A report on solution of second order ordinary differential equation using fourth order Runge-Kutta method

*Arjun Darbha*

Contents

[1 Introduction 4](#_Toc352109643)

[2 Numerical Methods 5](#_Toc352109644)

[3 Solution to the problem 9](#_Toc352109645)

[4 References 20](#_Toc352109646)

[5 Appendix 20](#_Toc352109647)

# 1 Introduction

Ordinary differential equations (ODE) can be solved both analytically and numerically. However, it is not possible to formulate an analytical solution in every case. Also, numerical solutions are easier to obtain in most of the cases and some of these methods have a very good accuracy. Unlike analytical solutions, in order to formulate a numerical solution, one has to be provided with an initial/boundary value in order to start progressing from that given point. Numerical solutions for ordinary differential equations are actually classified into two types of problems,

* Initial-Value Problems (IVP)
* Boundary-Value Problems (BVP)

In case of IVP’s the value of dependent variable at a particular value of independent value is provided to start iterations. BVP’s however, are more constrained where dependent values are provided at two different values of independent variables. General form of an nth order differential equation is represented by Eq. (1)

*an(x)yn + an-1(x)yn-1 + … + a1(x)y1 + a0y = g(x)* Eq.1

Here ai(x), i = 0,1,2,…n and g(x) are known functions of x and yi = , i = 1,2,…n. Eq.1 is also a linear ODE. If the ai is a function of x and y, then Eq.1 transforms into a non-linear ODE. However, this report is limited to solving a linear ODE, (also an IVP) which is the scope of the problem statement. There are numerous techniques using which an IVP could be solved, some of them are

* Euler’s method
* Heun’s method
* Runge-Kutta (RK) method

The first two methods, developed in 18th century, actually form a subset of the RK method, but they have been developed independently and RK method evolved much later in the early 1900’s.

The ODE that is solved in this report is a simple model of a suspension system given by Eq.2

Eq.2

Where t is time, m is mass of the moving parts of the system, B is a measure of damping in the system, k is the stiffness of the suspension system and f is a constant force applied at t=0.

# 2 Numerical Methods

A brief description on Euler’s and Heun’s method is provided at this point, which enables the reader to appreciate the idea behind the RK methods.

**2.1 Euler Method**

This is the simplest and the earliest techniques used for solving the ODEs. This method is not as accurate as the other methods discussed, but, it forms the basis of the idea of every other numerical method used for solving ODEs. Also, this method provides a direct solution to the first order ODE. But, solution to higher ODEs can be obtained by converting them into first order ODEs and then, using Euler method.

Consider a first order ODE

Eq.3

With initial conditions

y(x0) = y0

the idea is obtain a solution at a point xi such that

xi+1 = xi + h

where h is the step size which is an input parameter. Also, lower the step size, higher the accuracy of the solution. The next step is to calculate the value of function at yi+1 which is given by

yi+1 = yi + hf(xi,yi) Eq.4

From Eq 4, it can be clearly seen that the next value of y is obtained using the Taylor series by neglecting the higher order terms. This fact can be well appreciated by the graphical illustration presented in Fig.1. It is evident that the red line (Euler’s solution) gradually diverges from the blue line (Actual solution) . Hence this method yields good results only for extremely small values of h, which increases the computation time.

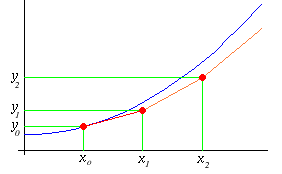


Fig 1. Graphical Illustration of Euler’s method

There have been many improvements suggested to improve the solution generated by Euler method. Some of them are Heun’s method, improved polygon method, Runge-Kutta methods etc. Heun’s and improved polygon method have a meager benefit when compared to Euler’s method and are just an improvement over the basic idea of Euler’s method. Runge-Kutta methods follow a more generalized approach and are discussed in detail in subsequent sections

**2.2 Runge-Kutta methods**

As already discussed higher order Taylor series expansions can be used to reduce the truncation error, but they tend to increase the computation time. Moreover, higher order terms require higher order derivatives, which may not be available in every case. Runge-Kutta (RK) methods do not require higher order derivatives of f(x,y) and produce results equivalent higher order Taylor formulas.

Not unlike Euler’s method, RK methods require the value of function at initial point, f(x0,y0), to start with the solution scheme. However the function is evaluated several times at each step which makes this method computationally intensive. Runge-Kutta methods are defined by order of RK methods. It has to be realized that order of RK method is not same as order of the Taylor series. The difference should be clear from the equations presented in the subsequent parts of the section. Also, the method discussed pertains to two dimensions only. However, the same idea can be extended to multiple dimensions. A general RK method can be stated

*yi+1 = yi + hα(xi,,yi,h)* Eq.5

Here, *α(xi,,yi,h)* is the increment function, and this calculates the average slope between two points xi and xi+1. The increment function α can be expressed as

*α(xi,,yi,h) = c1k1 + c2k2 + … + cnkn* Eq.6

In Eq.6 c1,c2,…cn are constants and k1, k2,…kn are recurrence relations which are given by

*k1 = f(xi,yi)* Eq.7

*k2 = f(xi + p2h, yi + a21hk1)* Eq.8

*k3 = f(xi + p3h, yi + a31hk1 + a32hk2)* Eq.9

Following from the above discussion, a general nth order RK equation can be written in the following form

Eq.10

As already mentioned Eq.10 is a general equation n determines the order of RK method. The order of interest for solving the problem statement is fourth order Runge-Kutta method the details of which are presented in the subsequent section.

**2.3 Fourth order Runge-Kutta method (RK4)**

This is the most widely used method for solving Ordinary Differential Equations. It has also been determined that the accuracy/computation-time ratio is highest for this method. From general Eq.10, for RK4, it can be deduced that,

yi+1 = yi + (h/6)(k1 + 2k2 + 2k3 + k4) Eq.11

where

k1 = f(xi,yi)

k2 = f(xi + 0.5h, yi + 0.5hk1)

k3 = f(xi + 0.5h, yi + 0.5hk2)

k4 = f(xi + h, yi + hk3)

The idea behind deriving these equations is discussed here. A second order RK method is solved and the same idea can be extended to RK4.

For second order from Eq.10,

Eq.12

from Eq.7 and Eq.8 it can be seen that

Eq.13

Expanding over one can obtain the following expression,

Eq.14

Also from TSE of we get

Eq.15

can be expressed as

Eq.16

Therefore Eq.15 transforms into

Eq.17

By comparing Eq.17 to Eq.14 we have,

Eq.18

Eq.19

Eq.20

Eq.19,20,21 yield more than a single value for each of the variable involved based on assigning a value to one of the variables.

The same idea can be extended to RK4 where TSE involves expansion to 4 terms with O(h5).

# 3 Solution to the problem

With idea of RK methods it is evident that that Eq.2 can be solved by RK method. However RK method solves a first order ODE. Hence Eq.10 needs to be split into two first order equations.

Let

Eq.21

then Eq.2 transforms to

Eq.22

The conditions for which this system needs to be solved is for m=1, k=4, f=1 and x=0, and =0 when t = 0. This means that this is a second order ODE and also an IVP and hence an RK method can be used to solve this problem.

