Euler.m

```
function Euler(m,c,g,t0,v0,tn,n)
% make empty matrices
    time =[];
    velocity =[];
% 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
    t=t+h;
    v=v+(g-c/m*v)*h;
    time(i+1) = t;
    velocity(i+1) = v;
end
    time(i+1) = t
    velocity(i+1) = v
```

Command Window

New to MATLAB? See resources for Getting Started.

```
>> Euler ( 68.1 , 12.5 , 9.81, 0 , 0 , 12 , 6 )

time =

     0     2     4     6     8    10    12


velocity =

         0   19.6200   32.0374   39.8962   44.8700   48.0179   50.0102
```

Editor - \\home.uvic.ca\devroopb\My Documents\MATLAB\Euler.m

Euler.m

```matlab
function Euler(m,c,g,t0,v0,tn,n)
% make empty matrices
    time =[];
    velocity =[];
% 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
    t=t+h;
    v=v+(g-c/m*v)*h;
    time(i+1) = t;
    velocity(i+1) = v;
end
    time(i+1) = t
    velocity(i+1) = v
```

Command Window

New to MATLAB? See resources for Getting Started.

```
>> Euler (70, 12, 9.81, 0, 0, 12, 15)

time =

         0    0.8000    1.6000    2.4000    3.2000    4.0000    4.8000    5.6000    6.4000    7.2000    8.0000    8.8000    9.6000   10.4000   11.2000   12.0000


velocity =

         0    7.8480   14.6197   20.4627   25.5044   29.8547   33.6083   36.8472   39.6418   42.0532   44.1339   45.9293   47.4784   48.8151   49.9684   50.9636

>> 
```

Editor - \\home.uvic.ca\devroopb\My Documents\MATLAB\Euler.m

Euler.m

```
function Euler(m,c,g,t0,v0,tn,n)
% make empty matrices
    time =[];
    velocity =[];
% 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
    t=t+h;
    v=((g*m)/c)*(1-exp((0-(c*t)/m)));
    time(i+1) = t;
    velocity(i+1) = v;
end
    time(i+1) = t
    velocity(i+1) = v
```

