

NEEP 602 -- Engineering Problem Solving II
Spring 2005
Exercise 1

A single, first-order equation

We can solve these types of problems using a wide variety of computer tools, including Excel, but let's use Matlab in this case. A generic first order ordinary differential equation can be written as follows:

$$\frac{dy}{dt} = f(t, y)$$
$$y(0) = A$$

where f is an arbitrary, nonlinear function of the dependent and independent variables and A is a constant. Our first model problem of this type is a model for the velocity of an object falling in air. There is a gravitational acceleration term and a drag term due to air resistance. The differential equation is

$$m \frac{dy}{dt} = mg - Dy^{2.5} = m f(t, y)$$
$$y(0) = 0$$

where m is the mass of the object, y is the velocity, and D is the drag coefficient.

Matlab uses an adaptive Runge-Kutta algorithm to solve such equations. As you'll recall, the fourth-order Runge-Kutta algorithm is:

$$k_1 = h * f(t_n, y_n)$$
$$k_2 = h * f(t_n + \frac{h}{2}, y_n + \frac{k_1}{2})$$
$$k_3 = h * f(t_n + \frac{h}{2}, y_n + \frac{k_2}{2})$$
$$k_4 = h * f(t_n + h, y_n + k_3)$$
$$y_{n+1} = y_n + \frac{k_1 + 2(k_2 + k_3) + k_4}{6}$$

where h is the time step. If we know y at a particular time t , then we can use this algorithm to estimate y at a time $t+h$. We can then just apply a series of these steps to determine y at any desired time. Matlab is a bit more sophisticated in that it adjusts the time step h as it goes along in order to achieve a desired accuracy.

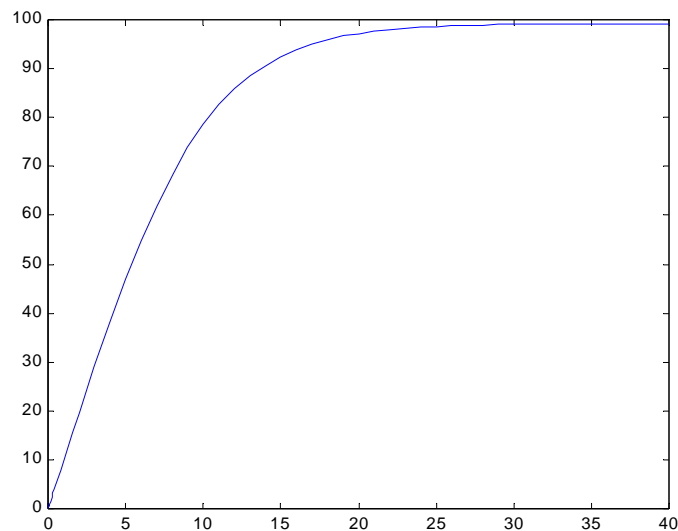
In order to solve the above equation, we merely need to provide Matlab with a function which gives $f(t,y)$. Given an initial value for y , Matlab will then solve for y over the desired time interval. To provide the function, we create an m-file called `rkfunc.m`. This file will contain the following Matlab statements:

```
function f=rkfunc(t,y)
velocity=y;
mass=10;
D=0.001;
g=9.81;
f=g-D*velocity^2.5/mass;
```

Here f returns the value for the derivative (dy/dt) at the given values of t and y . These statements must be saved in a file called `rkfunc.m`. Then, we can type the following into the Matlab command window and obtain a plot of the solution for $y(t)$.

```
vinitial=0;
timeinterval=[0 40];
[t y]=ode45('rkfunc',timeinterval,vinitial);
plot(t,y)
```

Your plot should look like this (it shows the velocity as a function of time over the requested time interval: $0 < t < 40$):



Two first-order equations

Sometimes we need to solve two coupled differential equations. Our next exercise is to write a routine for solving such a system, in this case a typical predator-prey problem. If we let z be the number of rabbits in an environment, and y be the number of foxes, then we can model the evolution of the animal populations by the following equations:

$$\frac{dz}{dt} = z - \frac{zy}{100} = f(t, y, z)$$

$$\frac{dy}{dt} = -y + \frac{zy}{50} = g(t, y, z)$$

The first equation gives the rate of change of the rabbit population. There is a growth term (z) and a death term which depends on the product of z and y , implying that the more foxes there are, the higher the death rate of rabbits. The second equation gives the rate of change of the fox population. There is a death term (more foxes implies more competition for a limited food supply) and a growth term which depends on the zy product (more rabbits means more available food and thus an increase in the number of foxes).

