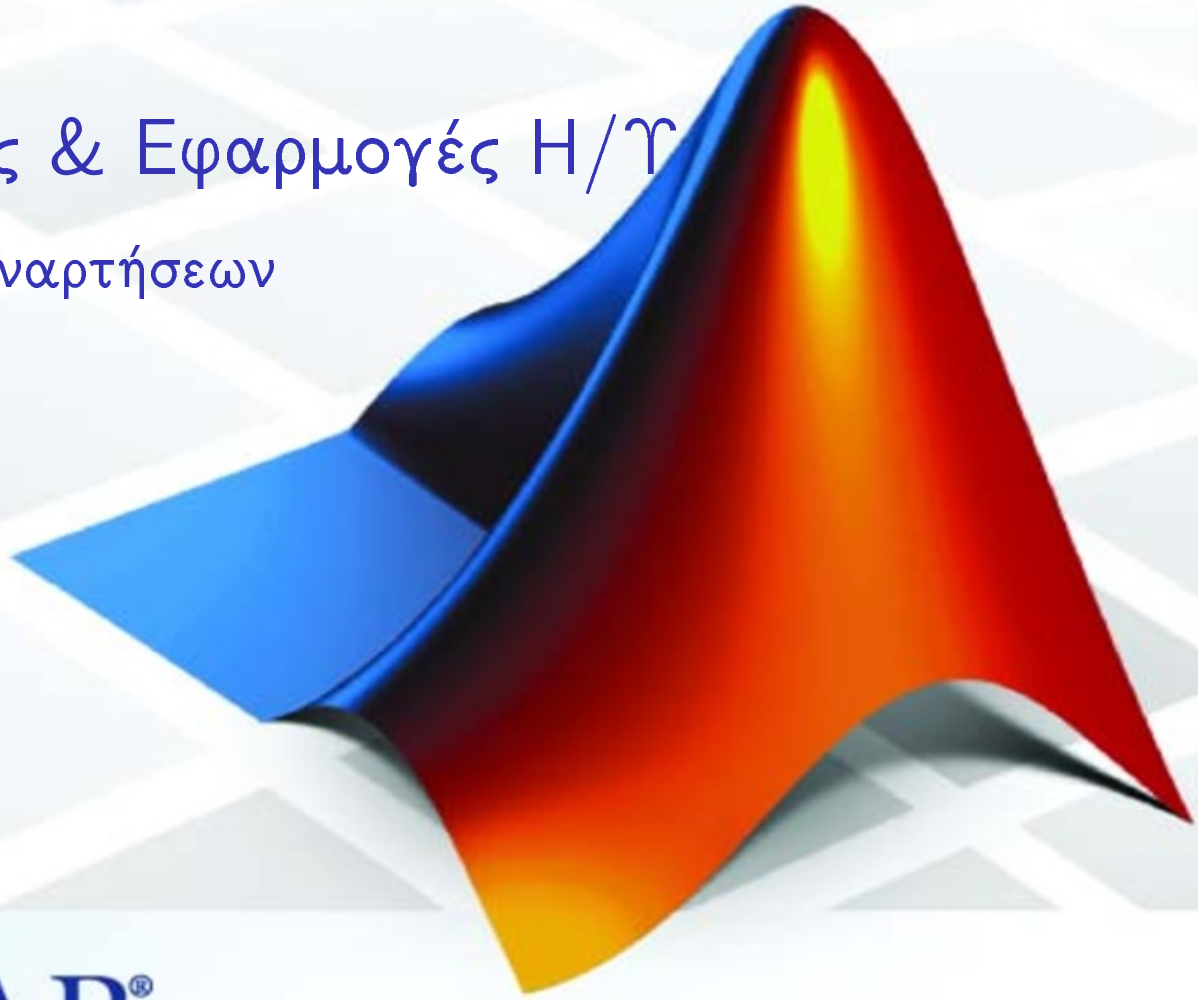
Προγραμματισμός & Εφαρμογές Η/Υ

Ορισμός και χρήση συναρτήσεων



Π. Οικονόμου

Τμήμα Πολιτικών Μηχανικών

2020–2021

MATLAB®

Σχέδιο Σημειώσεων

- Ορισμός συναρτήσεων
- Χρήση συναρτήσεων
- Υπό-συναρτήσεις
- Αναδρομή
- σύγκριση script και function .m αρχείων
- Παραδείγματα

