Solving systems of ODEs with SCILAB's function 'ode'

By Gilberto E. Urroz, April 2014

Recap on the operation of SCILAB function 'ode'

Function 'ode' in SCILAB can be used to solve a single, first-order, ODE of the form: dy/dx = f(x,y), subject to initial conditions y = y0 at x = x0. When using a Runge-Kutta, 4th-order, method for the solution, the function call is:

```
y = ode("rk",y0,x0,x,f)
```

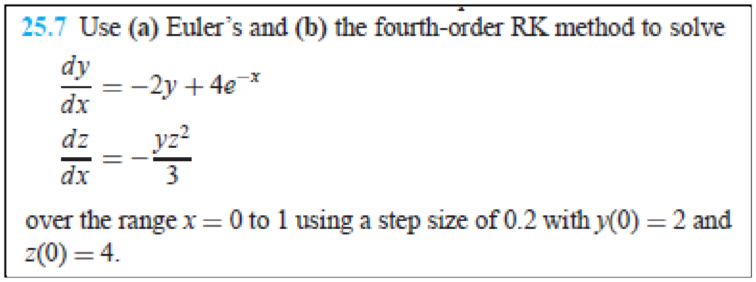
where x is a vector of values of the independent variable. The vector x is defined before the call to 'ode', by using, for example, x = [x0:h:xn], where h is the increment in x, and xn is the upper limit of the solution domain x0 < x < xn.

Here's an example of the use of SCILAB function 'ode' to solve the ODE dy/dx = sin(y), with initial conditions y = 1 at x = 0, in the range 0 < x < 5, with increment h = 0.5. Try the following commands in a SCILAB console:

```
--> function [z] = f(x,y)
-->   z = sin(y);
--> endfunction;

--> x = [0:0.5:5];

--> y = ode("rk",1,0,x,f);

-->[x' y']
 ans  =

   0.    1.
   0.5   1.466404
   1.    1.956295
   1.5   2.3660758
   2.    2.6559113
   2.5   2.8433129
   3.    2.9598255
   3.5   3.0311532
   4.    3.0745647
   4.5   3.1009285
   5.    3.1169264
```

Function 'ode' for solving a system of two, first-order, ODEs

Suppose we want to solve the system of equations proposed in problem 25.7 of the Chapra/Canale, 6th-edition, textbook, namely:



25.7 Use (a) Euler's and (b) the fourth-order RK method to solve

$$\frac{dy}{dx} = -2y + 4e^{-x}$$

$$\frac{dz}{dx} = -\frac{yz^2}{3}$$

over the range $x = 0$ to 1 using a step size of 0.2 with $y(0) = 2$ and $z(0) = 4$.

Taking $y_1 = y$ and $y_2 = z$, the system of equations can be converted into a single ODE:

$$\frac{dY}{dx} = F(x, y) \quad \text{subject to} \quad Y = Y_0 \text{ at } x = x_0$$

by using: $Y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$, $F(x, Y) = \begin{pmatrix} f_1(x, Y) \\ f_2(x, Y) \end{pmatrix} = \begin{pmatrix} -2 \cdot y_1 + 4 \cdot \exp(-x) \\ -\dfrac{y_1 \cdot y_2^{\,2}}{3} \end{pmatrix}$, $Y_0 = \begin{pmatrix} 2 \\ 4 \end{pmatrix}$, and $x_0 = 0$

Using SCILAB, we can use function 'ode' to solve this problem by specifying:

Y = ode("rk",Y0,x0,x,F)

The solution domain, vector x, can be specified, for this case as:

x = [0.0:0.2:1.0]

While the SCILAB function that defines the right-hand side of the vector ODE will be defined as follows:

```
function [Z] = F(x,Y)
   Z(1) = -2*Y(1)+4*exp(-x);
   Z(2) = -Y(1)*Y(2)^2/3;
end function
```

A SCILAB script for solving a system of two, first-order, ODEs using 'ode'

