10 – Solutions to Ordinary Differential Equations

An ordinary differential equation has only one independent variable. A system of ordinary differential equations consists of several differential equations each with the same independent variable. An example of an ordinary differential equation sets the derivative of velocity with respect to time equal to acceleration.

$$\frac{dv}{dt} = a \tag{10.1}$$

Including the ordinary differential of distance with time as velocity makes a system of ordinary differential equations.

$$\frac{ds}{dt} = v \qquad \frac{dv}{dt} = a \tag{10.2}$$

Another way this system may be written involves the expression of acceleration as the second differential of distance with time.

$$\frac{d^2s}{dt^2} = a \tag{10.3}$$

where

$$v = \frac{ds}{dt} \tag{10.4}$$

Both of the forms described by Eqs. (10.2) and (10.3) require two constants of integration to solve. These are well known to be the velocity and distance at some time, usually at the initial time.

This system can be made more interesting if the acceleration (net force divided by the mass) bears a functional relationship to velocity and distance as is the case for a falling meteorite. As it enters the Earth's atmosphere it encounters drag that is related to both elevation and velocity. In this case

$$\frac{ds}{dt} = v \qquad \frac{dv}{dt} = f(s, v) \tag{10.5}$$

The acceleration function can even be a function of the independent variable, time, as is the case for a returning space capsule with a time-deployed parachute.

$$\frac{ds}{dt} = v \qquad \frac{dv}{dt} = f(t, s, v) \tag{10.6}$$

In general, a system of three ordinary equations may be written as

$$\frac{dx}{dt} = f_x(t, x, y, z), \qquad \frac{dy}{dt} = f_y(t, x, y, z), \qquad \frac{dz}{dt} = f_z(t, x, y, z) \tag{10.7}$$

10.1 – Euler's Method

Every engineering and science student quickly realizes that a slope function may be used to approximate the position of the slope's dependent variable providing a starting value for the variable is known. For example, for the function

$$\frac{dy}{dx} = f(x,y) = 1 + 0.2y + x \qquad y_0 = 2 \text{ when } x_0 = 0 \tag{10.8}$$

an approximation of y(1.5) is found as shown in Table 1 for a step size (extrapolation distance) of 0.5. Although Euler's Method is easy, the step size and its effect on the result need to be examined. Table 2 repeats the integration using a step size of 0.01 and has a very different result (5.092 vs. 10.232...).

Table 1. Euler Example with step size of 0.5

x	y	y' = 1 + 0.2*y + x	Δy = y'*Δx
0.0	2.000	1.4000	0.7000
0.5	2.700	2.0400	1.0200
1.0	3.720	2.7440	1.3720
1.5	5.092	3.5184	1.7592

Table 2. Euler Example with step size of 0.01

x	**y**	y' = 1 + 0.2*y + x	Δy = y'*Δx
0	2	1.4	0.014
0.01	2.014	1.4128	0.014128
0.02	2.028128	1.4256256	0.014256256
0.03	2.042384256	1.438476851	0.014384769
..	..	..	..
..	..	..	..
2.48	10.12244296	5.504488591	0.055044886
2.49	10.17748784	5.525497569	0.055254976
2.50	10.23274282	5.546548564	0.055465486

The order of error for Euler's Method is found by recognizing that each extrapolation of the dependent variable y is

$$y_{i+1} = y_i + h * f(x_i, y_i) \tag{10.9}$$

The next term in this first order Taylor Series contains the error and is

$$e_i = \frac{f''(x_i, y_i)}{2} h^2 \tag{10.10}$$

The accumulated error is the summation of all these terms over the interval from (a to b):

$$e_N = \sum_{i=0}^{N} \frac{f''(x_i, y_i)c_i}{2} h^2 = \frac{Nf''(x,y)c}{2} h^2 = \frac{Nhf''(x_i, y_i)c}{2} h = \frac{(b-a)f''(x_i, y_i)c}{2} h = O(h)\,. \quad (10.11)$$

Therefore, the order of the error is related directly to the step size. In the above example, the error is approximately 50 times smaller for a step size of 0.01than for a step size of 0.5. If a method were devised that had an error on the order of h^2, the error would be reduced by approximately $(1/50)^2$. Methods with errors on the order of h^4 are presented in the following section.

