Jaime Frade
Computational Finance: Dr. Kopriva
Homework #6

1. Show that once can approximate the intial value problem for $y' = F(t,y)$ by

$$Y_{n+1} = Y_n + \Delta t\, F(t_{n+1/2}, Y_{n+1/2}) \tag{1}$$

where $t_{n+1/2} = t_n + \dfrac{\Delta t}{2}$.

NOTE: For Midpoint Rule over $[a,b]$

$$\int_a^b f(x)dx = (b-a)f\left(\frac{a+b}{2}\right) \tag{2}$$

Integrating $y' = F(t,y)$ over $[t_{n+1}, t_n]$, by using the midpoint quadrature will obtain:

$$\begin{aligned}\int_{t_n}^{t_{n+1}} y'dt &= \int_{t_n}^{t_{n+1}} F(t,y)dt \\ &= (t_{n+1}-t_n)F(t_{n+1/2}, y(t_{n+1/2})) + \tau_n \qquad \text{From (2)} \\ &= \Delta t\, F(t_{n+1/2}, y(t_{n+1/2})) + \tau_n \qquad \text{From } \Delta t = t_{n+1}-t_n\end{aligned} \tag{3}$$

where τ_n is the local truncation error. Define $\Delta t = (t_{n+1} - t_n)$ and $t_{n+1/2} = t_n + \dfrac{\Delta t}{2}$

From (3), obtain the following:

$$y(t_{n+1}) - y(t_n) = \Delta t\, F(t_{n+1/2}, y(t_{n+1/2})) + \tau_n \tag{4}$$

Define $y(t_n) = y_n \approx Y_n$, $y(t_{n+1}) = y_{n+1} \approx Y_{n+1}$, and $y(t_{n+1/2}) = y_{n+1/2} \approx Y_{n+1/2}$.
From above, can redefine (4) to obtain the following:

$$Y_{n+1} = Y_n + \Delta t\, F(t_{n+1/2}, Y_{n+1/2}) \tag{5}$$

Show that this approximation is second order.
I will use a-posterior approach using the Talyor Series. Let $\Delta t^* = \dfrac{\Delta t}{2}$.

$$y_{n+1} = y_{n+1/2} + \Delta t^*\, y'_{n+1/2} + (\Delta t^*)^2\,\frac{y''_{n+1/2}}{2!} + (\Delta t^*)^3\,\frac{y'''(\xi)}{3!} \tag{6}$$

$$y_n = y_{n+1/2} - \Delta t^*\, y'_{n+1/2} + (\Delta t^*)^2\,\frac{y''_{n+1/2}}{2!} - (\Delta t^*)^3\,\frac{y'''(\xi)}{3!} \tag{7}$$

Taking the difference between (6) and (7), will obtain:

$$\begin{aligned} y_{n+1} - y_n &= \Delta t\, y'_{n+1/2} + (\Delta t^*)^3 \frac{y'''(\xi)}{3} \\ &= \Delta t\, F(t_{n+1/2}, Y_{n+1/2}) + \underbrace{(\Delta t^*)^3 \frac{y'''(\xi)}{3}}_{=LTE}\end{aligned} \tag{8}$$

Comparing (8) and (4), obtain the following for the local truncation error(LTE):

$$\tau_n = O((\Delta t)^3) \tag{9}$$

The approximation when using (5) is second order from above (9)

2. To solve (5), one needs the value for $Y_{n+1/2}$. This can be approximated using Euler's Method to give the following scheme.

$$\begin{aligned} Y^*_{n+1/2} &= Y_n + \frac{\Delta t}{2}\, F(t_n, Y_n) \qquad (10)\\ Y_{n+1} &= Y_n + \Delta t\, F(t_{n+1/2}, Y^*_{n+1/2}) \qquad (11)\end{aligned}$$

Show that this method is second order. Let $\Delta t^* = \dfrac{\Delta t}{2}$.

