Introduction to computational programming

Chapter 4 Exercise

Forward and Backward Euler Methods

Solutions

David M. Rosenberg

University of Chicago

Committee on Neurobiology

Version control information:
Last changed date: 2009-11-17 20:43:48 -0600 (Tue, 17 Nov 2009)
Last changes revision:246
Version: Revision 246
Last changed by: David M. Rosenberg

November 18, 2009

1. Consider the simple SIS epidemiology model introduced in the section 3.2, where S stands for the number of individuals susceptible to the infection, and I is the number of infected individuals. Remember that the total population is constant, $N = I + S$, and thus the two ODEs can be reduced to one:

$$\dot{I} = \beta(N - I)I - \gamma I$$

β is the infectivity parameter (rate of infection per encounter) and γ is the recovery rate of infected individuals, and both are positive numbers.

(a) Find the fixed points of this system, and describe when they are biologically relevant. Explain what this means for your predictions for the course of an epidemic, and how it depends on the value of the parameters.

$$\begin{aligned} \dot{I} = 0 &\Rightarrow \beta(N - I)I = \gamma I \\ \beta(N - I) &= \gamma \\ \beta I &= \beta N - \gamma \\ I &= \frac{\beta N - \gamma}{\beta} \end{aligned}$$

Fixed points exist for this model when $I = 0$ and when $I = \frac{\beta N - \gamma}{\gamma}$.

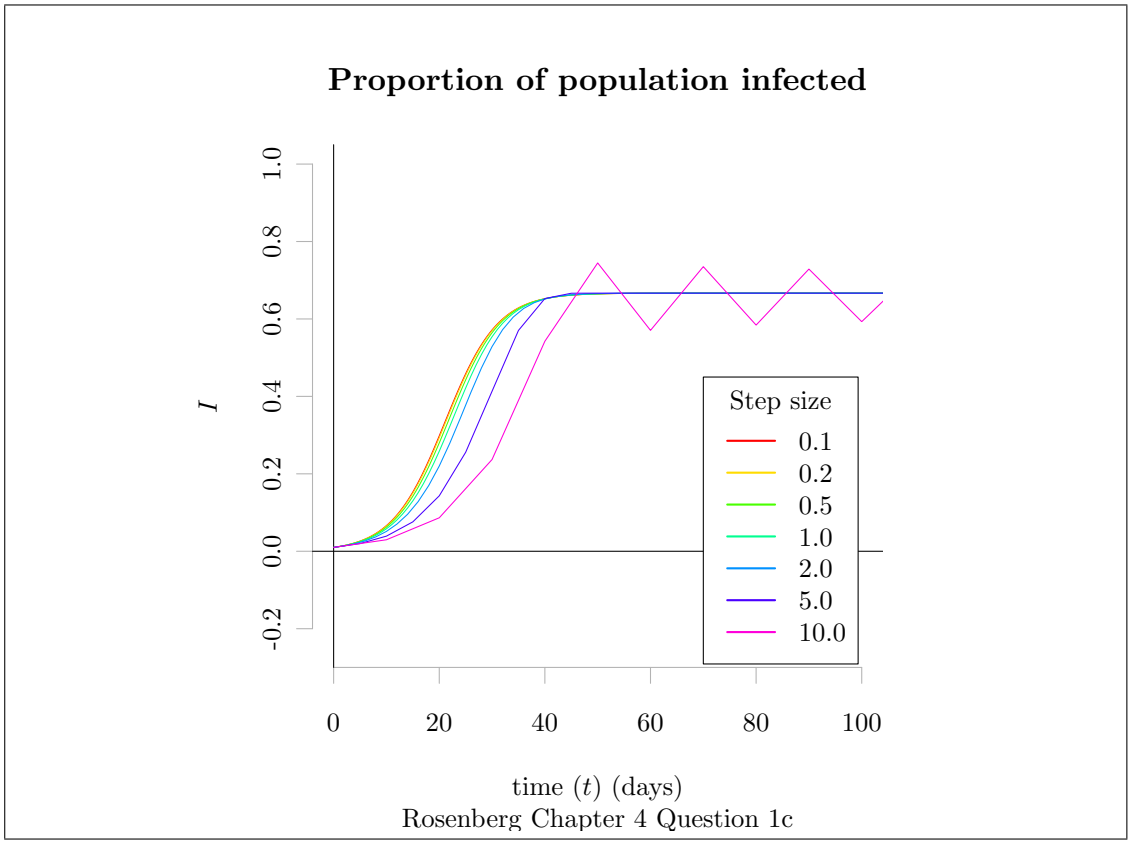
(b) Determine the stability of the fixed points analytically. Again, describe the implications for epidemiological prediction.