10.2 - Runge-Kutta Methods

Runge-Kutta is a method of extrapolating the dependent variable in a first order differential equation or all the dependent equations in a system of first order ordinary differential equations. The general procedure is always the same but may be employed to achieve different levels of accuracy. Of course, it is true that additional computational effort is needed to achieve higher accuracy.

The Runge-Kutta method consists of using an estimate or the average of several estimates, of the change in the dependent variable over a small step in the independent variable. This estimation procedure requires knowing the function that describes the slope of each dependent equation. Also required to find new values of the dependent at the end of a step in the independent is the initial value for each of the dependent variables. In the simplest form of Runge-Kutta, a single estimate of slope is used to determine the change in the dependent variable. More complex forms use multiple estimates of slope to determine changes in the dependent variable. Of course, if the problem involves two interdependent differential equations, then an estimate for each of the variables must be determined in order to evaluate the slopes. The following example illustrates such as case.

Example 1a

Consider the two slope functions

$$\frac{\partial x_0}{\partial t} = f_x(t, x_0, x_1) = \frac{x_0}{t} + 0.002 * x_1 \qquad \text{at t = 0 } x_0 = 0 \quad (10.12)$$

$$\frac{\partial x_1}{\partial t} = f_{x1}(t, x_0, x_1) = x_0 + 0.002 x_1 t \qquad \text{at t = 0 } x_1 = 1 \quad (10.13)$$

Solution

Notice that the equations require a simultaneous solution since they are both functions of x_0 and x_1.

The First Order Runge-Kutta solution is

$$x_{0,i+1} = x_{0,i} + k_{1x_0}$$
$$x_{1,i+1} = x_{1,i} + k_{1x_1}$$

Where each k value is an estimate of the change in the dependent variable (the rise) and the Δt is the step size (the run).

$$k_{1x_0} = f_{x_0}(t, x_0, x_1)\Delta t$$
$$k_{1x_1} = f_{x_1}(t, x_0, x_1)\Delta t$$

Since this form uses only one estimate of slope for each dependent variable, it is called a First Order Runge-Kutta approximation. It is the same as the Euler Method and has an error in the result on the order of h^1.

A second order approximation with error on the order of h^2 would have two estimates of slope. The most common locations for these two estimates are at the beginning and the ending of the step. The estimate at the start of the increment is the same as for the First Order Runge-Kutta Method. The estimate at the end of the step requires an estimate for the dependent variables as shown below.

The Second Order Runge-Kutta solution is

$$\mathrm{x}_{0,\mathrm{i}+1} = \mathrm{x}_{0,\mathrm{i}} + \tfrac{1}{2}\left(\mathrm{k}_{1\mathrm{x}_0} + \mathrm{k}_{2\mathrm{x}_0}\right)$$
$$\mathrm{x}_{1,\mathrm{i}+1} = \mathrm{x}_{1,\mathrm{i}} + \tfrac{1}{2}\left(\mathrm{k}_{1\mathrm{x}_1} + \mathrm{k}_{2\mathrm{x}_1}\right)$$

where

$$k_{1x_0} = f_{x_0}(t, x_0, x_1)\Delta t$$
$$k_{1x_1} = f_{x_1}(t, x_0, x_1)\Delta t$$
$$k_{2x_0} = f_{x_0}(t + \Delta t, x_0 + k_{1x_0}, x_1 + k_{1x_1})\Delta t$$
$$k_{2x_1} = f_{x_1}(t + \Delta t, x_0 + k_{1x_0}, x_1 + k_{1x_1})\Delta t$$

