1}{\text{mg insulin/100 ml extracellular fluid}}\right)$$
$$\left(\frac{1}{\text{mg glucose/100 ml extracellular fluid}}\right)$$
$$\left(\frac{\text{mg insulin}}{\text{100 ml extracellular fluid}}\right)\left(\frac{\text{mg glucose}}{\text{100 ml extracellular fluid}}\right)$$
$$= \text{mg glucose/hr}$$

$D_d G$:

$$\left(\frac{\text{mg glucose}}{\text{hr}}\right)\left(\frac{1}{\text{mg glucose/100 ml extracellular fluid}}\right)$$
$$\left(\frac{\text{mg glucose}}{\text{100 ml extracellular fluid}}\right)$$
$$= \text{mg glucose/hr}$$

$M_u(G - G_k)$:

$$\left(\frac{\text{mg glucose}}{\text{hr}}\right)\left(\frac{1}{\text{mg glucose/100 ml extracellular fluid}}\right)$$
$$\left(\frac{\text{mg glucose}}{\text{100 ml extracellular fluid}}\right)$$
$$= \text{mg glucose/hr}$$

$C_g \dfrac{dG}{dt}$:

$$(\text{number of 100 ml extracellular volumes})$$
$$\left(\frac{\text{mg glucose}}{\text{100 ml extracellular fluid}}\frac{1}{\text{hr}}\right)$$
$$= \text{mg glucose/hr}$$

The sign of the derivative $\frac{dG}{dt}$ is an important consideration because it determines if the glucose $G(t)$ is increasing with t $\left(\frac{dG}{dt} > 0\right)$, which might indicate that diabetes is a significant problem, that is, a condition of *hyperglycemia*; or decreasing with t $\left(\frac{dG}{dt} < 0\right)$, which might indicate that diabetes is not a problem or under control; or possibly a condition of excessively low glucose, that is, a condition of *hypoglycemia*.

- As all of the constants and variables are in hours, the timescale for the solution of eqs. (2.1b) and (2.1c) will also be in hours; this follows particularly from the derivative $C_g \frac{dG}{dt}$ which has the time units of h^{-1}. For the subsequent calculations, this timescale is taken as $0 \leq t \leq 12$ h, that is, $G(t), I(t)$ will be calculated for 12 hr.

In summary, this check for the consistency of units throughout eqs. (2.1b) and (2.1c) is essential to ensure that the solution $G(t)$ is correct. Also, these units provide additional insight into the physical/chemical significance of each term.

For example, the coefficient G_g in the nonlinear term $G_g IG$ is particularly noteworthy because it is a direct indicator of the effect of insulin (I) on glucose level. If G_g is lower than normal, this could be interpreted as a condition for which insulin is less effective in determining glucose level than normal, that is, a resistance to insulin or Type II diabetes.

We now go through a similar analysis for the insulin mass balance, which leads to a second ODE for $I(t)$ that is used in eqs. (2.1b) and (2.1c); that is, the two ODE are *simultaneous* or *coupled*.

2.2.2 Insulin Balance

The insulin mass balance can be stated in words as

$$\begin{matrix}\text{rate of change} \\ \text{of insulin}\end{matrix} = - \begin{matrix}\text{first-order insulin} \\ \text{reduction}\end{matrix} + \begin{matrix}\text{pancreas insulin} \\ \text{release rate}\end{matrix} \qquad (2.2a)$$

If eq. (2.2a) is applied to 100 of the extracellular fluid, a two-part ODE results.

$$C_i\frac{dI}{dt} = -A_a I,\ G < G_0 \tag{2.2b}$$

$$C_i\frac{dI}{dt} = -A_a I + B_b(G - G_0),\ G \geq G_0 \tag{2.2c}$$

We can then make a term-by-term comparison between eq. (1.2a) and eqs. (1.2b) and (1.2c).

Eq. (2.2a)	Eqs. (2.2b), (2.2c)	Comments
rate of change of insulin	$C_{id}I/dt$	$dI/dt < 0$, $I(t)$ decreasing with t $dI/dt > 0$, $I(t)$ increasing with t
first-order insulin reduction	$-A_a I$	decrease in $I(t)$ for $A_a > 0$
pancreas insulin release rate	$B_b(G - G_0)$	positive function for $G \geq G_0$ zero function for $G < 0$

(2.2d)

Numerical values and units for the model parameters for eqs. (2.2a), (2.2b), and (2.2c) are as follows:

I: extracellular insulin (mg insulin/100 ml extracellular fluid)

C_i: insulin capacitance $= E_x/100$ (number of 100 ml extracellular volumes)

A_a: first-order insulin reduction rate

$$\left(\frac{\text{mg insulin}}{\text{h}}\right)\left(\frac{1}{\text{mg insulin/100 ml extracellular fluid}}\right)$$

G_0: pancreas threshold (mg glucose/100 extracellular fluid)

B_b: pancreas insulin release rate

$$\left(\frac{\text{mg insulin}}{\text{h}}\right)\left(\frac{1}{\text{mg glucose/100 ml extracellular fluid}}\right)$$

We can note some interesting and important features of eqs. (2.2b) and (2.2c).

- The dependent variable $I(t)$ is the insulin concentration in units of mg insulin/100 ml of the extracellular fluid; these units are designated subsequently as mIml.
- The ODEs also have a variable coefficient in the sense that the term $B_b(G - G_0)$ is switched on to account for insulin production by the pancreas.
- Each term in eqs. (2.1b) and (2.1c) has the unit mg insulin/hr: $A_a I$:

$$\left(\frac{\text{mg insulin}}{\text{hr}}\right)\left(\frac{1}{\text{mg insulin/100 ml extracellular fluid}}\right)$$
$$\left(\frac{\text{mg insulin}}{\text{100 ml extracellular fluid}}\right)$$
$$= \text{mg insulin/hr}$$

$B_b(G - G_0)$:

$$\left(\frac{\text{mg insulin}}{\text{hr}}\right)\left(\frac{1}{\text{mg glucose/100 ml extracellular fluid}}\right)$$
$$\left(\frac{\text{mg glucose}}{\text{100 ml extracellular fluid}}\right)$$
$$= \text{mg insulin/hr}$$

$C_i \dfrac{dI}{dt}$:

$$\text{(number of 100 ml extracellular volumes)}$$
$$\left(\frac{\text{mg insulin}}{\text{100 ml extracellular fluid}}\frac{1}{\text{hr}}\right)$$
$$= \text{mg insulin/hr}$$

The sign of the derivative $\dfrac{dI}{dt}$ is an important consideration because it determines if the insulin $I(t)$ is increasing with t $\left(\dfrac{dI}{dt} > 0\right)$, which might indicate that diabetes is not a significant problem, or decreasing with t $\left(\dfrac{dI}{dt} < 0\right)$, which might indicate that diabetes is a problem.

- Since all of the constants and variables are in hours, the timescale for the solution of eqs. (2.2b) and (2.2c) will also be in hours as required to be consistent with the simultaneous solution of eqs. (2.1b) and (2.1c).

In summary, this check for the consistency of units throughout eqs. (2.2b) and (2.2c) is essential to ensure that the solution $I(t)$ is correct. Also, these units provide additional insight into the physical/chemical significance of each term.

For example, the coefficient B_b in the term $B_b(G - G_0)$ is noteworthy because it is a direct indicator of pancreatic insulin production. If B_b is lower than normal, this could be interpreted as a condition of below normal insulin production, that is, Type I diabetes. Variations in B_b are studied in the subsequent computer analysis.

Eqs. (2.1) and (2.2) constitute the mathematical model for the glucose tolerance test. We now consider a computer solution of these equations.

2.3 Computer Analysis of the Mathematical Model

Eqs. (2.1) and (2.2) constitute a 2×2 ODE system (two equations in two unknowns). These equations are *not stiff*, and therefore, an *explicit integrator* can be used for their solution.[1] In other words, the derivatives $\frac{dG}{dt}, \frac{dI}{dt}$ of eqs. (2.1) and (2.2) produce solutions $G(t), I(t)$ on approximately the same timescale (one dependent variable does not move at a much higher rate than the other). This is a somewhat loose description of a nonstiff ODE problem.

2.3.1 ODE Integration by `lsoda`

We start the discussion of the numerical solution of eqs. (2.1) and (2.2) with the library integrator `lsoda` in the R utility `ode`.[2]

[1]The concepts of stiffness and explicit integration are discussed in detail in [2], Appendix C, and [4].

[2]Numerical methods for initial-value ODEs are discussed in Chapter 1.

