MA 322: Lab Assignment #7

Due on Sunday, October 10, 2015

Jiten Chandra Kalita

Silvi Pandey-130123045

Contents

PROBLEM 1

The one-dimensional radiation problem is described by :-

$$T' = -\alpha(T^4 - T_a^4) = f(t,T)$$

$$T(0) = T_0 = 2500, T_a = 250, \alpha = 2.0 * 10^{-12}$$

(a) Find the exact solution of the above problem at t = 1,2,3...10 using secant method

(b) Solve the ODE numerically by the

(i) Euler explicit

(ii) Euler implicit

(iii) Modified Euler

(iv) Fourth order Runge-Kutta method

using Δt = 2 , 1 , 0.05 , 0.025 and 0.01.

(c) Plot graph for each of the methods.

(d) Check for the rate of convergence of the methods

SOLUTION

(a) CODE

```
#include <bits/stdc++.h>

using namespace std;
double T0 = 2500;
double Ta = 250;
double alpha = 2*pow(10,-12);
double function(double t,double T);
double functionDer(double t,double pre,double prepre);
double secantMethod(double t);
int main()
{
    double t,valT,valFunc;
    FILE *fp;
    fp = fopen("ExactVal.txt","w");
    for(t=1;t<=10;t++)
    {
        valT = secantMethod(t);
        valFunc = function(t,valT);
     fprintf(fp,"%.0lf %lf\n",t,valT);
        printf("For t: %.0lf , T: %lf f(%.0lf): %lf\n",t,valT,t,valFunc);
    }
    fclose(fp);
}
double functionDer(double t,double pre,double prepre)
{
    return (function(t,pre)-function(t,prepre))/(pre-prepre);
}
double function(double t,double T)
{
    double val = ((T0-Ta)*(T+Ta))/((T-Ta)*(T0+Ta));
    return atan(T/Ta) - atan(T0/Ta) + 0.5*log(val) - 2*alpha*Ta*Ta*Ta*t;
}
double secantMethod(double t)
```

```
{
    int maxIter = 100000;
    int Iter = 0;
    double prepre,pre,curr;
    prepre = T0;
    pre = T0+1;
    while(maxIter--)
    {
        curr = pre - function(t,pre)/functionDer(t,pre,prepre);
        prepre = pre;
        pre = curr;
        if(fabs(curr-pre)/fabs(pre) <= 0.000001)
            break;
    }
    return curr;
}
```

OUTPUT

For t: 1 , T: 2421.820304 f(1): 0.000004
For t: 2 , T: 2343.640608 f(2): 0.000018
For t: 3 , T: 2265.460913 f(3): 0.000042
For t: 4 , T: 2187.281217 f(4): 0.000079
For t: 5 , T: 2109.101521 f(5): 0.000131
For t: 6 , T: 2030.921825 f(6): 0.000202
For t: 7 , T: 1952.742129 f(7): 0.000295
For t: 8 , T: 1874.562433 f(8): 0.000415
For t: 9 , T: 1796.382738 f(9): 0.000568
For t: 10 , T: 1718.203042 f(10): 0.000762

Where f is the error .