Engineers and scientists commonly use the Fourth Order Runge-Kutta solution. The Fourth Order Runge-Kutta solution is

$$\mathrm{x}_{0,\mathrm{i}+1} = \mathrm{x}_{0,\mathrm{i}} + \tfrac{1}{6}\left(\mathrm{k}_{1\mathrm{x}_0} + 2\mathrm{k}_{2\mathrm{x}_0} + 2\mathrm{k}_{3\mathrm{x}_0} + \mathrm{k}_{4\mathrm{x}_0}\right)$$
$$\mathrm{x}_{1,\mathrm{i}+1} = \mathrm{x}_{1,\mathrm{i}} + \tfrac{1}{6}\left(\mathrm{k}_{1\mathrm{x}_1} + 2\mathrm{k}_{2\mathrm{x}_1} + 2\mathrm{k}_{3\mathrm{x}_1} + \mathrm{k}_{4\mathrm{x}_1}\right)$$

where

$$k_{1x_0} = f_{x_0}(t, x_0, x_1)\Delta t$$
$$k_{1x_1} = f_{x_1}(t, x_0, x_1)\Delta t$$
$$k_{2x_0} = f_{x_0}(t + \frac{\Delta t}{2}, x_0 + \frac{k_{1x_0}}{2}, x_1 + \frac{k_{1x_1}}{2})\Delta t$$
$$k_{2x_1} = f_{x_1}(t + \frac{\Delta t}{2}, x_0 + \frac{k_{1x_0}}{2}, x_1 + \frac{k_{1x_1}}{2})\Delta t$$
$$k_{3x_0} = f_{x_0}(t + \frac{\Delta t}{2}, x_0 + \frac{k_{2x_0}}{2}, x_1 + \frac{k_{2x_1}}{2})\Delta t$$
$$k_{3x_1} = f_{x_1}(t + \frac{\Delta t}{2}, x_0 + \frac{k_{2x_0}}{2}, x_1 + \frac{k_{2x_1}}{2})\Delta t$$
$$k_{4x_0} = f_{x_0}(t + \Delta t, x_0 + k_{3x_0}, x_1 + k_{3x_1})\Delta t$$
$$k_{4x_1} = f_{x_1}(t + \Delta t, x_0 + k_{3x_0}, x_1 + k_{3x_1})\Delta t$$

In the fourth order approximation, each k value is used to estimate the next slope. A total of six estimates of the slope are used to determine the change in the dependent variables: one at the start of the step (k_1), four at the middle of the step ($2k_2 + 2k_3$), and one at the end of the step (k_4). This provides an excellent approximation of slope across the step leading to an error in the result on the order of h^4.

10.3 – ODE Computational Options

The manual computation using any of the previous methods is a daunting task, especially for the very accurate 4th Order Runge-Kutta Method. Therefore, computational software is used. Among the more commonly used are MathCad® by MathSoft® and MATLAB® by Math Works®. The examples used here are primarily MathCad® and MATLAB® examples.

Example 2

Consider the three slope functions and their corresponding initial values.

Functions	Initial values
$fx' = ax - byt$	$x_0 = 1.0$
$fy' = cy + xt$	$y_0 = 1.2$
$fz' = ft$	$z_0 = 2.0$

where

$a = 0.02$

$b = 0.025$

$c = 0.005$

$f = 0.015$

The MathCad solution of this system by 4th Order RK is shown below.