Συναρτήσεις

Το MATLAB, όπως έχουμε δει, έχει πληθώρα ενσωματωμένων συναρτήσεων, οι οποίες είναι έτοιμες από το ίδιο το MATLAB (π.χ. η factor).

Η χρήση αυτών των συναρτήσεων γίνεται πληκτρολογώντας απλά το όνομά τους στο command window, ακολουθώντας φυσικά τη σωστή τους σύνταξη.

Πληροφορίες για τις συναρτήσεις αυτές μπορούμε να πάρουμε πληκτρολογώντας τις εντολές

```
>> help function_name
>> type function_name
```

Το MATLAB όμως επιτρέπει τη δημιουργία συναρτήσεων και από τον χρήστη με την βοήθεια των .m αρχείων.

Συναρτήσεις

Η εντολή function

Για τη δημιουργία μιας νέας συνάρτησης χρειαζόμαστε ένα αρχείο .m, στο οποίο στην πρώτη γραμμή του να πληκτρολογήσουμε

```
function [output_parameter_list] = function-name(input_parameter_list)
```

- η εντολή **function** πρέπει να εμφανίζεται στην αρχή της πρώτης γραμμής
- Αν η συνάρτηση επιστρέφει περισσότερα από ένα στοιχείο, τότε αυτά πρέπει υποχρεωτικά να δηλώνονται μέσα σε αγκύλες σαν διάνυσμα **([output_parameter_list])**
- το όνομα της συνάρτησης **(function_name)** πρέπει να είναι το ίδιο με το όνομα του αρχείου .m
- οι παράμετροι της συνάρτησης **([input_parameter_list])** πρέπει να δηλώνονται κατά την εκτέλεση της συνάρτησης.

function .m αρχείο – Παράδειγμα

```
1  function [m,sd] = stat(x)
2  %stat(x) returns two values,
3  %namely the mean value and
4  %the standard deviation of
5  %the elements in x
6  %
7  %x must be a vector with
8  %at least two elements
9
10 s=size(x);
11       if min(s)~=1
12          m=NaN;
13          sd=NaN;
14          disp('please enter a vector')
15          return
16      end
17
18      if max(s)==1
19          m=NaN;
20          sd=NaN;
21          disp('please enter a vector with at least two elements')
22          return
23      end
24
25 n = length(x);
26 m = sum(x)/n;
27 sd = sqrt(sum((x-m).^2/(n-1)));
28
29 end
```

