# Swapnil Daxini (V00861672) CSC 349A Assignment 1

## Contents



## Question 1 Part A

```
type Euler.m
```

```
function Euler(m,c,g,t0,v0,tn,n)
% print headings and initial conditions
fprintf('values of t approximations v(t)\n')
fprintf('%8.3f',t0),fprintf('%19.4f\n',v0)
% compute step size h
h=(tn-t0)/n;
% set t,v to the initial values
t=t0;
v= v0;
% compute v(t) over n time steps using Euler's method
for i=1:n
v=v+(g-c/m*v)*h;
t=t+h;
fprintf('%8.3f',t),fprintf('%19.4f\n',v)
end
```

## Question 1 Part B

Use Euler Equation with constants m = 86.2, c = 12.5, v0 = 0 t0=0, tn = 12, n = 15, g = 9.81

```
Euler(86.2, 12.5, 9.81, 0, 0, 12, 15)
```

```
values of t approximations v(t)
   0.000             0.0000
   0.800             7.8480
   1.600            14.7856
   2.400            20.9183
   3.200            26.3396
   4.000            31.1319
   4.800            35.3684
   5.600            39.1133
   6.400            42.4238
   7.200            45.3502
   8.000            47.9372
   8.800            50.2240
   9.600            52.2456
  10.400            54.0326
  11.200            55.6123
  12.000            57.0088
```

## Question 1 Part C

```
Euler(86.2, 12.5, 3.71, 0, 0, 12, 15)
```

```
values of t approximations v(t)
   0.000             0.0000
   0.800             2.9680
   1.600             5.5917
   2.400             7.9110
   3.200             9.9612
   4.000            11.7737
   4.800            13.3758
   5.600            14.7921
   6.400            16.0441
   7.200            17.1508
   8.000            18.1292
   8.800            18.9940
   9.600            19.7585
  10.400            20.4343
  11.200            21.0318
  12.000            21.5599
```

## Question 1 Part D

```
true_v = velocity(86.2, 9.81, 12.5, 12)

% Relative error is given by abs(1-approx_v/true_v)

approx_v = 57.0088
relative_error = abs(1-(approx_v/true_v))
```

```
true_v =

   55.7775


approx_v =

   57.0088


relative_error =

    0.0221
```

## Question 2 Part A

```
type Euler2.m
```

```
function Euler2(m,k,g,t0,v0,tn,n)
% print headings and initial conditions
fprintf('values of t approximations v(t)\n')
fprintf('%8.3f',t0),fprintf('%19.4f\n',v0)
% compute step size h
h=(tn-t0)/n;
% set t,v to the initial values
t=t0;
v= v0;
% compute v(t) over n time steps using Euler's method
for i=1:n
v=v+(g-k/m*v^2)*h;
t=t+h;
fprintf('%8.3f',t),fprintf('%19.4f\n',v)
end
```

## Question 2 Part B

```
Euler2(73.5, 0.234, 9.81, 0, 0, 18, 72)
```

```
values of t approximations v(t)
   0.000             0.0000
   0.250             2.4525
   0.500             4.9002
   0.750             7.3336
   1.000             9.7433
   1.250            12.1202
   1.500            14.4558
   1.750            16.7420
   2.000            18.9714
   2.250            21.1374
   2.500            23.2343
   2.750            25.2572
   3.000            27.2019
   3.250            29.0655
   3.500            30.8456
   3.750            32.5408
   4.000            34.1505
   4.250            35.6748
   4.500            37.1143
   4.750            38.4705
   5.000            39.7450
   5.250            40.9402
   5.500            42.0587
   5.750            43.1033
   6.000            44.0770
   6.250            44.9832
   6.500            45.8252
   6.750            46.6063
   7.000            47.3300
   7.250            47.9995
   7.500            48.6182
   7.750            49.1894
```

```
     8.000          49.7161
     8.250          50.2013
     8.500          50.6480
     8.750          51.0588
     9.000          51.4363
     9.250          51.7831
     9.500          52.1013
     9.750          52.3933
    10.000          52.6609
    10.250          52.9062
    10.500          53.1309
    10.750          53.3366
    11.000          53.5249
    11.250          53.6971
    11.500          53.8547
    11.750          53.9988
    12.000          54.1305
    12.250          54.2509
    12.500          54.3608
    12.750          54.4613
    13.000          54.5531
    13.250          54.6369
    13.500          54.7134
    13.750          54.7833
    14.000          54.8471
    14.250          54.9053
    14.500          54.9584
    14.750          55.0069
    15.000          55.0512
    15.250          55.0915
    15.500          55.1284
    15.750          55.1620
    16.000          55.1926
    16.250          55.2206
    16.500          55.2461
    16.750          55.2693
    17.000          55.2905
    17.250          55.3099
    17.500          55.3275
    17.750          55.3436
    18.000          55.3583
```