**Algorithm**

* Initialize the parameters t, ∆t, initial position x0, and total time
* Start loop with loop counter as the time-step
* Calculate the values of k1, k2, k3, k4 which are the vital parameters of RK4 method
* Update the value of x at the end of each time step
* Stop loop when t is calculated for all time steps

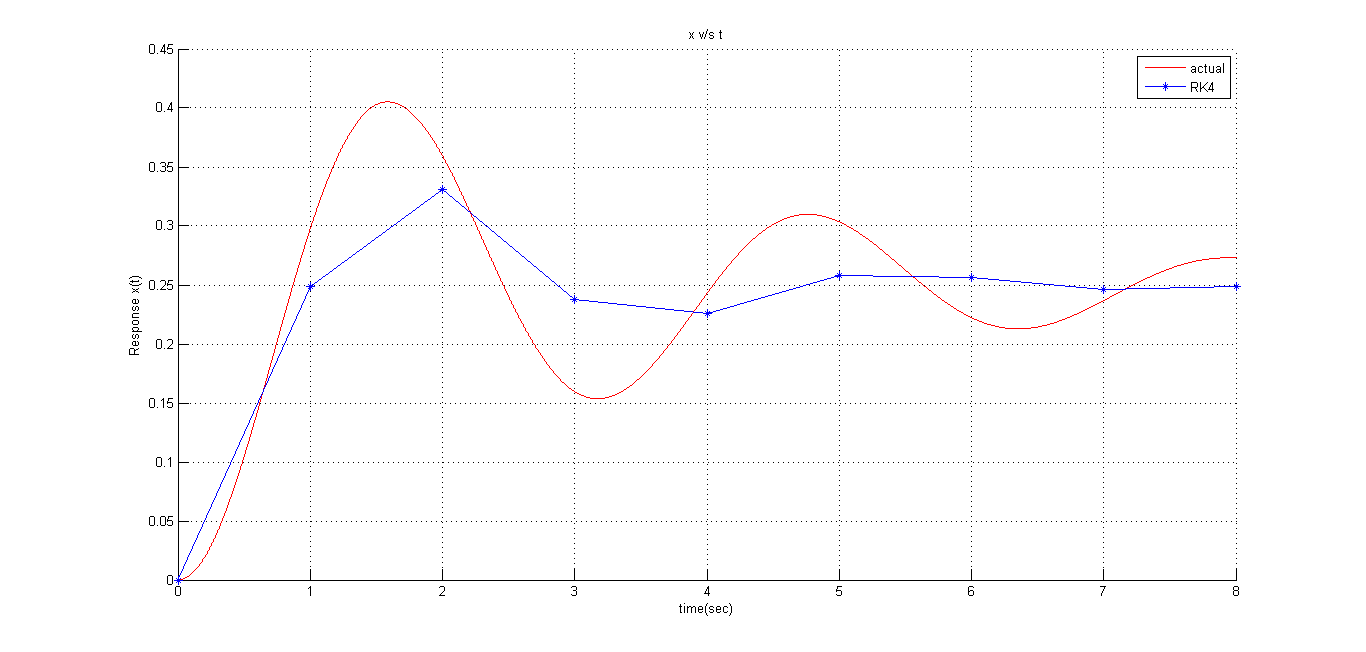
**Case 1**

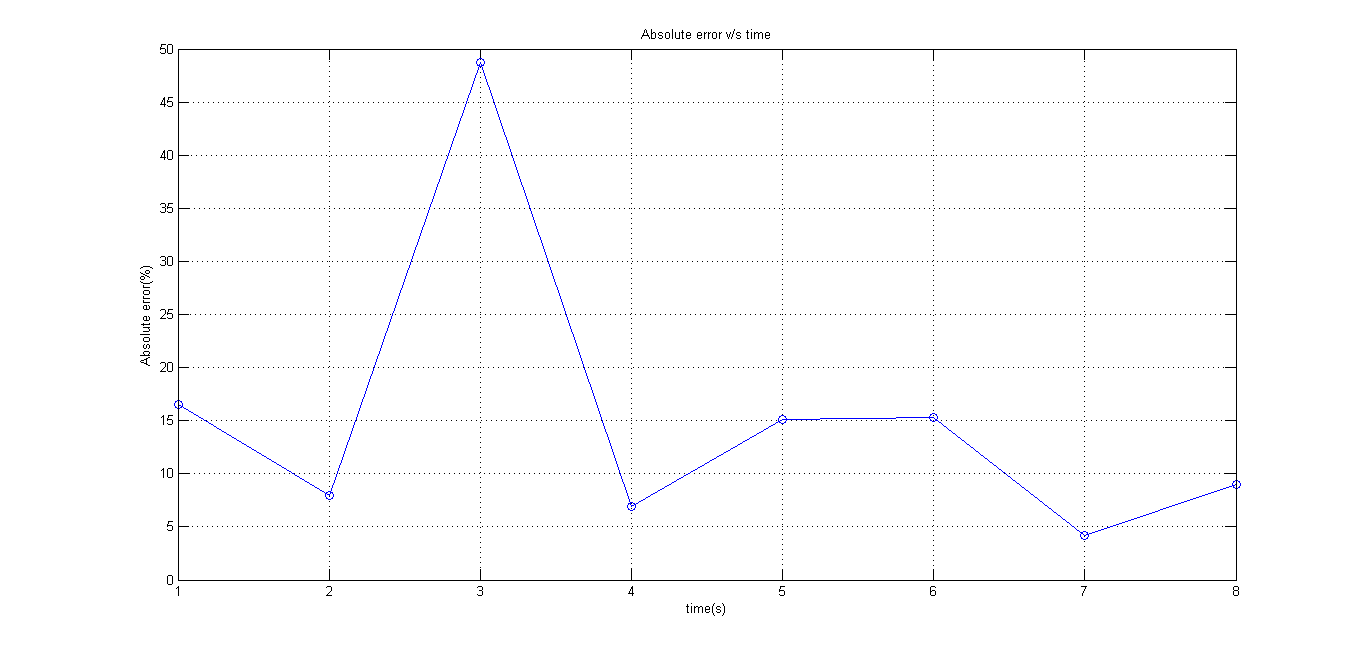
In this case, . The analytical solution for this case is given by the equation

