INTEGRATION OF ORDINARY DIFFERENTIAL EQUATIONS

% Many processes or systems involving the rates of change (derivatives) of variables can be described by one or more equations which describe the behavior of the derivatives as functions of the variable and independent variable (eg. time). These equations are called differential equations and are presented here treated to first order (first derivative) since higher order equations can be reduced to first order ones.

EXAMPLES of first order Differential Equations

$$dy/dt = ky$$

$$dy/dt = \sin(t)$$

% In both of the examples we say we have solved these equations when we solve for y as a function of t. ie. y =f(t) is a solution to above.

% Normally we move the differentials around and then integrate to get the solution. In all cases we usually need to know the starting values of t ,y or dy/dt to solve completely. ie. the initial conditions.

BASIC NUMERICAL APPROACH

% The common numerical approaches to the problem are to predict the slopes of the y=f(t) curve (The differential equation itself is a statement of the behavior of the derivative) starting at the initial conditions and then use the derived slope to predict the next value of y which is then used to predict the next slope which is used to predict the next value of y which is then used to continue the process until all values of y for the range of t are predicted.
% The different techniques used to obtain the curve y =f(t) differ in the ways slopes used to obtain y are calculated.

MATLAB ODE23() and ODE45() FUNCTIONS – RUNGE-KUTTA METHOD

% MATLAB provides two functions to solve differential equations, namely, **ode23()** and **ode45()** both of which use a technique called the **Runge-Kutta** method to determine slope averages and then predict values of y that satisfy the original differential equations. In using these **ode** functions we set up the differential equations as function M-files and then use the 'ode..' function to obtain the y and t values of the solution (Given as column vectors)
% The **ode..()** functions need several arguments as seen in
» help ode45

ODE45 Integrate a system of ordinary differential equations using 4th and 5th order Runge-Kutta formulas. See also ODE23 and ODEDEMO.M.

[T,Y] = ODE45('yprime', T0, Tfinal, Y0)

integrates the system of ordinary differential equations described by the M-file YPRIME.M over the interval T0 to Tfinal and using initial conditions Y0....or use

[T, Y] = ODE45(F, T0, Tfinal, Y0)

INPUT: F - String containing name of user-supplied problem description.

% To Set up function M-file called 'yprime.m'

Use in M-file: y = yprime(t,y) where F = 'yprime'.

t - Time (scaler). NOTE: SHOULD BE USED IN ARGUMENT EVEN IF t IS NOT EXPLICITLY ON RIGHT SIDE OF O.D.E.

y - Solution column-vector.

yprime - Returned derivative column-vector; yprime(i) = dy(i)/dt.
IT ACTUALLY IS THE DERIVATIVE

t0 - Initial value of t.

tfinal- Final value of t.

y0 - Initial value column-vector.

tol - The desired accuracy. (Default: tol = 1.e-6).

trace - If nonzero, each step is printed. (Default: trace = 0).

OUTPUT:
T - Returned integration time points (row-vector).
Y - Returned solution, one solution column-vector per tout-value.

The result can be displayed by: plot(tout, yout).

Illustrative example:

% In the radioactive decay of a an initial mass of material m_o.
It can be shown that the mass m left after a time t is obtained with

$$dm/dt = - ßm$$

Initial condition is at t = 0 m = m_o

and it is further known ß = ln(2)/T_h T_h is half-life of the substance.

we solve this by getting m = f(t)

SOLUTION I: INTEGRATION the exact solution.
% A simple integration gives an exact solution in this case of

$$m = m_o \exp(-ßt)$$

SOLUTION II: NUMERICAL APPROACH

% We will now do a numerical solution and compare our answer with the exact solution. Given an isotope of cobalt with T_h = 5.3 years
And assuming the initial amount m_o = 4.0 we restate the problem as