A main program with a call to ode is listed next. A discussion of this main program then follows the listing.

```
#
# Glucose Tolerance Test
#
# The glucose tolerance test is used clinically to
# evaluate the ability of the pancreas to release insulin
# in response to a large dose of glucose given either
# orally or intravenously.  The normal pancreas releases
# enough insulin to lower the plasma glucose within a few
# hours, sometimes to the point of hyperglycemia.  The
# deficiency of insulin release, characteristic of Type 1
# diabetes, also termed juvenile-onset diabetes, prolongs
# the fall of glucose for many hours (1).
#
# (1)  Randall, James E., Microcomputers and Physiological
# Simulation, Addison-Wesley Publishing Company, Inc.,
# Reading, MA, 1980, p69.
#
# The following differential equations can be used to
# compute the glucose and insulin levels as a function
# of time:
#
#    Cg*dG/dt = Q + In - (Gg*I*G) - Dd*G, G < Gk      (1)
#
#    Cg*dG/dt = Q + In - (Gg*I*G) - Dd*G - Mu*(G - Gk),
#       G >= Gk (2)
#
#    Ci*dI/dt = -Aa*I, G < G0                         (3)
#
#    Ci*dI/dt = -Aa*I + Bb*(G - G0), G >= G0          (4)
#
# where
#
#    Symbol        Parameters/Variables     Normal Values
#
#      Ex     extracellular space             15000 ml
#
#      Cg     glucose capacitance = Ex/100     150 ml
#
```

```
#      Ci     insulin capacitance = Ex/100     150 ml
#
#      Q      liver release of glucose        8400 mG/hr
#
#      Gt     glucose infusion rate           80000 mG/hr
#
#      In     glucose infusion, Gt or 0
#
#      Dd     first-order glucose loss    24.7 mG/hr/mGml
#
#      Gg     controlled glucose loss 13.9 mG/hr/mGml/mIml
#
#      Gk     renal threshold                 250 mGml
#
#      Mu     renal loss rate              72 mG/hr/mGml
#
#      G0     pancreas threshold              51 mGml
#
#      Bb     insulin release rate        14.3 mI/hr/mGml
#
#      Aa     first-order insulin rate     76 mI/hr/mGml
#
#      G      extracellular glucose       81 mGml (t = 0)
#
#      I      extracellular insulin      5.7 mIml (t = 0)
#
#      t      time                              hr
#
# The mass and concentration units are:
#
#      mG     milligrams of glucose
#
#      mI     milligrams of insulin
#
#      mGml   milligrams of glucose/100 ml extracellular
#                fluid
#
#      mIml   milligrams of insulin/100 ml extracellular
```

```
#                 fluid
#
#      ml     milliliter = cubic centimeter = 0.001 liter
#
# The glucose infusion function, In, is given by:
#
#             In = Gt, 0 <= t < 0.5
#
#             In = 0, 0.5 <= t <= 12
#
# Equations (1) to (4) are integrated for four cases:
#
#    (1)  (ncase = 1)
#
#         normal pancreatic sensitivity (Bb = 14.3)
#         (without glucose infusion, Gt = 0)
#
#    (2)  (ncase = 2)
#
#         normal pancreatic sensitivity (Bb = 14.3)
#         (with glucose infusion)
#
#    (3)  (ncase = 3)
#
#         reduced pancreatic sensitivity (Bb = 0.2(14.3))
#         (with glucose infusion)
#
#    (4)  (ncase = 4)
#
#         elevated pancreatic sensitivity (Bb = 2.0(14.3))
#         (with glucose infusion)
#
# Library of R ODE solvers
  library("deSolve")
#
# ODE routine
  setwd("c:/R/bme_ode/chap2")
  source("glucose_1.R")
```

```
#
# Vectors, matrices for the graphical output
  nout=49
  Gplot=matrix(0,nrow=nout,ncol=4)
  Iplot=matrix(0,nrow=nout,ncol=4)
  tplot=rep(0,nout)
#
# Step through four cases
  for(ncase in 1:4){
#
# Select the case parameters
  if(ncase==1){Bb=14.3;         Gt=0}
  if(ncase==2){Bb=14.3;     Gt=80000}
  if(ncase==3){Bb=0.2*14.3; Gt=80000}
  if(ncase==4){Bb=2.0*14.3; Gt=80000}
#
# Model parameters
  Ex=15000; Cg=150; Ci=150; Q=8400; Dd=24.7;
   Gg=13.9; Gk=250;  Mu=72;  G0=51;   Aa=76;
#
# Initialize counter for calls to glucose_1
  ncall=0
#
# Initial condition
  yini=c(81.14,5.671)
  yini
#
# t interval
  times=seq(from=0,to=12,by=12/(nout-1))
#
# ODE integration
  out=ode(y=yini,times=times,func=glucose_1,parms=NULL)
#
# ODE numerical solution
  for(it in 1:nout){
    if(it==1){
    cat(sprintf(
    "\n ncase = %2d \n\n        t      In       G
```

```
      I",ncase))}
#
#   Glucose infusion function
    t=times[it]
    if((t>=0)&(t<=0.51)){In=Gt}
    if( t>0.51)         {In=0 }
    cat(sprintf("\n %8.2f%8.0f%8.2f%8.3f",
                out[it,1],In,out[it,2],out[it,3]))
  }
#
# Store solution for plotting
  Gplot[,ncase]=out[,2]
  Iplot[,ncase]=out[,3]
  if(ncase==1)tplot=out[,1]
#
# Calls to glucose_1
  cat(sprintf("\n\n ncall = %5d\n\n",ncall))
#
# Next case
}
#
# Single plot for G
  par(mfrow=c(1,1))
#
# G, ncase = 1
  plot(tplot,Gplot[,1],xlab="t (hr)",
  ylab="G(t) (mg glucose/100 ml) vs t",
  xlim=c(0,12),ylim=c(0,300),type="b",lty=1,pch="1",lwd=2,
  main="Extracellular glucose, G(t), ncase = 1,2,3,4")
#
# G, ncase = 2
  lines(tplot,Gplot[,2],type="b",lty=1,pch="2",lwd=2)
#
# G, ncase = 3
  lines(tplot,Gplot[,3],type="b",lty=1,pch="3",lwd=2)
#
# G, ncase = 4
  lines(tplot,Gplot[,4],type="b",lty=1,pch="4",lwd=2)
```

```
#
# Single plot for I
  par(mfrow=c(1,1))
#
# I, ncase = 1
  plot(tplot,Iplot[,1],xlab="t (hr)",
  ylab="I(t) (mg insulin/100 ml) vs t",
  xlim=c(0,12),ylim=c(0,25),type="b",lty=1,pch="1",
     lwd=2,
  main="Extracellular insulin, I(t), ncase = 1,2,3,4")
#
# I, ncase = 2
  lines(tplot,Iplot[,2],type="b",lty=1,pch="2",lwd=2)
#
# I, ncase = 3
  lines(tplot,Iplot[,3],type="b",lty=1,pch="3",lwd=2)
#
# I, ncase = 4
  lines(tplot,Iplot[,4],type="b",lty=1,pch="4",lwd=2)
```

Listing 2.1 Main program for the numerical integration of eqs. (2.1) and (2.2).

We can note the following points about Listing 2.1.

- A block of comments documents the model of eqs. (2.1) and (2.2) (they are not repeated here to conserve space). In particular, four cases are explained, for `ncase = 1` to `ncase = 4`, in which the pancreas sensitivity parameter B_b is varied. The details of these four cases are discussed in the comments and subsequently.
- The R library of the ODE integrators, `deSolve`, is accessed for the solution of eqs. (2.1) and (2.2).

```
#
# Library of R ODE solvers
  library("deSolve")
```

- The ODE routine `glucose_1` with the programming of eqs. (2.1) and (2.2) is accessed through `setwd` (set working directory) and `source` (to specify the file name).

```
#
# ODE routine
  setwd("c:/R/bme_ode/chap2")
  source("glucose_1.R")
```

- Two 2D matrices for plotting $G(t)$ from eq. (2.1) and $I(t)$ from eq. (2.2) are declared (preallocated) with the `matrix` utility for `nout=49` values of t and `ncol=4` cases (described previously in the comments).

```
#
# Vectors, matrices for the graphical output
  nout=49
  Gplot=matrix(0,nrow=nout,ncol=4)
  Iplot=matrix(0,nrow=nout,ncol=4)
  tplot=rep(0,nout)
```

The vector for `nout=49` values of t is also declared with the `rep` utility. Forty-nine output values were selected to give (i) three points for the glucose infusion function ($t = 0, 0.25, 0.5$) in the output and (ii) enough points for the plots without crowding.

- Four cases are programmed with a `for`. For each case, the model parameters `Bb, Gt` are defined numerically.

```
#
# Step through four cases
  for(ncase in 1:4){
#
# Select the case parameters
  if(ncase==1){Bb=14.3;         Gt=0}
  if(ncase==2){Bb=14.3;     Gt=80000}
  if(ncase==3){Bb=0.2*14.3; Gt=80000}
  if(ncase==4){Bb=2.0*14.3; Gt=80000}
```

Note in particular that for `ncase = 1`, `Gt=0` so that no glucose infusion takes place. This corresponds to the normal condition without a glucose tolerance test. `ncase = 2,3,4` then corresponds to the response to a glucose infusion of `Gt=80000` (mg glucose) for three different values of the pancreas sensitivity parameter B_b.

- The remaining parameters in eqs. (2.1) and (2.2) are defined numerically.

```
#
# Model parameters
  Ex=15000; Cg=150; Ci=150; Q=8400; Dd=24.7;
   Gg=13.9; Gk=250;  Mu=72;  G0=51;   Aa=76;
```

These parameter values are available to the subordinate ODE routine `glucose_1` (discussed subsequently).

- The counter for the number of calls to `glucose_1` is initialized.

```
#
# Initialize counter for calls to glucose_1
  ncall=0
```

- The ICs for eqs. (2.1) and (2.2) are specified to start the solution.

```
#
# Initial condition
  yini=c(81.14,5.671)
  yini
```

Note the use of the R utility `c` to place the two ICs in a vector, `yini`, that is, `G(t=0)=81.14, I(t=0)=5.671`. The use of the name `yini` on a separate line displays the two numerical values for confirmation.

- The interval in t is defined as $0 \leq t \leq 12$ with the 49 values $t = 0, 12/(49-1) = 0.25, 0.50, \ldots, 12$ placed in the vector `times` (using the R utility `seq`).

```
#
# t interval
  times=seq(from=0,to=12,by=12/(nout-1))
```

- The ODE solution is computed by a call to `ode` that is available in `deSolve`.

```
#
# ODE integration
  out=ode(y=yini,times=times,func=glucose_1,
     parms=NULL)
```

Note the use of the IC vector `yini` and the output values of t in `times` as the input arguments `y` and `times` to `ode`. Also, the ODE routine `glucose_1` with the programming of eqs. (2.1) and (2.2) is an input to `ode` through the argument `func`. In other words, `y,times,func` are reserved names. The numerical solution is returned from `ode` as a 2D array, `out`. The argument `parms` is unused.

- The numerical ODE solution in `out` is displayed for the `nout=49` values of t with a `for` in `it`. For `it=1` corresponding to $t = 0$, a heading for the solution is displayed.

```
#
# ODE numerical solution
  for(it in 1:nout){
    if(it==1){
    cat(sprintf(
    "\n ncase = %2d \n\n        t      In       G
       I",ncase))}
#
#   Glucose infusion function
    t=times[it]
    if((t>=0)&(t<=0.51)){In=Gt}
    if( t>0.51)        {In=0 }
    cat(sprintf("\n %8.2f%8.0f%8.2f%8.3f",
                out[it,1],In,out[it,2],out[it,3]))
  }
```

The infusion function `In` is computed to be included in the numerical output. `if((t>=0)&(t<=0.51))In=Gt` is used in place of `if((t>=0)&(t<=0.5))In=Gt` to avoid the test of equality `t=0.5`, which is unreliable in floating point arithmetic.

`out[it,1]` has the 49 values of t. `out[it,2]`, `out[it,3]` have the 49 values of $G(t)$ and $I(t)$, respectively.

- The solution is placed in two 2D arrays for plotting. Note the use of `,` to include all (49) values of the first subscript. The second subscript is for each of the four solutions (set by the previous `for(ncase in 1:4)`).

```
#
# Store solution for plotting
  Gplot[,ncase]=out[,2]
  Iplot[,ncase]=out[,3]
  if(ncase==1)tplot=out[,1]
```

- The number of calls to `glucose_1` is displayed.

```
#
# Calls to glucose_1
  cat(sprintf("\n ncall = %5d\n\n",ncall))
#
# Next case
}
```

The concluding `}` terminates the `for` in `ncase`.

- A composite plot with the four solutions for $G(t)$ is produced. For the first solution (`ncase=1` with `Gplot[,1]` vs `tplot`), the plotting is similar to that in Listing 1.2 and therefore the details (input arguments to `plot`) are not repeated here.

```
#
# Single plot for G
  par(mfrow=c(1,1))
#
# G, ncase = 1
  plot(tplot,Gplot[,1],xlab="t (hr)",
  ylab="G(t) (mg glucose/100 ml) vs t",
  xlim=c(0,12),ylim=c(0,300),type="b",lty=1,pch="1",
     lwd=2,
  main="Extracellular glucose, G(t), ncase = 1,2,3,4")
```

`pch="1"` specifies the character `1` for the plot points in the first solution.

- The solutions for `ncase=2,3,4` follow from the `lines` utility. The plot characters are `2,3,4`.
- A similar set of statements provides the composite plot for $I(t)$.