(b)
CODE

```
#include<bits/stdc++.h>

using namespace std;
double alpha = 2*pow(10,-12);
double T0 =2500;
double Ta = 250;
double function(double T);
double functionSecant(double delt,double Tpre,double preT);
double functionDerSecant(double delt,double Tpre,double preT,double prepreT);
double secantMethod(double delt,double Tpre);
int main()
{
    double delt[] = {2,1,0.05,0.025,0.01};
    double currT,preT;
    FILE *fp1,*fp2,*fp3,*fp4;
    fp1 = fopen("ExplicitEuler.txt","w");
    fp2 = fopen("ImplicitEuler.txt","w");
    fp3 = fopen("ModifiedEuler.txt","w");
    fp4 = fopen("RungeKutta.txt","w");
    cout<<"Euler Explicit\n";
    for(int j=0;j<5;j++)
    {
        preT = T0;
        printf("For delt = %lf\n",delt[j]);
        for(int i=1;i<=10;i++)
        {
            currT = preT + function(preT)*delt[j];
            if(delt[j] == 1)
          fprintf(fp1,"%d %lf\n",i,currT);
            printf("T(%lf) : %lf\n",delt[j]*i,currT);
            preT = currT;
        }
        cout<<endl;
    }
    fclose(fp1);
    cout<<"Euler Implicit\n";
    for(int j=0;j<5;j++)
    {
        preT = T0;
        printf("For delt = %lf\n",delt[j]);
        for(int i=1;i<=10;i++)
        {
            currT = secantMethod(delt[j],preT);
         if(delt[j] == 1)
          fprintf(fp2,"%d %lf\n",i,currT);
            printf("T(%lf) : %lf\n",delt[j]*i,currT);
            preT = currT;
        }
        cout<<endl;
    }
    fclose(fp2);
    double halfp;
    cout<<"Modified Euler\n";
    for(int j=0;j<5;j++)
    {
        preT = T0;
        printf("For delt = %lf\n",delt[j]);
        for(int i=1;i<=10;i++)
        {
            halfp = preT + delt[j]*0.5*function(preT);
            currT = preT + delt[j]*function(halfp);
         if(delt[j] == 1)
          fprintf(fp3,"%d %lf\n",i,currT);
            printf("T(%lf) : %lf\n",delt[j]*i,currT);
            preT = currT;
        }
        cout<<endl;
    }
    fclose(fp3);
    cout<<"Fourth order Runge Kutta\n";
    double dely1,dely2,dely3,dely4,val;
    for(int j=0;j<5;j++)
    {
        preT = T0;
        printf("For delt = %lf\n",delt[j]);
        for(int i=1;i<=10;i++)
        {
            dely1 = delt[j]*function(preT);
            dely2 = delt[j]*function(preT+dely1*0.5);
            dely3 = delt[j]*function(preT+dely2*0.5);
            dely4 = delt[j]*function(preT+dely3);
            val = (dely1 + 2*dely2 + 2*dely3 + dely4)/6;
            currT = preT + val;
        if(delt[j] == 1)
          fprintf(fp4,"%d %lf\n",i,currT);
            printf("T(%lf) : %lf\n",delt[j]*i,currT);
            preT = currT;
        }
        cout<<endl;
    }
    fclose(fp4);
}
double secantMethod(double delt,double Tpre)
{
    double currT,prepreT,preT;
    preT = Tpre;
    prepreT = preT + 10;
    int MaxIter = 100;
    while(MaxIter--)
    {
        currT = preT -functionSecant(delt,Tpre,preT)/functionDerSecant(delt,Tpre,preT,
        prepreT);
        if(fabs(currT-preT)/fabs(preT) <= 0.000001)
            break;
     prepreT = preT;
        preT = currT;
    }
    return currT;
}
double functionDerSecant(double delt,double Tpre,double preT,double prepreT)
{
    return (functionSecant(delt,Tpre,prepreT) -functionSecant(delt,Tpre,preT))/
    (prepreT-preT);
}
double functionSecant(double delt,double Tpre,double preT)
{
    return (preT - Tpre) + alpha*delt*(pow(preT,4) - pow(Ta,4));
}
double function(double T)
{
    return -alpha*(pow(T,4)-pow(Ta,4));
}
```

