수치해석 00반
201402414 장수훈

2.7절

```
>> m = [83.6 60.2 72.1 91.1 92.9 65.3 80.9];
>> vt = [53.4 48.5 50.9 55.7 54 47.7 51.1];
>> g = 9.81;
>> cd = g*m./vt.^2
cd =

   0.28760    0.25106    0.27300    0.28806    0.31253    0.28154    0.30393

>> cdavg=mean(cd),cdmin=min(cd),cdmax=man(cd)
cdavg =  0.28539
cdmin =  0.25106
error: 'man' undefined near line 1 column 36
>> cdavg=mean(cd),cdmin=min(cd),cdmax=max(cd)
cdavg =  0.28539
cdmin =  0.25106
cdmax =  0.31253
```

ex 3.1

scriptdemo.m

```
g=9.81; m=68.1; t=12; cd=0.25;
v=sqrt(g*m/cd)*tanh(sqrt(g*cd/m)*t)
```

```
>> scriptdemo
v =  50.617
>>
```

ex3.2

```
function v = freefall(t,m,cd)
  %freefall : bungee velocrit with second-orde
  %v = freefall(t.m.cd) computers the free-fa
  % second-order drag
  %input :
  % t = time(s)
  % m = mass (kg)
  % cd = second-order drag coefficient (kg/m)
  % output:
  % v = downward velocrit (m/s)
  g = 9.81; %acceleration of gravity
  v = sqrt(g*m/cd)*tanh(sqrt(g*cd/m)*t);
endfunction
```

```
>> freefall(12,68.1,0.25)
ans =  50.617
>> help freefall
'freefall' is a function from the file C:\Users\mi

freefall : bungee velocrit with second-order drag
v = freefall(t.m.cd) computers the free-fall veloc
 second-order drag
input :
 t = time(s)
 m = mass (kg)
 cd = second-order drag coefficient (kg/m)
 output:
 v = downward velocrit (m/s)


Additional help for built-in functions and operato
available in the online version of the manual.  Us
'doc <topic>' to search the manual index.

Help and information about Octave is also availabl
at https://www.octave.org and via the help@octave.
mailing list.
>> lookfor bungee
 v = freefall(t.m.cd) compute: bungee velocrit wit
  t =t :-order drag rs the free-fall velocity of a
  output:cond-order drag coefficient (kg/m)
                    v = downward velocrit (m/s)
```

ex3.3

```
function freefalli
  %freefalli : interactive bungee velocity
  %freefalli interactive computation of the
  % free-fall velocity of an object with secoi

  g = 9.81; %acceleration of gravity
  m = input ('Mass (kg) : ');
  cd = input ('Drag coefficient (kg/m) : ');
  t = input ( 'Time (s) : ');
  disp (' ')
  disp ('Velociry (m/s) : ')
  disp (sqrt(g*m/cd)*tanh(sqrt(g*cd/m)*t))
endfunction
```

```
>> freefalli
Mass (kg) : 68.1
Drag coefficient (kg/m) : 0.25
Time (s) : 12

Velociry (m/s) :
 50.617
>>
```

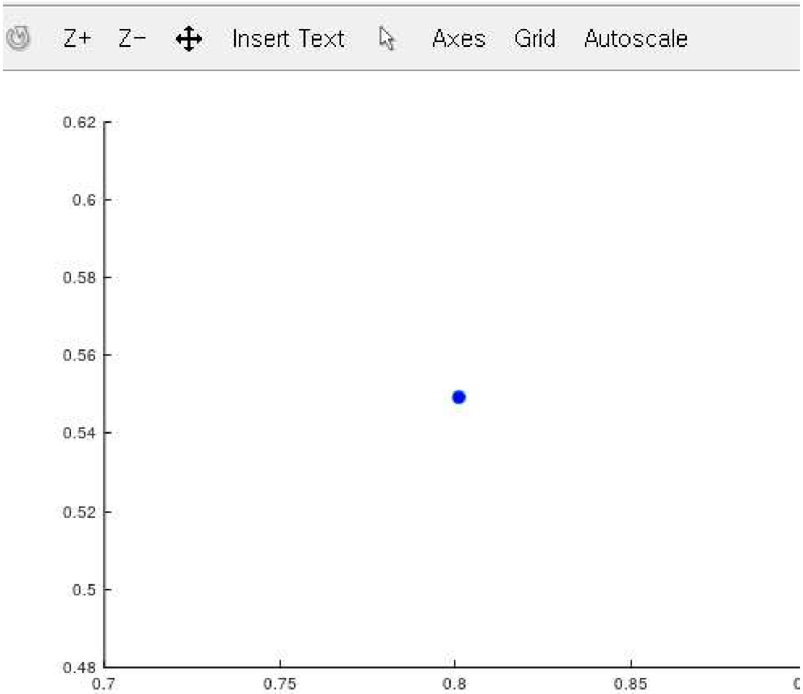
ex3.5

```
function fout = factor(n)
  %factor(n) :
  % computes the product of all the intefers i
  x = 1;
  for i = 1:n
    x = x*i;
  end
  fout = x;
endfunction
```

```
error: called from
    factor at line 5 column 3
>> factor(5)
ans =  120
>>
```