To check the solution in SCILAB, type, save, and execute the following script:

```
function [Z]=F(x,Y)
  Z(1)=-2*Y(1)+4*exp(-x);
  Z(2)=-Y(1)*Y(2)^2/3;
endfunction
x = [0.0:0.2:1.0];
x0 = 0;
Y0 = [2;4];
Y = ode("rk",Y0,x0,x,F);
disp("x   y1   y2")
disp([x' Y']);
```

The corresponding SCILAB output is:

```
x   y1   y2

  0.    2.        4.
  0.2   1.9342829   2.6191791
  0.4   1.7826223   1.9762211
  0.6   1.5928581   1.6164249
  0.8   1.3935228   1.3923823
  1.    1.2008472   1.2428446
```

Some notes on function F(x,Y):

(1) When defining a function such as F(x,Y), above, the second argument, Y, is a column vector (in this case) of 2 elements. If you need to evaluate this function for some reason, you could use, for example:

```
--> F(2.0,[5.0,-2.0])
 ans  =

  - 9.4586589
  - 6.6666667
```

In this case, x = 2.0, and Y = [5.0;-2.0]. The semi-colon separating the two elements of Y defines a column vector. Notice that the result of the function evaluation is also a column vector.

(2) Here is another way to evaluate function F(x,Y):

```
--> xx = 2.5; YY = [2.0,1.0]; F(xx,YY)
 ans  =

 - 3.67166
 - 0.6666667
```

Some notes on the results from the 'ode' call:

(1) The result from the evaluation of function 'ode' in the script above, is the following matrix:

```
--> Y
 Y  =

   2.   1.9342829   1.7826223   1.5928581   1.3935228   1.2008472
   4.   2.6191791   1.9762211   1.6164249   1.3923823   1.2428446
```

whereas, the input vector x is:

```
--> x
 x  =

   0.   0.2   0.4   0.6   0.8   1.
```

(2) The command **disp([x' Y'])** puts together the transpose of x and the transpose of Y into the matrix:

```
--> disp([x' Y'])

   0.    2.         4.
   0.2   1.9342829   2.6191791
   0.4   1.7826223   1.9762211
   0.6   1.5928581   1.6164249
   0.8   1.3935228   1.3923823
   1.    1.2008472   1.2428446
```

Extracting elements from a matrix in SCILAB

As indicated above, the results of the solution for the system of two ODEs, shown earlier, is a matrix Y of two rows and six columns. The first row in Y represents the values of y1, while those in the second column of Y are the values of y2. In this section, we'll learn how to extract those rows from the matrix Y, but before we get to that point, we'll learn how to manipulate elements of a matrix in SCILAB.

EXAMPLE - In the SCILAB console, create the following random matrix M:

```
--> M = int(100*rand(5,4))
 M  =

   93.   56.   43.   4.
   21.   48.   26.   48.
   31.   33.   63.   26.
   36.   59.   40.   41.
   29.   50.   91.   28.
```

The command **rand(5,4)** produces a matrix of 5 rows and 4 columns whose elements are random numbers in the range [0,1]. By multiplying 'rand(5,4)' times 100, then the elements will now be values in the range [0,100], but they will include decimal numbers. To convert those numbers into integers, thus eliminating the decimal part, you can use function 'int'.

Thus, in creating that random matrix M we used the following commands:

```
* rand(n,m): produces a random matrix of n rows and m columns, whose elements are uniform-
             ly-distributed random numbers in the range [0,1]
* int(x):    extracts the integer part of the floating-point number x
```

Next, we show the commands needed to extract individual elements, rows, columns, and sub-matrices out of matrix M:

```
--> M(1,1) //extracts element in row 1, column 1
 ans  =

   93.

--> M(3,2) // extracts element in row 3, column 2
 ans  =

   33.

--> M(2,:) // extracts all elements in row 2
 ans  =

   21.   48.   26.   48.

--> M(:,3) // extracts all elements in column 3
 ans  =

   43.
   26.
   63.
   40.
   91.

--> M(2:4,2:3) // extracts submatrix in rows 2 to 4, columns 2 to 3
 ans  =

   48.   26.
   33.   63.
   59.   40.
```