OUTPUT

Euler Explicit

```
For delt = 2.000000
T(2.000000) : 2343.765625
T(4.000000) : 2223.078626
T(6.000000) : 2125.397688
T(8.000000) : 2043.788762
T(10.000000) : 1974.012648
T(12.000000) : 1913.290381
T(14.000000) : 1859.703691
T(16.000000) : 1811.874488
T(18.000000) : 1768.780668
T(20.000000) : 1729.644115


For delt = 1.000000
T(1.000000) : 2421.882812
T(2.000000) : 2353.082061
T(3.000000) : 2291.773242
T(4.000000) : 2236.609328
T(5.000000) : 2186.568703
T(6.000000) : 2140.859012
T(7.000000) : 2098.853963
T(8.000000) : 2060.050413
T(9.000000) : 2024.038418
T(10.000000) : 1990.479813


For delt = 0.050000
T(0.050000) : 2496.094141
T(0.100000) : 2492.212636
T(0.150000) : 2488.355220
T(0.200000) : 2484.521634
T(0.250000) : 2480.711620
T(0.300000) : 2476.924925
T(0.350000) : 2473.161300
T(0.400000) : 2469.420501
T(0.450000) : 2465.702285
T(0.500000) : 2462.006415


For delt = 0.025000
T(0.025000) : 2498.047070
T(0.050000) : 2496.100236
T(0.075000) : 2494.159465
T(0.100000) : 2492.224723
T(0.125000) : 2490.295978
T(0.150000) : 2488.373197
T(0.175000) : 2486.456349
T(0.200000) : 2484.545400
T(0.225000) : 2482.640320
```

```
T(0.250000) : 2480.741077


For delt = 0.010000
T(0.010000) : 2499.218828
T(0.020000) : 2498.438632
T(0.030000) : 2497.659410
T(0.040000) : 2496.881160
T(0.050000) : 2496.103879
T(0.060000) : 2495.327566
T(0.070000) : 2494.552219
T(0.080000) : 2493.777834
T(0.090000) : 2493.004411
T(0.100000) : 2492.231947
```

Euler Implicit

```
For delt = 2.000000
T(2.000000) : 2373.145960
T(4.000000) : 2267.431887
T(6.000000) : 2177.517153
T(8.000000) : 2099.773878
T(10.000000) : 2031.642170
T(12.000000) : 1971.258198
T(14.000000) : 1917.228793
T(16.000000) : 1868.489072
T(18.000000) : 1824.209295
T(20.000000) : 1783.732059


For delt = 1.000000
T(1.000000) : 2430.244100
T(2.000000) : 2367.426413
T(3.000000) : 2310.442637
T(4.000000) : 2258.420959
T(5.000000) : 2210.662659
T(6.000000) : 2166.600257
T(7.000000) : 2125.767391
T(8.000000) : 2087.776725
T(9.000000) : 2052.303498
T(10.000000) : 2019.073062


For delt = 0.050000
T(0.050000) : 2496.118345
T(0.100000) : 2492.260632
T(0.150000) : 2488.426606
T(0.200000) : 2484.616012
T(0.250000) : 2480.828603
T(0.300000) : 2477.064132
```

```
T(0.350000) : 2473.322358
T(0.400000) : 2469.603043
T(0.450000) : 2465.905952
T(0.500000) : 2462.230854


For delt = 0.025000
T(0.025000) : 2498.053147
T(0.050000) : 2496.112338
T(0.075000) : 2494.177540
T(0.100000) : 2492.248721
T(0.125000) : 2490.325848
T(0.150000) : 2488.408889
T(0.175000) : 2486.497813
T(0.200000) : 2484.592588
T(0.225000) : 2482.693183
T(0.250000) : 2480.799567


For delt = 0.010000
T(0.010000) : 2499.219803
T(0.020000) : 2498.440578
T(0.030000) : 2497.662325
T(0.040000) : 2496.885039
T(0.050000) : 2496.108720
T(0.060000) : 2495.333365
T(0.070000) : 2494.558972
T(0.080000) : 2493.785540
T(0.090000) : 2493.013065
T(0.100000) : 2492.241546
```