$$
\begin{aligned}
f(I) &= \beta(N-I)I - \gamma I \\
f'(I) &= (\beta N - \gamma) - 2\beta I \\
f'(0) &= \beta N - \gamma \\
\text{Suppose } I^* &= 0 \\
f'(I^*) &< 0 \\
(\beta N - \gamma) - 2\beta(0) &< 0 \\
\gamma &> \beta N \\
\text{Suppose } I^* &= \frac{\beta N - \gamma}{\beta} \\
f'(I^*) &< 0 \\
(\beta N - \gamma) - 2\beta\frac{\beta N \gamma}{\beta} &< 0 \\
\gamma - \beta N &< 0 \\
\gamma &< \beta N
\end{aligned}
$$

Thus there are two fixed points, $I = 0$ and $I = \frac{\beta N - \gamma}{\beta}$. The former is stable when $\gamma > \beta N$ and unstable otherwise. Conversely, the latter is stable when $\gamma < \beta N$ and unstable otherwise.

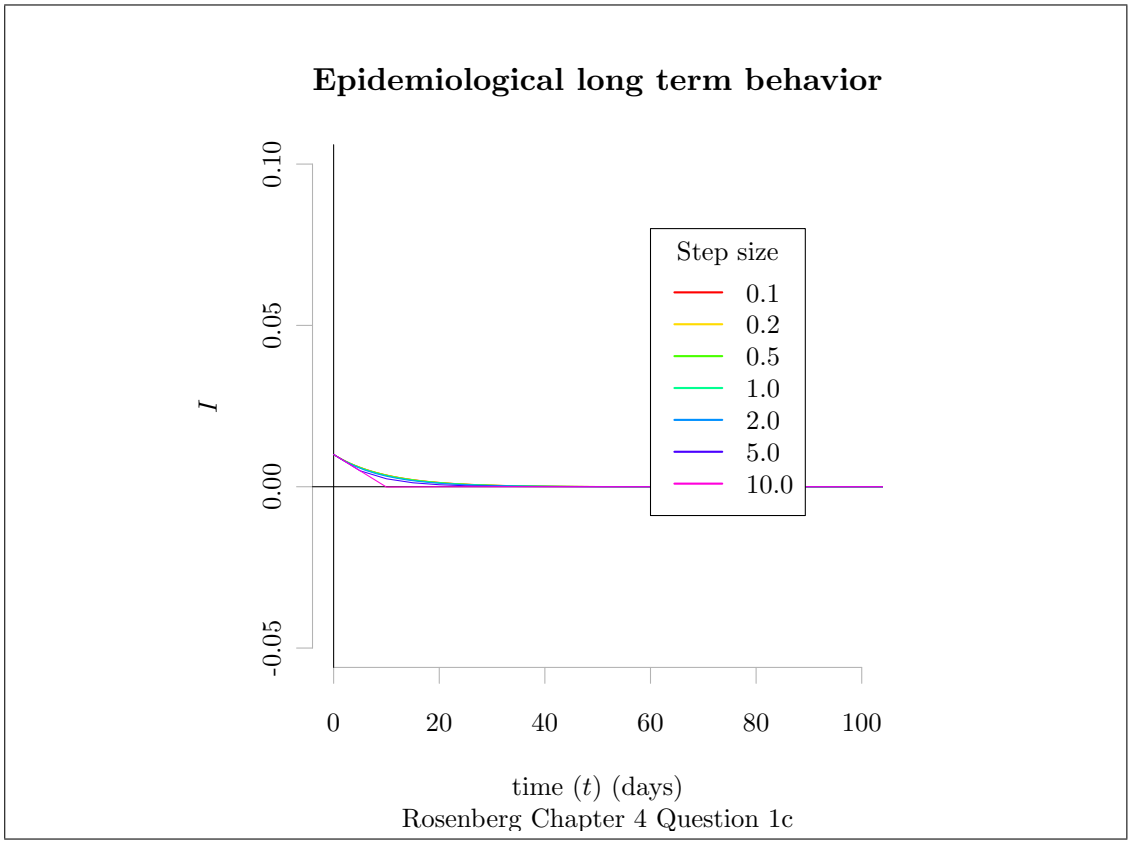
(c) Let $N = 1$ (this means that I and S are the fractions of infected and susceptible, respectively), $\beta = 0.3/day$, $\gamma = 0.1/day$. Use your implementation of Forward Euler to solve this equation for the fraction of infected over 365 days, starting with $I = 0.01$. Vary the time step Δt from small (0.1 day) to large (10 days) and report what effect this has on your numerical solution. At what time step is the behavior consistent with the behavior predicted by the theoretical analysis above?