EXTRACTING ROWS FROM THE ODE SYSTEM SOLUTION - Thus, to extract the individual rows out of matrix Y, obtained earlier in solving the system of two ODEs, we can use, for example:

```
--> y1 = Y(1,:)
 y1  =

   2.   1.9342829   1.7826223   1.5928581   1.3935228   1.2008472

--> y2 = Y(2,:)
 y2  =

   4.   2.6191791   1.9762211   1.6164249   1.3923823   1.2428446
```

After extracting the individual rows of matrix Y, we could use them to show the solutions to the system of two ODEs as separate tables, e.g.,

```
--> [x' y1']
 ans  =

   0.    2.
   0.2   1.9342829
   0.4   1.7826223
   0.6   1.5928581
   0.8   1.3935228
   1.    1.2008472
```

```
--> [x' y2']
 ans  =

   0.    4.
   0.2   2.6191791
   0.4   1.9762211
   0.6   1.6164249
   0.8   1.3923823
   1.    1.2428446
```

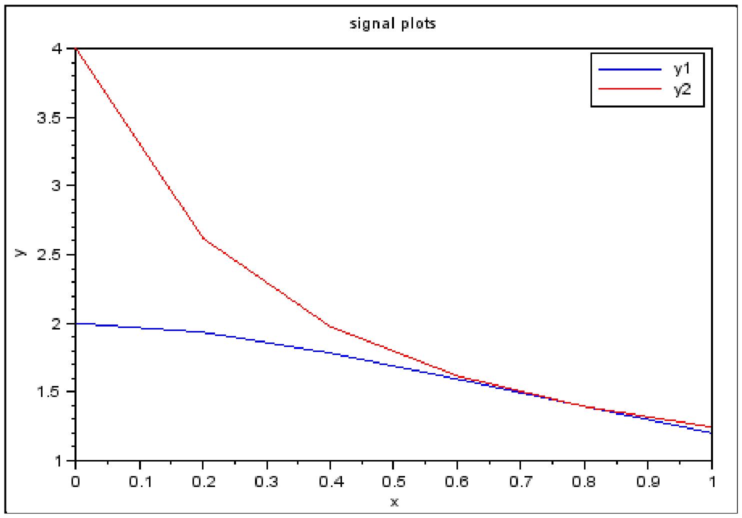
Plotting signals and phase portraits

When you solve a system of ODEs (such as in the example above where we used SCILAB function 'ode' to solve for Y, given x), plots of the individual dependent variables, such as y1 and y2, as functions of the independent variable, such as x, are called signal plots). On the other hand, plots of the individual dependent variables, with respect to one other dependent variable, are called phase portraits.

For the results of the solution found above, after extracting the rows y1 and y2 out of the solution matrix, Y, the signal plots can be shown in a single graph by using:

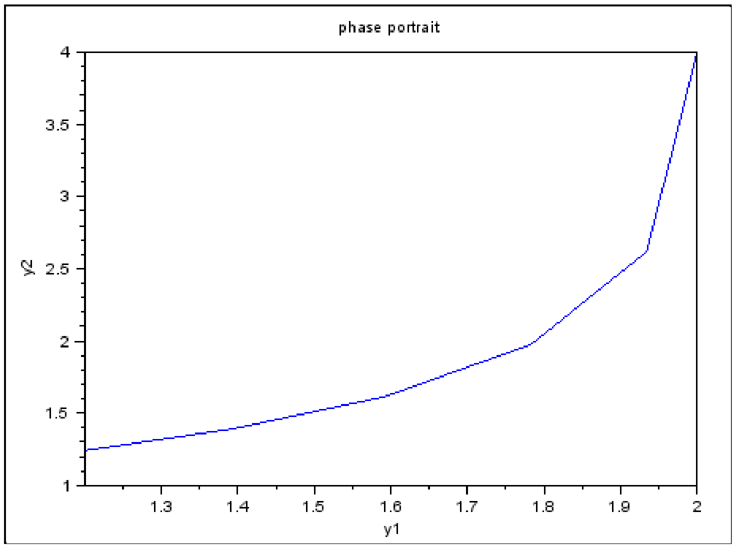
```
--> plot(x,y1,'-b',x,y2,'-r');legend('y1','y2');xtitle('signal plots','x','y');
```

