PH2150A Scientific Computing Skills Team Project
Warm up exercise for numerical solution of differential equations

For the projects involving numerical solutions to differential equations (i.e. the pendulum and planetary motion problems), the following warm-up exercise should prove useful. Consider a sample of N radioactive nuclei. The time derivative of N is proportional to the number of nuclei present, i.e.

$$\frac{dN}{dt} = -\frac{N}{\tau} \ . \tag{1}$$

Here τ, the mean lifetime of the nuclei, is a specified constant. Suppose we want to find $N(t)$, given that at $t = 0$, N_0 nuclei are present. For this example, the exact solution is easy to find. It is

$$N(t) = N_0 e^{-t/\tau} \ . \tag{2}$$

Suppose, however, that we wanted to find the solution numerically. This can be done by using the solution at the time t and the differential equation (1) to write down an approximate expression for N at a slightly later time $t + \Delta t$. This is obtained by expanding $N(t)$ to first order in a Taylor series about t, giving

$$\begin{aligned} N(t+\Delta t) &\approx N(t) + \frac{dN(t)}{dt}\Delta t \\ &= N(t) - \frac{N(t)}{\tau}\Delta t \ . \end{aligned} \tag{3}$$

Consider taking small steps in time of length Δt, labelled by an index n, so that

$$t_n = n\Delta t, \quad n = 0, 1, \dots \ , \tag{4}$$

and

$$N_n = N(t_n) \ . \tag{5}$$

The number of nuclei present at a time $t_{n+1} = t_n + \Delta t$ is therefore

$$\begin{aligned} N_{n+1} &\approx N_n + \left(\frac{dN}{dt}\right)_n \Delta t \\ &= N_n - \frac{N_n}{\tau}\Delta t \ . \end{aligned} \tag{6}$$

Equation (6) can easily be implemented on a computer to determine $N(t)$, as in the following piece of Java code:

```java
/**
 *  Sample program to illustrate the Euler method for numerical solution
 *  of a first-order differential equation.
 *  Author:  Glen Cowan, RHUL Physics Dept.
 *  Date:    7 November 2001
 */

public class Nuclei
{
  public static void main (String[] argv)
  {
    int numstep = 500;
    double dt = 0.01;
    double N = 100000.;
    double tau = 1.;
    double t = 0.;
    for(int n = 0; n < numstep; n++)
    {
      t = (double)n * dt;
      N = N * (1 - dt/tau);
      System.out.println(t  + "    " + N);
    }
  }
}
```

This shows how to solve a single 1st order differential equation using the Euler method. You can make a plot of the values produced and compare them to the exact solution given by equation (2). Your next step is to generalize this technique in order to solve a system of n 1st order equations, or equivalently, a single nth order equation. Then the Euler method can be extended to the Runge-Kutta method, as described in your lab scripts.

G. Cowan
19 November, 2001