```
> dotI <- function(i, n=1, beta=0.3, gamma=0.1) {
+   return(beta * i * (n-i) - gamma * i)
+ }
> runForwardEuler1 <- function(diffEq, deltaT, tRange=c(0, 365), i0=0.01) {
+   tVals <- seq(from=tRange[1], to=tRange[2], by=deltaT);
+   iVals <- tVals;
+   iVals[1] <- i0;
+   for (ii in 2:length(tVals)) {
+     iVals[ii] <- iVals[ii - 1] + deltaT * dotI(i=iVals[ii - 1])
+   }
+   result <- list(t=tVals, i=iVals);
+ }
> tSteps <- c(0.1, 0.2, 0.5, 1, 2, 5, 10);
> colorList <- rainbow(n=length(tSteps));
> plot(c(0, 100), c(-0.25, 1), type='n', bty='n',
+      fg=grey(0.7), sub='Rosenberg Chapter 4 Question 1c',
+      main='Proportion of population infected',
+      xlab='time ($t$) (days)', ylab='$I$');
> abline(h=0);
> abline(v=0);
> for (tt in 1:length(tSteps)) {
+   res <- runForwardEuler1(dotI, deltaT=tSteps[tt]);
+   lines(res$t, res$i, col=colorList[tt]);
+ }
> legend(c('0.1', '0.2', '0.5', '1.0', '2.0', '5.0', '10.0'),
+        x=70, y=0.45, lwd=2, col=colorList, title='Step size',
+        bg='white')
```