Modified Euler

```
For delt = 2.000000
T(2.000000) : 2362.398496
T(4.000000) : 2250.455756
T(6.000000) : 2156.911291
T(8.000000) : 2077.094837
T(10.000000) : 2007.851661
T(12.000000) : 1946.964139
T(14.000000) : 1892.821346
T(16.000000) : 1844.220548
T(18.000000) : 1800.242702
T(20.000000) : 1760.171468


For delt = 1.000000
T(1.000000) : 2426.651906
T(2.000000) : 2361.187059
```

```
T(3.000000) : 2302.238025
T(4.000000) : 2248.751789
T(5.000000) : 2199.902092
T(6.000000) : 2155.029954
T(7.000000) : 2113.602260
T(8.000000) : 2075.182256
T(9.000000) : 2039.408106
T(10.000000) : 2005.977029


For delt = 0.050000
T(0.050000) : 2496.106332
T(0.100000) : 2492.236811
T(0.150000) : 2488.391175
T(0.200000) : 2484.569168
T(0.250000) : 2480.770536
T(0.300000) : 2476.995032
T(0.350000) : 2473.242409
T(0.400000) : 2469.512426
T(0.450000) : 2465.804845
T(0.500000) : 2462.119432


For delt = 0.025000
T(0.025000) : 2498.050120
T(0.050000) : 2496.106310
T(0.075000) : 2494.168536
T(0.100000) : 2492.236766
T(0.125000) : 2490.310968
T(0.150000) : 2488.391109
T(0.175000) : 2486.477157
T(0.200000) : 2484.569080
T(0.225000) : 2482.666848
T(0.250000) : 2480.770428


For delt = 0.010000
T(0.010000) : 2499.219316
T(0.020000) : 2498.439607
T(0.030000) : 2497.660870
T(0.040000) : 2496.883102
T(0.050000) : 2496.106303
T(0.060000) : 2495.330470
T(0.070000) : 2494.555600
T(0.080000) : 2493.781693
T(0.090000) : 2493.008744
T(0.100000) : 2492.236754
```

Runge Kutta

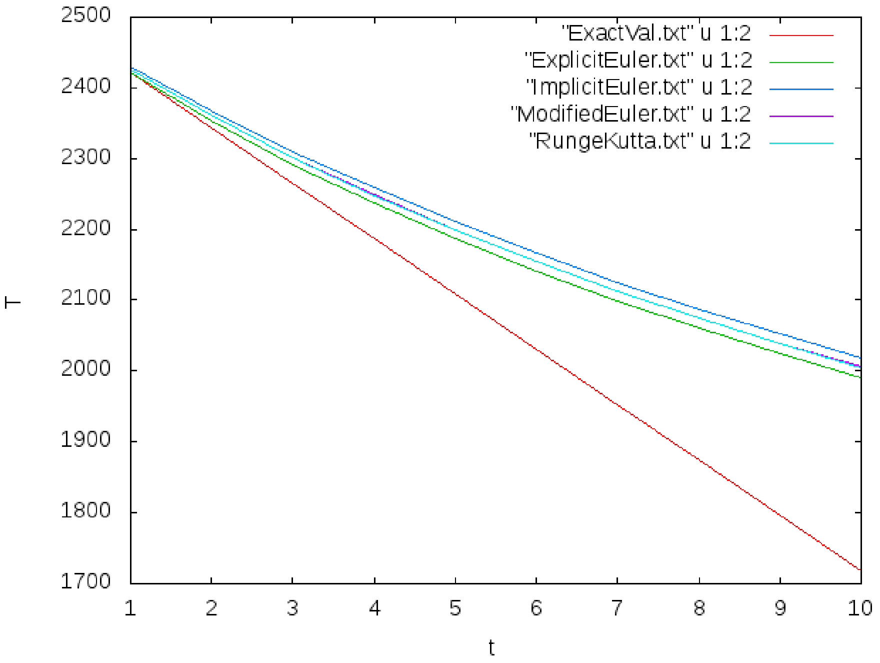
```
For delt = 2.000000
T(2.000000) : 2360.829563
T(4.000000) : 2248.246808
T(6.000000) : 2154.470302
T(8.000000) : 2074.611442
T(10.000000) : 2005.415952
T(12.000000) : 1944.618038
T(14.000000) : 1890.582525
T(16.000000) : 1842.094198
T(18.000000) : 1798.227583
T(20.000000) : 1758.263114


For delt = 1.000000
T(1.000000) : 2426.434864
T(2.000000) : 2360.829975
T(3.000000) : 2301.790735
T(4.000000) : 2248.247297
T(5.000000) : 2199.362653
T(6.000000) : 2154.470779
T(7.000000) : 2113.033845
T(8.000000) : 2074.611883
T(9.000000) : 2038.840832
T(10.000000) : 2005.416352


For delt = 0.050000
T(0.050000) : 2496.106302
T(0.100000) : 2492.236751
T(0.150000) : 2488.391087
T(0.200000) : 2484.569051
T(0.250000) : 2480.770393
T(0.300000) : 2476.994861
T(0.350000) : 2473.242212
T(0.400000) : 2469.512203
T(0.450000) : 2465.804597
T(0.500000) : 2462.119159


For delt = 0.025000
T(0.025000) : 2498.050116
T(0.050000) : 2496.106302
T(0.075000) : 2494.168525
T(0.100000) : 2492.236751
T(0.125000) : 2490.310949
T(0.150000) : 2488.391087
T(0.175000) : 2486.477131
T(0.200000) : 2484.569051
```

```
T(0.225000) : 2482.666816
T(0.250000) : 2480.770393
```