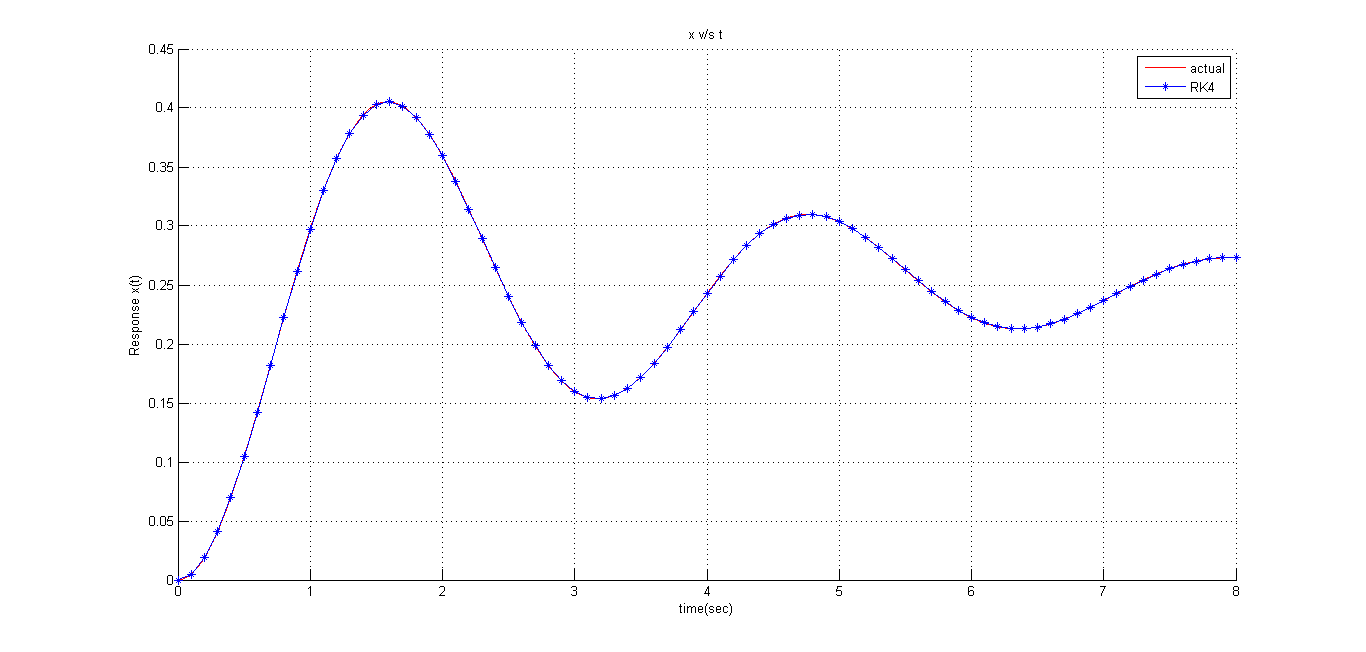
Eq.23

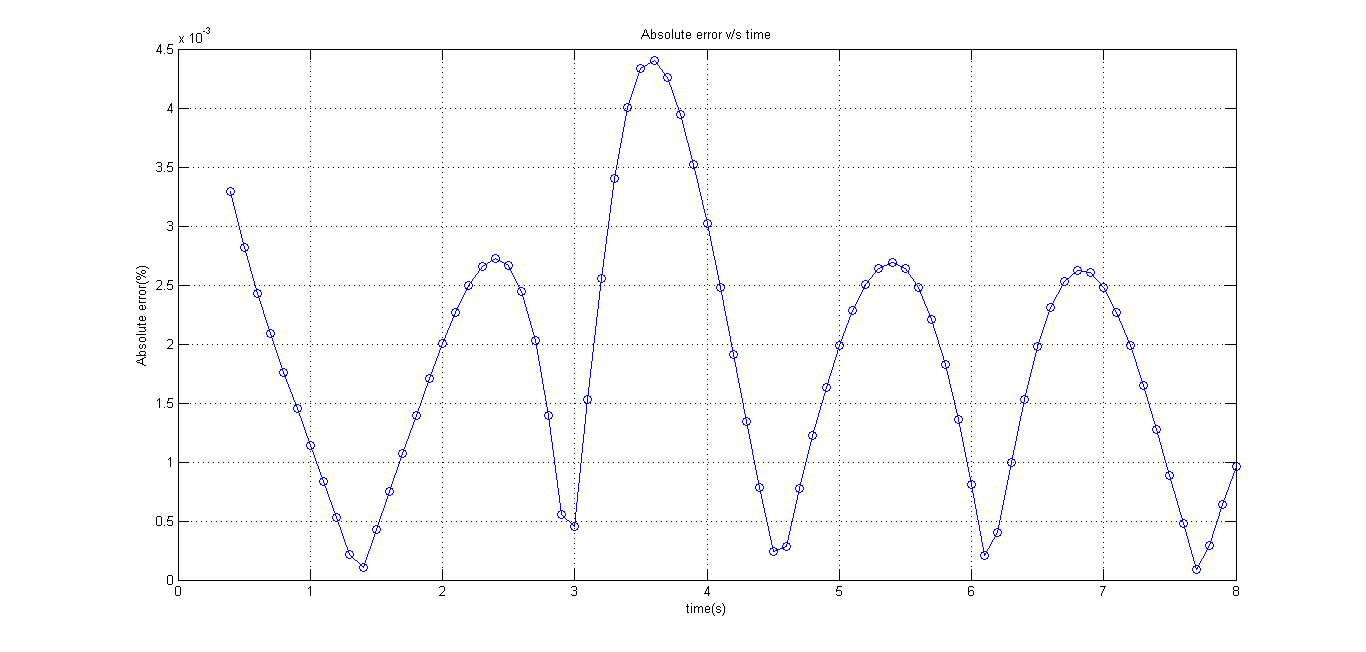
where, , ,

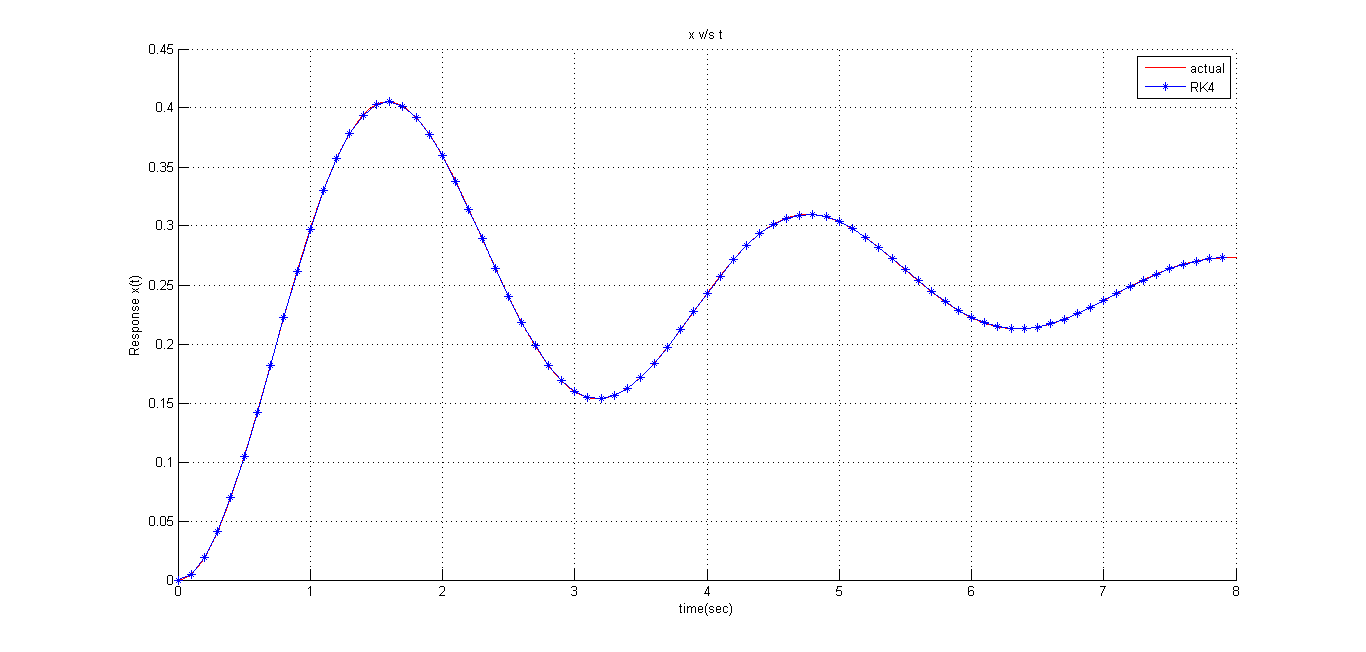
With Eq.23 as the actual solution and ∆t = 1,0.1,0.01,0.001 Fig 2,4,6,8 are presented as x v/s t plots. Fig. 3,5,7,9 denote percentage absolute value of error as a function of time. It is evident from these figures that the error is reduced as ∆t decreases. However, the pattern of error distribution is the same with error maximizing at the points of maxima/minima on the actual solution