Command Window

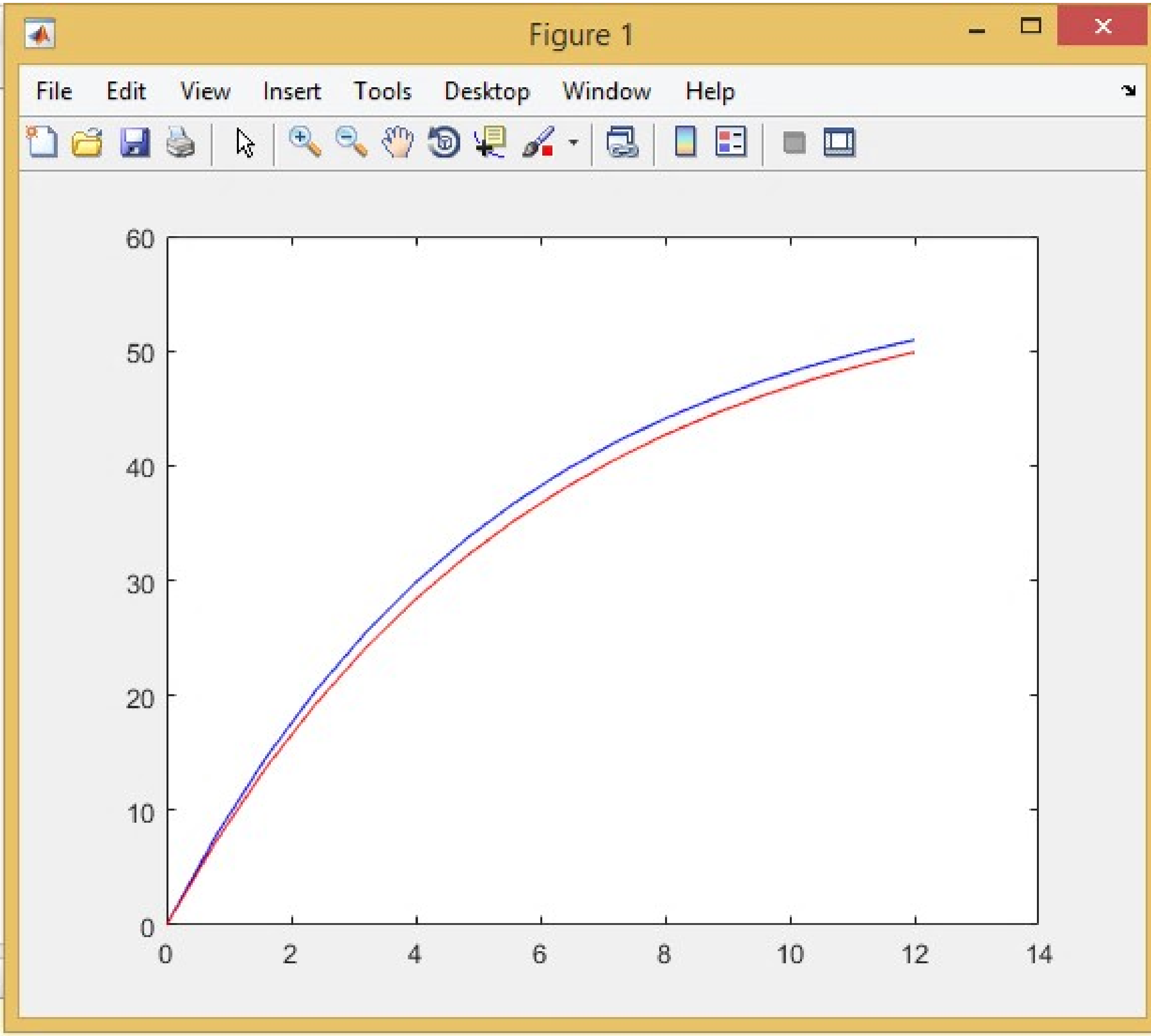
New to MATLAB? See resources for Getting Started.

```
>> Euler (70, 12, 9.81, 0, 0, 12, 15)

time =

         0    0.8000    1.6000    2.4000    3.2000    4.0000    4.8000    5.6000    6.4000    7.2000    8.0000    8.8000    9.6000   10.4000   11.2000   12.0000


velocity =

         0    7.3336   13.7274   19.3018   24.1618   28.3990   32.0932   35.3140   38.1219   40.5701   42.7045   44.5654   46.1877   47.6022   48.8354   49.9106

>> 
```

Editor - \\home.uvic.ca\devroopb\My Documents\MATLAB\Euler.m

Euler.m

```matlab
function Euler(m,c,g,t0,v0,tn,n)
% make empty matrices
    time = [];
    velocity1 =[];
    velocity2 =[];
% compute step size h
    h=(tn-t0)/n;
% set t,v to the initial values
    t=t0;
    v=v0;
    v2=v0;
% compute v(t) over n time steps using Euler's method
for i=1:n
    t=t+h;
    v=v+(g-c/m*v)*h;
    v2=((g*m)/c)*(1-exp((-c*t)/m));
    time(i+1) = t;
    velocity1(i+1) = v;
    velocity2(i+1) = v2;
end
plot(time, velocity1,'b', time, velocity2, 'r')
```



Command Window

New to MATLAB? See resources for Getting Started.