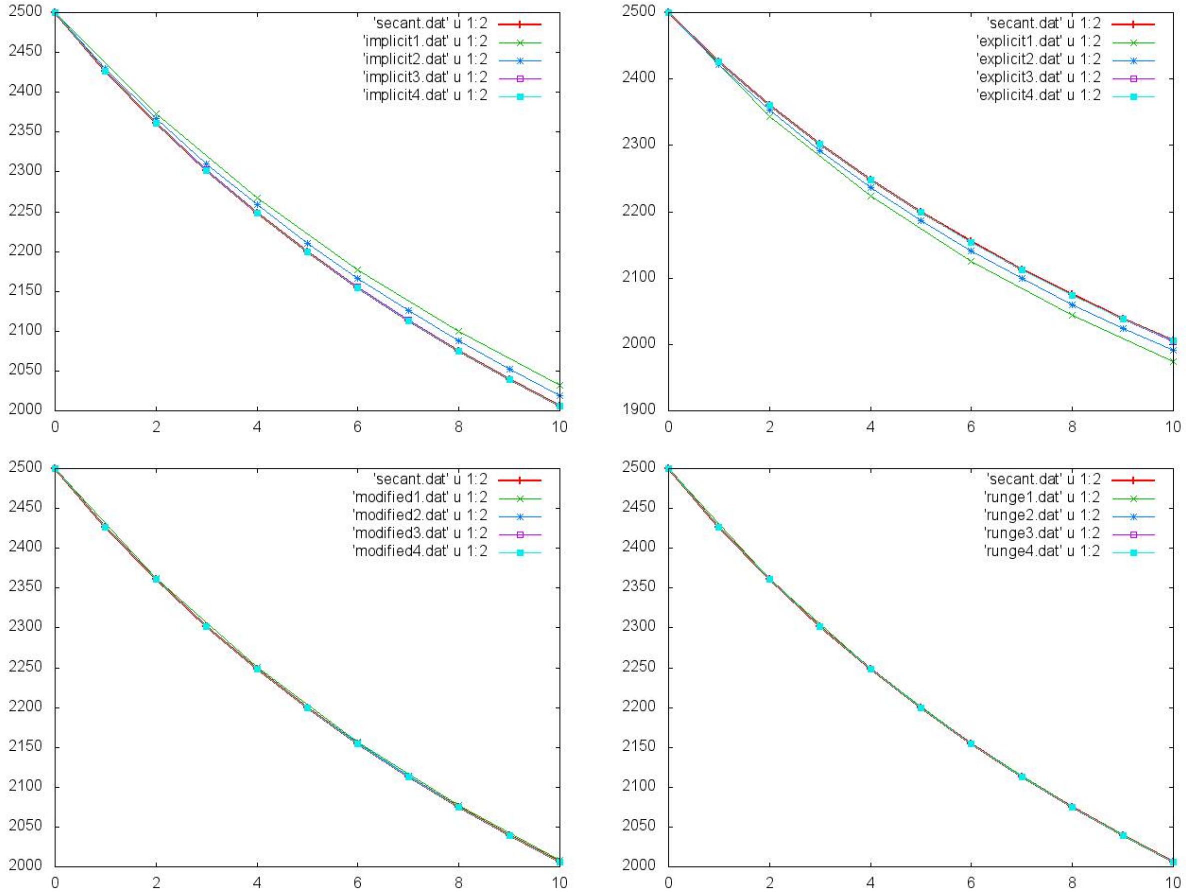
```
For delt = 0.010000
T(0.010000) : 2499.219316
T(0.020000) : 2498.439606
T(0.030000) : 2497.660869
T(0.040000) : 2496.883102
T(0.050000) : 2496.106302
T(0.060000) : 2495.330469
T(0.070000) : 2494.555599
T(0.080000) : 2493.781691
T(0.090000) : 2493.008742
T(0.100000) : 2492.236751
```

