Tom Green reg no.:1202807

Lab 4

1) **Starting With the outline Matlab program Euler.m use the method to compute a solution (up to t=1.5) of**

$$\frac{dy}{dt} = -t^2 y, \text{ with } y(0) = 1.0$$

Do this for N=10, N=100, N=1000 and N=10000

The lines of code that I completed in this file were:
```
h = (t_end-t_begin)/N;
y(k+1) = y(k)+yd*h;
truey = exp(-(t_end^3)/3);
```

Then the results that I got were:

N=10

```
Final value of t:
    1.5000


Final value of y:
    0.3449


Number of steps:
    10

True value of y:
    0.3247


Difference between computed and true values:
    0.0203
```

N=100

```
Final value of t:
    1.5000


Final value of y:
    0.3265


Number of steps:
   100

True value of y:
    0.3247


Difference between computed and true values:
    0.0018
```

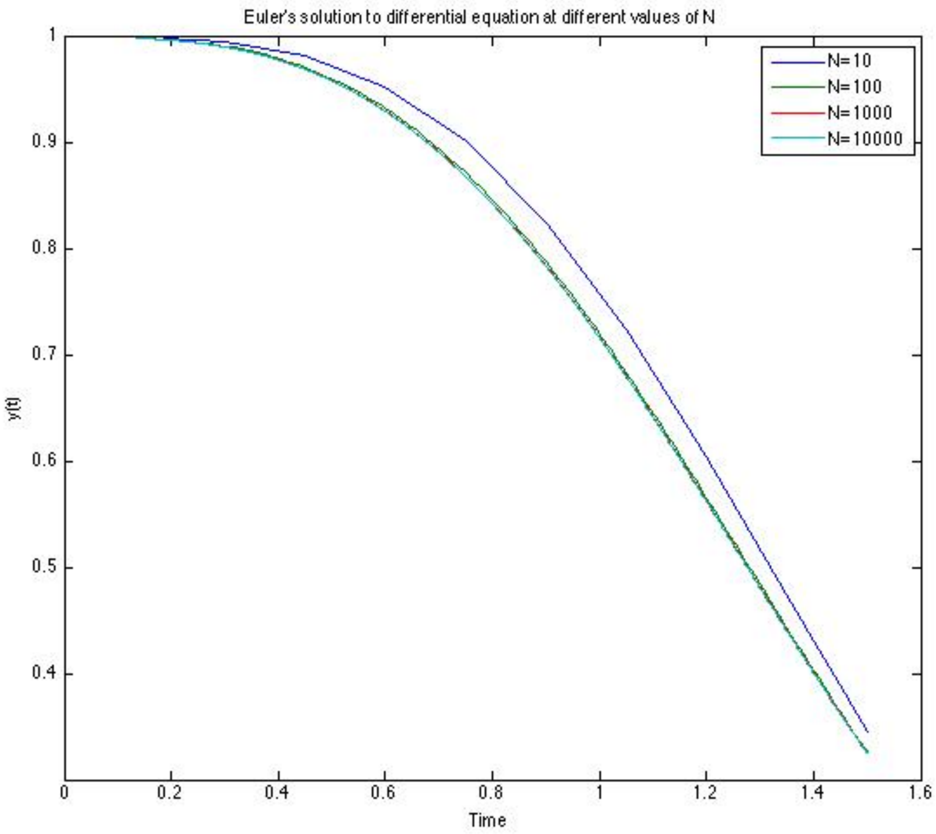
Tom Green reg no.:1202807

N=1000

```
Final value of t:
    1.5000


Final value of y:
    0.3248


Number of steps:
        1000

True value of y:
    0.3247


Difference between computed and true values:
   1.7825e-04
```

N=10,000

```
Final value of t:
    1.5000


Final value of y:
    0.3247


Number of steps:
       10000

True value of y:
    0.3247


Difference between computed and true values:
   1.7807e-05
```



I was then able to plot the reulsts for the four different vaues of N on the plot above.