From (10), can approximate with a local trunction error using the talyor series as:

$$y^*_{n+1/2} = y_n + (\Delta t^*) y'_n + \tau^* \qquad (12)$$

I will use a-posterior approach using the Talyor Series for $y_{n+1/2} \approx Y_{n+1/2}$.

$$\begin{aligned} y_{n+1/2} &= \underbrace{y_n + \Delta t^*\, y'_n}+(\Delta t^*)^2\, \frac{y''_n}{2!} + O((\Delta t^*)^3)\\ &= Y^*_{n+1/2} + \tau^* + (\Delta t^*)^2\, \frac{y''_n}{2!} + O((\Delta t^*)^3) \qquad \text{From (12)} \qquad (13)\end{aligned}$$

Using Talyor Series on to expand $F(t_{n+1/2}, Y^*_{n+1/2})$, from (11).

$$\begin{aligned} F(t_{n+1/2}, Y^*_{n+1/2}) &= F(t_{n+1/2}, Y_{n+1/2}) + (Y^*_{n+1/2} - \underbrace{Y_{n+1/2}}_{=(13)})\frac{\partial F}{\partial Y} + O(\underbrace{(Y^*_{n+1/2} - Y_{n+1/2})^2}_{(\Delta t^2)^2})\\ &= F(t_{n+1/2}, Y_{n+1/2}) + \left[Y^*_{n+1/2} - Y^*_{n+1/2} - (\Delta t^*)^2\, \frac{Y''_n}{2!} + O((\Delta t^*)^3)\right]\frac{\partial F}{\partial Y} + O((\Delta t)^4)\\ &= F(t_{n+1/2}, Y_{n+1/2}) + \left[-(\Delta t^*)^2\, \frac{Y''_n}{2!} + O((\Delta t^*)^3)\right]\frac{\partial F}{\partial Y} + O((\Delta t)^4) \qquad (14)\end{aligned}$$

Multiplying both sides of (14) by (Δt), will then subsitute into (11).

$$\begin{aligned} \Delta t\, F(t_{n+1/2}, Y^*_{n+1/2}) &= \Delta t\, F(t_{n+1/2}, Y_{n+1/2}) + \Delta t\, \left[-(\Delta t^*)^2\, \frac{Y''_n}{2!} + O((\Delta t^*)^3)\right]\frac{\partial F}{\partial Y} + O((\Delta t)^5)\\ &= \Delta t\, F(t_{n+1/2}, Y_{n+1/2}) + O((\Delta t^*)^3) \qquad (15)\end{aligned}$$

From (11) and using (14) from above, will obtain the local truncation error for this method of approximation

$$\begin{aligned} y_{n+1} - y_n &= \underbrace{\Delta t\, F(t_{n+1/2}, Y^*_{n+1/2})}_{=(15)} \qquad \text{From (11)}\\ &= \Delta t\, F(t_{n+1/2}, Y_{n+1/2}) + O((\Delta t^*)^3) \qquad (16)\end{aligned}$$

Obtain the following for the local truncation error(LTE) for the approximation which is second order:

$$\tau_n = O((\Delta t)^3) \qquad (17)$$

2. To solve (5), one needs the value for $Y_{n+1/2}$. This can be approximated using Euler's Method to give the following scheme.

$$\begin{aligned} Y^*_{n+1/2} &= Y_n + \frac{\Delta t}{2} F(t_n, Y_n) && (18) \\ Y_{n+1} &= Y_n + \Delta t \, F(t_{n+1/2}, Y^*_{n+1/2}) && (19) \end{aligned}$$

Find the region of absolute stability. Let $\Delta t^* = \dfrac{\Delta t}{2}$.