The ODE routine, `glucose_1`, called by `ode` is in Listing 2.2.

```
  glucose_1=function(t,y,parms) {
#
# Assign state variables
  G=y[1];
  I=y[2];
#
# Glucose infusion function
  if((t>=0)&(t<=0.51)){In=Gt}
  if( t>0.51)        {In=0 }
#
# ODEs
#
# Glucose equations
  if(G< Gk){dGdt=(1/Cg)*(Q+In-(Gg*I*G)-Dd*G)}
  if(G>=Gk){dGdt=(1/Cg)*(Q+In-(Gg*I*G)-Dd*G-Mu*(G-Gk))}
#
# Insulin equations
  if(G< G0){dIdt=(1/Ci)*(-Aa*I)}
  if(G>=G0){dIdt=(1/Ci)*(-Aa*I+Bb*(G-G0))}
#
# Calls to glucose_1
  ncall <<- ncall+1
#
# Return derivative vector
  return(list(c(dGdt,dIdt)))
}
```

Listing 2.2 ODE routine `glucose_1` for the eqs. (2.1) and (2.2).

We can note the following details about `glucose_1`.

- The function is defined.

```
glucose_1=function(t,y,parms) {
```

- The dependent variable vector `y` is expressed as two problem-oriented variables `G,I` to facilitate the programming of eqs. (2.1) and (2.2).

```
#
# Assign state variables
  G=y[1];
  I=y[2];
```

  Note that `y` is an input vector (RHS argument) to `glucose_1`. It has two elements as specified by the number of ICs in the main program of Listing 2.1.
- The glucose infusion function `In` is defined at a particular value of the independent variable t (an input or RHS arguments to `glucose_1`).

```
#
# Glucose infusion function
  if((t>=0)&(t<=0.51)){In=Gt}
  if( t>0.51)         {In=0 }
```

  `In` equals `Gt` for $0 \leq t \leq 0.5$ h and zero thereafter.
- Eqs. (2.1b) and (2.1c) are programmed in a straightforward manner, including the switch based on `Gk` to add the term `-Mu*(G-Gk)`.

```
#
# Glucose equations
  if(G< Gk){dGdt=(1/Cg)*(Q+In-(Gg*I*G)-Dd*G)}
  if(G>=Gk){dGdt=(1/Cg)*(Q+In-(Gg*I*G)-Dd*G-Mu*
     (G-Gk))}
```

  Note that in order to calculate the derivative $dG/dt =$ `dGdt`, all of the RHS variables and parameters must be defined numerically. For $t = 0$, `G,I` are available from the ICs set previously in the main program of Listing 2.1. For $t > 0$, `G,I` are available through the input argument `y`. All of the parameters were defined numerically in Listing 2.1.

This coding illustrates the ease of numerically including the time-dependent switch for the two forms of eqs. (2.1b) and (2.1c). Also, the nonlinear term `-(Gg*I*G)` is easily included. These features would be difficult to accommodate analytically.

- Eqs. (2.2b) and (2.2c) are programmed in a similar way, including the switch based on `G0` to include the term $B_b(G - G_0)$.

```
#
# Insulin equations
  if(G< G0){dIdt=(1/Ci)*(-Aa*I)}
  if(G>=G0){dIdt=(1/Ci)*(-Aa*I+Bb*(G-G0))}
```

This is the point at which the variation in `Bb` (changes in the pancreas sensitivity) for the four cases `ncase = 1,2,3,4` enters the numerical solution.

- The number of calls to `glucose_1` is incremented

```
#
# Calls to glucose_1
  ncall <<- ncall+1
```

with the return of the value of `ncall` to the main program of Listing 2.1 using `<<-`.

- Finally, the two derivatives $dG/dt, dI/dt$ are returned as a list (as required by the ODE integrators in `deSolve`).

```
#
# Return derivative vector
  return(list(c(dGdt,dIdt)))
}
```

The final `}` concludes `glucose_1`.

The numerical and graphical outputs from Listings 2.1 and 2.2 follows. Abbreviated numerical output for `ncase = 1,2,3,4` is in Table 2.1.

TABLE 2.1 Abbreviated output from the routines of Listings 2.1 and 2.2.

```
ncase =  1

       t      In       G       I
    0.00       0   81.14   5.671
    0.25       0   81.14   5.671
    0.50       0   81.14   5.671
    0.75       0   81.14   5.671
    1.00       0   81.14   5.671
     .                      .
     .                      .
     .                      .
 Output for t = 1.25 to 10.75 removed
     .                      .
     .                      .
     .                      .
   11.00       0   81.14   5.671
   11.25       0   81.14   5.671
   11.50       0   81.14   5.671
   11.75       0   81.14   5.671
   12.00       0   81.14   5.671

ncall =    95

ncase =  2

       t      In       G       I
    0.00   80000   81.14   5.671
    0.25   80000  201.71   7.100
    0.50   80000  286.72  10.687
    0.75       0  217.19  13.881
    1.00       0  159.88  15.265
     .                      .
     .                      .
     .                      .
```

TABLE 2.1 (*Continued*)

```
 Output for t = 1.25 to 10.75 removed
     .                      .
     .                      .
     .                      .
   11.00       0   80.90   5.681
   11.25       0   80.92   5.674
   11.50       0   80.96   5.670
   11.75       0   80.99   5.666
   12.00       0   81.03   5.664

ncall =   358

ncase =  3

       t      In       G       I
    0.00   80000   81.14   5.671
    0.25   80000  204.22   5.420
    0.50   80000  305.96   5.704
    0.75       0  265.14   6.070
    1.00       0  233.03   6.230
     .                      .
     .                      .
     .                      .
 Output for t = 1.25 to 10.75 removed
     .                      .
     .                      .
     .                      .
   11.00       0  129.31   2.894
   11.25       0  129.31   2.900
   11.50       0  129.29   2.906
   11.75       0  129.26   2.911
   12.00       0  129.22   2.915

ncall =   297
```

(*continued*)

TABLE 2.1 (*Continued*)

```
ncase =  4

      t       In        G        I
   0.00   80000    81.14    5.671
   0.25   80000   198.66    9.167
   0.50   80000   265.50   16.454
   0.75       0   172.61   21.905
   1.00       0   108.34   23.153
     .                        .
     .                        .
     .                        .
 Output for t = 1.25 to 10.75 removed
     .                        .
     .                        .
     .                        .
  11.00       0    69.47    6.905
  11.25       0    69.49    6.911
  11.50       0    69.50    6.917
  11.75       0    69.50    6.922
  12.00       0    69.49    6.926

ncall =   422
```

We can note the following details for this output.

- For `ncase = 1`, the solution does not change from the ICs. This implies that the derivatives $dG/dt \approx 0, dI/dt \approx 0$ can be confirmed by numerically evaluating the RHS of eqs. (2.1c) and (2.2c) with $I_n = 0$ (no glucose infusion). First, for eq. (2.1c),

$$C_g \frac{dG}{dt} = Q + I_n - G_g IG - D_d G - M_u(G - G_k);\ G \geq G_k$$

$$\frac{dG}{dt} = \frac{1}{C_g}(Q + I_n - G_g IG - D_d G - M_u(G - G_k));\ G \geq G_k$$

$$\frac{dG}{dt} = \frac{1}{150}(8400 + 0 - (13.9)(5.671)(81.14) - (24.7)(81.14))$$

$$= -0.00115$$

Here, we have used the IC $G = 81.18$, $I = 5.671$, and because $G = 81.14 < G_k = 250$, the term with M_u is dropped. This small value of the derivative at $t = 0$ is actually the largest value during the solution $0 \leq t \leq 12$. So eq. (2.1c) essentially remains at the IC $G = 81.14$ as reflected in the constant (time invariant) solution for `ncase = 1`.

- The number of calls to `ode_1`, `ncall = 95` is relatively small because the solution does not change.
- A similar analysis of eq. (2.2c) again indicates a small value for $\dfrac{dI}{dt}$.

$$
\begin{aligned}
C_i \frac{dI}{dt} &= -A_a I + B_b(G - G_0),\ G \geq G_0 \\
\frac{dI}{dt} &= \frac{1}{C_i}(-A_a I + B_b(G - G_0)),\ G \geq G_0 \\
\frac{dI}{dt} &= \frac{1}{150}(-(76)(5.671) + (14.3)(81.14 - 51)) \\
&= 0.0000,\ G \geq G_0
\end{aligned}
$$

so that eqs. (2.2) also remains at the IC $I = 5.671$. Note that this requires the pancreatic insulin production, $B_b(G - G_0)$, is not zero.

- The infusion of glucose will, therefore, cause the ODE system to depart from the steady state. That is, $I_n(t) \neq 0$ in eqs. (2.1b) and (2.1c) will drive the ODE system away from equilibrium. This is reflected in the solutions for `ncase = 2,3,4`. For `ncase = 2`, the abbreviated solution is

```
ncase =  2

       t       In         G         I
    0.00    80000     81.14     5.671
    0.25    80000    201.71     7.100
    0.50    80000    286.72    10.687
    0.75        0    217.19    13.881
    1.00        0    159.88    15.265
```

```
      .                      .
      .                      .
      .                      .
 Output for t = 1.25 to 10.75 removed
      .                      .
      .                      .
      .                      .
   11.00       0   80.90   5.681
   11.25       0   80.92   5.674
   11.50       0   80.96   5.670
   11.75       0   80.99   5.666
   12.00       0   81.03   5.664

ncall =   358
```

We can note the following details about this numerical output.

— The glucose infusion, I_n, is 80,000 for $0 \leq t \leq 0.5$ and zero thereafter, which follows from the programming in Listing 2.1

```
if(ncase==2){Bb=14.3; Gt=80000}
```

— The solution undergoes a transient (departure from the ICs) and then approaches the same steady state as for `ncase = 1` as reflected in the output at $t = 12$: `12.0 0 81.03 5.664`. In other words, the model returns to a normal condition as a response to the glucose infusion $I_n(t)$.

— The number of calls to `glucose_1` is `358` which is larger than for `ncase=1` because smaller steps in t are required because of the changing solution.

- For `ncase = 3`, the abbreviated solution is

```
ncase =  3

       t       In        G        I
    0.00    80000    81.14    5.671
    0.25    80000   204.22    5.420
    0.50    80000   305.96    5.704
    0.75        0   265.14    6.070
    1.00        0   233.03    6.230
```

```
    .                      .
    .                      .
    .                      .
Output for t = 1.25 to 10.75 removed
    .                      .
    .                      .
    .                      .
  11.00       0  129.31   2.894
  11.25       0  129.31   2.900
  11.50       0  129.29   2.906
  11.75       0  129.26   2.911
  12.00       0  129.22   2.915

ncall =   297
```

We can note the following details about this numerical output.

— For `ncase = 3`, the insulin release rate B_b is decreased by a factor of 0.2; that is, from the main program in Listing 2.1 (Figs. 2.1 and 2.2)

```
if(ncase==3){Bb=0.2*14.3; Gt=80000}
```

— The solution undergoes a transient (departure from the ICs) and then approaches a new steady state (different than for `ncase = 1,2`) as reflected in the output at $t = 12$: `12.0 0 129.22 2.915`. As expected (when B_b is decreased), the final glucose level $G(t)$ is higher than for `ncase = 1,2` and the insulin level, $I(t)$, is lower.

