Introduction to computational programming
Chapter 4 Exercise
Numerical methods for ODEs

David M. Rosenberg
University of Chicago
Committee on Neurobiology

Version control information:
Last changed date: 2009-10-26 10:44:14 -0500 (Mon, 26 Oct 2009)
Last changes revision:176
Version: Revision 176
Last changed by: David M. Rosenberg

October 29, 2009

Part I
Tutorial

Overview

1 Forward Euler

Pseudocode for Forward Euler

```
1. Define the derivative function for the ODE $f(x,t)$
2. Choose the step size $\Delta t$, and number of iterations $Niter$
3. Choose an initial point $(t_0, x_0)$, and initialize two arrays, $T$ and $X$
4. Let $i = 1$, and repeat the following while $i < Niter$:
    (a) store $X[i] + \Delta t * f(X[i], T[i])$ in $X[i+1]$
    (b) store $T[i] + \Delta t$ in $T[i+1]$
    (c) increase $i$ by 1
```

We can rewrite this as:

```
define F(x : REAL, t : REAL)            // Derivative of x(t)
var dt : REAL
var n_iter: INT
var t_init, x_init: REAL
var X, T : REAL ARRAY [0..n_iter]
T[0] ← t_init
X[0] ← x_init

for i : INT 0 to n_iter
begin
    X[i] ← X[i-1] + dt * F(X[i-1], T[i-1])
    T[i] ← T[i-1] + dt
end
```

Pseudocode listing 1 – Forward Euler

Let us consider the differential equation $\dot{x} = -ax$. This gives us

$$\begin{aligned} f(x,t) &= -ax \\ y_{n+1} &= y_n + \Delta t \cdot f(y_n, t_n) \\ &= y_n + \Delta t \cdot -ay_n \\ &= y_n(1 - a\Delta t) \end{aligned}$$

In order to ensure a stable solution, we will consider

$$\begin{aligned} y_{n+1} &= (x_n + \epsilon)(1 - a\Delta t) \\ &= x_n(1 - a\Delta t) + \epsilon(1 - a\Delta t) \end{aligned}$$

To keep the error term from growing, we set

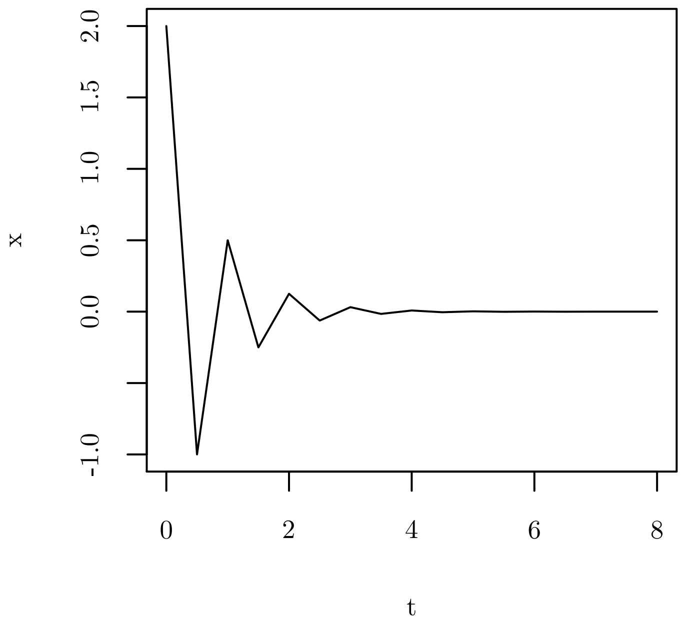
$$\begin{aligned} |1 - a\Delta t| &< 1 \\ -1 < 1 - a\Delta t &< 1 \\ 2 > a\Delta t &> 0 \\ 0 < \Delta t &< \frac{2}{a} \end{aligned}$$

Thus if we set $\Delta t < \frac{2}{a}$ when $a > 0$, we can be assured that our numerical solution will be stable. Let us further consider this model when $s = 3$ with initial condition $x_0 = 2, t_0 = 0$. We can then reasonably set our step size to $\Delta t = \frac{1}{2}$. Thus, we could approximate the value of x at $t = 8$ by:

Thus we can see that the numerical solution via the forward Euler algorithm with the given conditions is (approximately) zero for $t = 8$.

```
> f <- function(x, t) {
+   return (-3 * x);
+ }
> forwardEuler <- function(f, max_iter,
   step, x0, t0) {
+   xvals <- tvals <- numeric(length=(
    max_iter+1));
+   xvals[1] <- x0;
+   tvals[1] <- t0;
+   for (ii in 1:max_iter) {
+     xvals[ii+1] <- xvals[ii] +
+         step * f(xvals[ii], tvals[ii]);
+     tvals[ii+1] <- tvals[ii] + step
+   }
+   return(data.frame(t=tvals, x=xvals));
+ }
> step_size <- 1/2;
> nSteps <- 8 / step_size;
> feuler1 <- forwardEuler(f, nSteps, 0.5,
     2, 0);
> feuler1$x[length(feuler1$x)];

[1] 3.051758e-05
```



2 Backward Euler

Pseudocode for Backward Euler

```
1. Define the derivative function for the ODE $f(x,t)$
2. Choose the step size $\Delta t$, and number of iterations $Niter$
3. Choose an initial point $(t_0, x_0)$, and initialize two arrays, $T$ and $X$
4. Let $i = 1$, and repeat the following while $i < Niter$:
    (a) store $T[i] + \Delta t$ in $T[i+1]$
    (b) solve $X[i] + \Delta t * f(X[i+1], T[i+1]) = X[i+1]$ for $X[i+1]$ and store the value
    (c) increase $i$ by 1
```

```
define F(x : REAL, t : REAL)          // Derivative of x(t)
var dt : REAL
var n_iter: INT
var t_init, x_init: REAL
var X, T : REAL ARRAY [0..n_iter]
T[0] ← t_init
X[0] ← x_init

for i : INT 0 to n_iter
begin
    T[i] ← T[i-1] + dt
    X[i] ← (solve X[i] - dt f[X[i], T[i]] = X[i-1], X[i])
end
```

Pseudocode listing 2 – Backward Euler

$$
\begin{aligned}
f(x,t) &= -ax \\
y_{n+1} &= y_n + \Delta t \cdot f(y_{n+1}, tn+1) \\
y_{n+1} &= y_n + \Delta t(-a y_{n+1}) \\
y_{n+1}(1 + a\Delta t) &= y_n \\
y_{n+1} &= \frac{y_n}{1 + a\Delta t}
\end{aligned}
$$

$$
\begin{aligned}
y_n &= x_n + \epsilon \\
y_{n+1} &= \frac{x_n + \epsilon}{1 + a\Delta t} \\
&= \frac{x_n}{1 + a\delta t} + \frac{\epsilon}{1 + a\Delta t} \\
|\frac{1}{1 + a\Delta t}| &< 1 \\
-1 &< \frac{1}{1 + a\Delta t} < 1 \text{ Suppose a} > 0 \\
\frac{1}{1 + a\Delta t} &< 1 \\
1 + a\Delta t &> 1 \\
a\Delta t &> 0
\end{aligned}
$$

Thus the solution will always be stable.

Part II

Exercises

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.

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

   (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?

   (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.

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.

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

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

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

   (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.

(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.

(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.