(d) Now repeat this numerical experiment with the parameters changed to $\beta = 0.1/day$ and $\gamma = 0.2/day$. Once again, try different step sizes and report at what step size the instability of the method becomes apparent, and the numerical solution behaves qualitatively different from the theoretical prediction.

```
> dotI2 <- function(i, n=1, beta=0.1, gamma=0.2) {
+   return(beta * i * (n-i) - gamma * i)
+ }
> runForwardEuler2 <- function(diffEq, deltaT, tRange=c(0, 365), i0=0.01) {
+   tVals <- seq(from=tRange[1], to=tRange[2], by=deltaT);
+   iVals <- tVals;
+   iVals[1] <- i0;
+   for (ii in 2:length(tVals)) {
+     iVals[ii] <- iVals[ii - 1] + deltaT * dotI2(i=iVals[ii - 1])
+   }
+   result <- list(t=tVals, i=iVals);
+ }
> tSteps <- c(0.1, 0.2, 0.5, 1, 2, 5, 10);
> colorList <- rainbow(n=length(tSteps));
> plot(c(0, 100), c(-0.05, 0.1), type='n', bty='n',
+      fg=grey(0.7), main='\\textbf{Epidemiological long term behavior}',
+      sub='Rosenberg Chapter 4 Question 1c',
+      xlab='time ($t$) (days)', ylab='$I$');
> abline(h=0);
> abline(v=0);
> for (tt in 1:length(tSteps)) {
+   res <- runForwardEuler2(dotI2, deltaT=tSteps[tt]);
+   lines(res$t, res$i, col=colorList[tt]);
+ }
> legend(c('0.1', '0.2', '0.5', '1.0', '2.0', '5.0', '10.0'),
+        x=60, y=0.08, lwd=2, col=colorList, title='Step size',
+        bg='white')
```



2. Suppose a signaling molecule (whose concentration is denoted by S) binds a receptor molecule (with concentration R) to produce a complex (concentration C). The rate of the binding reaction is k_+, and the rate of dissociation of the complex into S and R is k_-. Further, let us assume that the concentration of the signaling molecule is constant (it is maintained at some level by physiological mechanisms).

   (a) Write down an ODE model for the concentration of the receptor molecule.

$$\begin{aligned}\dot{R} = f(R,t) &= k_2(R_0 - R) - R\,(k_1 S) \\ &= -(k_1 S + k2)R + k_2 R_0\end{aligned}$$

   (b) Analyze the model in the usual fashion: find the equilibria, and determine their stability, as a function of the rate parameters.

$$
\begin{aligned}
R^* (k_1 S + k_2) &= k_2 R_0 \\
R^* &= \frac{k_2 R_0}{k_1 S + k_2} \\
f(R,t) &= -(k_1 S + k_2) R - k_2 R_0 \\
\frac{d}{dR} f(R,t) &= k_2 - k_1 S \\
\frac{d}{dR} f(R^*,t) &= k_2 - k_1 S
\end{aligned}
$$

Thus there is one fixed point at $R = \frac{k_2 R_0}{k_1 S - k_2}$. It is stable when $k_2 < k_1 S$ and unstable otherwise.