— The new final equilibrium values correspond to $dG/dt \approx 0, dI/dt \approx 0$. This is confirmed by a numerical evaluation of the RHS of eqs. (2.1c) and (2.2c).

$$\frac{dG}{dt} = \frac{1}{150}(8400 + 0 - (13.9)(2.915)(129.22) \\ - (24.7)(129.22)) = -0.1836$$

This derivative at $t = 12$ is not zero but is small (and would become smaller with t beyond 12). For a comparison, the derivative $\frac{dG}{dt}$ at $t = 0$ is

$$\frac{dG}{dt} = \frac{1}{150}(8400 + 80000 - (13.9)(5.663)(81.027) \\ - (24.7)(81.027)) = 533.47$$

This large initial derivative is not unusual (frequently ODEs exhibit their largest derivatives initially), and therefore, the solution changes most rapidly at the IC.

— The insulin level $I(t)$ reaches the final value 2.915 since the derivatives is effectively zero,

$$\frac{dI}{dt} = \frac{1}{150}(-(76)(2.915) + (0.2)(14.3)(129.22 - 51)) \\ = 0.0144, \; G \geq G_0$$

Note the reduced value of B_b used in this calculation, (0.2)(14.3).

— The number of calls to `glucose_1` is `297`, which is different from that for `ncase=2`, reflecting the variable step method in `lsoda` of `ode`.

In conclusion, for `ncase = 3`, the reduced pancreatic sensitivity causes the glucose level $G(t)$ to reach a value higher than the normal value (because of the decreased insulin release rate B_b)—a condition of *hyperglycemia*. Similarly, the insulin level $I(t)$ reaches a value lower than the normal value.

- For `ncase = 4`, the abbreviated solution is

```
 ncase =  4

        t      In        G        I
     0.00   80000    81.14    5.671
     0.25   80000   198.66    9.167
     0.50   80000   265.50   16.454
     0.75       0   172.61   21.905
     1.00       0   108.34   23.153
       .                        .
```

```
      .                    .
      .                    .
  Output for t = 1.25 to 10.75 removed
      .                    .
      .                    .
      .                    .
    11.00       0   69.47   6.905
    11.25       0   69.49   6.911
    11.50       0   69.50   6.917
    11.75       0   69.50   6.922
    12.00       0   69.49   6.926

 ncall =   422
```

We can note the following details about this numerical output.

— For `ncase = 4`, the insulin release rate B_b is increased by a factor of 2; that is, from the main program in Listing 1.1

```
if(ncase==4){Bb=2.0*14.3; Gt=80000}
```

— The solution undergoes a transient (departure from the ICs) and then then approaches a new steady state (different than for `ncase = 1,2,3`) as reflected in the output at $t = 12$: `12.0 0 69.49 6.926`. As expected (when B_b is increased), the glucose level $G(t)$ is lower than that for `ncase = 1,2,3` and the insulin level, $I(t)$, is higher.

— The new final equilibrium values correspond to $dG/dt \approx 0, dI/dt \approx 0$. This is confirmed by a numerical evaluation of the RHS of eqs. (2.1c) and (2.2c).

$$\begin{aligned}\frac{dG}{dt} &= \frac{1}{150}(8400 + 0 - (13.9)(6.926)(69.49) - (24.7)(69.49)) \\ &= -0.0420\end{aligned}$$

The derivative at $t = 0$ is the same as for `ncase = 2,3` because the ICs are the same, that is,

$$\begin{aligned}\frac{dG}{dt} =& \frac{1}{150}(8400 + 80000 - (13.9)(5.663)(81.027) \\ &- (24.7)(81.027)) = 533.47\end{aligned}$$

— The insulin level $I(t)$ reaches the final value 6.926 because the derivative is effectively zero,

$$\begin{aligned}\frac{dI}{dt} &= \frac{1}{150}(-(76)(6.926) + (2)(14.3)(69.49 - 51)) \\ &= 0.0167, \ G \geq G_0\end{aligned}$$

Note the increased value of B_b used in this calculation, (2)(14.3).

— The number of calls to `glucose_1` is `422`, which is different than to `ncase=1,2,3` as might be expected considering the variable step algorithm in `lsoda`.

In conclusion, for `ncase = 4`, the increased pancreatic sensitivity causes the glucose level $G(t)$ to reach a value lower than the normal value (because of the increased insulin release rate B_b)—a condition of *hypoglycemia*. Similarly, the insulin level $I(t)$ reaches a value higher than the normal value.

The solutions for `ncase = 1,2,3,4` can be visualized through the following composite plots produced by the main program of Listing 2.1. We observe in these plots that the solutions have the properties discussed previously: (i) for `ncase = 1`, $G(t), I(t)$ are unchanged with t and dG/dt, dI/dt remain at zero, and (ii) for `ncase = 2,3,4`, the solutions approach different final values.

These plots facilitate the overall visualization of the solutions for `ncase = 1,2,3,4` and elucidate the effect of the pancreatic sensitivity, B_b. Also, all of the RHS terms in eqs. (2.1b), (2.1c), (2.2b), and (2.2c) could be computed and plotted individually to gain additional insight into the features of the solutions in Figs. 2.1 and 2.2. For example, the relative contributions of the individual RHS terms to the LHS derivatives could be investigated, and the effect of changes in the model parameters would indicate the sensitivity of the solutions to the parameter values.

We now consider an alternative numerical ODE integration using the fixed step RKF45 algorithm discussed in Chapter 1. The intent is to demonstrate that essentially the same solution is produced as the preceding solution from `ode`.

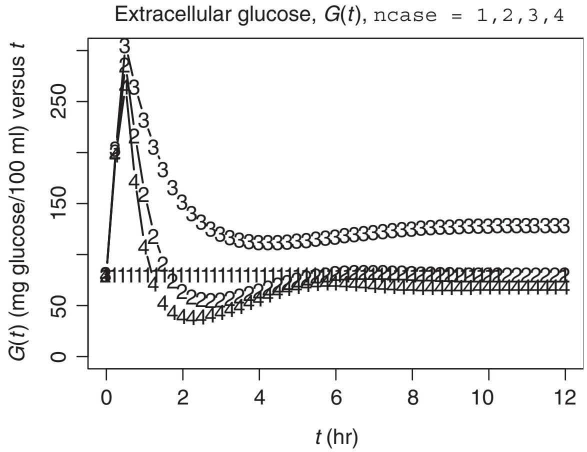

Figure 2.1 $G(t)$ for `ncase = 1,2,3,4`.

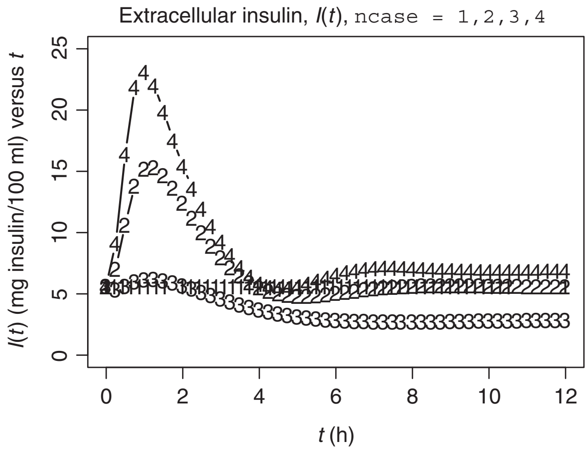

Figure 2.2 $I(t)$ for `ncase = 1,2,3,4`.