Once we assume an initial number of foxes and rabbits (i.e. initial values for z and y), we can then solve this system to plot z and y as a function of time. To do this, we need a Runge-Kutta algorithm for two coupled equations. This algorithm is:

$$k_1 = h * g(t_n, y_n, z_n)$$

$$l_1 = h * f(t_n, y_n, z_n)$$

$$k_2 = h * g(t_n + \frac{h}{2}, y_n + \frac{k_1}{2}, z_n + \frac{l_1}{2})$$

$$l_2 = h * f(t_n + \frac{h}{2}, y_n + \frac{k_1}{2}, z_n + \frac{l_1}{2})$$

$$k_3 = h * g(t_n + \frac{h}{2}, y_n + \frac{k_2}{2}, z_n + \frac{l_2}{2})$$

$$l_3 = h * f(t_n + \frac{h}{2}, y_n + \frac{k_2}{2}, z_n + \frac{l_2}{2})$$

$$k_4 = h * g(t_n + h, y_n + k_3, z_n + l_3)$$

$$l_4 = h * f(t_n + h, y_n + k_3, z_n + l_3)$$

$$y_{n+1} = y_n + \frac{k_1 + 2(k_2 + k_3) + k_4}{6}$$

$$z_{n+1} = z_n + \frac{l_1 + 2(l_2 + l_3) + l_4}{6}$$

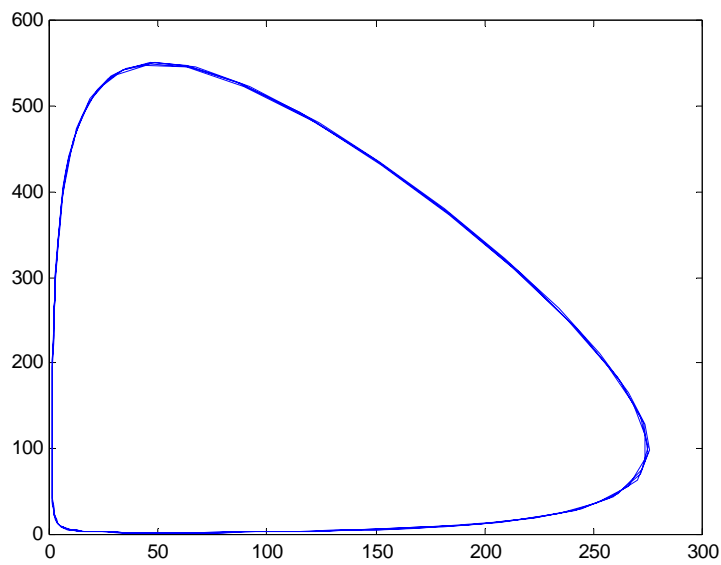
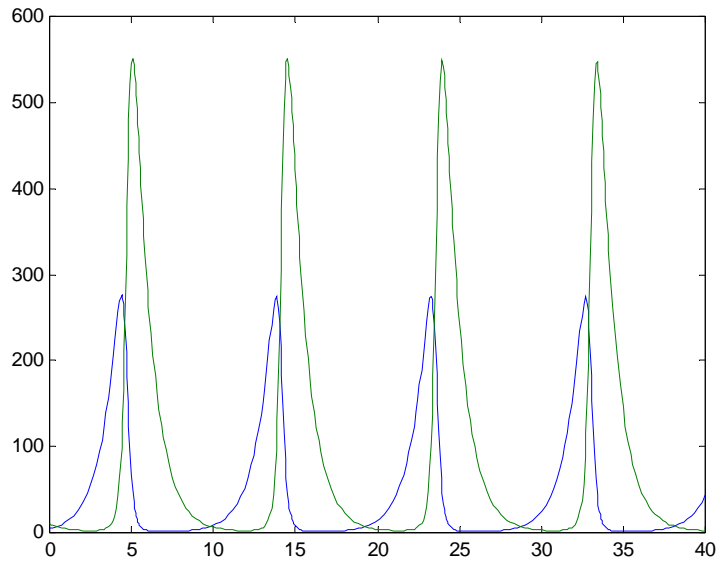
The Matlab file to define the equations for this case is:

```
function f=rkfunc(t,w)
y=w(1);
z=w(2);
f=[-y+z*y/50; z-z*y/100];
```

Note that the variable w is a vector containing values for both y and z and that f now returns both functions defining our system of equations. The commands to solve the equations are:

```
timeinterval=[0 40];  
initial=[5 10];  
[t w]=ode45('rkfunc',timeinterval,initial);  
plot(t,w)  
plot(w(:,1),w(:,2))
```

This uses initial values of 5 rabbits and 10 foxes. The two plots we obtain are:



The first plot shows y and z as a function of time and the second plot shows y vs. z .

A system of three equations

Lorenz simplified a complicated model of the weather down to a system of three equations:

$$\begin{aligned}\frac{dx}{dt} &= -3(x - y) \\ \frac{dy}{dt} &= -xz + rx - y \\ \frac{dz}{dt} &= xy - z\end{aligned}$$

Solve this system using a value of $r=17$ and any initial values of the dependent variables that you would like. Compare to the solution for $r=26$ using the same initial values.

Second-order equations

To solve second order equations, we convert them to a system of two first order equations and then solve as described above. Suppose we want to solve:

$$\begin{aligned}\frac{d^2 y}{dt^2} &= t + y + y \frac{dy}{dt} \\ y(0) &= A \\ \frac{dy}{dt}(0) &= B\end{aligned}$$

To convert this into two first order equations, we just introduce an intermediate variable as:

$$z = \frac{dy}{dt}$$

Then we obtain

$$\frac{dz}{dt} = \frac{d^2 y}{dt^2}$$

and our second order equation becomes:

$$\begin{aligned}\frac{dy}{dt} &= z \\ \frac{dz}{dt} &= t + y + yz \\ y(0) &= A \\ z(0) &= B\end{aligned}$$

This can then be solved just as described above.