Suppose $y' = \lambda \, y$. Using (18) and (19) above, obtain the following:

$$\begin{aligned} Y_{n+1/2} &= Y_n + \Delta t^* \, \lambda \, Y_n && (20) \\ Y_{n+1} &= Y_n + \Delta t \, \lambda \, Y_{n+1/2} && (21) \end{aligned}$$

Subsitituting (20) into $Y_{n+1/2}$ of (21), obtain the following:

$$\begin{aligned} Y_{n+1} &= Y_n + \Delta t \, \lambda \, (Y_n + \Delta t^* \, \lambda \, Y_n) \\ &= Y_n \left(1 + \Delta t \, \lambda + \frac{(\Delta t \, \lambda)^2}{2} \right) && (22) \end{aligned}$$

For the solution no to blow up as $n \to \infty$, then $\left| 1 + \Delta t \, \lambda + \dfrac{(\Delta t \, \lambda)^2}{2} \right| \leq 1$

A method is <u>absolutely stable</u> for a time step Δt if $||Y_{n+1}|| \leq ||Y_n||$, when applied to

$$\begin{aligned} y' &= \lambda \, y \\ y(0) &= y_0 \end{aligned}$$

To find the region of absolute stabilty. Let $\Delta t \, \lambda = x + iy$. From above,

$$\begin{aligned} \left| 1 + \Delta t \, \lambda + \frac{(\Delta t \, \lambda)^2}{2} \right| &\leq 1 \\ \left| 1 + x + iy + \frac{(x+iy)^2}{2} \right| &\leq 1 \\ \left| 1 + x + iy + \frac{(x^2 + 2ixy - y^2)}{2} \right| &\leq 1 \\ \left| 1 + x + iy + \frac{x^2}{2} + ixy - \frac{y^2}{2} \right| &\leq 1 \\ \left| 1 + x + \frac{x^2}{2} - \frac{y^2}{2} + i(xy + y) \right| &\leq 1 \\ \left(1 + x + \frac{x^2}{2} - \frac{y^2}{2} \right)^2 + (xy + y)^2 &\leq 1 \\ \left[\frac{1}{2} \left(x^2 + 2x + \frac{1}{2} - y^2 \right) \right]^2 + y^2(x+1)^2 &\leq 1 \\ \frac{1}{4} \left[(x+1)^2 + \frac{1}{2} + \frac{1}{2} - y^2 \right]^2 + y^2(x+1)^2 &\leq 1 \\ \frac{1}{4} \left[(x+1)^2 + (1 - y^2) \right]^2 + y^2(x+1)^2 &\leq 1 \\ (x+1)^4 + 2(x+1)^2(1-y^2) + (1-y^2)^2 + 4y^2(x+1)^2 &\leq 4 \\ (x+1)^4 + (1-y^2) + 2(x+1)^2(1+y^2) &\leq 4 \end{aligned}$$

0. Executive Summary

In this assignment I developed a program which solve ordinary differential equations using the midpoint method of intergration. I will run several functions with known solutions to test my program. The accuracy of the program will be determined by checking the errors in the solutions and how the errors behave in each test case.

I will then apply my program to an equation which determines the value of a bond as a function of time and T be its maturity date. The estimated bond price was determined to be 780.023.

1. Statement of Problem

The problem in this assignment involves solving ordinary differential equation using the midpoint rule in integration. This rule was shown already to be second order, (17).

Using the scheme derived by the Euler Method from problem 2, (10) and (11) can solve the main problem of this homework.

The value of the bond is determined using the following ordinary differential equation:

$$\frac{dV}{dt} + K(t) = r(t)V \tag{23}$$

where $V(t)$, is the value of a bond as a function of time and T be its maturity date. $K(t)$ is a continuous coupon parment which is made furing the life of the bond. $r(t)$ is the bank interest rate that one would get during this time.

Suppose the following conditions apply:

$$\begin{aligned} T &= 5 && (24)\\ V(5) &= \$1,000.00 \quad \Rightarrow \text{ boundary condition} && (25)\\ r(t) &= 0.05 + 0.02\sin(\pi t) && (26)\\ K(t) &= T - t && (27) \end{aligned}$$

<u>**Question:**</u> Assuming (24) $\rightarrow$ (27), using the second order midpoint method **calculate** the value of the bond today, i.e., at time $t = 0$, accurate to <u>one cent</u>.