2.3.2 ODE Integration by RKF45

A main program with the RKF45 algorithm programmed as an in-line integrator is given in Listing 2.3.

```
#
# Glucose Tolerance Test
#
# Documentation comments removed
```

```
#
# ODE routine
  setwd("c:/R/bme_ode/chap2")
  source("glucose_2.R")
#
# Vectors, matrices for the graphical output
  nout=49
  Gplot=matrix(0,nrow=nout,ncol=4)
  Iplot=matrix(0,nrow=nout,ncol=4)
  dGplot=matrix(0,nrow=nout,ncol=4)
  dIplot=matrix(0,nrow=nout,ncol=4)
  tplot=rep(0,nout)
#
# Select rkf45 format
#
# nint = 1: No error estimation
#
# nint = 2: With error estimation
  nint=2
#
# Step through four cases
  for(ncase in 1:4){
#
# Select the case parameters
  if(ncase==1){Bb=14.3;          Gt=0}
  if(ncase==2){Bb=14.3;     Gt=80000}
  if(ncase==3){Bb=0.2*14.3; Gt=80000}
  if(ncase==4){Bb=2.0*14.3; Gt=80000}
#
# Model parameters
  Ex=15000; Cg=150; Ci=150; Q=8400; Dd=24.7;
   Gg=13.9; Gk=250;  Mu=72;  G0=51;   Aa=76;
#
# Initial condition
  t=0; ncall=0
  y=c(81.14,5.671)
  ee=c(0,0)
  cat(sprintf(
  "\n\n\n ncase = %2d \n\n        t        In         G
     I",ncase))
  if(nint==2){
```

```
  cat(sprintf("\n                          e1
     e2" ))}
#
# Parameters for t integration
  nt=10;tout=0.25;h=tout/nt
#
# rkf45 integration
  for(i1 in 1:nout){
#
#   Glucose infusion function
    if((t>=0)&(t<=0.51)){In=Gt}
    if(t>0.51        )  {In=0}
#
#   Solution output
    cat(sprintf("\n %8.2f%10.2f%10.4f%10.4f",t,In,y[1],
       y[2]))
    if(nint==2){
    cat(sprintf("\n                     %8.4f  %8.4f",
                                          ee[1],ee[2]))}
#
#   Store solution for plotting
    Gplot[i1,ncase]=y[1]
    Iplot[i1,ncase]=y[2]
    if(ncase==1)tplot[i1]=t
#
#   nt rkf45 steps
    for(i2 in 1:nt){
    if(nint==1){
      yb=y; tb=t;
      rk1=glucose_2(tb,yb)*h
      y=yb+0.25*rk1;
      t=tb+0.25*h;
      rk2=glucose_2(t,y)*h
      y=yb+(3/32)*rk1+(9/32)*rk2;
      t=tb+(3/8)*h;
      rk3=glucose_2(t,y)*h
      y=yb+(1932/2197)*rk1-(7200/2197)*rk2+(7296/2197)
         *rk3;
      t=tb+(12/13)*h;
      rk4=glucose_2(t,y)*h
      y=yb+(439/216)*rk1-8*rk2+(3680/ 513)*rk3-(845/4104)
```

```
         *rk4;
      t=tb+h;
      rk5=glucose_2(t,y)*h
      y=yb-(8/27)*rk1+2*rk2-(3544/2565)*rk3+(1859/4104)
         *rk4-(11/40)*rk5;
      t=tb+0.5*h;
      rk6=glucose_2(t,y)*h
      y=yb+(16/135)*rk1+(6656/12825)*rk3+(28561/56430)
         *rk4-(9/50)*rk5+
           (2/55)*rk6;
      t=tb+h;
    }
    if(nint==2){
      yb=y; tb=t;
      rk1=glucose_2(tb,yb)*h
      y=yb+0.25*rk1;
      t=tb+0.25*h;
      rk2=glucose_2(t,y)*h
      y=yb+(3/32)*rk1+(9/32)*rk2;
      t=tb+(3/8)*h;
      rk3=glucose_2(t,y)*h
      y=yb+(1932/2197)*rk1-(7200/2197)*rk2+(7296/2197)
         *rk3;
      t=tb+(12/13)*h;
      rk4=glucose_2(t,y)*h
      y=yb+(439/216)*rk1-8*rk2+(3680/ 513)*rk3-( 845/4104)
         *rk4;
      t=tb+h;
      rk5=glucose_2(t,y)*h
      y=yb-(8/27)*rk1+2*rk2-(3544/2565)*rk3+(1859/4104)
         *rk4-(11/40)*rk5;
      t=tb+0.5*h;
      rk6=glucose_2(t,y)*h
#
#     Fourth order step
      y4=yb+(25/216)*rk1+( 1408/2565)*rk3+(2197/4104)
         *rk4-(1/5)*rk5;
#
#     Fifth order step
      y=yb+(16/135)*rk1+(6656/12825)*rk3+(28561/56430)
         *rk4-(9/50)*rk5+
```

```
           (2/55)*rk6;
      t=tb+h;
#
#     Truncation error estimate
      ee=y-y4
    }
    }
 }
#
# Store derivatives for plotting
  cat(sprintf("\n\n Derivatives, ncase = %2d\n",ncase))
  cat(sprintf("\n        t   dG/dt   dI/dt"))
  for(it in 1:nout){
    dgdi=glucose_2(tplot[it],c(Gplot[it,ncase],Iplot[it,
       ncase]))
    cat(sprintf("\n %8.2f%8.2f%8.2f",tplot[it],dgdi[1],
       dgdi[2]))
    dGplot[it,ncase]=dgdi[1]
    dIplot[it,ncase]=dgdi[2]
  }
#
# Next case
}
#
# Single plot for G
  par(mfrow=c(1,1))
#
# G, ncase = 1
  plot(tplot,Gplot[,1],xlab="t (hr)",
  ylab="G(t) (mg glucose/100 ml) vs t",
  xlim=c(0,12),ylim=c(0,300),type="b",lty=1,pch="1",
     lwd=2,
  main="Extracellular glucose, G(t), ncase = 1,2,3,4")
#
# G, ncase = 2
  lines(tplot,Gplot[,2],type="b",lty=1,pch="2",lwd=2)
#
# G, ncase = 3
  lines(tplot,Gplot[,3],type="b",lty=1,pch="3",lwd=2)
#
# G, ncase = 4
```

```
  lines(tplot,Gplot[,4],type="b",lty=1,pch="4",lwd=2)
#
# Single plot for I
  par(mfrow=c(1,1))
#
# I, ncase = 1
  plot(tplot,Iplot[,1],xlab="t (hr)",
  ylab="I(t) (mg insulin/100 ml) vs t",
  xlim=c(0,12),ylim=c(0,25),type="b",lty=1,pch="1",lwd=2,
  main="Extracellular insulin, I(t), ncase = 1,2,3,4")
#
# I, ncase = 2
  lines(tplot,Iplot[,2],type="b",lty=1,pch="2",lwd=2)
#
# I, ncase = 3
  lines(tplot,Iplot[,3],type="b",lty=1,pch="3",lwd=2)
#
# I, ncase = 4
  lines(tplot,Iplot[,4],type="b",lty=1,pch="4",lwd=2)
#
# Single plot for dG/dt
  par(mfrow=c(1,1))
#
# dG/dt, ncase = 1
  plot(tplot,dGplot[,1],xlab="t (hr)",
  ylab="dG(t)/dt (mg glucose/100 ml)/hr vs t",
  xlim=c(0,12),ylim=c(-400,600),type="b",lty=1,pch="1",
     lwd=2,
  main="dG(t)/dt, ncase = 1,2,3,4")
#
# dG/dt, ncase = 2
  lines(tplot,dGplot[,2],type="b",lty=1,pch="2",lwd=2)
#
# dG/dt, ncase = 3
  lines(tplot,dGplot[,3],type="b",lty=1,pch="3",lwd=2)
#
# dG/dt, ncase = 4
  lines(tplot,dGplot[,4],type="b",lty=1,pch="4",lwd=2)
#
# Single plot for dI/dt
  par(mfrow=c(1,1))
```

```
#
# dI/dt, ncase = 1
  plot(tplot,dIplot[,1],xlab="t (hr)",
  ylab="dI(t)/dt (mg Insulin/100 ml)/hr vs t",
  xlim=c(0,12),ylim=c(-10,40),type="b",lty=1,pch="1",
     lwd=2,
  main="dI(t)/dt, ncase = 1,2,3,4")
#
# dI/dt, ncase = 2
  lines(tplot,dIplot[,2],type="b",lty=1,pch="2",lwd=2)
#
# dI/dt, ncase = 3
  lines(tplot,dIplot[,3],type="b",lty=1,pch="3",lwd=2)
#
# dI/dt, ncase = 4
  lines(tplot,dIplot[,4],type="b",lty=1,pch="4",lwd=2)
```

Listing 2.3 Main program with in-line RKF45 for the numerical integration of eqs. (2.1) and (2.2).

Listing 2.3 is similar to Listing 1.11. Therefore, only the details that are different are considered next.

- The documentation at the beginning of Listing 2.1 has been removed to conserve space.
- A series of vector and matrices are declared (preallocated) for plotting the solution.

  ```
  #
  # Vectors, matrices for the graphical output
    nout=49
    Gplot=matrix(0,nrow=nout,ncol=4)
    Iplot=matrix(0,nrow=nout,ncol=4)
    dGplot=matrix(0,nrow=nout,ncol=4)
    dIplot=matrix(0,nrow=nout,ncol=4)
    tplot=rep(0,nout)
  ```

  The use of these arrays was explained with Listings 1.11 and 2.1. Briefly, `dGplot, dIplot` have been added to plot the derivatives dG/dt of eqs. (2.1b) and (2.1c) and dI/dt of eqs. (2.2b) and (2.2c).

- The estimated RKF45 error `ee` is computed and displayed numerically with `nint=2`, as discussed in Listing 1.11. This estimated error is also initialized to zero as part of the ICs (`ee=c(0,0)`).
- The ODE integration using `ode` (and `lsoda` of Listing 2.1) is replaced with the RKF45 integration of Listing 1.11.

```
#
# Parameters for t integration
  nt=10;tout=0.25;h=tout/nt
#
# rkf45 integration
  for(i1 in 1:nout){
#
#   Glucose infusion function
    if((t>=0)&(t<=0.51)){In=Gt}
    if(t>0.51        )  {In=0}
#
#   Solution output
    cat(sprintf("\n %8.2f%10.2f%10.4f%10.4f",t,In,
       y[1],y[2]))
    if(nint==2){
    cat(sprintf("\n                  %8.4f  %8.4f",
                                    ee[1],ee[2]))}
#
#   Store solution for plotting
    Gplot[i1,ncase]=y[1]
    Iplot[i1,ncase]=y[2]
    if(ncase==1)tplot[i1]=t
#
#   nt rkf45 steps
    for(i2 in 1:nt){
    if(nint==1){
      yb=y; tb=t;
      rk1=glucose_2(tb,yb)*h
         .
         .
  Complete RKF45 coding is in Listing 2.3
         .
         .
#
#     Fifth order step
```

```
      y=yb+(16/135)*rk1+(6656/12825)*rk3+(28561/56430)
         *rk4-(9/50)*rk5+
           (2/55)*rk6;
      t=tb+h;
#
#     Truncation error estimate
      ee=y-y4
    }
    }
 }
```

Note the following in particular.

— The use of the two `for` loops with indices `i1` and `i2` (as discussed with Listing 1.11).

— The output interval in t is 0.25 and the integration step is, therefore, `h = tout/nt = 0.25/10 = 0.025`. This integration step was selected through `nt` to give good resolution in t (enough plotted points in t, e.g., 49 set previously) and a stable and accurate numerical solution (with $h = 0.025$).

— The use of `glucose_2` rather than `glucose_1` in RKF45 (these two routines differ in only the `return` statement with the inclusion of `list` for `glucose_1`). `glucose_2` is in Listing 2.6.

— After `nt` integration steps, the derivatives $dG/dt, dI/dt$ are computed by a call to the ODE routine `glucose_2` and displayed numerically (note in particular the input arguments to `glucose_2` which follow from the first line of Listing 2.2). These derivatives are also stored for subsequent plotting.

```
  #
  # Store derivatives for plotting
    cat(sprintf("\n\n Derivatives, ncase = %2d\n",
       ncase))
    cat(sprintf("\n        t   dG/dt   dI/dt"))
    for(it in 1:nout){
      dgdi=glucose_2(tplot[it],c(Gplot[it,ncase],
         Iplot[it,ncase]))
      cat(sprintf("\n %8.2f%8.2f%8.2f",tplot[it],
```

```
            dgdi[1],dgdi[2]))
          dGplot[it,ncase]=dgdi[1]
          dIplot[it,ncase]=dgdi[2]
        }
      #
      # Next case
      }
```

- Plots are added for dG/dt and dI/dt.

```
    #
    # dG/dt, ncase = 1
      plot(tplot,dGplot[,1],xlab="t (hr)",
      ylab="dG(t)/dt (mg glucose/100 ml)/hr vs t",
      xlim=c(0,12),ylim=c(-400,600),type="b",lty=1,
         pch="1",lwd=2,
      main="dG(t)/dt, ncase = 1,2,3,4")
          .                        .
          .                        .
    #
    # dI/dt, ncase = 1
      plot(tplot,dIplot[,1],xlab="t (hr)",
      ylab="dI(t)/dt (mg Insulin/100 ml)/hr vs t",
      xlim=c(0,12),ylim=c(-10,40),type="b",lty=1,pch="1",
         lwd=2,
      main="dI(t)/dt, ncase = 1,2,3,4")
          .                        .
          .                        .
```

The ODE routine `glucose_2` differs from `glucose_1` of Listing 2.2 in one line.

```
Glucose 1
#
# Return derivative vector
  return(list(c(dGdt,dIdt)))

Glucose 2
#
# Return derivative vector
  return(c(dGdt,dIdt))
```

`list` is required by `ode` (which calls `glucose_1` in Listing 2.1) and is not required by RKF45 (which calls `glucose_2` in Listing 2.3).

Abbreviated numerical output from Listing 2.3 is in Table 2.2.

We can note the following details of this output.

- For `ncase = 1`, $G(t)$ and $I(t)$ are constant and the derivatives $dG(t)/dt$ and $dI(t)/dt$ are zero as expected (see also Table 2.1).
- For all four solutions (`ncase=1,2,3,4`), the estimated error is zero to four figures after the decimal. This implies that the solutions are accurate to four figures after the decimal.
- All four solutions appear to have reached an equilibrium condition for $t \to 12$ (note the small derivatives).

The graphical output for $G(t)$ and $I(t)$ (Figs. 2.3 and 2.4) is the same as in Figs. 2.1 and 2.2. The graphical output for the derivatives follows in Figs. 2.3 and 2.4.

A comparison of the solutions from Listings 2.1 and 2.3 (Table 2.3) gives an indication of the accuracy of the preceding numerical solutions (in Tables 2.1 and 2.2).