(c) Find an analytical solution for the model, and comment on how it relates to the stability analysis. (Hint: use an integrating factor)

$$
\begin{aligned}
\dot{R} + (k_1 S + k_2) R &= k_2 R_0 \\
\mu &= e^{\int (k_1 S + k_2) dt} \\
&= e^{t(k_1 S + k_2)} \\
\mu' &= (k_1 S + k_2) e^{t(k_1 S + k_2)} \\
(R \cdot \mu)' &= \mu k_2 R_0 \\
R \cdot \mu + C &= \frac{k_2 R_0}{k_1 S + k_2} e^{-t(k_1 S + k_2)} \\
R &= \frac{1}{\mu} \left(\frac{k_2 R_0}{k_1 S + k_2} e^{t(k_1 S + k_2)} - C \right) \\
R &= \frac{k_2 R_0}{k_1 S + k_2} - C e^{-t(k_2 + k_1 S)} \\
R_0 &= \frac{k_2 R_0}{k_1 S + k_2} - C e^0 \\
C &= \frac{k_2 R_0}{k_1 S + k_2} - R_0 \\
C &= \frac{R_0 k_1 S}{k_1 S + k_2} \\
R &= \frac{k_2 R_0}{k_1 S + k_2} - \frac{R_0 k_1 S}{k_1 S + k_2} e^{-t(k_1 S + k_2)}
\end{aligned}
$$

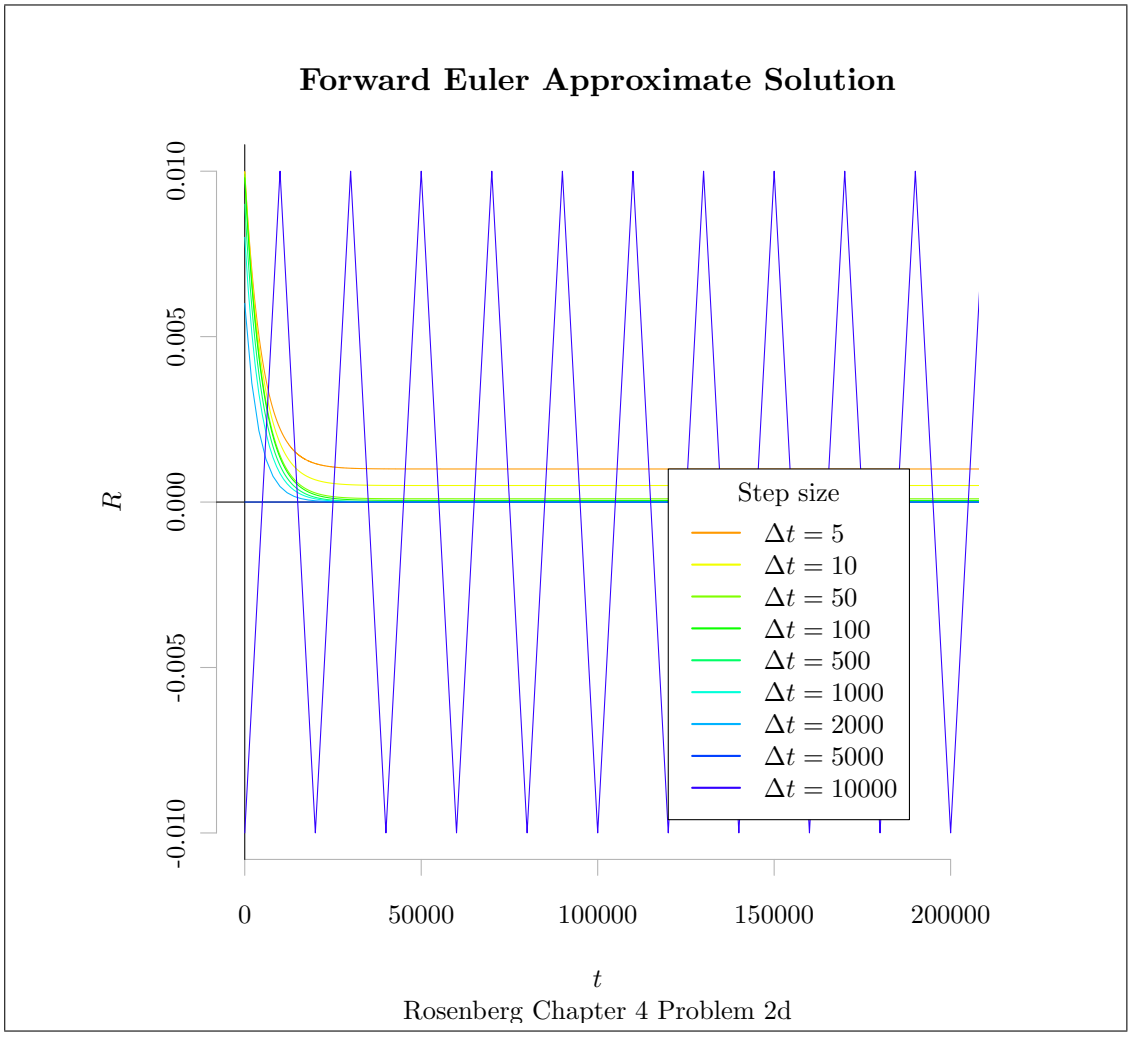
From the formula it is clear that the solution will approach the stable equilibrium value $R^* = \frac{k_2 R_0}{k_1 S + k_2}$ as $t \to \infty$.