Fig 2. x v/s t graph ∆t = 1

Fig 3. Absolute error v/s time for ∆t = 1

Fig 4. x v/s t ∆t = 0.1

Fig 5 Absolute error v/s time for ∆t = 0.1

Fig 6. x v/s t ∆t = 0.01

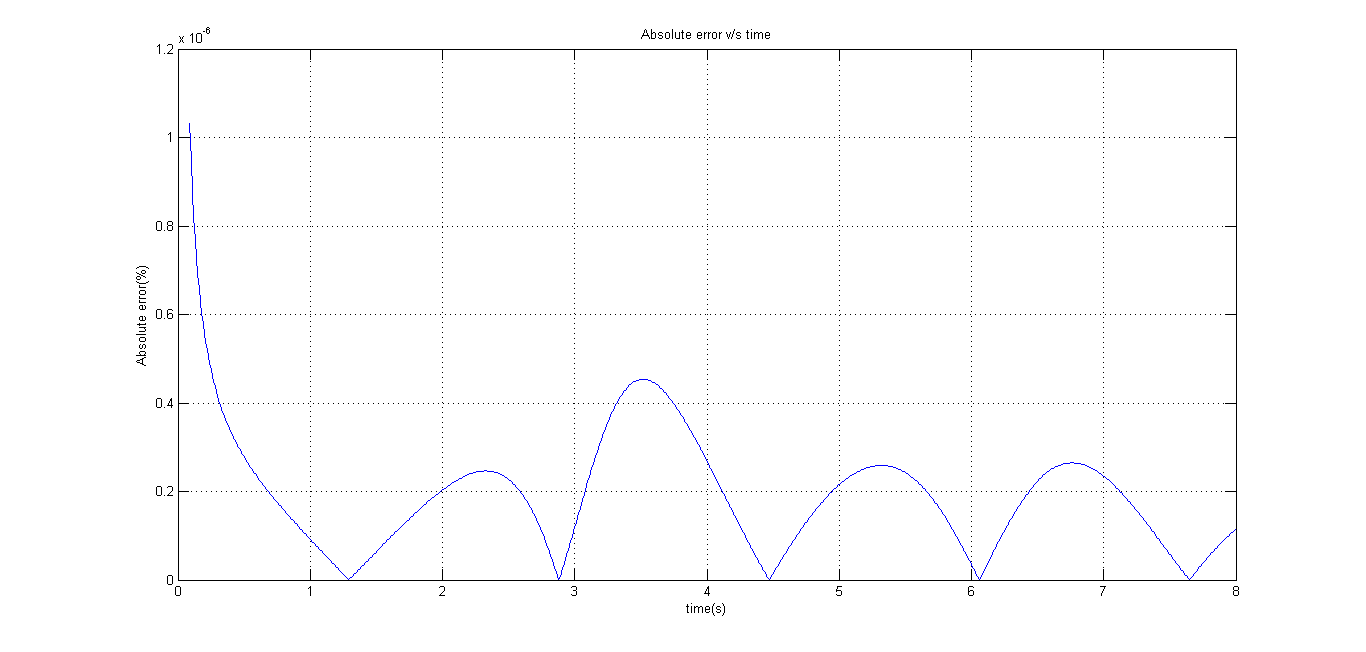
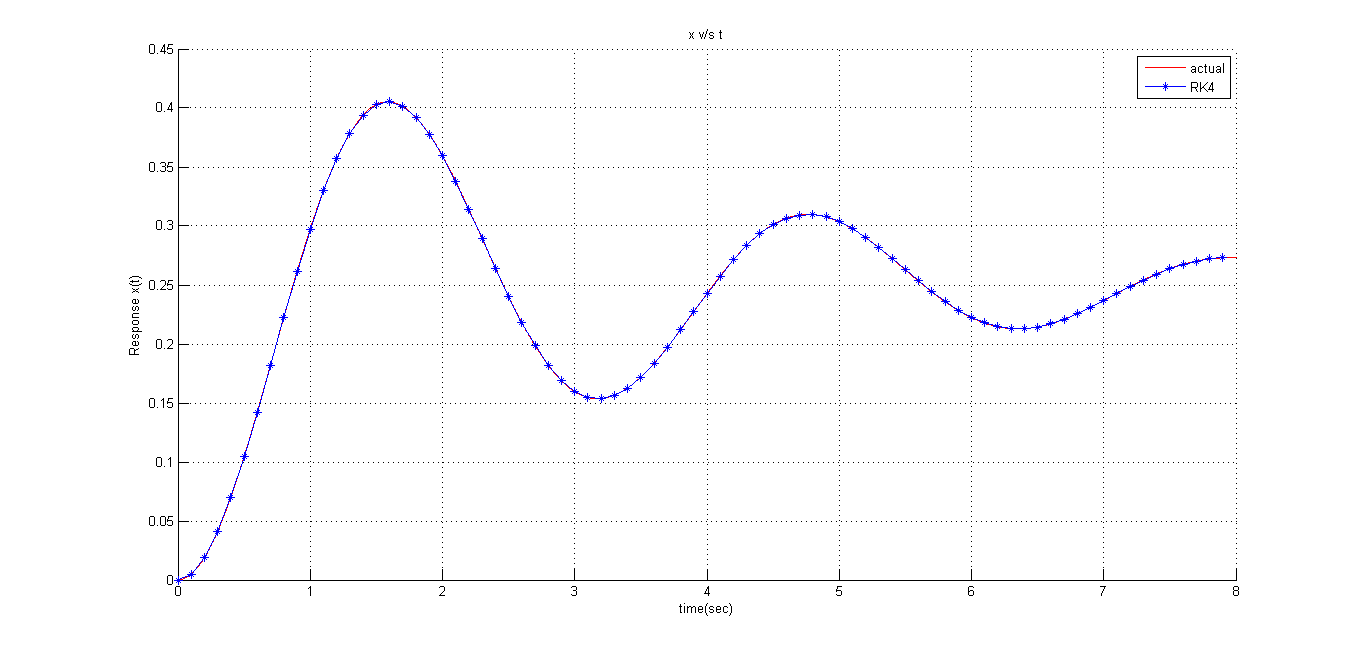