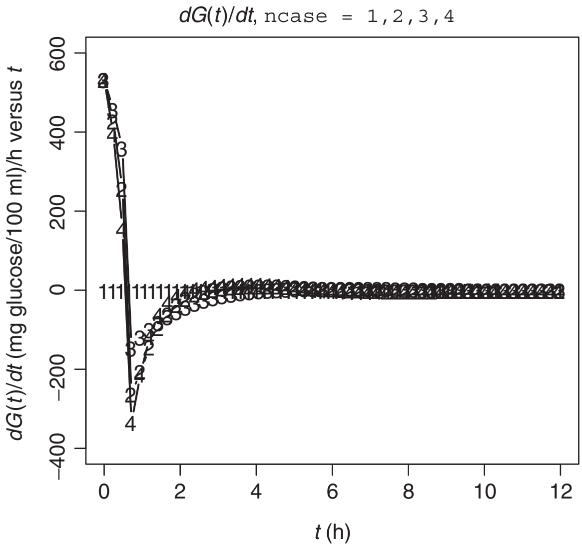

Figure 2.3 $dG(t)/dt$ for `ncase = 1,2,3,4`.

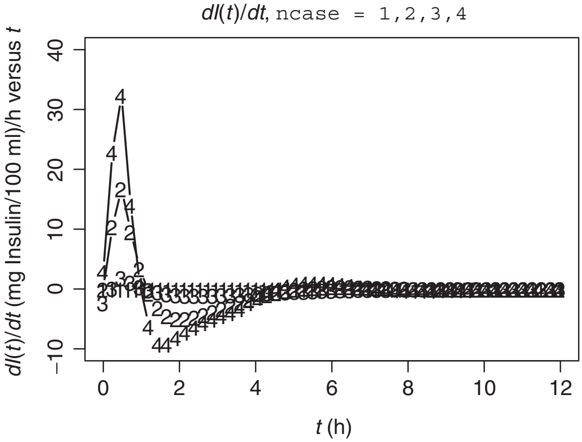

Figure 2.4 $dI(t)/dt$ for `ncase = 1,2,3,4`.

TABLE 2.2 Abbreviated output from of Listing 2.3.

```
 ncase =  1

       t        In         G         I
                          e1        e2
    0.00      0.00   81.1400    5.6710
                      0.0000    0.0000
    0.25      0.00   81.1397    5.6710
                     -0.0000    0.0000
    0.50      0.00   81.1395    5.6710
                     -0.0000    0.0000
    0.75      0.00   81.1393    5.6710
                     -0.0000    0.0000
    1.00      0.00   81.1392    5.6710
                     -0.0000    0.0000
      .                            .
      .                            .
      .                            .
  Output for t = 1.25 to 10.75 removed
      .                            .
      .                            .
      .                            .
   11.00      0.00   81.1392    5.6709
```

TABLE 2.2 (*Continued*)

```
                          0.0000   -0.0000
    11.25       0.00     81.1392    5.6709
                          0.0000    0.0000
    11.50       0.00     81.1392    5.6709
                          0.0000    0.0000
    11.75       0.00     81.1392    5.6709
                         -0.0000    0.0000
    12.00       0.00     81.1392    5.6709
                          0.0000   -0.0000

 Derivatives, ncase =  1

        t   dG/dt   dI/dt
     0.00   -0.00    0.00
     0.25   -0.00    0.00
     0.50   -0.00   -0.00
     0.75   -0.00   -0.00
     1.00   -0.00   -0.00
       .               .
       .               .
       .               .
  Output for t = 1.25 to 10.75 removed
       .               .
       .               .
       .               .
    11.00    0.00   -0.00
    11.25    0.00   -0.00
    11.50    0.00   -0.00
    11.75    0.00    0.00
    12.00    0.00    0.00

 ncase =  2

        t         In          G          I
                             e1         e2
     0.00   80000.00   81.1400    5.6710
                          0.0000    0.0000
```

(*continued*)

TABLE 2.2 (*Continued*)

```
    0.25  80000.00  201.7138    7.0999
                      0.0000   -0.0000
    0.50  80000.00  286.7212   10.6868
                      0.0000   -0.0000
    0.75      0.00  222.5755   14.0297
                     -0.0000    0.0000
    1.00      0.00  162.8538   15.4879
                     -0.0000    0.0000
      .                           .
      .                           .
      .                           .
  Output for t = 1.25 to 10.75 removed
      .                           .
      .                           .
      .                           .
   11.00      0.00   80.8916    5.6812
                     -0.0000   -0.0000
   11.25      0.00   80.9191    5.6748
                     -0.0000   -0.0000
   11.50      0.00   80.9520    5.6697
                      0.0000   -0.0000
   11.75      0.00   80.9871    5.6661
                      0.0000   -0.0000
   12.00      0.00   81.0219    5.6637
                      0.0000   -0.0000

 Derivatives, ncase =  2

       t   dG/dt   dI/dt
    0.00  533.33    0.00
    0.25  423.41   10.77
    0.50  240.55   17.06
    0.75 -270.02    9.25
    1.00 -204.55    2.82
      .               .
      .               .
      .               .
  Output for t = 1.25 to 10.75 removed
```

TABLE 2.2 (*Continued*)

```
     .              .
     .              .
     .              .
   11.00    0.09   -0.03
   11.25    0.12   -0.02
   11.50    0.14   -0.02
   11.75    0.14   -0.01
   12.00    0.14   -0.01

 ncase =  3

        t         In          G          I
                             e1         e2
     0.00  80000.00   81.1400     5.6710
                        0.0000     0.0000
     0.25  80000.00  204.2189     5.4200
                        0.0000     0.0000
     0.50  80000.00  305.9657     5.7036
                        0.0000     0.0000
     0.75      0.00  271.4036     6.1015
                       -0.0000    -0.0000
     1.00      0.00  237.6941     6.2824
                       -0.0000     0.0000
     .                           .
     .                           .
     .                           .
  Output for t = 1.25 to 10.75 removed
     .                           .
     .                           .
     .                           .
    11.00      0.00  129.3243     2.8929
                        0.0000     0.0000
    11.25      0.00  129.3232     2.8994
                       -0.0000     0.0000
    11.50      0.00  129.3050     2.9051
                       -0.0000     0.0000
    11.75      0.00  129.2737     2.9100
                       -0.0000     0.0000
```

(*continued*)

TABLE 2.2 (*Continued*)

```
   12.00      0.00  129.2328    2.9141
                     -0.0000    0.0000

 Derivatives, ncase =  3

       t   dG/dt   dI/dt
    0.00  533.33   -2.30
    0.25  453.14    0.18
    0.50  350.37    1.97
    0.75 -152.42    1.11
    1.00 -121.52    0.38
     .               .
     .               .
     .               .
  Output for t = 1.25 to 10.75 removed
     .               .
     .               .
     .               .
   11.00    0.04    0.03
   11.25   -0.04    0.02
   11.50   -0.10    0.02
   11.75   -0.15    0.02
   12.00   -0.18    0.02

 ncase =  4

       t        In         G         I
                          e1        e2
    0.00  80000.00   81.1400    5.6710
                      0.0000    0.0000
    0.25  80000.00  198.6589    9.1666
                      0.0000   -0.0000
    0.50  80000.00  265.5024   16.4543
                      0.0000   -0.0000
    0.75      0.00  177.1371   22.1836
                     -0.0000    0.0000
    1.00      0.00  110.0692   23.5312
                      0.0000   -0.0000
```

TABLE 2.2 (*Continued*)

```
     .                            .
     .                            .
     .                            .
  Output for t = 1.25 to 10.75 removed
     .                            .
     .                            .
     .                            .
   11.00      0.00   69.4686    6.9036
                      0.0000   -0.0000
   11.25      0.00   69.4911    6.9097
                      0.0000    0.0000
   11.50      0.00   69.5005    6.9158
                      0.0000    0.0000
   11.75      0.00   69.4998    6.9213
                      0.0000    0.0000
   12.00      0.00   69.4917    6.9259
                      0.0000    0.0000

 Derivatives, ncase =  4

        t   dG/dt   dI/dt
    0.00  533.33    2.87
    0.25  387.87   23.51
    0.50  133.34   32.56
    0.75 -337.31   12.81
    1.00 -202.14   -0.66
     .               .
     .               .
     .               .
  Output for t = 1.25 to 10.75 removed
     .               .
     .               .
     .               .
   11.00    0.12    0.02
   11.25    0.06    0.02
   11.50    0.02    0.02
   11.75   -0.02    0.02
   12.00   -0.04    0.02
```

TABLE 2.3 Comparison of the solutions from Listings 2.1 and 2.3.

Table 2.1, `ncase=4` (Listing 2.1)	$t = 0.5$	$G(t) = 265.50$	$I(t) = 16.454$
Table 2.2, `ncase=4` (Listing 2.3)	$t = 0.5$	$G(t) = 265.5024$	$I(t) = 16.4543$
Table 2.1, `ncase=4` (Listing 2.1)	$t = 12$	$G(t) = 69.49$	$I(t) = 6.926$
Table 2.2, `ncase=4` (Listing 2.3)	$t = 12$	$G(t) = 69.4917$	$I(t) = 6.9259$

This comparison suggests that the numerical solutions are accurate to at least four figures. In the case of the RKF45 solutions (Table 2.2), this accuracy is achieved with $h = 0.025$ and there was no indication of impending instability. To investigate this further, Listing 2.2 is modified by changing `nt=10` to `nt=1` so that $h = 0.025$ is changed to $h = 0.25$. The abbreviated numerical output is in Table 2.4.

The details listed in Table 2.5 are obtained by comparing the solutions in Tables 2.2 and 2.4.

The beginning results in Table 2.5 are for $t = 0.75$ rather than for $t = 0.50$ of Table 2.3 because the estimated errors `ee(1),ee(2)` are larger (see Table 2.4, `ncase=4`).

Table 2.3 suggests that the `nt=10` solution from Listing 2.3 is of relatively high accuracy. If we consider it to be essentially an exact solution, then the exact error $G(t)$ for the `nt=1` solution at $t = 0.75$ is $177.1371 - 170.7425 = 6.3946$. The estimated error is $ee(1) = 1.7644$ so that the error is underestimated. This is not entirely unexpected because of the large step for `nt=1`, $h = 0.25$, which is the output interval, that is, there is only one integration step for each output where the solution is changing most rapidly. An important point to note is that the error is only estimated by comparing the fourth-order and fifth-order solutions from RKF45, and the error estimate might be unreliable if h is large. Experience has indicated that when h is small enough to produce an accurate solution, the error estimate is also accurate and is reliable enough to adjust the integration step (in a variable step method such as RKF45 in `ode` of `deSolve`).