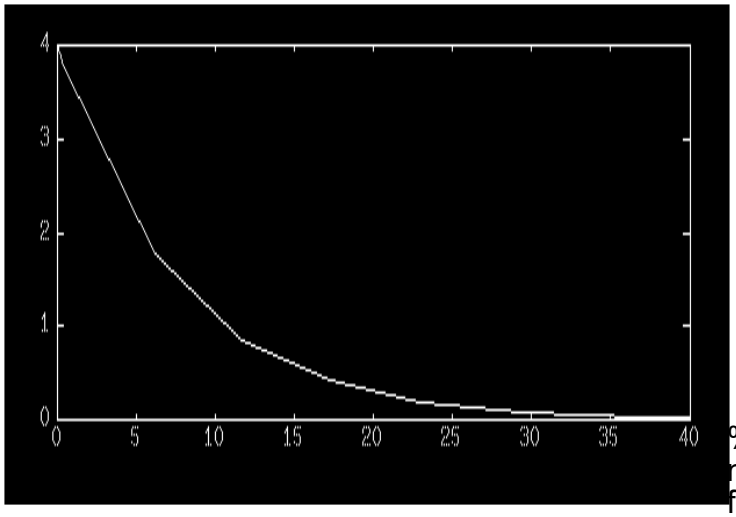
dm/dt =ßm m_o = 4.0 and ß = ln (2) /5.3 =0.131 or

$$dm/dt = 0.131\ m$$
$$mo = 4.0$$

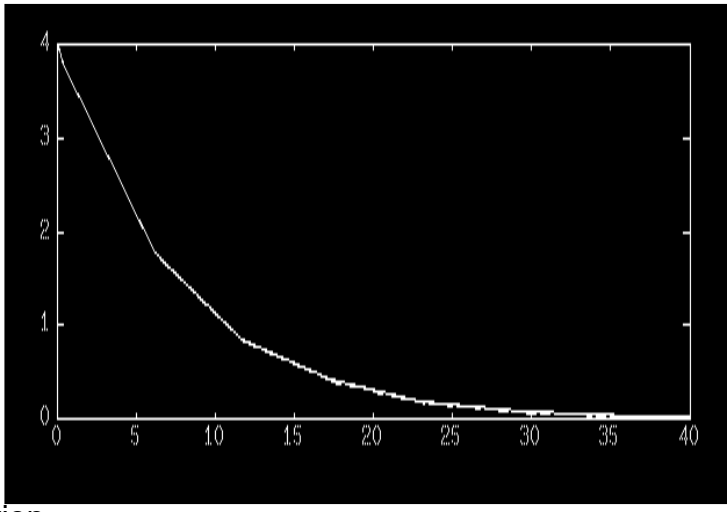
% We first set up the function m-file mprime.m
» type mprime
function y=mprime(t,m)
y = -0.131*m
We know starting time is 0 and starting mass is 4.0 and arbitrarily pick
t = 40 yrs for final value of t
We then use the ode45 MATLAB routine as

» [T M]=ode45('mprime',0,40,4);

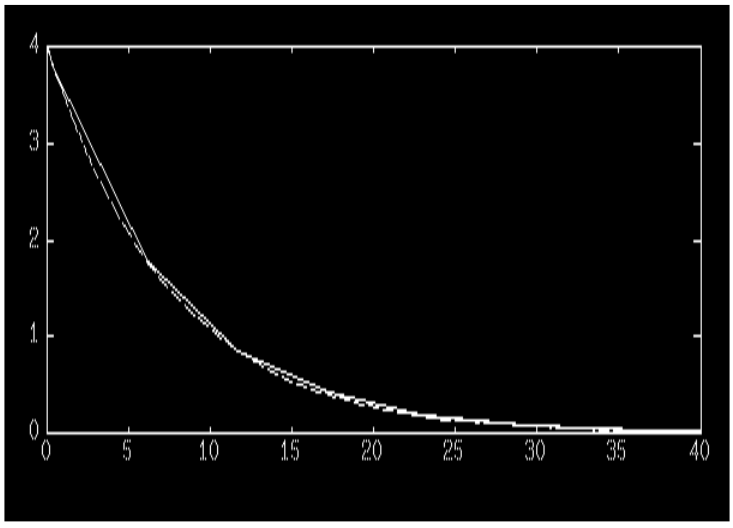
% The values of T and the Solution M are visualized with the plot
» plot(T,M)