(c)

For $\Delta t = 1$



Four Plots :



EXPLANATION

(a) As evident from graph,the lesser the Δt,the better the results.

(b) Explicit Euler gives better results than other methods.This can be observed by looking at the first plot for $\Delta t = 1$

(d) Yes the rates of convergence as nearly the same as the theoretical convergence order/rate

PROBLEM 2

(a) Solve the ODE of numerically by the :-

(a) Adams-Bashforth explicit method

(b) Adams-Bashforth implicit method

using Δt = 2 , 1 , 0.05 , 0.025 , 0.01

(b) Plot graphs

(c) Check for the convergence order

SOLUTION
(a)
CODE

```
#include<bits/stdc++.h>

using namespace std;
double alpha = 2*pow(10,-12);
double T0 =2500;
double Ta = 250;
double function(double T);
int main()
{
    //ADAMS BASHFORTH EXPLICIT
    double delt[] = {2,1,0.05,0.025,0.01};
    cout<<"Adams Bashforth Explicit\n";
    double val,preT,currT;
    double fn,fn1,fn2,fn3;
    for(int j=0;j<5;j++)
    {
        preT = T0;
        printf("For delt = %lf\n",delt[j]);
        fn1 = preT + function(preT)*delt[j];
        printf("T(%lf) : %lf\n",delt[j]*1,fn1);
        fn2 = fn1 + function(fn1)*delt[j];
        printf("T(%lf) : %lf\n",delt[j]*2,fn2);
        fn3 = fn2 + function(fn2)*delt[j];
        printf("T(%lf) : %lf\n",delt[j]*3,fn3);
        for(int i=4;i<=13;i++)
        {
            val = 55*function(fn3)-59*function(fn2)+37*function(fn1)-9*function(preT);
            currT = fn3 + delt[j]*val/24;
            printf("T(%lf) : %lf\n",delt[j]*i,currT);
            preT = fn1;
            fn1 = fn2;
            fn2 = fn3;
            fn3 = currT;
        }
        cout<<endl;
    }
    //ADAMS BASHFORTH IMPLICIT
    cout<<"Adams Bashforth Implicit\n";
    for(int j=0;j<5;j++)
    {
        preT = T0;
        printf("For delt = %lf\n",delt[j]);
        fn1 = preT + function(preT)*delt[j];
        printf("T(%lf) : %lf\n",delt[j]*1,fn1);
        fn2 = fn1 + function(fn1)*delt[j];
        printf("T(%lf) : %lf\n",delt[j]*2,fn2);
        fn3 = fn2 + function(fn2)*delt[j];
        printf("T(%lf) : %lf\n",delt[j]*3,fn3);
        for(int i=4;i<=13;i++)
        {
            val = 9*function(fn3)+19*function(fn2)-5*function(fn1)+function(preT);
            currT = fn3 + delt[j]*val/24;
            printf("T(%lf) : %lf\n",delt[j]*i,currT);
            preT = fn1;
            fn1 = fn2;
            fn2 = fn3;
            fn3 = currT;
        }
        cout<<endl;
    }

}
double function(double T)
{
    return -alpha*(pow(T,4)-pow(Ta,4));
}
```