TABLE 2.4 Abbreviated output from Listing 2.3, `nt=1`.

```
 ncase =  1

       t         In          G          I
                            e1         e2
    0.00       0.00    81.1400     5.6710
                        0.0000     0.0000
    0.25       0.00    81.1397     5.6710
                       -0.0000     0.0000
    0.50       0.00    81.1395     5.6710
                       -0.0000     0.0000
    0.75       0.00    81.1393     5.6710
                       -0.0000     0.0000
    1.00       0.00    81.1392     5.6710
                       -0.0000     0.0000
     .                            .
     .                            .
     .                            .
  Output for t = 1.25 to 10.75 removed
     .                            .
     .                            .
     .                            .
   11.00       0.00    81.1392     5.6709
                        0.0000    -0.0000
   11.25       0.00    81.1392     5.6709
                        0.0000    -0.0000
   11.50       0.00    81.1392     5.6709
                        0.0000    -0.0000
   11.75       0.00    81.1392     5.6709
                        0.0000    -0.0000
   12.00       0.00    81.1392     5.6709
                        0.0000    -0.0000

 Derivatives, ncase =  1

       t   dG/dt   dI/dt
    0.00   -0.00    0.00
    0.25   -0.00    0.00
```

(*continued*)

TABLE 2.4 (*Continued*)

```
     0.50   -0.00   -0.00
     0.75   -0.00   -0.00
     1.00   -0.00   -0.00
       .               .
       .               .
       .               .
  Output for t = 1.25 to 10.75 removed
       .               .
       .               .
       .               .
    11.00    0.00   -0.00
    11.25    0.00   -0.00
    11.50    0.00   -0.00
    11.75    0.00    0.00
    12.00    0.00    0.00

 ncase =  2

        t        In         G         I
                           e1        e2
     0.00  80000.00   81.1400    5.6710
                       0.0000    0.0000
     0.25  80000.00  201.7134    7.1000
                       0.0046   -0.0001
     0.50  80000.00  286.7229   10.6879
                       0.0307   -0.0003
     0.75      0.00  222.0304   14.1499
                       0.8084    0.0183
     1.00      0.00  162.1029   15.5785
                      -0.0014    0.0002
       .                           .
       .                           .
       .                           .
  Output for t = 1.25 to 10.75 removed
       .                           .
       .                           .
       .                           .
    11.00      0.00   80.8909    5.6811
                      -0.0000   -0.0000
```

TABLE 2.4 (*Continued*)

```
   11.25      0.00   80.9186    5.6747
                     -0.0000   -0.0000
   11.50      0.00   80.9518    5.6696
                     -0.0000   -0.0000
   11.75      0.00   80.9871    5.6660
                      0.0000   -0.0000
   12.00      0.00   81.0220    5.6636
                      0.0000   -0.0000

 Derivatives, ncase =  2

       t   dG/dt   dI/dt
    0.00  533.33    0.00
    0.25  423.40   10.77
    0.50  240.52   17.06
    0.75 -271.69    9.14
    1.00 -204.71    2.70
     .               .
     .               .
     .               .
  Output for t = 1.25 to 10.75 removed
     .               .
     .               .
     .               .
   11.00    0.09   -0.03
   11.25    0.12   -0.02
   11.50    0.14   -0.02
   11.75    0.14   -0.01
   12.00    0.14   -0.01

 ncase =  3

       t        In         G         I
                          e1        e2
    0.00  80000.00   81.1400    5.6710
                      0.0000    0.0000
    0.25  80000.00  204.2186    5.4200
                      0.0010    0.0000
```

(*continued*)

TABLE 2.4 (*Continued*)

```
    0.50  80000.00  306.1217    5.7042
                      0.0033   -0.0000
    0.75      0.00  275.8744    6.1299
                      0.3163    0.0039
    1.00      0.00  240.8850    6.3242
                      0.0011    0.0000
     .                           .
     .                           .
     .                           .
  Output for t = 1.25 to 10.75 removed
     .                           .
     .                           .
     .                           .
   11.00      0.00  129.3329    2.8922
                      0.0000    0.0000
   11.25      0.00  129.3327    2.8988
                      0.0000    0.0000
   11.50      0.00  129.3150    2.9046
                      0.0000    0.0000
   11.75      0.00  129.2838    2.9096
                     -0.0000    0.0000
   12.00      0.00  129.2428    2.9138
                     -0.0000    0.0000

 Derivatives, ncase =  3

       t   dG/dt   dI/dt
    0.00  533.33   -2.30
    0.25  453.14    0.18
    0.50  350.17    1.97
    0.75 -158.55    1.18
     .               .
     .               .
     .               .
  Output for t = 1.25 to 10.75 removed
     .               .
     .               .
     .               .
   11.00    0.04    0.03
```

TABLE 2.4 (*Continued*)

```
   11.25   -0.04    0.02
   11.50   -0.10    0.02
   11.75   -0.15    0.02
   12.00   -0.18    0.02

 ncase =  4

       t         In         G          I
                           e1         e2
    0.00  80000.00   81.1400     5.6710
                      0.0000     0.0000
    0.25  80000.00  198.6600     9.1672
                      0.0084    -0.0007
    0.50  80000.00  265.4178    16.4550
                      0.0276    -0.0017
    0.75      0.00  170.7245    22.3869
                      1.7644     0.0303
    1.00      0.00  106.3199    23.4860
                     -0.0014     0.0003
      .                            .
      .                            .
      .                            .
  Output for t = 1.25 to 10.75 removed
      .                            .
      .                            .
      .                            .
   11.00      0.00   69.4710     6.9041
                      0.0000    -0.0000
   11.25      0.00   69.4923     6.9102
                      0.0000    -0.0000
   11.50      0.00   69.5008     6.9163
                      0.0000    -0.0000
   11.75      0.00   69.4993     6.9217
                      0.0000     0.0000
   12.00      0.00   69.4908     6.9263
                      0.0000     0.0000
```

(*continued*)

TABLE 2.4 (*Continued*)

```
 Derivatives, ncase =  4

        t   dG/dt   dI/dt
     0.00  533.33    2.87
     0.25  387.86   23.51
     0.50  133.51   32.55
     0.75 -326.28   11.48
     1.00 -192.90   -1.35
      .               .
      .               .
      .               .
  Output for t = 1.25 to 10.75 removed
      .               .
      .               .
      .               .
    11.00    0.11    0.02
    11.25    0.06    0.02
    11.50    0.01    0.02
    11.75   -0.02    0.02
    12.00   -0.04    0.02
```

2.3.3 ODE Integration with RKF45 in a Separate Routine

We now consider a variation of Listing 2.3 in which RKF45 is placed in a separate routine. This is a worthwhile approach as it makes the programming easier to follow (more modular). First, the main program that parallels Listing 2.3 is in Listing 2.4.

```
#
# Glucose Tolerance Test
#
# Documentation comments removed
#
# ODE routine
  setwd("c:/R/bme_ode/chap2")
  source("glucose_2.R")
  source("rkf45.R")
#
```

TABLE 2.5 Comparison of the solutions from Listing 2.3, `nt=1,10`.

Table 2.2, ncase=4, `nt=10`	$t = 0.75$	$G(t) = 177.1371$	$I(t) = 22.1836$
		$ee(1) = -0.0000$	$ee(2) = 0.0000$
Table 2.4, ncase=4, `nt=1`	$t = 0.75$	$G(t) = 170.7245$	$I(t) = 22.3869$
		$ee(1) = 1.7644$	$ee(2) = 0.0303$
Table 2.2, ncase=4, `nt=10`	$t = 12$	$G(t) = 69.4917$	$I(t) = 6.9259$
		$ee(1) = 0.0000$	$ee(2) = 0.0000$
Table 2.4, ncase=4, `nt=1`	$t = 12$	$G(t) = 69.4908$	$I(t) = 6.9263$
		$ee(1) = 0.0000$	$ee(2) = 0.0000$

```
# Select RKF45 method
#
# nint = 1: No error estimation
#
# nint = 2: With error estimation
  nint=2
#
# Vectors, matrices for the graphical output
  nout=49
  Gplot=matrix(0,nrow=nout,ncol=4)
  Iplot=matrix(0,nrow=nout,ncol=4)
  tplot=rep(0,nout)
#
# Step through four cases
  for(ncase in 1:4){
#
# Select the case parameters
  if(ncase==1){Bb=14.3;         Gt=0}
  if(ncase==2){Bb=14.3;     Gt=80000}
  if(ncase==3){Bb=0.2*14.3; Gt=80000}
  if(ncase==4){Bb=2.0*14.3; Gt=80000}
#
# Model parameters
  Ex=15000; Cg=150; Ci=150; Q=8400; Dd=24.7;
   Gg=13.9; Gk=250;  Mu=72;  G0=51;   Aa=76;
#
# Initial condition
  t=0; ncall=0
```

```
  y=c(81.14,5.671)
  ee=c(0,0)
  cat(sprintf(
  "\n ncase = %2d \n\n        t        In          G
     I",ncase))
  if(nint==2){
  cat(sprintf("\n                           e1
     e2" ))}
#
# Parameters, functions for t integration
  nt=1;tout=0.25;h=tout/nt
#
# rkf45 integration
  for(i1 in 1:nout){
#
#   Glucose infusion function
    if((t>=0)&(t<=0.5)){In=Gt}
    if(t>0.5        )  {In=0}
#
#   Solution output
    cat(sprintf("\n %8.2f%10.2f%10.4f%10.4f",t,In,y[1],
       y[2]))
    if(nint==2){
    cat(sprintf("\n                     %8.4f  %8.4f",
                                          ee[1],ee[2]))}
#
#
#   Store solution for plotting
    Gplot[i1,ncase]=y[1]
    Iplot[i1,ncase]=y[2]
    if(ncase==1)tplot[i1]=t
#
#   rkf45 integration over nt points
    yout=rkf45(nt,h,t,y)
    y=yout; t=t+tout
 }
#
# Calls to glucose_2
  cat(sprintf("\n\n ncall = %5d\n\n",ncall))
#
# Next case
```

```
}
#
# Single plot for G
  par(mfrow=c(1,1))
#
# G, ncase = 1
  plot(tplot,Gplot[,1],xlab="t (hr)",
  ylab="G(t) (mg glucose/100 ml) vs t",
  xlim=c(0,12),ylim=c(0,300),type="b",lty=1,pch="1",lwd=2,
  main="Extracellular glucose, G(t), ncase = 1,2,3,4")
#
# G, ncase = 2
  lines(tplot,Gplot[,2],type="b",lty=1,pch="2",lwd=2)
#
# G, ncase = 3
  lines(tplot,Gplot[,3],type="b",lty=1,pch="3",lwd=2)
#
# G, ncase = 4
  lines(tplot,Gplot[,4],type="b",lty=1,pch="4",lwd=2)
#
# Single plot for I
  par(mfrow=c(1,1))
#
# I, ncase = 1
  plot(tplot,Iplot[,1],xlab="t (hr)",
  ylab="I(t) (mg insulin/100 ml) vs t",
  xlim=c(0,12),ylim=c(0,25),type="b",lty=1,pch="1",lwd=2,
  main="Extracellular insulin, I(t), ncase = 1,2,3,4")
#
# I, ncase = 2
  lines(tplot,Iplot[,2],type="b",lty=1,pch="2",lwd=2)
#
# I, ncase = 3
  lines(tplot,Iplot[,3],type="b",lty=1,pch="3",lwd=2)
#
# I, ncase = 4
  lines(tplot,Iplot[,4],type="b",lty=1,pch="4",lwd=2)
```

Listing 2.4 Main program with call to a separate ODE integration routine for RKF45.