Fig 7 Absolute error v/s time for ∆t = 0.01

Fig 8. x v/s t ∆t = 0.001

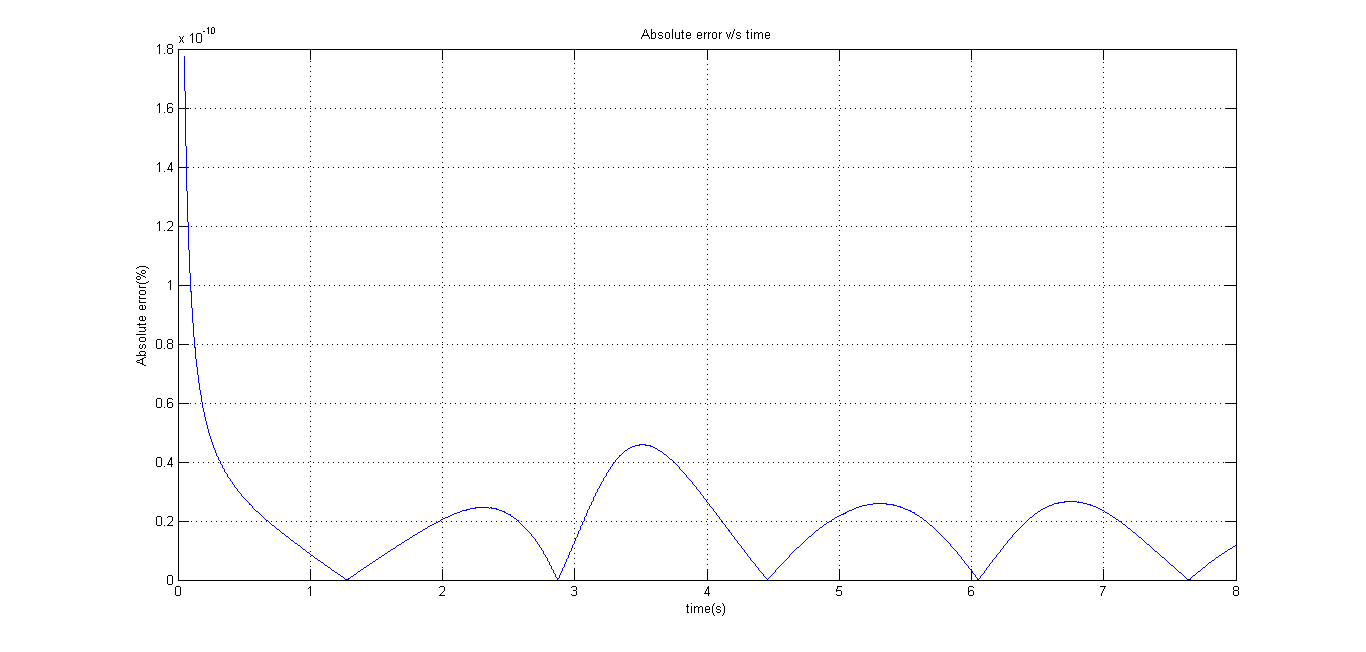


Fig 9. Absolute error v/s time for ∆t = 0.001

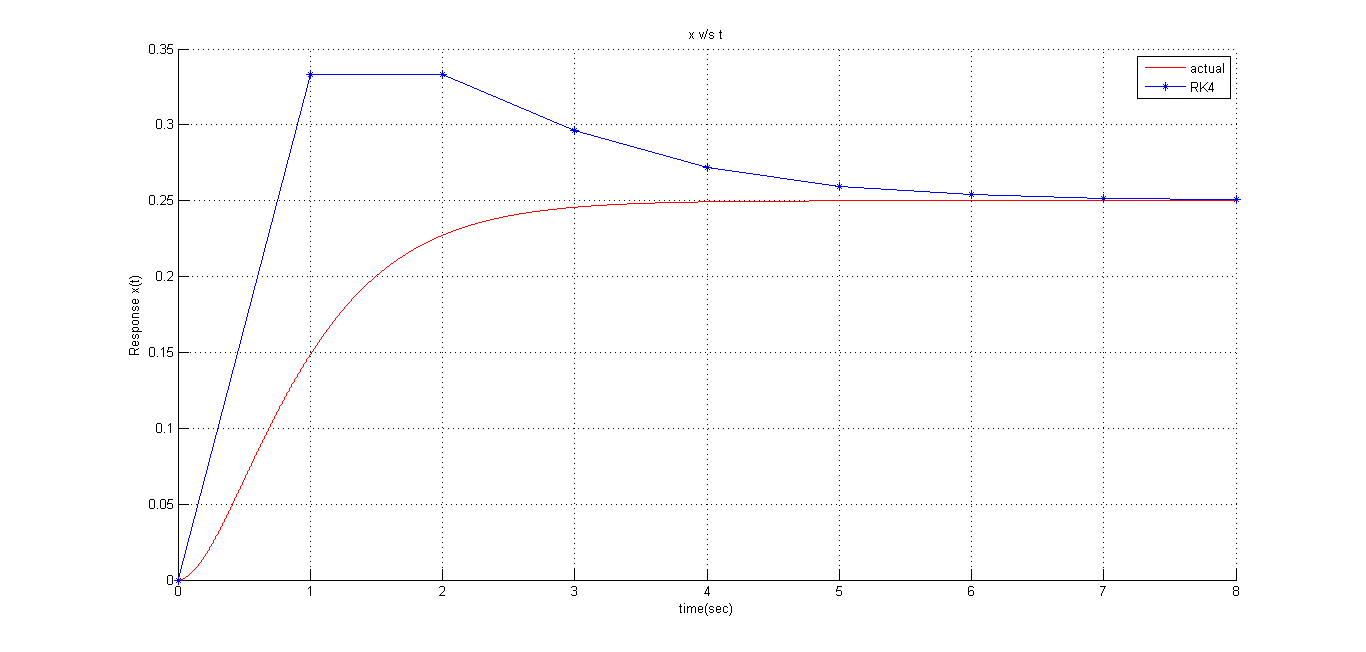
**Case 2**

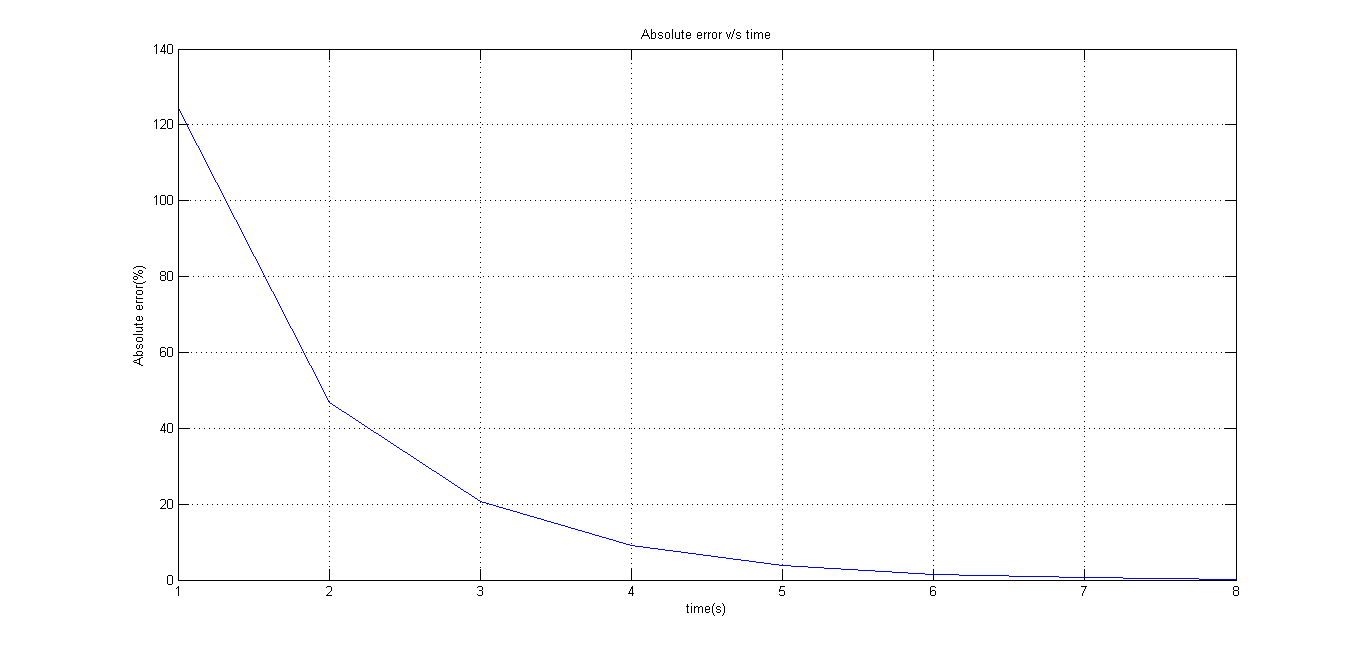
In this case, . The analytical solution for this case is given by the equation