OUTPUT

Adams Bashforth Explicit

```
For delt = 2.000000
T(2.000000) : 2343.765625
T(4.000000) : 2223.078626
T(6.000000) : 2125.397688
T(8.000000) : 2051.038304
T(10.000000) : 1984.141440
T(12.000000) : 1926.904237
T(14.000000) : 1874.428607
T(16.000000) : 1827.819753
T(18.000000) : 1785.140921
T(20.000000) : 1746.378708
T(22.000000) : 1710.676125
T(24.000000) : 1677.773024
T(26.000000) : 1647.239041


For delt = 1.000000
T(1.000000) : 2421.882812
T(2.000000) : 2353.082061
T(3.000000) : 2291.773242
T(4.000000) : 2239.299572
T(5.000000) : 2190.964123
T(6.000000) : 2146.923786
T(7.000000) : 2106.011263
T(8.000000) : 2068.122913
T(9.000000) : 2032.778686
T(10.000000) : 1999.750537
T(11.000000) : 1968.761351
T(12.000000) : 1939.611211
T(13.000000) : 1912.112837
```

```
For delt = 0.050000
T(0.050000) : 2496.094141
T(0.100000) : 2492.212636
T(0.150000) : 2488.355220
T(0.200000) : 2484.533441
T(0.250000) : 2480.734942
T(0.300000) : 2476.959654
T(0.350000) : 2473.207217
T(0.400000) : 2469.477419
T(0.450000) : 2465.770022
T(0.500000) : 2462.084790
T(0.550000) : 2458.421494
T(0.600000) : 2454.779905
T(0.650000) : 2451.159800
```

```
For delt = 0.025000
T(0.025000) : 2498.047070
T(0.050000) : 2496.100236
T(0.075000) : 2494.159465
T(0.100000) : 2492.227724
T(0.125000) : 2490.301943
T(0.150000) : 2488.382111
T(0.175000) : 2486.468183
T(0.200000) : 2484.560131
T(0.225000) : 2482.657923
T(0.250000) : 2480.761527
T(0.275000) : 2478.870913
T(0.300000) : 2476.986049
T(0.325000) : 2475.106906
```

```
For delt = 0.010000
T(0.010000) : 2499.218828
T(0.020000) : 2498.438632
T(0.030000) : 2497.659410
T(0.040000) : 2496.881645
T(0.050000) : 2496.104847
T(0.060000) : 2495.329015
T(0.070000) : 2494.554148
T(0.080000) : 2493.780241
T(0.090000) : 2493.007295
T(0.100000) : 2492.235305
T(0.110000) : 2491.464272
T(0.120000) : 2490.694191
T(0.130000) : 2489.925062
```