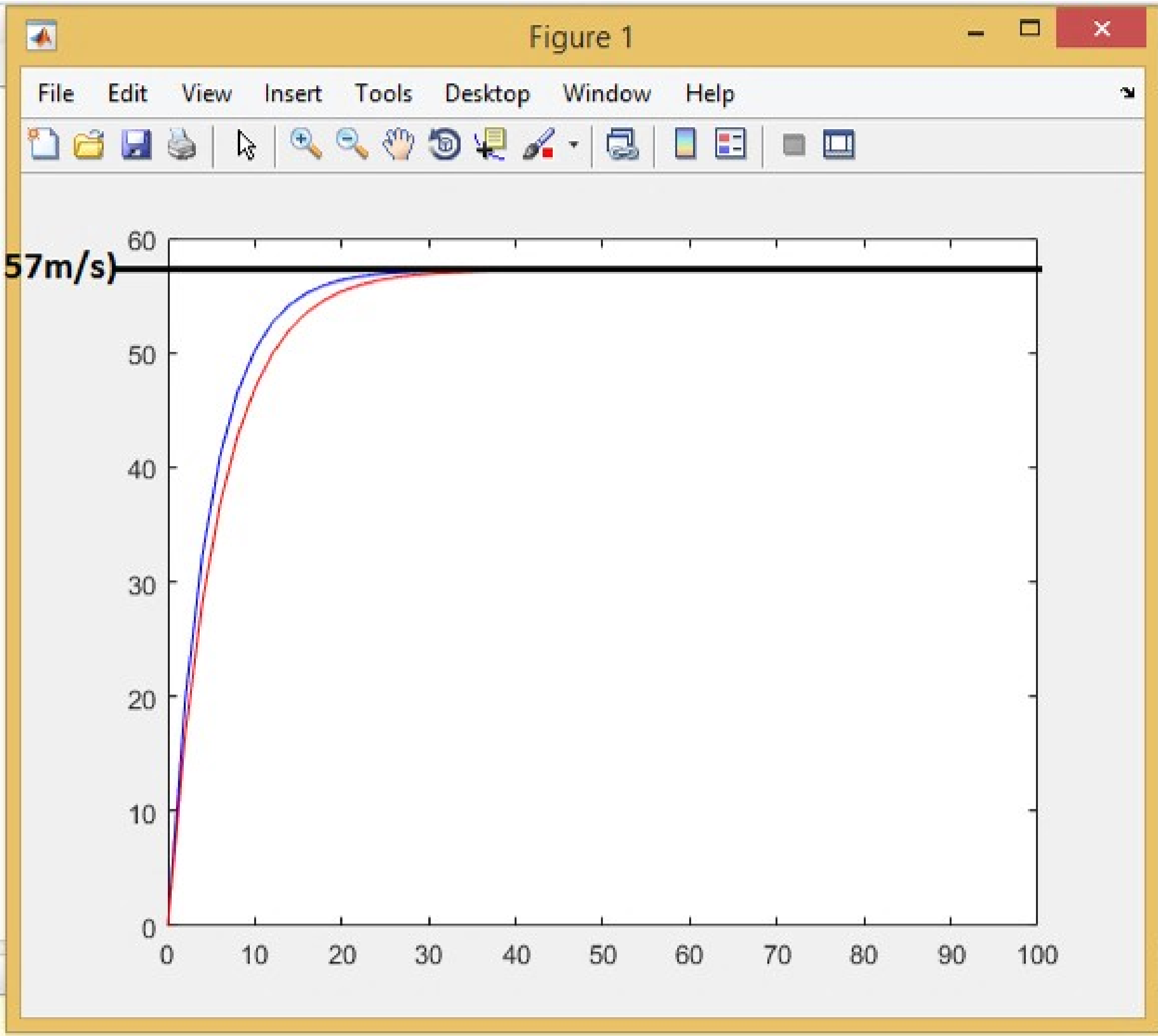
```
>> Euler (70, 12, 9.81, 0, 0, 12, 15)
>>
```

Editor - \\home.uvic.ca\devroopb\My Documents\MATLAB\Euler.m

Euler.m

```
function Euler(m,c,g,t0,v0,tn,n)
% make empty matrices
    time = [];
    velocity1 =[];
    velocity2 =[];
% compute step size h
    h=(tn-t0)/n;
% set t,v to the initial values
    t=t0;
    v=v0;
    v2=v0;
% compute v(t) over n time steps using Euler's method
for i=1:n
    t=t+h;
    v=v+(g-c/m*v)*h;
    v2=((g*m)/c)*(1-exp((-c*t)/m));
    time(i+1) = t;
    velocity1(i+1) = v;
    velocity2(i+1) = v2;
end
plot(time, velocity1,'b', time, velocity2, 'r')
```

Terminal velocity (around 57m/s)



Command Window

New to MATLAB? See resources for Getting Started.

```
>> Euler (70, 12, 9.81, 0, 0, 12, 15)
>> Euler (70, 12, 9.81, 0, 0, 100, 50)
>>
```