2) **Let d be the three digit integer formed by taking the last three digits of your registration number in order and define $\propto = 0.001 * d$. Starting with the outline Matlab program Heun.m, use the method to compute a solution (up to t=3.0) of**

$$\frac{dy}{dt} = t^2 - y \text{ with } y(-3.0) = \alpha$$

(The analytical solution of this IVP is $y(t) \equiv t^2 - 2t + 2 + (\alpha - 17)e^{-(t+3)}$. Do this for N=10, N=100, N=1000 and N=10000

To verify that $y(t) \equiv t^2 - 2t + 2 + (\alpha - 17)e^{-(t+3)}$ is in fact a solution of the IVP we first differentiate the function:

$$y(t) \equiv t^2 - 2t + 2 + (\alpha - 17)e^{-(t+3)}$$
$$y'(t) \equiv 2t + (17 - \alpha)e^{-(t+3)}$$

Then we know that this differentiated function should equal $t^2 - y$

$$\frac{dy}{dt} = t^2 - y$$
$$2t + (17-\alpha)e^{-(t+3)} = t^2 - y$$
$$2t + (17-\alpha)e^{-(t+3)} = t^2 - \left(t^2 - 2t + 2 + (\alpha - 17)e^{-(t+3)}\right)$$
$$2t + (17-\alpha)e^{-(t+3)} = 2t + (17-\alpha)e^{-(t+3)}$$

So both sides are equal and therefore it is a solution.

The lines of code that I changed in the Heun.m file were:

```
d=807;
alpha=0.01*d;
% Set up the initial data and the end value for 't':
t_begin = -3;
t_end = 3;
y0 = alpha;
h = (t_end-t_begin)/N;
k2 = f(tempy,tempt);
  y(k+1) = y(k)+h*(k1+k2)/2;
tt=t(N+1);
truey = tt.^2-2*tt+2+(alpha-17)*exp(-tt-3);
```

The results that I got were:

For N=10

```
Final value of t:
    3.0000


Final value of y:
    5.2176


Number of steps:
    10

True value of y:
    4.9779


Difference between computed and true values:
    0.2397
```

For N=100

```
Final value of t:
    3.0000


Final value of y:
    4.9796


Number of steps:
   100

True value of y:
```

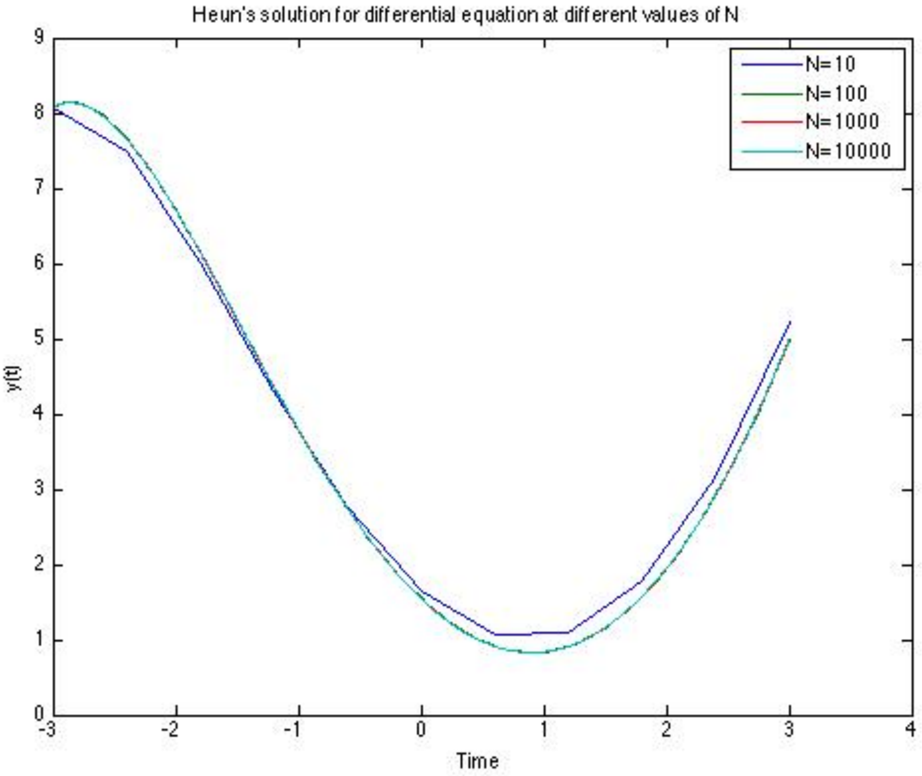
```
    4.9779


Difference between computed and true values:
    0.0018
```