The resulting signal plot is shown next:



Since there are only two signals, y1 and y2, in this case, there is a single phase portrait to be produced, i.e., the plot of y2 vs. y1 (or y1 vs. y2). Here we show the phase portrait of y2 vs. y1:

```
--> scf(); plot(y1,y2);xtitle('phaseportrait','y1','y2');
```



NOTES:
(1) Function 'scf();' produces a new graphics window so that the phase portrait is not drawn on top of the existing signal plots graphics window. If you use 'scf(n);' where n is an integer, graphics window number n will be open for producing plots.
(2) If plotting a single pair of vectors, as in this phase portrait, you don't need to specify the type of plot. You could simply use the default (blue continuous line). However, you can still specify the plot characteristics if you want, e.g., plot(y1,y2,'--')
(3) If you are plotting as single set of vectors, you don't need a 'legend' command, but you could still specify one if you so desire, e.g., legend('y2 vs. y1', 2)

SCILAB script for solving ODE system and producing plots

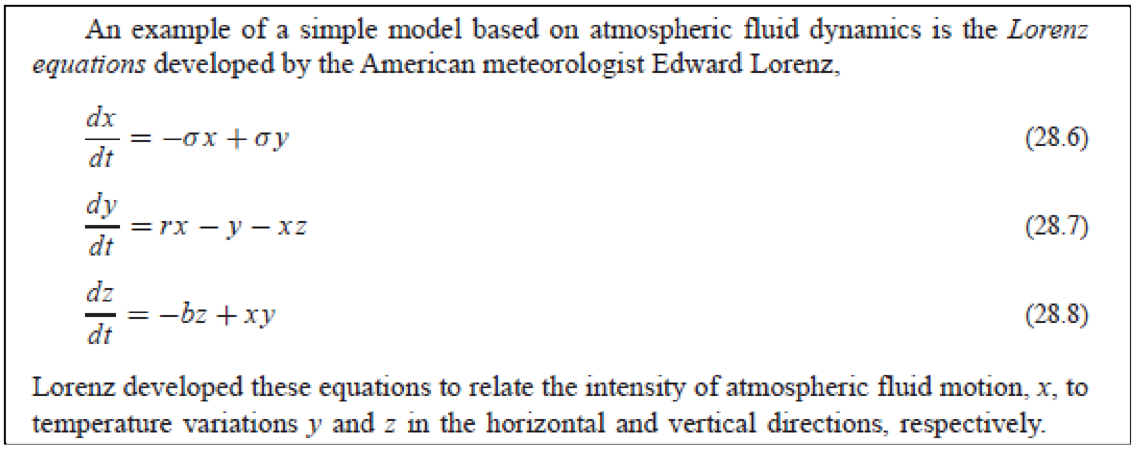
The following SCILAB script combines the commands used to produce the numerical solution to the system of ODEs, defined earlier, and those required to produce signal plots and phase portraits. I also added some comments to document the script:

```
//Script to solve dY/dx=F(x,Y), Y(x0)=Y0
function [Z]=F(x,Y)       //define F(x,Y)
  Z(1)=-2*Y(1)+4*exp(-x);
  Z(2)=-Y(1)*Y(2)^2/3;
endfunction
//ODE solution
x = [0.0:0.2:1.0];        //solution x domain
x0 = 0;                   //initial x value
Y0 = [2;4];               //initial Y value
Y = ode("rk",Y0,x0,x,F);  //solve ODE
disp("x   y1   y2")        //table title
disp([x' Y']);             //show result table
y1 = Y(1,:); y2 = Y(2,:);  //extract y1, y2C
//Produce signal plots
scf(); plot(x,y1,'b-',x,y2,'r-');
legend('y1','y2',1);
xtitle('signal plots','x','y');
//Produce phase portrait
scf(); plot(y1,y2);
xtitle('phase portrait','y1','y2');
```