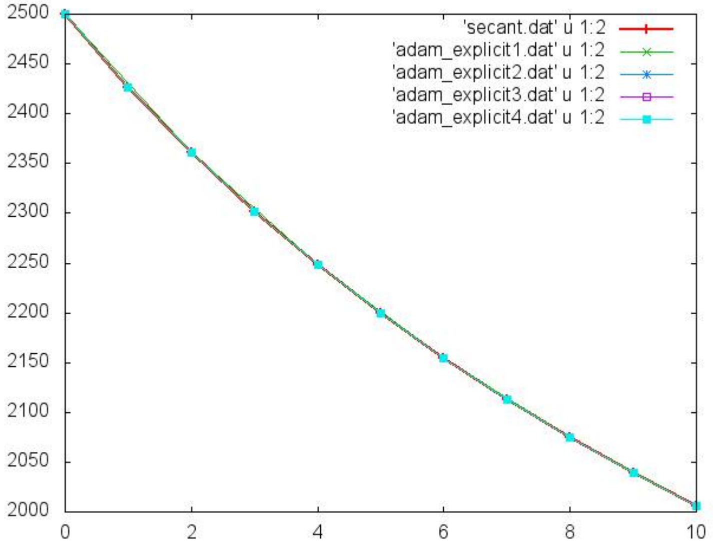
Adams Bashforth Implicit

```
For delt = 2.000000
T(2.000000) : 2343.765625
T(4.000000) : 2223.078626
T(6.000000) : 2125.397688
T(8.000000) : 2036.096958
T(10.000000) : 1961.037192
T(12.000000) : 1897.378796
T(14.000000) : 1842.042871
T(16.000000) : 1793.207362
T(18.000000) : 1749.589643
T(20.000000) : 1710.240993
T(22.000000) : 1674.449641
T(24.000000) : 1641.667157
T(26.000000) : 1611.461531


For delt = 1.000000
T(1.000000) : 2421.882812
T(2.000000) : 2353.082061
T(3.000000) : 2291.773242
T(4.000000) : 2233.629233
T(5.000000) : 2181.198368
T(6.000000) : 2133.758243
T(7.000000) : 2090.452929
T(8.000000) : 2050.673998
T(9.000000) : 2013.932842
T(10.000000) : 1979.832294
T(11.000000) : 1948.047216
T(12.000000) : 1918.307993
T(13.000000) : 1890.388594


For delt = 0.050000
T(0.050000) : 2496.094141
T(0.100000) : 2492.212636
T(0.150000) : 2488.355220
T(0.200000) : 2484.509741
T(0.250000) : 2480.687989
T(0.300000) : 2476.889767
T(0.350000) : 2473.114807
T(0.400000) : 2469.362863
T(0.450000) : 2465.633690
T(0.500000) : 2461.927047
T(0.550000) : 2458.242695
T(0.600000) : 2454.580401
T(0.650000) : 2450.939933


For delt = 0.025000
```

```
T(0.025000) : 2498.047070
T(0.050000) : 2496.100236
T(0.075000) : 2494.159465
T(0.100000) : 2492.221711
T(0.125000) : 2490.289974
T(0.150000) : 2488.364228
T(0.175000) : 2486.444439
T(0.200000) : 2484.530575
T(0.225000) : 2482.622605
T(0.250000) : 2480.720496
T(0.275000) : 2478.824216
T(0.300000) : 2476.933736
T(0.325000) : 2475.049023
```

```
For delt = 0.010000
T(0.010000) : 2499.218828
T(0.020000) : 2498.438632
T(0.030000) : 2497.659410
T(0.040000) : 2496.880674
T(0.050000) : 2496.102909
T(0.060000) : 2495.326114
T(0.070000) : 2494.550285
T(0.080000) : 2493.775421
T(0.090000) : 2493.001520
T(0.100000) : 2492.228580
T(0.110000) : 2491.456599
T(0.120000) : 2490.685574
T(0.130000) : 2489.915503
```

(b)


'secant.dat' u 1:2
'adam_explicit1.dat' u 1:2
'adam_explicit2.dat' u 1:2
'adam_explicit3.dat' u 1:2
'adam_explicit4.dat' u 1:2