In the problem above, the conditions of the function indicate that applying a change of variables will give the solution for the value of the bond. With the advice of a colluege who is more knowledgable with in differential equations, I will change the boundary condition to an intial condition. I have seeked his tutoring in this area of ordinary differential equations. I will setup all my test functions in similiar manners.

The Value of the bond is a function of time. Given $T = 5$, the $V(5) = 1000.00$. From (24)-(25), this is the value of the function at the final time. I will change this to the intial condition and work backwards to my solution.

Change of variables:

$$\begin{aligned} \tau &= T - t && \Rightarrow T \text{ constant!}\\ \frac{d}{dt} &= \frac{d}{d\tau} && \Rightarrow \text{Sub intp (23) to obtain:}\\ -\frac{dV}{dt} + K(T-\tau) &= r(T-\tau)V && \Rightarrow \text{Solve.}\\ \frac{dV}{dt} &= K(T-\tau) - r(T-\tau)V && \Rightarrow \text{Value of bond as a function of } \tau \quad (28)\\ V(0) &= 1000 && \Rightarrow \text{intial condition} \quad (29) \end{aligned}$$

2. **Description of the Mathematics**

The method of solving the ordinary differential equation integration was based on the midpoint rule. It was already discussed in the earlier section of this assignment that the method is second order. See ealier section of this paper to see details on the order and the region of absolute stability.

Error Estimation From lecture notes, can calculate the error in our estimation using this method. Therefore, the error obtained in the numerical result should correspond to this value. Let C be a constant

$$\begin{aligned}
\overline{Y}(t) &= Y(t) + O(\Delta t^r) \\
\overline{Y}_{\Delta t}(t) &= Y(t) + C\Delta t^r \qquad \text{Course error} \qquad (30) \\
\overline{Y}_{\frac{\Delta t}{2}}(t) &= Y(t) + C\left(\frac{\Delta t}{2}\right)^r \qquad \text{Fine error} \qquad (31)
\end{aligned}$$

Taking the difference between (30) − (31), will obtain an a formula for the error estimation.

$$\begin{aligned}
\overline{Y}_{\Delta t}(t) - \overline{Y}_{\frac{\Delta t}{2}}(t) &= C\Delta t^r\left(1 - \frac{1}{2^r}\right) \\
&= C\Delta t^r\left(\frac{2^r - 1}{2^r}\right) \\
E_{\Delta t} \approx C\Delta t^r &= \left(\frac{2^r}{2^r - 1}\right)\left(\overline{Y}_{\Delta t}(t) - \overline{Y}_{\frac{\Delta t}{2}}(t)\right) \qquad (32) \\
E_{\frac{\Delta t}{2}} &\approx \left(\frac{1}{2^r - 1}\right)\left(\overline{Y}_{\Delta t}(t) - \overline{Y}_{\frac{\Delta t}{2}}(t)\right) \qquad (33)
\end{aligned}$$

3. **Description of the Algorithm**

In this program there was a need to write a function which takes in a mathematical function, intial values at an intial time, the final time, and the lengths of each time step. With this input the, the function will solve the ordinary differential equation.

The following is the alogorithm which was used to write the program. It is based on the equations derived in part 2 of this assignment.

```
ODE (function, intial value, intial time, final time, $\Delta t$)
{
  $Y_n$ = intial value                  // ($y_0$)
  $t_n$ = intial time                   // ($t_0$)
  WHILE ($t_n$ < final time)
  {
    IF ((final time-$t_n$) < $\Delta t$
      $\Delta t$ = final time $-t_n$
    $t_{n+1/2} = t_n + \frac{1}{2}\Delta t$
    $Y_{n+1/2} = Y_n + \frac{1}{2}\,\Delta t\, F(t_n, Y_n)$
    $t_{n+1} = t_{n+1/2} + \frac{1}{2}\Delta t$
    $Y_{n+1} = Y_n + \Delta t\, F(t_{n+1/2}, Y_{n+1/2})$
    $t_n = t_{n+1}$
    $Y_n = Y_{n+1}$
  }
  RETURN $Y_n$
}
```