(d) Let $S = 10^{-6} M$, $k_+ = 10^{-3} M^{-1} s^{-1}$, $C = 0$, and $k_- = 10^{-5} s^{-1}$. Use the Forward Euler method to solve this model numerically, starting with $R = 10^{-4} M$.

Note: As per Dmitry's email, the following values may be used instead.

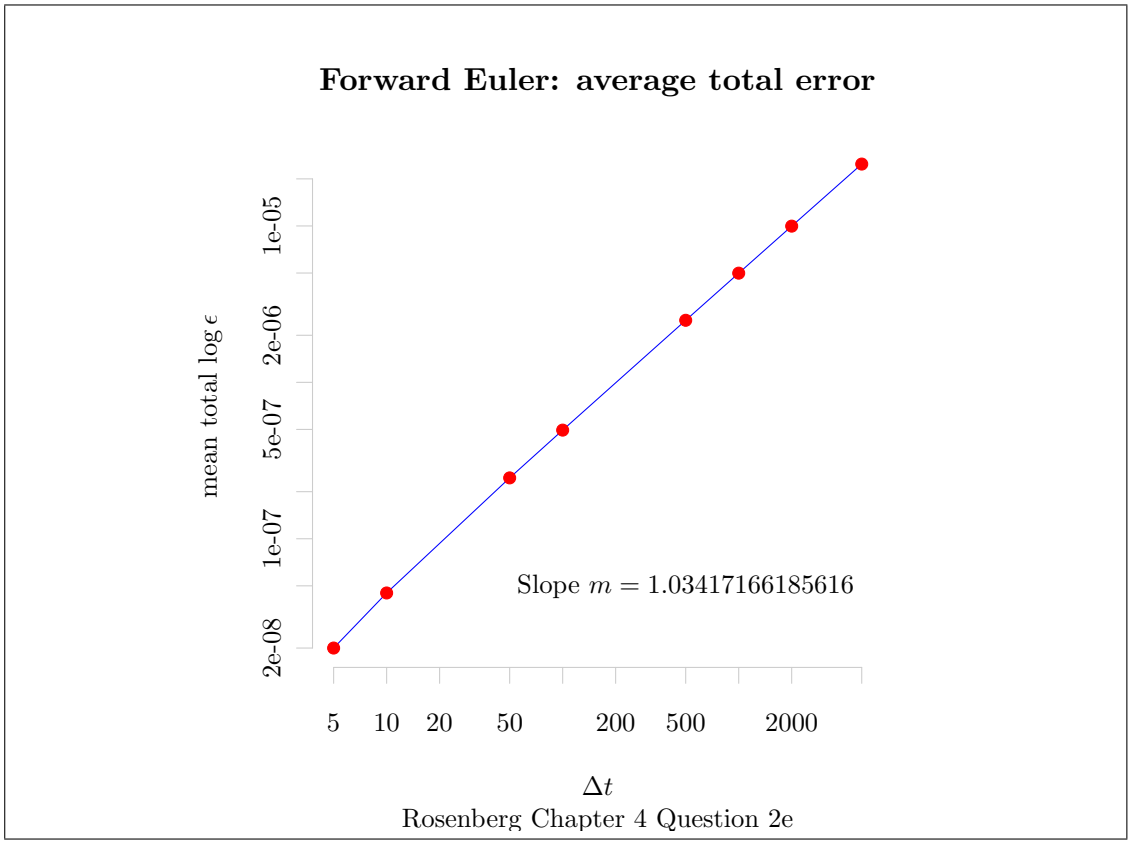
variable	value
S	10^{-2}
k_1	10^{-2}
k_2	10^{-4}
R_0	10^{-2}
C_0	0.00

```
> tSteps <- c(5, 10, 50, 100, 500, 1000, 2000, 5000, 10000)
> dotR <- function(R, S=1e-2, R0=1e-2, k1=1e-2, k2=1e-4) {
+   return((R * -k1 * S) + k2 * (R0 - R));
+ }
> runForwardEuler <- function(diffEq, tRange=c(0, 1e6),
+                             timeStepSize=100, R0=1e-2) {
+   tVals <- seq(from=tRange[1], to=tRange[2], by=timeStepSize);
+   rVals <- numeric(length=length(tVals));
+   rVals[1] <- R0;
+   for (ii in 1:length(tVals)) {
+     rVals[ii + 1] <- rVals[ii] + diffEq(R=rVals[ii] * timeStepSize);
+   }
+   return(list(t=tVals, r=rVals));
+ }
> result <- list();
> cols <- rainbow(n=length(tSteps), start=0.1, end=0.7);
> plot(c(0, 2e5), c(-0.01, 0.01), fg=grey(0.7), bty='n',
+      type='n', main='\\textbf{Forward Euler Approximate Solution}',
+      xlab='$t$', ylab='$R$', sub='Rosenberg Chapter 4 Problem 2d');
> abline(h=0, v=0);
> for (ii in 1:length(tSteps)) {
+   result[[ii]] <- runForwardEuler(dotR, timeStepSize=tSteps[ii])
+   lines(result[[ii]]$t, result[[ii]]$r[-1], col=cols[ii]);
+ }
> legend(12e4, y=0.001, bg='white', lwd=2, title='Step size',
+        col=cols, legend=paste('$\\Delta t =', as.character(tSteps), '$'))
```