Solution to three simultaneous ordinary differential equations

Constants

$a := 0.02 \qquad b := 0.025 \qquad c := 0.005 \qquad f := 0.015$

Functions

$fx(t,x,y,z) := a \cdot x - b \cdot y \cdot t \qquad fy(t,x,y,z) := c \cdot y + x \cdot t \qquad fz(t,x,y,z) := f \cdot t$

Initial Conditions and Slope Array

$$I := \begin{pmatrix} 1 \\ 1.2 \\ 2 \end{pmatrix} \begin{matrix} xo \\ yo \\ zo \end{matrix} \qquad D(t,w) := \begin{pmatrix} fx(t, w_0, w_1, w_2) \\ fy(t, w_0, w_1, w_2) \\ fz(t, w_0, w_1, w_2) \end{pmatrix} \begin{matrix} x' \\ y' \\ z' \end{matrix}$$

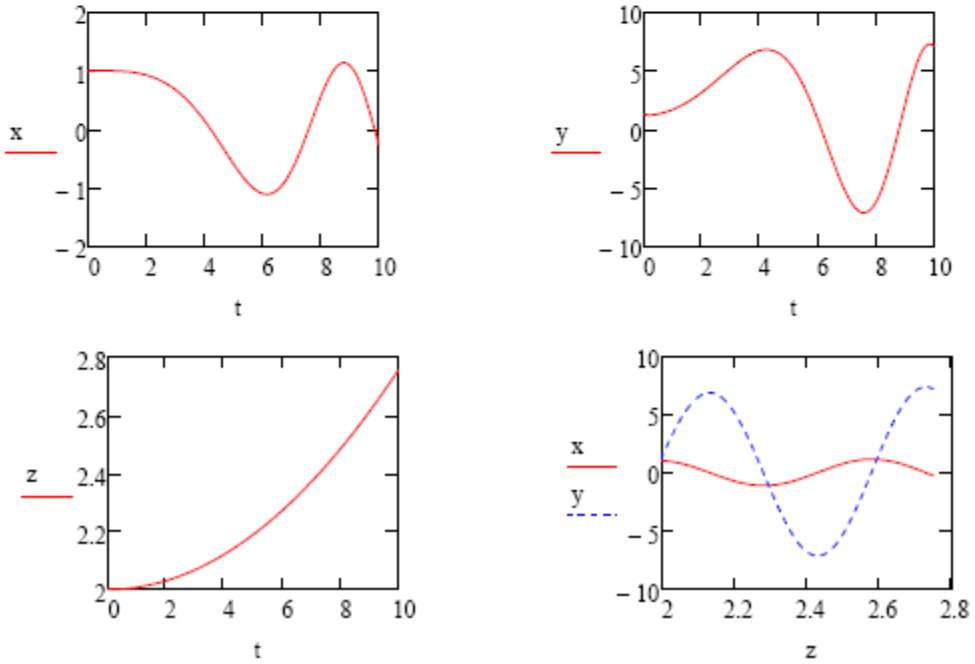
Integrate from t=0 to t=10 in 100 steps

$A := rkfixed(I, 0, 10, 100, D)$

Strip out A columns arrays for convenience

$t := A^{\langle 0 \rangle} \qquad x := A^{\langle 1 \rangle} \qquad y := A^{\langle 2 \rangle} \qquad z := A^{\langle 3 \rangle}$

Plot results



Example 2

Solve Example 2 using MATLAB®.

First write the slope function :

M.file ex2.m

```
function dw = react(t,w)
a=0.02;
b=0.025;
c=0.005;
f=0.015;
dw = zeros(3,1);  % a column vector
dw(1) = a*w(1)-b*w(2)*t;
dw(2) = c*w(2) + w(1)*t;
dw(3) = f*t;
```