As an exercise, you may want to modify this script so that the x increment is 0.1, rather than 0.2, and the maximum value of x is 10.0, and run the script again. To run the script: from the text editor (SCi-Notes) window, save the script, and use the menu option: 'Execute > ... file with no echo'

Assignment 6 - Problem [6]

Modify the script shown above so as to produce the numerical solution of the Lorenz equations shown below:



An example of a simple model based on atmospheric fluid dynamics is the *Lorenz equations* developed by the American meteorologist Edward Lorenz,

$$\frac{dx}{dt} = -\sigma x + \sigma y \tag{28.6}$$

$$\frac{dy}{dt} = rx - y - xz \tag{28.7}$$

$$\frac{dz}{dt} = -bz + xy \tag{28.8}$$

Lorenz developed these equations to relate the intensity of atmospheric fluid motion, x, to temperature variations y and z in the horizontal and vertical directions, respectively.

For your solution, use the values given in the textbook, namely, σ = 10, b = 2.666667, and r = 28, and use the initial conditions x = y = z = 5. Solve the system of equations in the interval 0 < t < 20, with an increment dt = 0.5, and produce a table of values of t, x, y, and z. Then, run the solution again, but now using an increment of dt = 0.01, and produce signal plots of x vs. t, y vs. t, and z vs. t, in the same set of axes. Also, produce phase portraits of x vs. y, y vs. z, and z vs. x, in separate plots. It is not necessary to report the solution for this second case (with dt = 0.01), only report the plots.

In your assignment report show the following:
(a) A copy of your script
(b) The results shown in the SCILAB script for dx = 0.5
(c) Printscreens of the single signal plot, and of the three phase portraits for dx = 0.01

Assignment 6 - Problem [7]

Solve problem 28.47 (see below) by modifying the script shown above that uses function 'ode' in SCILAB. First, you'll need to convert the second-order ODE described in the problem statement, into a system of two, first-order, ODEs, in order to use the modified script.

28.47 A forced damped spring-mass system (Fig. P28.47) has the following ordinary differential equation of motion:

$$m\frac{d^2x}{dt^2} + a\left|\frac{dx}{dt}\right|\frac{dx}{dt} + kx = F_o\sin(\omega t)$$

where x = displacement from the equilibrium position, t = time, $m = 2$ kg mass, $a = 5$ N/(m/s)^2, and $k = 6$ N/m. The damping term is nonlinear and represents air damping. The forcing function $F_o\sin(\omega t)$ has values of $F_o = 2.5$ N and $\omega = 0.5$ rad/sec. The initial conditions are

Initial velocity	$\frac{dx}{dt} = 0$ m/s
Initial displacement	$x = 1$ m

Solve this equation using a numerical method over the time period $0 \le t \le 15$ s. Plot the displacement and velocity versus time, and plot the forcing function on the same curve. Also, develop a separate plot of velocity versus displacement.

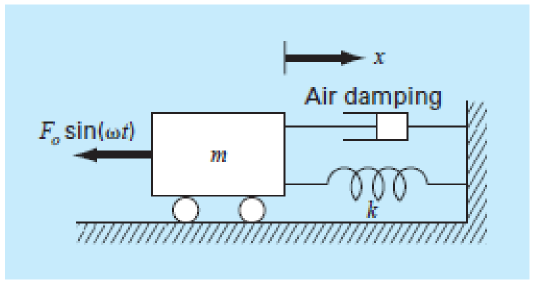


Figure P28.47

In your assignment report include the following:
(a) A write up showing how you converted the 2nd-order ODE into a system of two, 1st-order, ODEs.
(b) The SCILAB script used to produce the solution.
(c) The table of position x, and velocity dx/dt, as function of time t, produced by 'ode' (these results can be copied from the SCILAB console after the script is executed).
(d) The plot of x vs. t, dx/dt vs. t, required from the problem, in the same set of axes (i.e., signal plots).
(e) The phase portrait of dx/dt vs. x, required from the problem.