(e) Use the analytical solution you found to find the total error of the method at every time step. Vary the time step over several orders of magnitude, and report the total error (averaged over the time course of the simulation). Plot the total error vs time step, as a log-log plot, and report the slope of the curve. This is the order of the method.

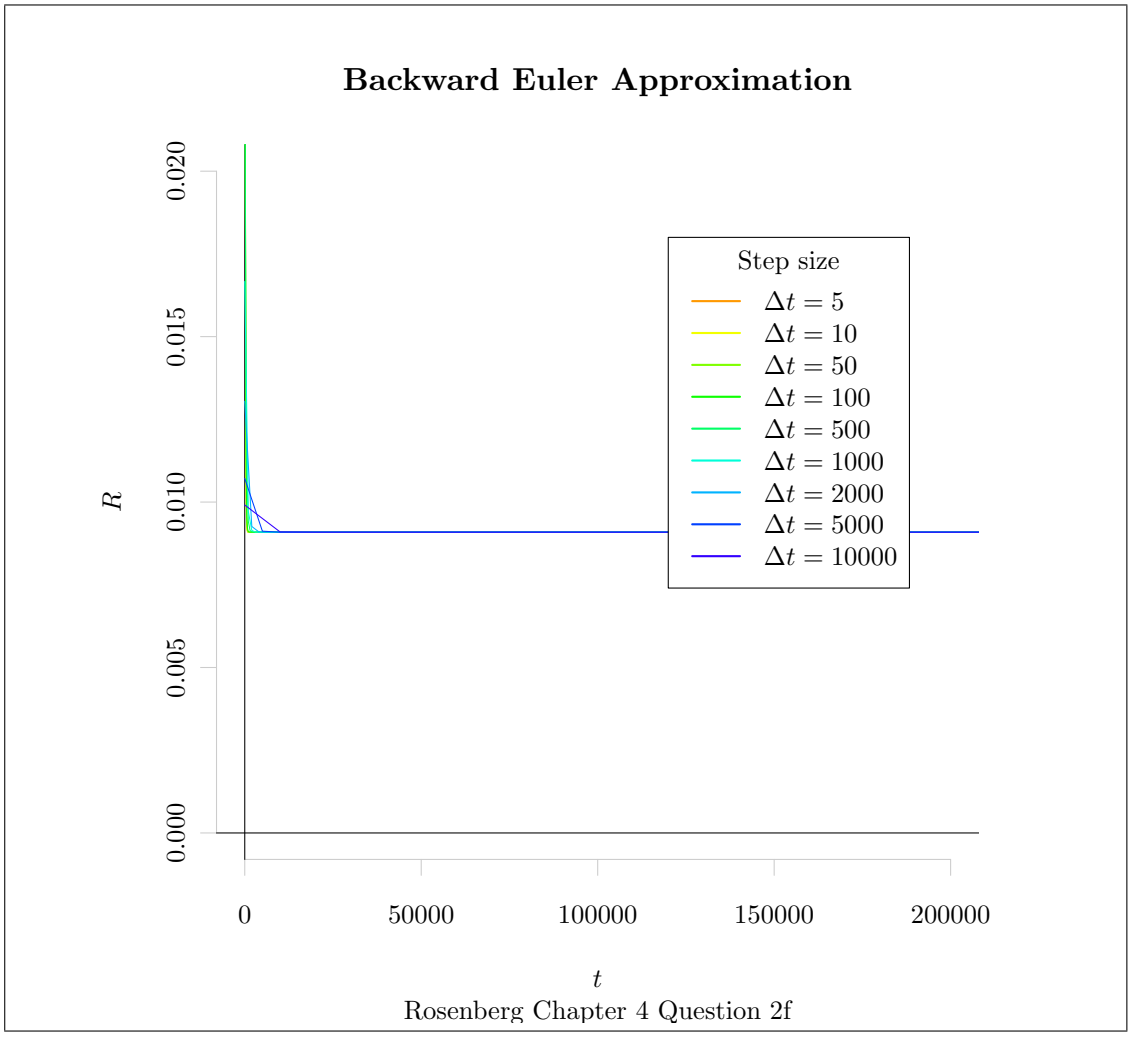
```
> runAnalytic <- function(t, S=1e-2, R0=1e-2, k1=1e-2, k2=1e-4) {
+   intConst <- (R0 * k2 - R0 * (k1 * S + k2)) / (k2 + k1 * S);
+   rVal <- (R0 * k2)/(k1 * S + k2) - intConst * exp(-t * (k2 + k1 * S));
+   return(rVal);
+ }
> tSteps <- c(5, 10, 50, 100, 500, 1000, 2000, 5000)
> errors <- numeric(length=length(tSteps))
> for (ii in 1:length(tSteps)) {
+   fEulerEst <- runForwardEuler(dotR, timeStepSize=tSteps[ii]);
+   analyticVals <- runAnalytic(t=fEulerEst$t);
+   totalError <- abs(fEulerEst$r[-1] - analyticVals)[length(analyticVals)] /
+                                       length(analyticVals)
+   errors[ii] <- totalError[length(totalError)];
+ }
> plot(tSteps, errors, type='l', log=c('xy'),
+      bty='n', fg=grey(0.8), xlab='$\\Delta t$',
+      ylab='mean total $\\log \\epsilon$',
+      sub='Rosenberg Chapter 4 Question 2e',
+      main='\\textbf{Forward Euler: average total error}',
+      col='blue');
> points(tSteps, errors, pch=19, col='red');
> slope <- mean(diff(log(errors)) / diff(log(tSteps)));
> text(x=500, y=5e-08,
+      paste('Slope $m=', as.character(slope), '$.', sep=''))
```



(f) Implement the Backward Euler method by writing down the numerical scheme with the future value of R in the derivative function, and then solving for the future value of R. Use the same parameters as above, and find the solution for the same time scale