Listing 2.4 is the same as Listing 2.3 except for the call to the separate ODE integration routine, `rkf45`, and the display of the number of calls to the ODE routine `glucose_2`.

```
#
#   rkf45 integration over nt points
    yout=rkf45(nt,h,t,y)
    y=yout; t=t+tout
 }
#
# Calls to glucose_2
  cat(sprintf("\n\n ncall = %5d\n\n",ncall))
```

$G(t)$ and $I(t)$ are again programmed as `y[1]` and `y[2]`, which are computed as the vector `yout` returned by `rkf45` (these are equated as `y=yout` right after the call to `rkf45`).

`rkf45` is listed in Listing 2.5.

```
    rkf45=function(nt,h,t,y) {
#
#   Function rkf45 implements a fourth order Runge Kutta
#   method embedded in a fifth order Runge Kutta method.
#   The ODE routine has to be renamed for a new
#   application.
#
#   The arguments are
#
#     Input
#
#       nt - number of rkf45 steps between the starting
#            and final points along the solution
#
#       h  - integration step (constant)
#
#       t  - initial value of the independent variable
#
#       y  - initial dependent variable vector
#
#     Output
#
#       y  - solution vector after nt integration steps
```

```
#
#   nt rkf45 steps
    for(i2 in 1:nt){
    if(nint==1){
      yb=y; tb=t;
      rk1=glucose_2(tb,yb)*h
      y=yb+0.25*rk1;
      t=tb+0.25*h;
      rk2=glucose_2(t,y)*h
      y=yb+(3/32)*rk1+(9/32)*rk2;
      t=tb+(3/8)*h;
      rk3=glucose_2(t,y)*h
      y=yb+(1932/2197)*rk1-(7200/2197)*rk2+(7296/2197)
         *rk3;
      t=tb+(12/13)*h;
      rk4=glucose_2(t,y)*h
      y=yb+(439/216)*rk1-8*rk2+(3680/ 513)*rk3-( 845/4104)
         *rk4;
      t=tb+h;
      rk5=glucose_2(t,y)*h
      y=yb-(8/27)*rk1+2*rk2-(3544/2565)*rk3+(1859/4104)
         *rk4-(11/40)*rk5;
      t=tb+0.5*h;
      rk6=glucose_2(t,y)*h
      y=yb+(16/135)*rk1+(6656/12825)*rk3+(28561/56430)
         *rk4-(9/50)*rk5+
           (2/55)*rk6;
      t=tb+h;
    }
    if(nint==2){
      yb=y; tb=t;
      rk1=glucose_2(tb,yb)*h
      y=yb+0.25*rk1;
      t=tb+0.25*h;
      rk2=glucose_2(t,y)*h
      y=yb+(3/32)*rk1+(9/32)*rk2;
      t=tb+(3/8)*h;
      rk3=glucose_2(t,y)*h
      y=yb+(1932/2197)*rk1-(7200/2197)*rk2+(7296/2197)
         *rk3;
      t=tb+(12/13)*h;
```

```
      rk4=glucose_2(t,y)*h
      y=yb+(439/216)*rk1-8*rk2+(3680/ 513)*rk3-( 845/4104)
         *rk4;
      t=tb+h;
      rk5=glucose_2(t,y)*h
      y=yb-(8/27)*rk1+2*rk2-(3544/2565)*rk3+(1859/4104)
         *rk4-(11/40)*rk5;
      t=tb+0.5*h;
      rk6=glucose_2(t,y)*h
#
#     Fourth order step
      y4=yb+(25/216)*rk1+( 1408/2565)*rk3+(  2197/4104)
         *rk4-( 1/5)*rk5;
#
#     Fifth order step
      y=yb+(16/135)*rk1+(6656/12825)*rk3+(28561/56430)
         *rk4-(9/50)*rk5+
           (2/55)*rk6;
      t=tb+h;
#
#     Truncation error estimate
      ee=y-y4
      ee <<- ee
    }
  }
      return(c(y))
}
```

Listing 2.5 Routine `rkf45`.

We can note the following details about `rkf45`.

- The input (RHS) arguments follow directly from Listing 2.4.

```
rkf45=function(nt,h,t,y)
```

- `nt` integration steps are taken within each call to `rkf45`. Two options are programmed: `nint=1` without error estimation and `nint=2` with error estimation.

```
for(i2 in 1:nt){
 if(nint==1){
```

- The programming of the RKF45 algorithm is the same as in Listing 2.3.
- The estimated error `ee` is returned to the main program of Listing 2.4 (note the use of `<<-` to provide the return to the higher level main program).

```
      ee <<- ee
```

- The solution vector is returned after `nt` steps along the solution.

```
     }
   }
       return(c(y))
  }
```

  The first brace `}` completes the `nint=2` option. The second brace completes the `for` in `i2`. The third brace completes `rkf45`.
- `rkf45` of Listing 2.5 can be considered to be a library routine that can be applied to other ODE systems. However, it does require the specification of an ODE routine, in this case `glucose_2`. Changing this name is easily accomplished with an editor. An alternative would be to pass the ODE routine to `rkf45` as an argument, as was done with `ode` (`func=glucose_1` in Listing 2.1).

The ODE routine `glucose_2` is in Listing 2.6.

```
  glucose_2=function(t,y) {
#
# Assign state variables:
  G=y[1];
  I=y[2];
#
# Glucose infusion function
  if((t>=0)&(t<=0.51)){In=Gt}
  if(t>0.51          ){In=0}
#
# ODEs
#
# Glucose equations
```

```
  if(G< Gk){dGdt=(1/Cg)*(Q+In-(Gg*I*G)-Dd*G)}
  if(G>=Gk){dGdt=(1/Cg)*(Q+In-(Gg*I*G)-Dd*G-Mu*(G-Gk))}
#
# Insulin equations
  if(G< G0){dIdt=(1/Ci)*(-Aa*I)}
  if(G>=G0){dIdt=(1/Ci)*(-Aa*I+Bb*(G-G0))}
#
# Calls to glucose_2
  ncall <<- ncall+1
#
# Return derivative vector
  return(c(dGdt,dIdt))
}
```

Listing 2.6 ODE routine `glucose_2`.

The essential detail is the `return`, that is, `return(list(c(dGdt, dIdt)))` in `glucose_1` of Listing 2.2 and `return(c(dGdt,dIdt))` in `glucose_2` of Listing 2.6. The output from Listings 2.4–2.6 is the same as in Table 2.2, with the addition of the number of calls to `glucose_2`.

```
 ncall =  2940
```

This number (which is the same for `ncase=1,2,3,4`) comes from the six derivative evaluations in `rkf45` of Listing 2.5, that is, `6*nout*nt = (6)(49)(10) = 2940`. This modest number of calls indicates that eqs. (2.1) and (2.2) are nonstiff (in contrast with `240000` for eqs. (1.1)).

If the main program in Listing 2.4 is executed with `nt=1`, the solution of Table 2.4 is produced, with the number of calls to `glucose_2`

```
ncall =   294
```

as expected (`2940/10`). In the case of eqs. (2.1) and (2.2), h was determined by accuracy rather than stability; even with the large value $h = 0.25$ for `nt=1`, the solution was stable but inaccurate (consider again Table 2.5). In order words, eqs. (2.1) and (2.2) are not stiff.

In the case of eqs. (1.1), h was constrained by stability to a small value because eqs. (1.1) are stiff. The small value of h then produced excessive accuracy (very small errors) and clearly the use of a stiff (implicit) integrator (such as `lsoda`) was very efficient in the sense that the stiff integrator required far fewer calls to the ODE routine than the nonstiff integrators such as the explicit Euler method through the RKF45 integrator.

In conclusion, the stiffness of an ODE system generally is not apparent and some experimentation with the choice of an integrator and the integration step is usually required. Physical considerations may immediately indicate that an ODE system is stiff, for example, some chemical kinetic models (with fast and slow reactions) have stiffness ratios of $10^9 - 10^{12}$ so that the required use of a stiff integrator is immediately apparent.

2.3.4 *h* Refinement

The preceding error analysis in which the investigation of h was carried out through the variation in `nt` is termed as *h refinement*. Generally, h is varied and the observed effect on the solution is used to infer the accuracy of the solution. This can be done without using an analytical solution (only the ODE is required, and it does not have to be differentiated to include additional terms in the Taylor series that represents the solution). However, the accuracy can only be inferred even with explicit error estimates such as from an embedded method. Also, as h is varied, instability may occur indicating that the stability limit of the method has been exceeded. In this case, h is determined by stability and not by accuracy, and the use of a stiff integrator may be required.

2.3.5 *p* Refinement

The preceding error analysis also indicates that varying the order of the integration method can provide an estimate of the integration error. As the order of a method is typically designated as $O(h^p)$, comparing solutions from algorithms of different orders is usually termed as *p refinement*. In the case of RKF45, the solutions for a fourth-order method and a fifth-order method were compared through the estimated error (`ee` in Listing 2.4).

Library integrators such as `lsoda` perform h and p refinement simultaneously and are, therefore, generally more complicated than the preceding explicit methods. Also, the source code is not readily available to study the details. Therefore, the basic explicit integrators considered previously (explicit Euler method through RKF45) can be useful as an introduction and can also be used to calculate solutions to nonstiff ODEs with good accuracy.

2.4 Conclusions

This concludes the discussion of the numerical integration of eqs. (2.1) and (2.2) by a variety of algorithms, both nonstiff (explicit) and stiff (implicit). An error analysis was performed in several ways without the use of an exact solution. Some experimentation, for example, with the step h (h refinement), was suggested by the results.

This need for numerical experimentation is generally true for a new ODE application. For example, additional tests could include changes in the method (p refinement) and variation of the error tolerances for a variable step method such as the BDF (backward differentiation formula methods) in `lsoda`. The solutions for a new ODE application should be viewed critically and tested thoroughly with some form of error analysis.

Once a solution of acceptable accuracy is computed, experimentation with the model can proceed, which is the ultimate objective of computer-based mathematical modeling. We, therefore, should keep in mind the famous statement by Richard Hamming [1]: *The purpose of computing is insight, not numbers.*

Eqs. 2.1 and 2.2 are an early model of the dynamics of the glucose tolerance test [3]. This test has been the subject of a series of papers and remains an active area of research. An example is given in [5].

References

[1] Hamming, R. W. (1962), *Numerical Methods for Scientists and Engineers*, McGraw-Hill Book Co., New York.

[2] Lee, H.J., and W.E. Schiesser (2004), *Ordinary and Partial Differential Equation Routines, in C, C++, Fortran, Java, Maple and Matlab*, CRC Press, Boca Raton, FL.

[3] Randall, J.E. (1980), *Microcomputers and Physiological Simulation*, Addison-Wesley Publishing Company, Inc., Reading, MA.

[4] Shampine, L.F., and S. Thompson (2007), Stiff systems, Scholarpedia, vol. 2, no. 3, p 2855; available at: `http://www.scholarpedia.org/article/Stiff_systems`.

[5] Vahidi, O., K.E. Kwok, R.B. Gopaluni, and L. Sum (2011), Developing a physiological model for type II diabetes mellitus, *Biochem. Eng. J.*, **55** (1), pp 7–16.