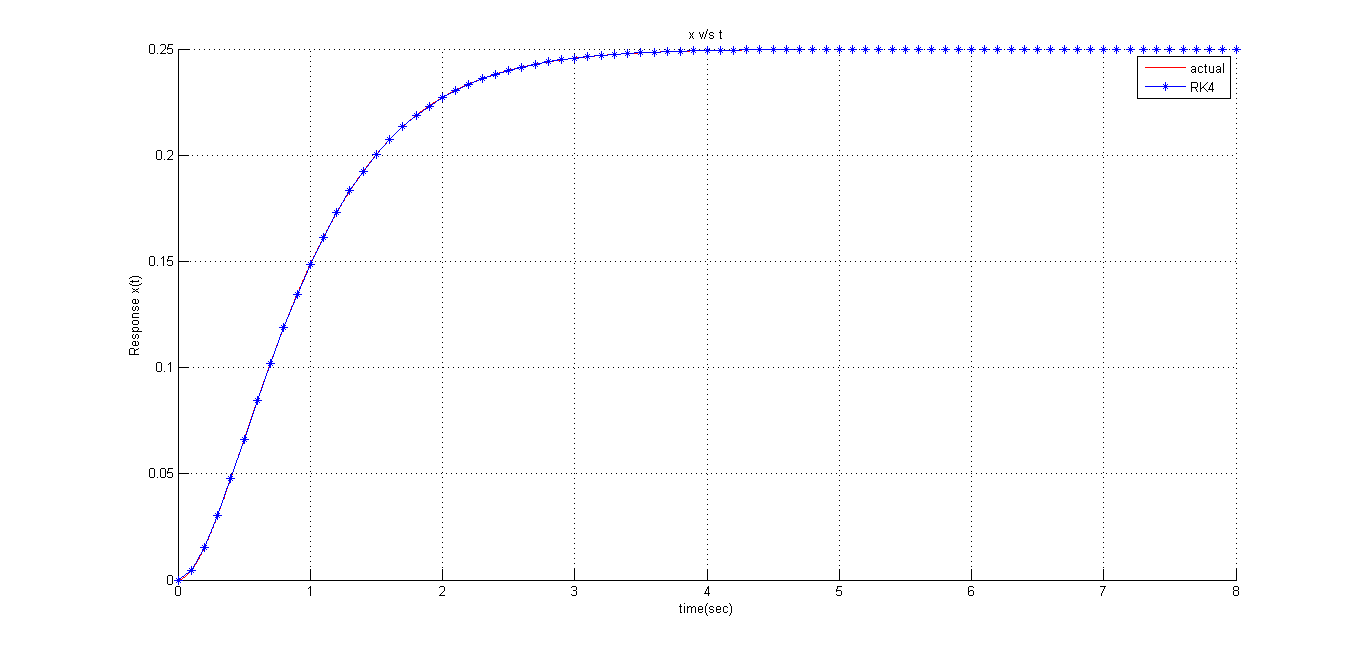
Eq.23

where, , ,

With Eq.23 as the actual solution and ∆t = 1,0.1,0.01,0.001 Fig 10,12,14,16 are presented as x v/s t plots. Fig. 11,13,15,17 denote percentage absolute value of error as a function of time. It is also evident from these figures that the error is reduced as ∆t decreases. Unlike the previous case, here, the error is not oscillating and always tending to zero.

Fig 10 x v/s t ∆t = 1

Fig 11 Absolute error v/s time for ∆t = 1

Fig 12 x v/s t ∆t = 0.1

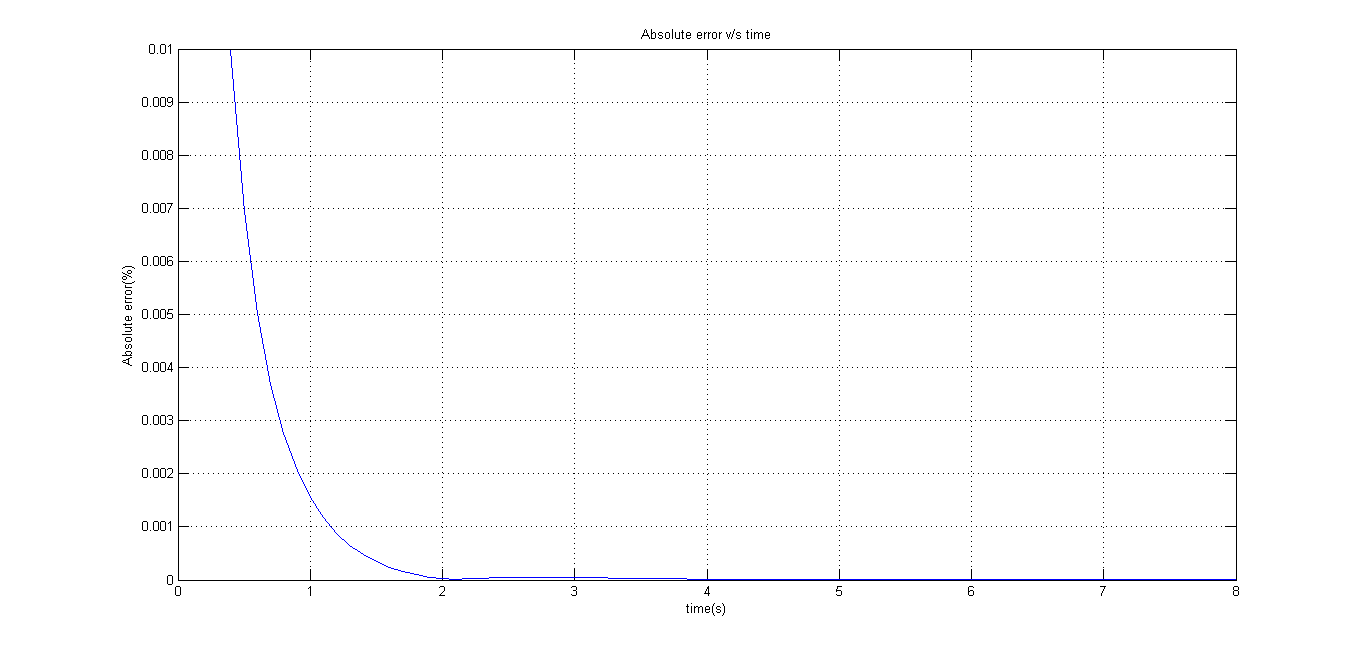
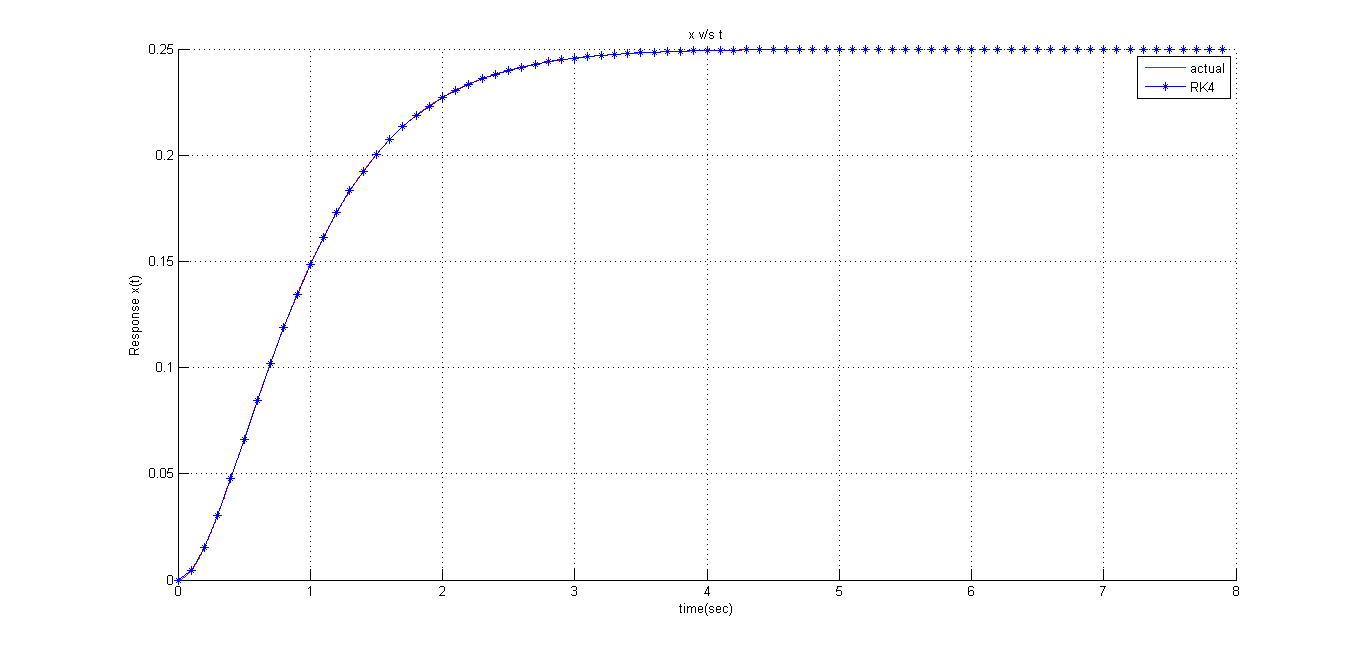