<----Used 100 seconds here so that terminal velocity is reached

Devroop Banerjee
V00837868

2. a) $(3 + 2/9)_{10}$

$\Rightarrow 3/3 = 1$ $\qquad$ $2/9 \times 3 = 2/3$

$2/3 \times 3 = 2$

$0 \times 3 = 0$

$\Rightarrow (1\times3^0)_3 + (0\times3^{-1})_3 + (2\times3^{-2})_3$

$\Rightarrow (1\cdot02\times3^0)_3$

$\Rightarrow 0\cdot102\times3^1$

$\Rightarrow$ 00011020 (Ans)

b) -55_{10}

$-55/3$	18	1
$-18/3$	6	0
$-6/3$	2	0
$-2/3$	0	2

$\Rightarrow -2001_3$

$\Rightarrow -0\cdot2001\times3^4$

$\Rightarrow$ 10112001 (Ans)

c) 00222222

$\Rightarrow 0.2222 \times 3^{8}$

$\Rightarrow (22220000)_3$ (Ans)

d) 01220001

$\Rightarrow 0.0001 \times 3^{-8}$

$\Rightarrow (0.000000000001)_3$ (Ans)

e) 3_{10}

$\Rightarrow 3/3$ 1 0

$\Rightarrow 1/3$ 0 1

$\Rightarrow 10_3$

$\Rightarrow 3^1$

9_{10}

$\Rightarrow 9/3$ 3 0

$\Rightarrow 3/3$ 1 0

$\Rightarrow 1/3$ 0 1

$\Rightarrow 100_3$

$\Rightarrow 3^2$

So the spacing for numbers between 3_{10} & 9_{10} is $[3^1, 3^2)$

Precision is 4. $3^{e+1-p} = 3^{2-4} = 3^{-2} =$ 0.1111

~~S6 [3^2 × 3^1) S~~

18/1/17 CSC 349A Assignment 1

Devroop Banerjee
V00837868

3) $e^{-2} \approx 1 - 2 + \frac{2^2}{2!} - \frac{2^3}{3!} + \frac{2^4}{4!} - \frac{2^5}{5!} + \frac{2^6}{6!}$

$\Rightarrow 1 - 2 + 2 - 1.3333 + 0.6666 - 0.2666 + 0.0888$ (1)

$\Rightarrow 0.15555.$

$$e^{-2} \approx \frac{1}{1 + 2 + \frac{2^2}{2!} + \frac{2^3}{3!} + \frac{2^4}{4!} + \frac{2^5}{5!} + \frac{2^6}{6!}}$$

$$\Rightarrow \frac{1}{1 + 2 + \frac{4}{2} + \frac{8}{6} + \frac{16}{24} + \frac{32}{120} + \frac{64}{720}}$$ (2)

$$\Rightarrow \frac{1}{7\frac{16}{45}} \Rightarrow 0.13595$$

Inverse of the Mclaurin series expansion of e^2 is obviously closer to the true value.

(1)

i	p_i	$\lvert\varepsilon_t\rvert = \frac{\lvert e^{-2} - p_i\rvert}{\lvert e^{-2}\rvert}$	$\lvert\varepsilon_a\rvert = \frac{\lvert p_i - p_{i-1}\rvert}{\lvert p_i\rvert}$
1	1	6.3891	X
2	-1	8.3891	2
3	1	6.3891	2
4	-0.3333	3.4630	4
5	0.3333	1.4630	2
6	0.06666	0.5074	4
7	0.15555	0.1494	0.57143

(2)

i	p_i	$\lvert\varepsilon_t\rvert = \frac{\lvert e^{-2} - p_i\rvert}{\lvert e^{-2}\rvert}$	$\lvert\varepsilon_a\rvert = \frac{\lvert p_i - p_{i-1}\rvert}{\lvert p_i\rvert}$
1	1	6.3891	X
2	0.3333	1.46302	2
3	0.2	0.47781	0.66666
4	0.15789	0.16666	0.26670
5	0.14286	0.05560	0.10521
6	0.13761	0.01681	0.03815
7	0.13595	0.004544	0.01221

Obviously this second method is a better approach. The error values are much lesser than the ones from the first method.