4. **Results**

Rationale for test cases

- Each test case 1-5 were chosen where the exact value was known, done so to compare with the numerical result and expected error.
- Cases 1-4: Each solution was setup with an intial condition such that $Y(0) = 1$
- Test cases 1 and 2 were chosen to be $\leq$ to the order of the method used. This was done so to illustrate that the error will be zero.
- Test case 3 was chosen to be same degree as the error, so therefore the error will be constant.
- The last case is the question of this assignment,
- I tried to find a similiar test case as in Test case Wilmott, pg 267, see (17.3), where I have the find the value of a sero coupon bond, with $K = 0$, this has made the dependence on T **explicit.** However, I spent more time trying to find an error estimation and no time. If I were to rerun, I would first solve the error estimation and find the value of a zero coupon bond.
- I tested a function with the based on trigonmetic function because the homework problem has interest fluctuating as this type of function.
-

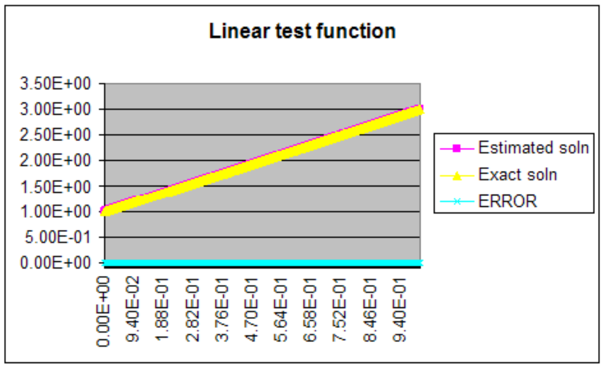
<u>Table 1</u>

Functions	Δt	final t	Estimated soln	Exact	**ERROR**
$\frac{dY}{dt} = 2$ $Y(0) = 1$ **Soln:** $Y(t) = 2t + 1$	0.001	1	3	3	9.2559631349e+061
$\frac{dY}{dt} = 2t + 4$ $Y(0) = 1$ **Soln:** $Y(t) = t^2 + 4t + 1$	0.001	1	6	6	9.2559631349e+061
$\frac{dY}{dt} = 3t^2 + 2t + 4$ $Y(0) = 1$ **Soln:** $Y(t) = t^3 + t^2 + 4t + 1$	0.001	1	7	7	9.2559631349e+061
$\frac{dY}{dt} = y$ $Y(0) = 1$ **Soln:** $Y(t) = e^t$	0.001	1	2.7182813758	2.7182818285	4.5270713356E-07
$\frac{dY}{dt} = 2\pi\cos(2\pi t)$ $Y(0) = 1$ **Soln:** $Y(t) = \sin(2\pi\, t) + 1$	0.001	1	1	1.0000000000e+000	-9.6496110100e-005
$\frac{dV}{dt} + K(t) = r(t)V$ $r(t) = 0.05 + 0.02\sin(\pi\, t)$ $K(t) = T - t$ $V(5) = 1000$	0.001	5		780.39626117	