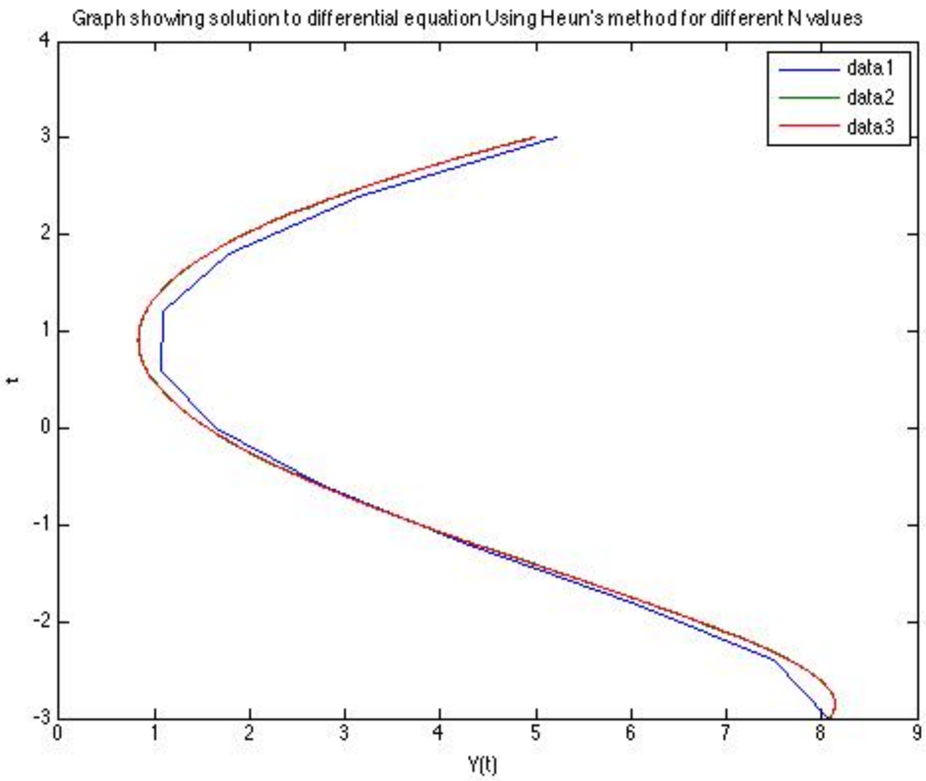
For N=1000

```
Final value of t:
    3.0000


Final value of y:
    4.9779


Number of steps:
        1000

True value of y:
    4.9779


Difference between computed and true values:
   1.7209e-05
```

For N=10000

```
Final value of t:
    3.0000


Final value of y:
    4.9779


Number of steps:
       10000

True value of y:
    4.9779


Difference between computed and true values:
   1.7164e-07
```





3) **Consider the model**

$\frac{dy}{dt} = \lambda y + \gamma t$, with $y(0) = y_0$ and $\lambda \neq 0$

(a) **Verify that the solution of the IVP is**

$$y(t) \equiv e^{\lambda t} y_0 + \gamma \lambda^{-2}\left(e^{\lambda t} - 1 - \lambda t\right)$$

$$y'(t) \equiv \lambda e^{\lambda t} y_0 + \gamma \lambda^{-2}\left(\lambda e^{\lambda t} - \lambda\right)$$
$$y'(t) \equiv \lambda e^{\lambda t} y_0 + \gamma \lambda^{-1}\left(e^{\lambda t} - 1\right)$$

$$\frac{dy}{dt} = \lambda y + \gamma t$$

$$\lambda e^{\lambda t} y_0 + \gamma \lambda^{-2}\left(\lambda e^{\lambda t} - \lambda\right) = \lambda y + \gamma t$$

$$\lambda e^{\lambda t} y_0 + \gamma \lambda^{-1}\left(e^{\lambda t} - 1\right) = \lambda[e^{\lambda t} y_0 + \gamma \lambda^{-2}\left(e^{\lambda t} - 1 - \lambda t\right)] + \gamma t$$

$$\lambda e^{\lambda t} y_0 + \gamma \lambda^{-1}\left(e^{\lambda t} - 1\right) = \lambda e^{\lambda t} y_0 + \gamma \lambda^{-1}\left(e^{\lambda t} - 1 - \lambda t\right) + \gamma t$$

$$\lambda e^{\lambda t} y_0 + \gamma \lambda^{-1}\left(e^{\lambda t} - 1\right) = \lambda e^{\lambda t} y_0 + \gamma \lambda^{-1}\left(e^{\lambda t} - 1\right) - \gamma t \ + \gamma t$$

$$\lambda e^{\lambda t} y_0 + \gamma \lambda^{-1}\left(e^{\lambda t} - 1\right) = \lambda e^{\lambda t} y_0 + \gamma \lambda^{-1}\left(e^{\lambda t} - 1\right)$$

Therefore they are the same, so this is a solution.