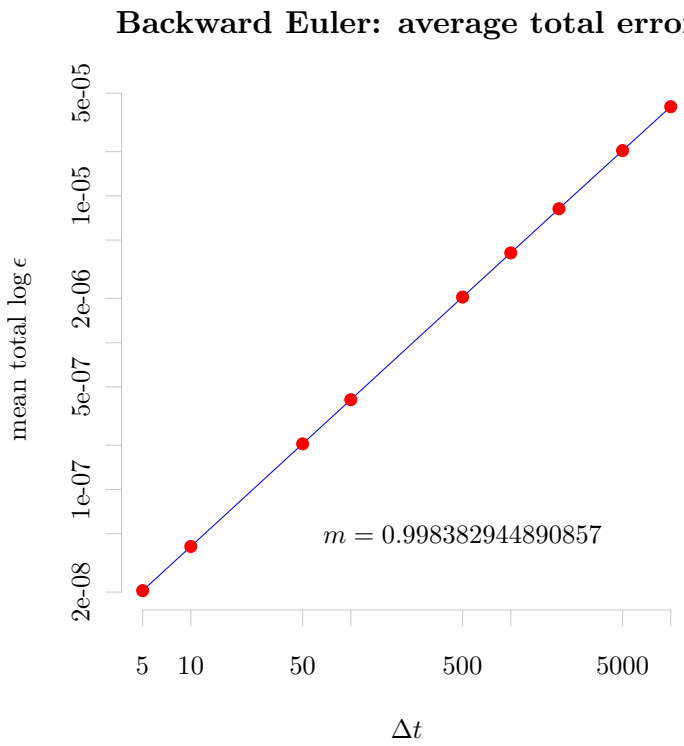
$$
\begin{aligned}
R_{n+1} &= R_n + \Delta t f(R_{n+1}, t_{n+1}) \\
&= R_n + \Delta t \left(-R_{n+1}(k_1 S + k_2) + k_2 R_0\right) \\
R_{n+1}\left(1 + \Delta t(k_1 S + k_2)\right) &= R_n + \Delta t k_2 R_0 \\
R_{n+1} &= \frac{R_n + \Delta t k_2 R_0}{1 + \Delta t(k_1 S + k_2)}
\end{aligned}
$$

```
> bEulerStepFun <- function(R, stepSize, S=10e-2, R0=10e-2,
+                           k1=10e-2, k2=10e-4) {
+   return( (R0 * stepSize * k2 + R) / (S * stepSize * k1 + stepSize * k2 + 1) )
+ }
> runBackwardEuler <- function(implicitFun, tRange=c(0, 1e6),
+                              timeStepSize=100, R0=10e-2) {
+   tVals <- seq(from=tRange[1], to=tRange[2], by=timeStepSize);
+   rVals <- numeric(length=length(tVals));
+   rVals[1] <- R0;
+   for (ii in 1:length(tVals)) {
+    rVals[ii + 1] <- implicitFun(R=rVals[ii], stepSize=timeStepSize);
+   }
+   return(list(t=tVals, r=rVals));
+ }
> result <- list();
> plot(c(0, 2e5), c(0, 0.02), type='n',
+      xlim=c(0, 2e5), bty='n', fg=grey(0.8), xlab='$t$',
+      ylab='$R$', sub='Rosenberg Chapter 4 Question 2f',
+      main='\\textbf{Backward Euler Approximation}');
> abline(h=0, v=0);
> tSteps <- c(5, 10, 50, 100, 500, 1000, 2000, 5000, 10000);
> cols <- rainbow(n=length(tSteps), start=0.1, end=0.7);
> for (ii in 1:length(tSteps)) {
+   result[[ii]] <- runBackwardEuler(bEulerStepFun, timeStepSize=tSteps[ii])
+   lines(result[[ii]]$t, result[[ii]]$r[-1], col=cols[ii]);
+ }
> legend(x=12e4, y=0.018, bg='white', lwd=2, title='Step size',
+        col=cols, legend=paste('$\\Delta t =', as.character(tSteps), '$'))
```



(g) Repeat the error analysis above for the Backward Euler algorithm: find the mean total error for different values of the time step, and plot the results as a log-log graph.

```
> tSteps <- c(5, 10, 50, 100, 500, 1000, 2000, 5000, 10000)
> errors <- numeric(length=length(tSteps))
> for (ii in 1:length(tSteps)) {
+   bEulerEst <- runBackwardEuler(bEulerStepFun, timeStepSize=tSteps[ii]);
+   analyticVals <- runAnalytic(t=bEulerEst$t);
+   totalError <- abs(bEulerEst$r[-1] - analyticVals)[length(analyticVals)] /
+                                        length(analyticVals)
+   errors[ii] <- totalError[length(totalError)];
+ }
> plot(tSteps, errors, type='l', log=c('xy'),
+      bty='n', fg=grey(0.8), xlab='$\\Delta t$',
+      ylab='mean total $\\log \\epsilon$',
+      main='Rosenberg Chapter 4 Question 2g',
+      col='blue');
> slope <- mean(diff(log(errors)) / diff(log(tSteps)));
> points(tSteps, errors, pch=19, col='red');
> text(500, 5e-08, paste('$m=', as.character(slope), '$', sep=''))
```



Rosenberg Chapter 4 Question 2g