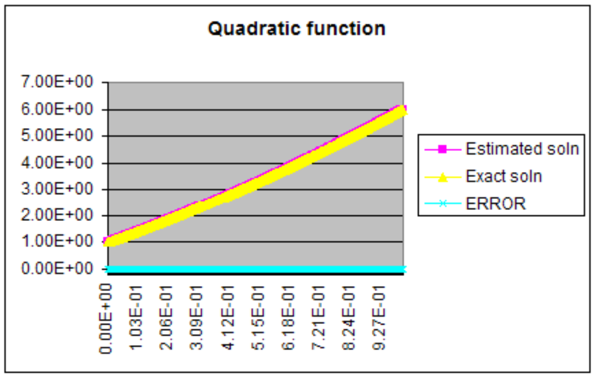
Rationale for graphs

- Each graph of the exact function for test cases 1-5.
- The numerical solution and the exact solution were graphed.
- From the graphs below, each numerical solution was obtained as the exact solution
- Test cases 1 and 2, were functions with degree less than or equal to the order of the method. Therefore, expect the error to be zero. This is illustrated by plotting the difference in the numerical and exact solution.
- Test case 3 illustrates the error is constant because the function is cubic. The plot in the error shows a constant increase in the error. The error is cumulative, and has a positive correlation with time.
- Each graph was obtained by printing the value obtained at each time step in the program. I then calculated the exact solution with the same input values. A comparsion was made for the error.
- Note graph 7-9. It was iteresting to see how error behaves as a function of time. I graphed the error at each period. You notice that the error is a maximum at the previous time step zero. I wanted to work more on this function.