ex 3.6

```
clc, clf, clear
g=9.81; theta0=45*pi/180; v0=5;
t(1)=0;x=0;y=0;
plot(x,y,'o','MarkerFaceColor', 'b', 'MarkerS
axis([0 3 0 0.8])
M(1)=getframe;
dt=1/128;
for j = 2:1000
    t(j)=t(j-1)+dt;
    x=v0*cos(theta0)*t(j);
    y=v0*sin(theta0)*t(j)-0.5*g*t(j)^2;
    plot(x,y,'o','MarkerFaceColor','b','Marke
    axis([0 3 0 0.8])
    M(j)=getframe;
    if y<=0, break, end
end
pause
movie(M,1)
```



ex 3.7

```
function quadroots(a, b, c)
  %quadroots: roots of quadratic equation
  %  quadroots(a, b, c) : real and complex roots
                          %of quadratic equation
  % input :
  %  a = second=order coefficient
  %  b = first=order coefficient
  %  c = zero=order coefficient
  % output :
  %  r1 = real part of first root
  %  r2 = imaginary part of first root
  %  r3 = real part of second root
  %  r4 = imaginary part of second root
  if a == 0
    %special cases
    if b ~= 0
      %sinfle root
      ri = -c / b
    else
      %trivial solution
      disp('Trivial solution. Try again')
    end

  else
    %quadratic formula
    d = b ^ 2 - 4 * a * c; %discriminant
    if d>=0
      %real roots
      r1 = (-b + sqrt(d)) / (2*a)
      r2 = (-b - sqrt(d)) / (2*a)
    else
    % complex roots
    r1 = -b / (2*a)
    i1 = sqrt(abs(d)) / (2*a)
      r2 = r1
      i2 = -i1
      end
    end
  endfunction
```

```
>> ex37(1,1,1)
r1 = -0.50000
i1 =  0.86603
r2 = -0.50000
i2 = -0.86603
```

ex 3.8

```
function favg = funcavg(a,b,n)
%  funcavg : average function height
%     favg = funcavg(a,b,n) : computes average value
%  of function over a range
% input :
% a = lower cound of range
% b = upper bound of range
% n = number of intervals
% output :
%    favg = average value of function
x = linspace(a,b,n);
y = func(x);
favg = mean(y);
end
function f = func(t)
f = sqrt(9.81*69.1/0.25)*tanh(sqrt(9.81*0.25/68.1)*t);
end
```

```
>> ex38(0,12,60)
warning: function
ans =  36.276
>>
```

3.6 절

```
function vend = velocity1(dt,ti,tf,vi)
% velocity1 : Euler solution for bungee velocity1
%    vend = velocity1(dt,ti,tf,vi)
%           Euler method solution of bungee
%           jumper velocity
% input :
%    dt = time step (s)
%    ti = initial time(s)
%    tf = final time (s)
%    vi = initial value of dependent variable (m/s)
% output :
%    vend = velocity at tf (m/s)
t = ti;
v = vi;
n = (tf - ti) /dt;
for i = 1:n
  dvdt = deriv(v);
 v = v + dvdt*dt;
t = t + dt;
end
vend = v;
end
function dv = deriv(v)
dv = 9.81 - (0.25/68.1)*v*abs(v);
end
```

```
>> velocity1(0.5,0,12,0)
ans =  50.926
>>
```