% We might try our other numerical technique 'ode23' function whose input arguments are the same:
» [T2 M2]=ode23('mprime',0,40,4); % The actual solution for the same T vector should be
% Mint=4.0*exp(-0.131*T); % We can see the Integrated solution with plot(T,Mint)



% This solution is strikingly like the ode45 solution
% Plotting the ode23 solution with the ode45 solution shows how close each are to each other
» plot(T,M,T2,M2)



% The 'ode45' solution is better than the 'ode23' solution in comparison

with the true integrated solution. It is left as a student exercise
to compare the ode23 and ode45 plots with the "true"integrated solution.
The solutions will be found to be almost identical, with differences showing up
near the end of the interval.

ODE Illustrative Example II

% Assume we are studying the concentration of Carbon Monoxide at some location then we get the following equation of the changes in time of the concentration, with C=Carbon Monoxide concentration:

dC/dt = .000034444 - .0015C

time interval = 0 to 2hrs at 10 min intervals. C_o = .0022

% Set up function file for dC/dt or 'cprime' (note t does not appear explicitly in the function but is needed as an argument for 'ode' MATLAB functions)

» type cprime

```
function y=cprime(t,C)
y=0.000034444 -0.0015*C;
```

% Get the numerical solution of C with

```
» [T,C]=ode45('cprime',0,120,0.0022);
```

% It is convenient to observe the time and CO_2 concentration with

```
» [T C]
ans =         0    0.0022
         1.2000    0.0022
        25.2000    0.0030
        49.2000    0.0037
        73.2000    0.0044
        97.2000    0.0050
       120.0000    0.0056
```

% The only draw back and it can be a big one is that we are stuck with the number of points and the step between the points based on the initial and final values of time t, which can be seen by looking at the function and in particular

```
% type ode45
...
hmax = (tfinal - t)/5;
hmin = (tfinal - t)/20000;
h = (tfinal - t)/100;
...
```

% If we need a finer look at our equations and better control on the step size we can modify the existing function at the above point and perhaps 'rename' the function or we can construct an M-file that follows a Runge -Kutta Algorithm.

% It is a good idea to get the numerical solution form ode23 for comparison especially the step sizes used. It is suggested that the student do this.

A MATLAB ODE DEMO

% A deeper understanding can be gotten by the supplied MATLAB demo which handles a second order equation (broken down to two first orders) and should be run by the student. It is presented here for study.

» odedemo

ODE23 and ODE45 are functions for the numerical solution of ordinary differential equations. They employ automatic step size Runge-Kutta-Fehlberg integration methods. ODE23 uses a simple 2nd and 3rd order pair of formulas for medium accuracy and ODE45 uses a 4th and 5th order pair for higher accuracy. This demo shows their use on a simple differential equation.

Consider the second order differential equation known as the van der Pol equation

y'' + (y^2 - 1)y' + y = 0

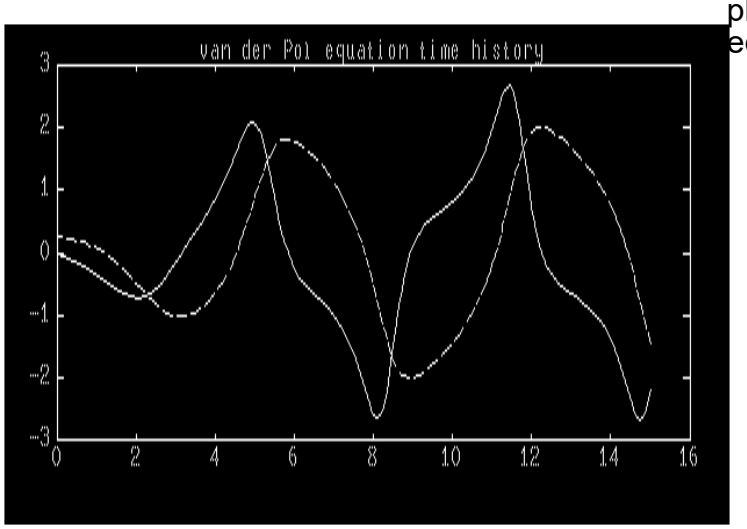
where the prime ' denotes the derivative operator. We can rewrite this as a system of first order differential equations:

y1' = y1(1 - y2^2) - y2

y2' = y1

To simulate a system, we create a function M-file that returns state derivatives, given state and time values. For this example, we've created a file called VDPOL.M. Here's what it looks like:

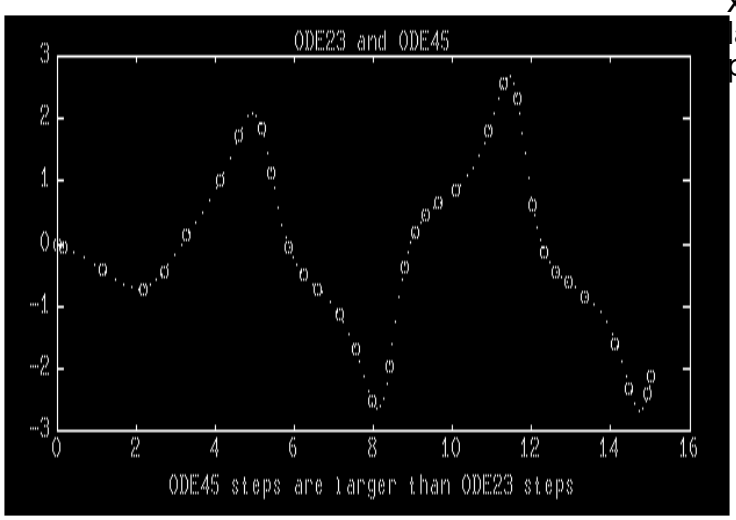
```
type vdpol
function yprime = vdpol(t,y);
        Vdpol(t,y) returns the state derivatives of the Van der Pol
        equation.  Used by ODEDEMO.
yprime = [(y(1) .* (1 - y(2).^2) - y(2)); y(1)];

% NOTE: This function m-file's value of yprime is an extension of previous
 ideas in that it is a row vector with 2 components. The first is the y1' and the
second is the y2'.
        To simulate the differential equation defined in VDPOL over the
        interval  0 < t < 20, we invoke ODE23:
t0 = 0;
tfinal = 15;
y0 = [0 0.25]'; % Define initial conditions.(NOTE two components)
[t,y] = ode23('vdpol',t0,tfinal,y0,tol,trace);
                                                    plot(t,y), title('van der Pol
                                                    equation time history'), pause
```



% NOTE: y1 AND y2 ARE PLOTTED AS FUNCTIONS OF TIME(t). IE. THE SOLUTION

We will now simulate VDPOL using ODE45, instead of ODE23.
ODE45 takes longer at each step, but also takes larger steps.
(On some computers ODE23 and or ODE45 may be hard-coded in MEX-files for speed, which makes time comparisons difficult).

```
[T,Y] = ode45('vdpol',t0,tfinal,y0);
plot(T,Y(:,1),'o',t,y(:,1),'.'), title('ODE23 and ODE45'),..
                                                    xlabel('ODE45 steps are
```




```
larger than ODE23 steps'),
pause
```

% A nice way to see the subtle differences in ode23 and ode45

CHECK OUT THE LATEST ODE DEMO IN MATLAB BY RUNING the "odedemo" program for additonal examples and clarifications

% **LABORATORY TASK**

41. (20 pts) First: REPRODUCE ALL THE NOTES AND SUGGESTIONS ABOVE ON O.D.E. as practice.

The concentration of Hydrochloric acid in the air of building 6S has been increasing in time. The Concentration, H, is governed by the **differential equation**

$$dH/dt = 9.7t^2-t/6$$

t is in hours

We want to know the concentration of the acid if we start with H=0 and time =0 after 500 hours. Write the MATLAB code necessary to find H(t).

NOTE: THIS IS A FICTITIOUS PROBLEM ANY RESEMBLANCE TO THE REAL WORLD IS STRICTLY COINCIDENTAL.