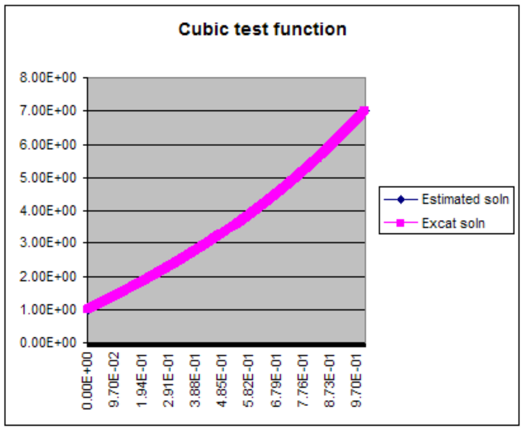
Graph 1



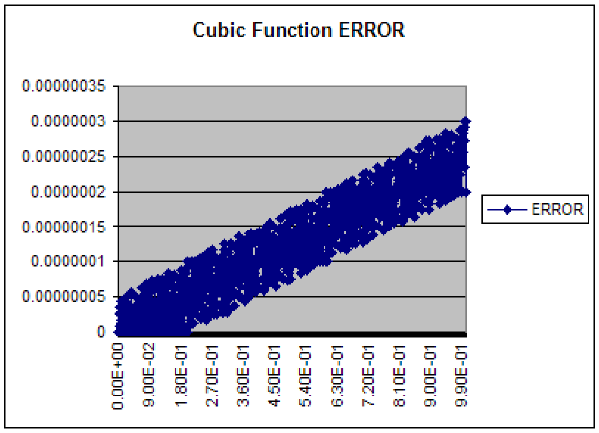
Graph 2



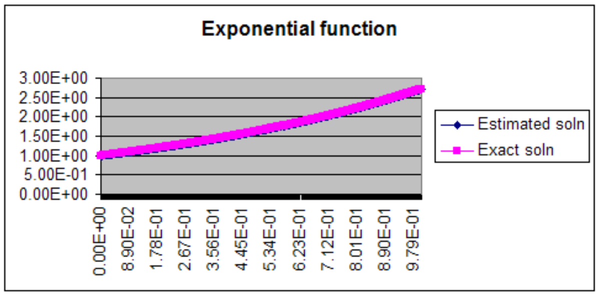
Graph 3



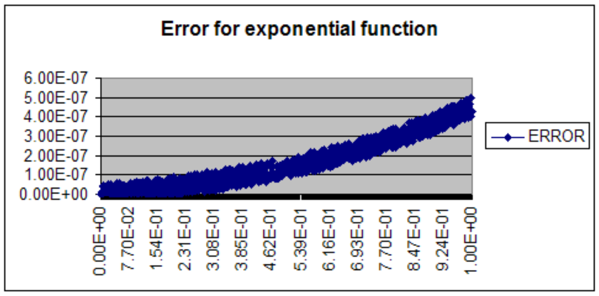
Graph 4



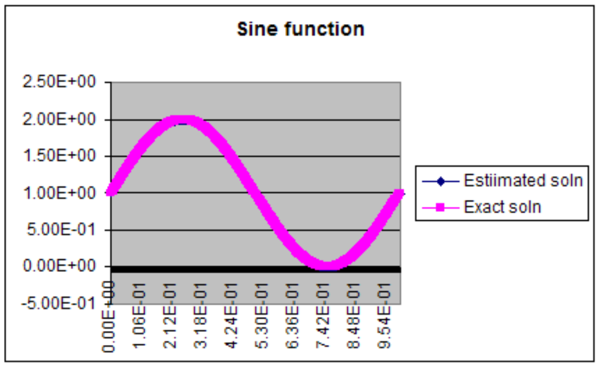
Graph 5



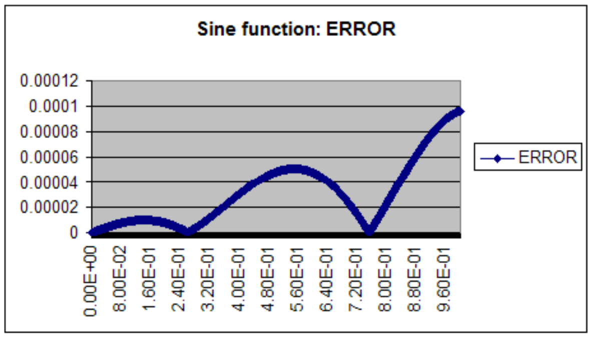
Graph 6



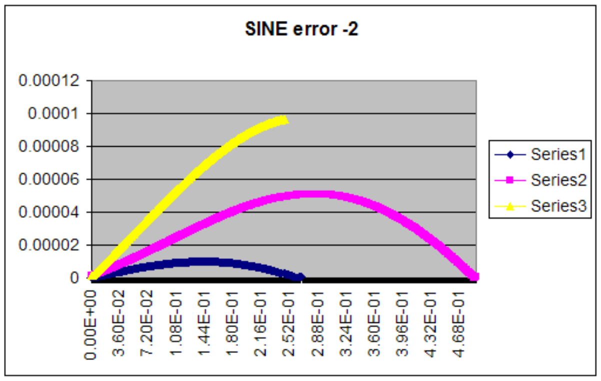
Graph 7



Graph 8



Graph 9



5. **Conclusion**

All the test case functions were chosen such that the solution can be easily calculated. From the table, the numerical solution of each test function were almost the same, all less than 10^{-7}. If I make the increment of time smaller, these error decrease.

As stated earlier, test cases 1 and 2 have errors close to machine epsilon for double precision because the method used in this assignment was second order. The solution obtained should be exact for these two cases. My program was *a little off*; this may be due to my choice for Δt or rounding errors.

I tried to look at the error estimation, however, my program was not working and I could not print out the error for each function. I wanted to look at the error of test case 4 and compare the error from the estimation to the error of the numerical result. I wanted to check the order of the error. If obtain this, I will email it to you?

For the solution to be accurate within one cent, I performed the error estimation. I was unable to compare with a exact solution, I do hope my tests proved my numerical result is acceptable. With a time increment of $\Delta t = 0.001$, I obtained the value of the bond today is \$780.396.