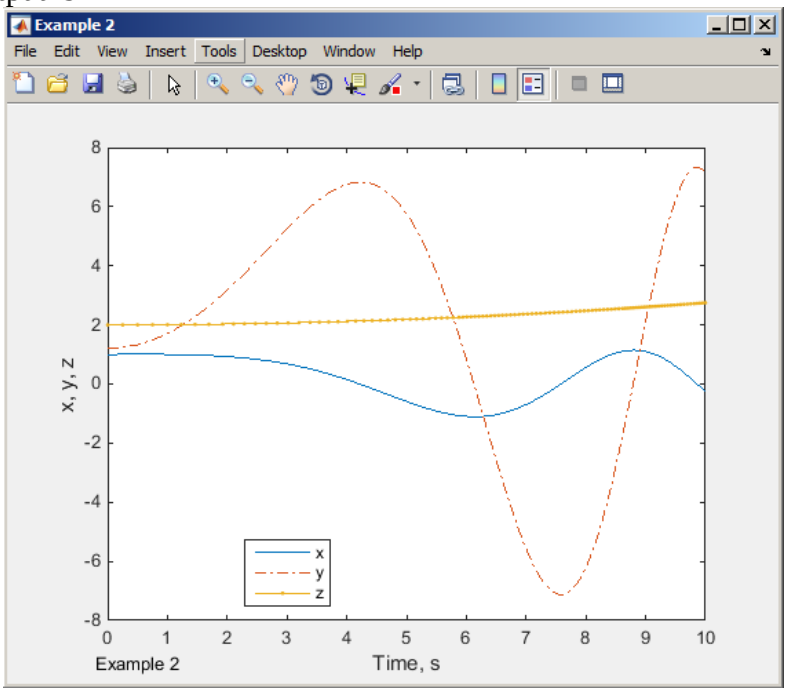
Then write the code for solving the problem using the *ode45* function.

M.file: Example2

```
figure('Name','Example 2','NumberTitle','off')
options = odeset('RelTol',1e-5,'AbsTol',[1e-5 1e-6 1e-6]);
[t,z] = ode45(@ex2,[0 10],[1 1.2 2],options);
plot(t,z(:,1),'-',t,z(:,2),'-.',t,z(:,3),'.-')

xlabel 'Time, s'
ylabel 'x, y, z '
legend('x','y','z', 'Location', 'best')
text(-0.2, -9.5, 'Example 2')  % placing text at bottom for title
```

The output is



Example 3

A chemical reaction converts A to B reversibly as shown

$$A \Leftrightarrow B$$

with a forward stoichiometric rate constant k_1 (1.0) and a reverse rate constant k_2 (0.1).

Therefore,

$$\frac{dA}{dt} = -k_1 A + k_2 B \qquad \text{and} \qquad \frac{dB}{dt} = k_1 A - k_2 B - k_3 B = -\frac{dA}{dt} - k_3 B$$

The species B reacts irreversibly to form C as follows:

$$B \Rightarrow C$$

The stoichiometric rate constant for the formation of C is k_3 (0.2). Find the maximum concentration of B, and plot the concentration of A, B, and C with time given the initial concentrations of A, B, and C are 1, 0, and 0. The units for concentration are molar and the units of the time constant are sec^{-1}.

Therefore,

$$\frac{dC}{dt} = k_3 B$$

First write the function called react here that will return the slope array for the three species.

M.file: react.m

```
function dy = react(t,y)
k1=1;
k2=0.1;
k3=0.2;
dy = zeros(3,1);  % a column vector
dy(1) = -k1*y(1) +k2*y(2);
dy(3) = k3 * y(2);
dy(2) = -dy(1) - dy(3);
end
```