b) **Show that, if Heun's method ($\alpha = 1$) and the modified Euler Method ($\alpha = \frac{1}{2}$) are each used with a step length of h and one step (from t=0) to compute an estimate of y(h), they produce the same estimate**

Let's start with the improved Euler Method:
$h_1, t_k \& \ y_k, k_1 = f(y_k, t_k)$

$$k_2$$

c) **Compare the estimate produced by the two methods with the value of the analytical solution at t=h and show that the leading term in the error is O(h^3)**

4) **Starting with the outline Matlab program Heun3rule.m which implements Heun's Third-order Rule, repeat the numerical experiments of Question 2 for N=10, N=100 and N=1000. Compare the results you obtain from the two methods**

The lines of code that I changed in this program were:

```
f = inline('t^2-y','y','t');
d=807;
alpha=0.01*d;
% Set up the initial data and the end value for 't':
t_begin = -3.;
t_end = 3.;
y0 = alpha;
h = (t_end-t_begin)/N;
k2 = f(tempy,tempt);
  tempy = y(k)+(2*h*k2)/3.0;
  tempt = t(k) + 2*h/3.0;
  k3 = f(tempy,tempt);
  y(k+1) = y(k)+h*(k1/4.0+3*k3/4.0);
truey = t(N+1)^2-2*t(N+1)+2+(alpha-17)*exp(-t(N+1)-3);
```

These are the results that I get:

For N=10

```
Final value of t:
    3.0000


Final value of y:
    4.9640


Number of steps:
    10

True value of y:
    4.9779


Difference between computed and true values:
   -0.0139
```

For N=100

```
Final value of t:
    3.0000


Final value of y:
    4.9779


Number of steps:
   100
```

```
True value of y:
    4.9779


Difference between computed and true values:
  -1.1079e-05
```

For N=1000

```
Final value of t:
    3.0000


Final value of y:
    4.9779


Number of steps:
        1000

True value of y:
    4.9779


Difference between computed and true values:
  -1.0805e-08
```

If we compare these results to the results from question 2 we can see that this method gets a more accurate value of y for a smaller value of N.

5)

The Code that I used for this question was

```
% 
#######################################
% Runge Kutta Fourth-order Rule for ODEs #
% #######################################


f = inline('(-
y/(r0*c)+(beta*cos(t))/(r0*c))/(1.+2.*0.8*y/r0)','y','t','c','r0','be
ta');  % This is the function on the RHS of the ODE
%fana = inline('t^2-2*t+2+(alpha-17.)*exp(-t-3.)','t','alpha'); %
analytical solution to differential equation
vprime = inline('beta*cos(t)','t','beta');  % applied voltage
r = inline('r0+0.8*i','i','r0');  % resistance as function of current

% Set up the initial data and the end value for 't':
t_begin = 0.;
t_end = 50.;
d=807;
beta=0.95+0.0001*d;

c = 1+0.001*d;
r0 = 18.;
```

```
y0 = 0.;


% Choose the number of points:
N = 1000;


% Set up the vectors to hold the 't' and 'y' values as they are
calculated:
t = zeros(N+1,1);
y = t;


% Put the initial data into the first elements of the vectors 't' and
'y':
t(1) = t_begin;
y(1) = y0;


% Calculate the step-length:
h = (t_end - t_begin)/N;



% Now carry out Heun's Third-order Rule:

for k = 1:N  % Be careful not to confuse 'k' (the loop-counter) with
k1, k2 or k3 (needed in Heun's Third-order Rule)
  k1 = f(y(k),t(k),c,r0,beta);
  tempy = y(k) + h*k1/2.0;
  tempt = t(k) + h/2.0;
  k2 = f( tempy, tempt, c, r0, beta );
  tempy = y(k) + h*k2/2.0;
  tempt = t(k) + h/2.0;
  k3 = f( tempy, tempt, c, r0, beta );
  tempy = y(k) + h*k3;
  tempt = t(k) + h;
  k4 = f( tempy, tempt, c, r0, beta );
  y(k+1) = y(k) + h*(k1+2.*k2+2.*k3+k4)/6.;
  t(k+1) = t(k) + h;
end


% Print the final values:
disp(' ')
disp(' ')
disp('Final value of t:')
disp(t(N+1))
disp(' ')
disp('Final value of y:')
disp(y(N+1))
disp(' ')
disp('Number of steps:')
disp(N)
plot(t,y);
```

The results that I got for this question were

Tom Green reg no.:1202807

Final value of t:
50.0000

Final value of y:
-0.0076

Number of steps:
1000

Finally the graph that I got for this question was