APPENDIX: 1 Main program

```
#include <iostream>
using namespace std;
#include "math.h"
#include "stdio.h"
#include "ode.h"
#include "funcs_ode.h"

void main()
{
double solution1;
double solution2;
double solution3;
double solution4;
double solution5;
double solution6;
double t_0, t_final, deltat;
double y_intial;
double true_exact;

FILE *file1;
FILE *file2;
FILE *file3;


t_0 = 0.0;


file1=fopen("odesolutions1.txt","w");   //prints soln at each time step
file2=fopen("odesolutions2.txt","w");  //prints final error
file3=fopen("odesolutions3.txt","w");  //prints error


deltat=0.001;  //change this!
y_intial=1.0;
t_final = 1.0;
true_exact=3.0;
solution1 = ODE(f1, y_intial, t_0, t_final, deltat, file1, file3);
fprintf(file2, "%17.10e %17.10e %17.10e %17.10e %17.10e \n", t_final,
deltat, true_exact, solution6, true_exact-solution6);

t_final = 1.0;
true_exact=6.0;
solution2 = ODE(f2, y_intial, t_0, t_final, deltat, file1, file3);
fprintf(file2, "%17.10e %17.10e %17.10e %17.10e %17.10e \n", t_final,
deltat, true_exact, solution6, true_exact-solution6);

t_final = 1.0;
true_exact=7.0;
solution3 = ODE(f3, y_intial, t_0, t_final, deltat, file1, file3);
fprintf(file2, "%17.10e %17.10e %17.10e %17.10e %17.10e \n", t_final,
deltat, true_exact, solution6, true_exact-solution6);
t_final = 1.0;
true_exact=exp(1);
solution4 = ODE(f4, y_intial, t_0, t_final, deltat, file1, file3);
fprintf(file2,"%17.10e %17.10e %17.10e %17.10e %17.10e \n", t_final,
deltat, true_exact, solution4, true_exact-solution4);


t_final = 1.0;
true_exact=1;
solution5 = ODE(f5, y_intial, t_0, t_final, deltat, file1, file3);
fprintf(file2, "%17.10e %17.10e %17.10e %17.10e %17.10e \n", t_final,
deltat, true_exact, solution5, true_exact-solution5);

t_0=0.0;
t_final = 5.0;
y_intial = 1000.0;


true_exact=0;
solution6 = ODE(f6, y_intial, t_0, t_final, deltat, file1, file3);
fprintf(file2, "%17.10e %17.10e %17.10e %17.10e %17.10e \n", t_final,
deltat, true_exact, solution6, true_exact-solution6);



cout << solution1 << endl;
cout << solution2 << endl;
cout << solution3 << endl;
cout << solution4 << endl;
cout << solution5 << endl;
cout << solution6 << endl;

}
```

APPENDIX: 2 Ordinary differential EQ.h

```
#include "stdio.h"

double ODE (double (*f) (double, double),  double y_i, double t_0, double t_final, double dt, FILE
{
double y_n, y_next1,y_mid;
double t_n, t_next1, t_mid;

float error;
float r=3.0;
float Tol=1.e-7;

y_n=y_i;
t_n=t_0;

fprintf(file1, "%15.7e %15.7e \n",t_n,y_n);

while(t_n < t_final)
{
if ((t_final-t_n)<dt)
dt = t_final - t_n;

t_mid = t_n + 0.5*dt;
y_mid = y_n + (0.5*dt)*f(t_n, y_n);

t_next1 = t_mid+0.5*dt;
y_next1 = y_n+dt*f(t_mid,y_mid);

t_n = t_next1;
y_n = y_next1;

fprintf(file1, "%15.7e %15.7e \n", t_n, y_n);
}

error = (1/(pow(2,r)-1.0))*(y_n-y_next1);
if (fabs(error)>Tol)
{
 dt = 0.5*dt;
}


fprintf(file1, "\n");

return y_n;
}
```

APPENDIX: 3 Test cases

```
#include "math.h"

//Linear function
double f1(double t, double y)
{
return 2;
}
//quadratic function
double f2(double t, double y)
{
return 2.0*t+4;
}

//cubic function
double f3(double t, double y)
{
return 3.0*pow(t,2)+2*t+4;
}



//exponential function
double f4(double t, double y)
{
return y;
}

//trig function
double f5(double t, double y)
{

float pi; //= 3.141592654;
pi = 3+sqrt(5);
pi = pi*3;
pi = pi/5;
return 2*pi*cos(2*pi*t);
}

//homework prob
double f6(double t, double y)
{
double Kt;
double r_t;
double V;
double T;
float pi; //= 3.141592654;
pi = 3+sqrt(5);
pi = pi*3;
pi = pi/5;

T =5.0;
r_t = 0.05+0.02*sin(pi*(T-t));
Kt=t;
V=Kt-r_t*y;

return V;
}
```