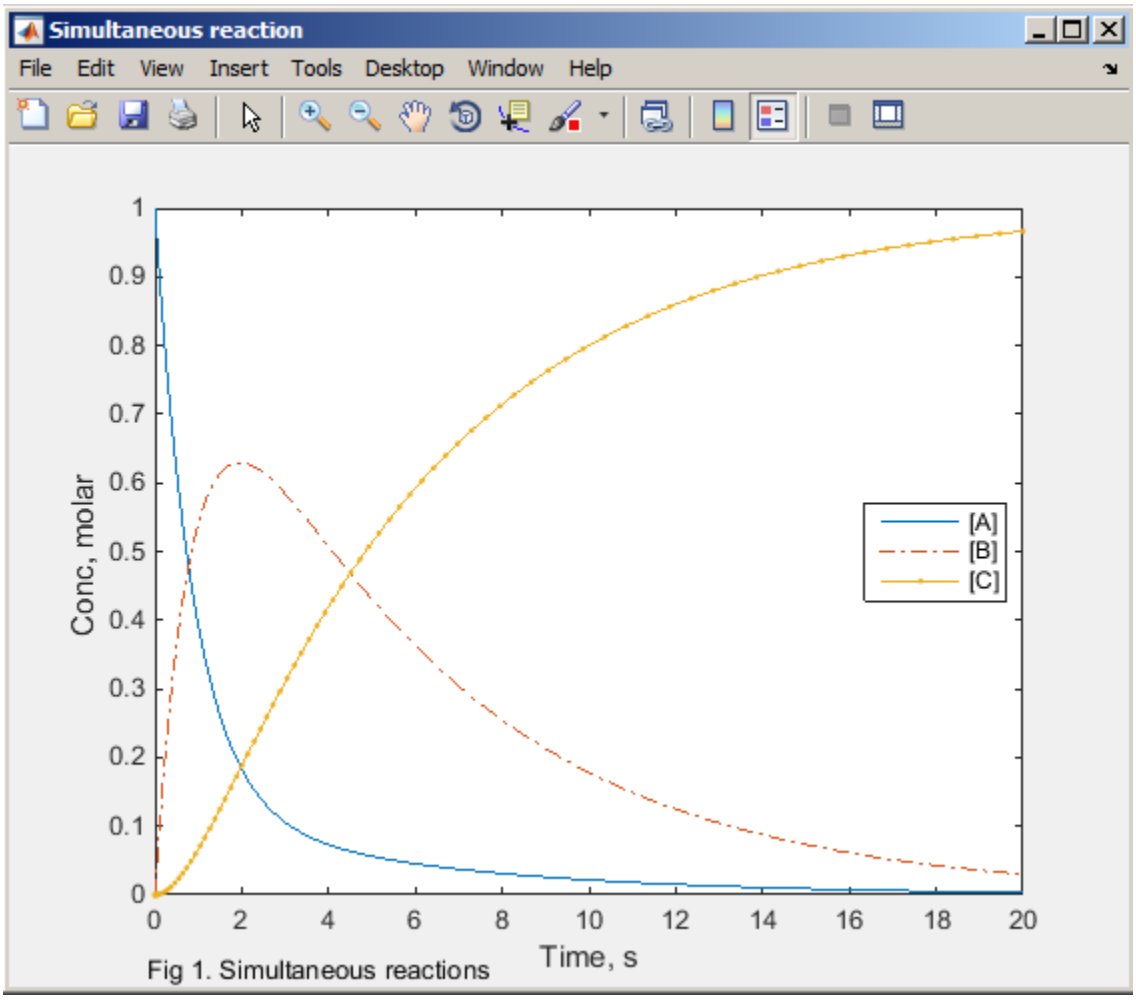
where y(1) = A, y(2) = B, and y(3) = C. The slopes of A, B, and C are dy(1), dy(2), and dy(3).

Now use the *ode45* MATLAB library subroutine to perform the 4th order Runge-Kutta integration.

M.file: SimReactions.m

```
figure('Name','Simultaneous reaction','NumberTitle','off')
options = odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-5 1e-5]);
[t,z] = ode45(@react,[0 20],[1 0 0],options);
plot(t,z(:,1),'-',t,z(:,2),'-.',t,z(:,3),'.-')
MaxB=max(z(:,2))% print maximum [B]
xlabel 'Time, s'
ylabel 'Conc, molar '
legend('[A]','[B]','[C]', 'Location', 'best')
text(-0.2, -0.11, 'Fig 1. Simultaneous reactions')  % placing text at
bottom for title
```

The output is




```
>> SimReactions

MaxB =

    0.6295
```