### Question 2 Part C

```
approx_v = 55.3583
true_v = velocity2nd(73.5, 9.81, 0.234, 18)

relative_error = abs(1-(approx_v/true_v))
```

```
approx_v =

   55.3583


true_v =

   55.3186


relative_error =

   7.1738e-04
```

### Question 3

```
% For this question, we can use the matlab function taylor which accepts a
% function and a value and expands it to a given order.

syms x;

f1 = exp(-x)
f2 = exp(x)

% Taylor series expansion of exp(-x) for n=1,2,3,4,5
t1 = taylor(f1, x, 'Order', 1);
t2 = taylor(f1, x, 'Order', 2);
t3 = taylor(f1, x, 'Order', 3);
t4 = taylor(f1, x, 'Order', 4);
t5 = taylor(f1, x, 'Order', 5);
t6 = taylor(f1, x, 'Order', 6);

% Taylor series expansion of exp(x) for n=1,2,3,4,5
g1 = taylor(f2, x, 'Order', 1);
g2 = taylor(f2, x, 'Order', 2);
g3 = taylor(f2, x, 'Order', 3);
g4 = taylor(f2, x, 'Order', 4);
```

```
g5 = taylor(f2, x, 'Order', 5);
g6 = taylor(f2, x, 'Order', 6);

% The taylor series are evaluated at x = 2 then convert to double
approx_1_1 = double(subs(t1, x, 2));
approx_1_2 = double(subs(t2, x, 2));
approx_1_3 = double(subs(t3, x, 2));
approx_1_4 = double(subs(t4, x, 2));
approx_1_5 = double(subs(t5, x, 2));
approx_1_6 = double(subs(t6, x, 2));

% The taylor series are evaluated at x = 2. The reciprocal of this result
% was calculated to approximate exp(-2)
approx_2_1 = double(1/subs(g1, x, 2));
approx_2_2 = double(1/subs(g2, x, 2));
approx_2_3 = double(1/subs(g3, x, 2));
approx_2_4 = double(1/subs(g4, x, 2));
approx_2_5 = double(1/subs(g5, x, 2));
approx_2_6 = double(1/subs(g6, x, 2));


true_value = exp(-2)

err_1_1 = relativeerror(true_value, approx_1_1);
err_1_2 = relativeerror(true_value, approx_1_2);
err_1_3 = relativeerror(true_value, approx_1_3);
err_1_4 = relativeerror(true_value, approx_1_4);
err_1_5 = relativeerror(true_value, approx_1_5);
err_1_6 = relativeerror(true_value, approx_1_6);

err_2_1 = relativeerror(true_value, approx_2_1);
err_2_2 = relativeerror(true_value, approx_2_2);
err_2_3 = relativeerror(true_value, approx_2_3);
err_2_4 = relativeerror(true_value, approx_2_4);
err_2_5 = relativeerror(true_value, approx_2_5);
err_2_6 = relativeerror(true_value, approx_2_6);


T = table([true_value;true_value;true_value;true_value;true_value;true_value], [approx_1_1;approx_1_2;approx_1_3;approx_1_4;approx_1_5;approx_1_6], [err_1_1;err_1_2
```

```
f1 =

exp(-x)


f2 =

exp(x)


true_value =

    0.1353
```

Note that Approximation1 represents the taylor expansion of exp(-2) while Approximation2 represents the taylor expansion of 1/exp(x).

```
T.Properties.VariableNames = {'TrueValue';'Approximation1';'RelativeError1';'Approximation2';'RelativeError2'};

T
```

```
T =

  6×5 table

    TrueValue    Approximation1    RelativeError1    Approximation2    RelativeError2
    _________    ______________    ______________    ______________    ______________

    0.13534             1          6.3891                   1             6.3891
    0.13534            -1          8.3891             0.33333              1.463
    0.13534             1          6.3891                 0.2            0.47781
    0.13534      -0.33333           3.463             0.15789            0.16669
    0.13534       0.33333           1.463             0.14286           0.055579
    0.13534      0.066667          0.5074             0.13761           0.016843
```

From the table we can conclude that using the approximation of 1/exp(2) is much better than the taylor approximation of exp(-2). The approximation with exp(-2) approaches the true more slower than 1/exp(2).

*Published with MATLAB® R2017a*