Fig 13 Absolute error v/s time for ∆t = 0.1

Fig 14 x v/s t ∆t = 0.01

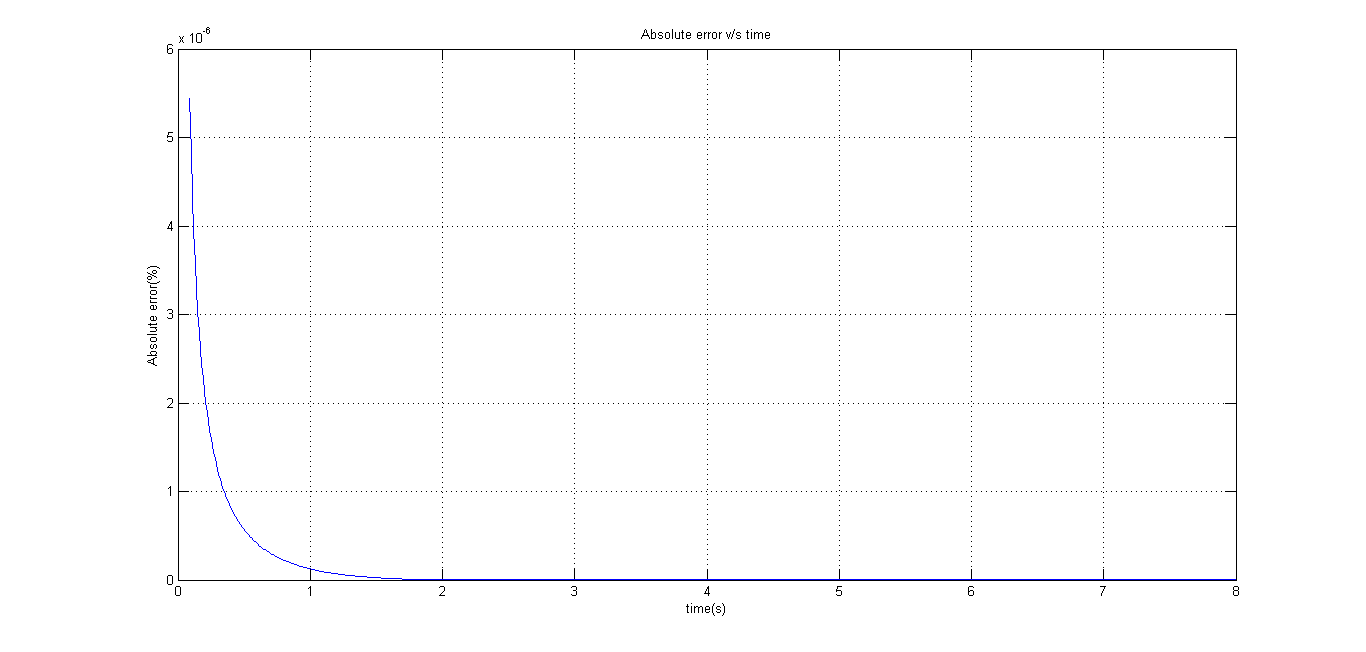
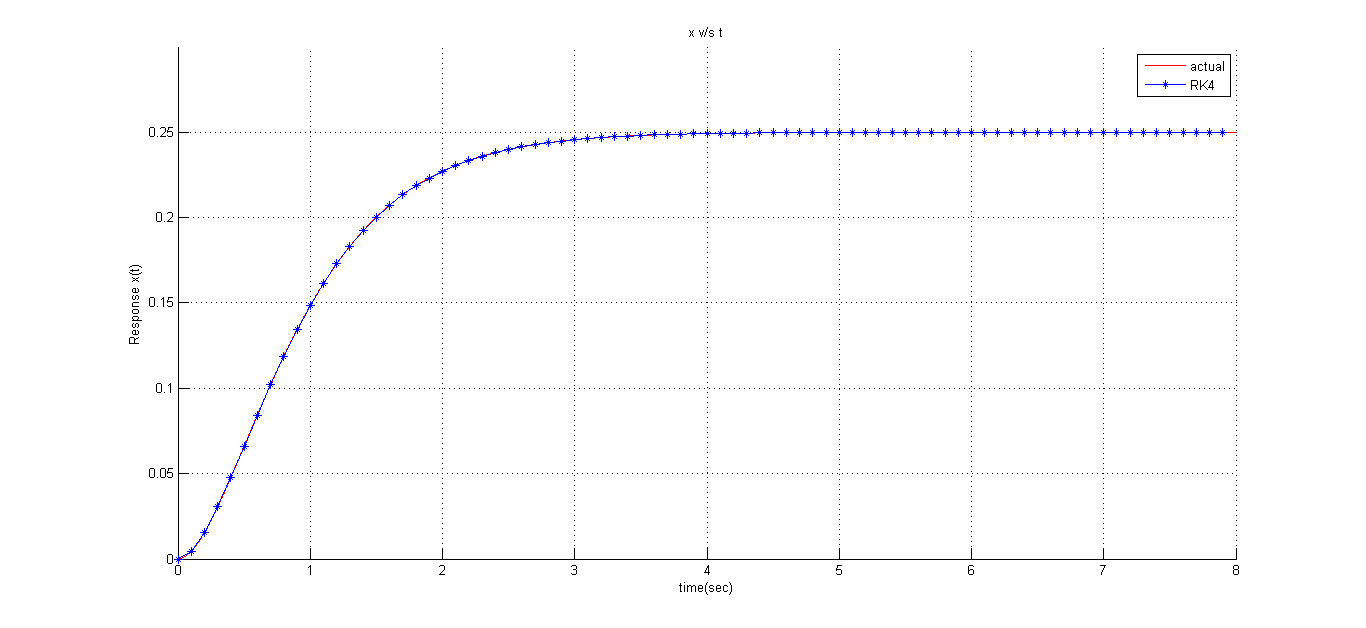
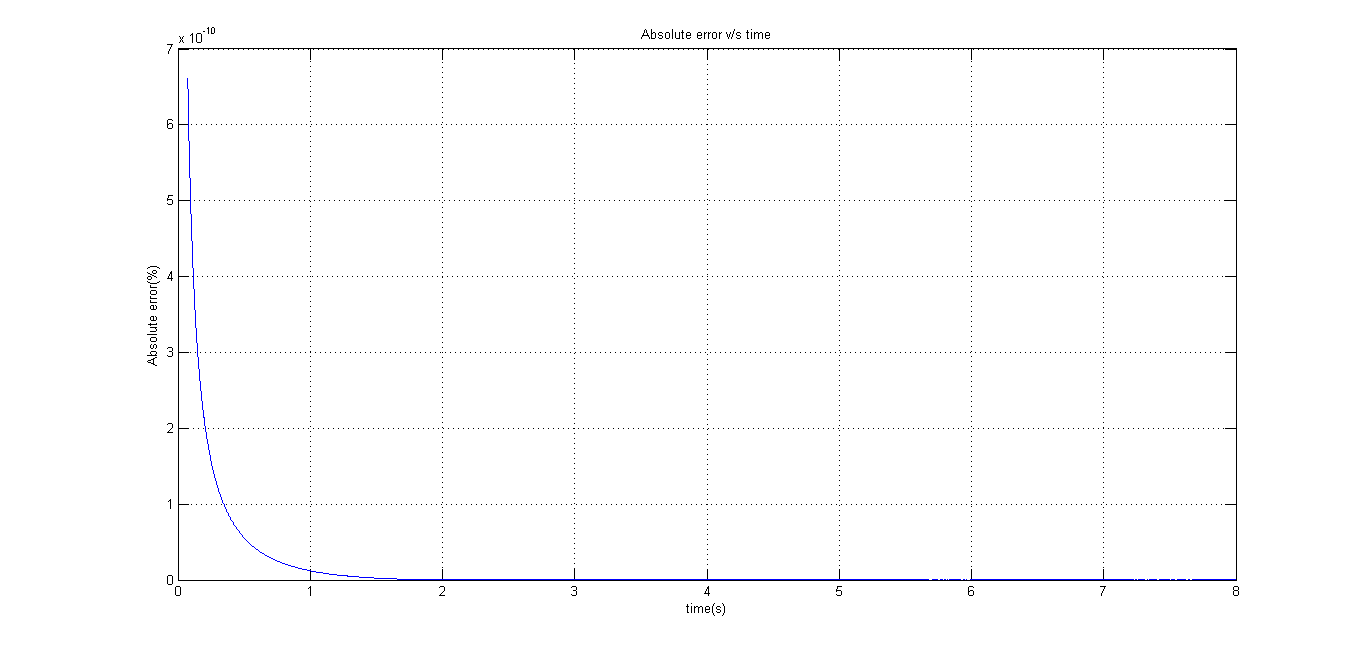
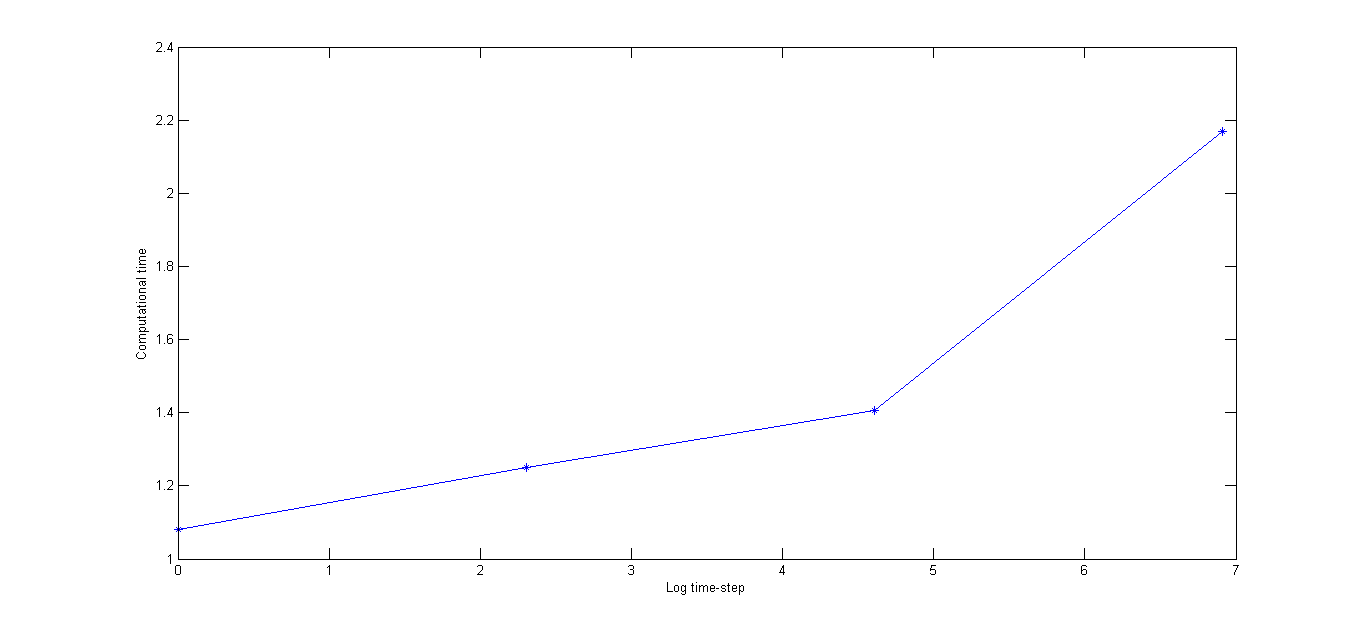


Fig 15 Absolute error v/s time for ∆t = 0.01

Fig 16 x v/s t ∆t = 0.001

Fig 17 Absolute error v/s time for ∆t = 0.001

Fig 18 Figure denoting computational time with varying time step

From Fig 18 it is evident that the computational time increases with decreasing time step. Time step is denoted on a logarithmic scale and computational time seems to increase exponentially with increasing time step.

# 4 References

* Applied numerical methods for engineers and scientists, S S Rao, Prentice Hall, 2002

# 5 Appendix

**5.1 Fourth order Runge Kutta algorithm**

%Parameters for initiating variables

h = 0.001; %step-size

t = 0:h:8;%Spatial variable

t\_act = 0:0.001:8;

x(1) = 0; %intial position

v(1) = 0; %initial velocity

% x\_e(1) = 0;

% v\_e(1) = 0;

for i = 1:length(t)-1

k1 = feval(@derivative1,x(i),v(i));

l1 = feval(@derivative2,x(i),v(i));

k2 = feval(@derivative1,x(i)+0.5\*h\*l1,v(i)+0.5\*h\*k1);

l2 = feval(@derivative2,x(i)+0.5\*h\*l1,v(i)+0.5\*h\*k1);

k3 = feval(@derivative1,x(i)+0.5\*h\*l2,v(i)+0.5\*h\*k2);

l3 = feval(@derivative2,x(i)+0.5\*h\*l2,v(i)+0.5\*h\*k2);

k4 = feval(@derivative1,x(i)+h\*l3,v(i)+h\*k3);

l4 = feval(@derivative2,x(i)+h\*l3,v(i)+h\*k3);

v(i+1) = v(i) + (h/6)\*(k1+(2\*k2)+(2\*k3)+k4);

x(i+1) = x(i) + (h/6)\*(l1+(2\*l2)+(2\*l3)+l4);

end

k = 4;

m = 1;

f = 1;

B = 2\*sqrt(m\*k);

omega\_n = sqrt(k/m);

zeta = B/(2\*sqrt(m\*k));

omega\_d = omega\_n\*(sqrt(1-zeta^2));

phi = atan(zeta/sqrt(1-zeta^2));

%case1

for j=1:length(t\_act)

x\_actual(j) = (f/k)\*(1-(1+omega\_n\*t\_act(j))\*exp(-omega\_n\*t\_act(j)));

% x\_actual(j) = (f/k)\*(1-sqrt(1/(1-(zeta^2)))\*exp(-zeta\*omega\_n\*t\_act(j))\*cos(omega\_d\*t\_act(j)-phi));

end

**5.2 derivative1 function**

function[ z ] = derivative1(x,v)

m=1;

k=4;

f=1;

B=2\*(sqrt(m\*k));

z = (1/m)\*(f-(B\*v)-(k\*x));

end

**5.3 derivative2 function**

function[z] = derivative2(x,v)

z = v;

end