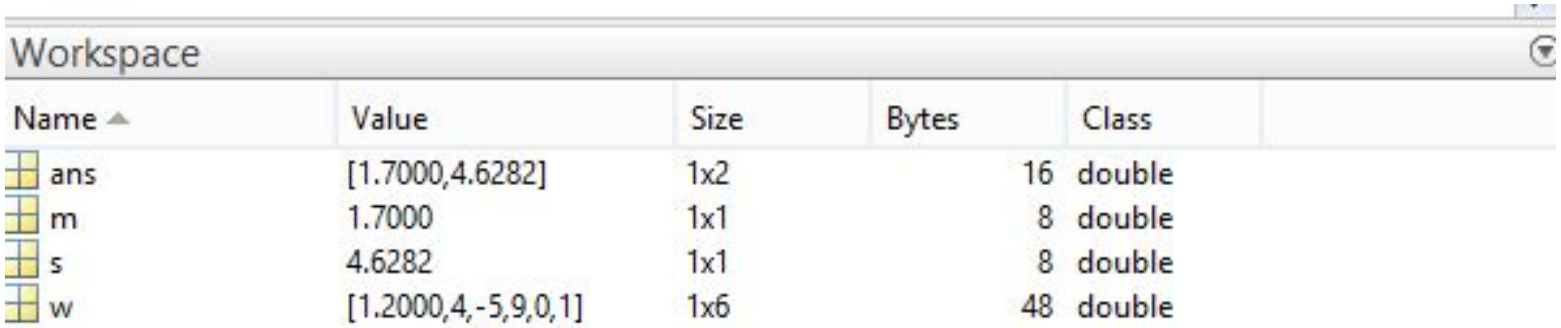
function .m αρχείο – Παράδειγμα

```
>> help stat
 stat(x) returns two values,
 namely the mean value and
 the standard deviation of
 the elements in x

 x must be a vector with
 at least two elements
```

```
>> type stat

function [m,sd] = stat(x)
%stat(x) returns two values,
%namely the mean value and
%the standard deviation of
%the elements in x
%
%x must be a vector with
%at least two elements
s=size(x);
     if min(s)~=1
        m=NaN;
        sd=NaN;
        disp('please enter a vector')
        return
    end

    if max(s)==1
        m=NaN;
        sd=NaN;
        disp('please enter a vector with at least two elements')
        return
    end

n = length(x);
m = sum(x)/n;
sd = sqrt(sum((x-m).^2/(n-1)));

end
```

function .m αρχείο – Παράδειγμα

```
>> w=[1.2, 4, -5, 9, 0, 1];
>> [m,s]=stat(w);
```



Workspace

Name	Value	Size	Bytes	Class
ans	[1.7000,4.6282]	1x2	16	double
m	1.7000	1x1	8	double
s	4.6282	1x1	8	double
w	[1.2000,4,-5,9,0,1]	1x6	48	double

```
>> [m,s]

ans =

    1.7000    4.6282
```

Υπό-συναρτήσεις

Σε ένα αρχείο .m μπορούμε να ορίσουμε παραπάνω από μια συνάρτηση.

Η πρώτη συνάρτηση, η οποία είναι και το όνομα του αρχείου .m, είναι η κύρια.

Οι επιπλέον συναρτήσεις μπορούν να γραφτούν μετά το τέλος της κύριας συνάρτησης. Οι συναρτήσεις αυτές, οι οποίες ονομάζονται υπό-συναρτήσεις, είναι προσβάσιμες μόνο από την κύρια συνάρτηση και από τις άλλες υπό-συναρτήσεις του αρχείου και δεν μπορούμε να τις καλέσουμε από το command window του MATLAB.

Υπό-συναρτήσεις – Παράδειγμα

Γράψτε μια συνάρτηση που θα υπολογίζει το εμβαδό ενός τριγώνου με βάση τις συντεταγμένες των κορυφών του.
Σημείωση: Αν γνωρίζουμε τις πλευρές a, b, c ενός τριγώνου το εμβαδό E του τριγώνου δίνεται από τη σχέση

$$E = \sqrt{S \cdot (S - a) \cdot (S - b) \cdot (S - c)}$$

όπου $S = \frac{a+b+c}{2}$.

Υπό-συναρτήσεις – Παράδειγμα - Λύση

```
1  function [TrArea] = triangleArea(points)
2  %calculates the are of a triangle
3  %given the coordinates of the three vertices
4  %
5  %give the coordinates in the form
6  %[xa,xb,xc;ya,yb,yc]
7  s = size(points);
8  if s(1) ~= 2 || s(2) ~= 3
9      TrArea = NaN;
10     disp('input error')
11     return
12 else
13      A = [points(1,1),points(2,1)];
14      B = [points(1,2),points(2,2)];
15      C = [points(1,3),points(2,3)];
16      a = side(B,C);
17      b = side(A,C);
18      c = side(A,B);
19      S = (a+b+c)/2;
20      TrArea = sqrt(S*(S-a)*(S-b)*(S-c));
21 end
22 end
23
24 function [x] = side(v1,v2)
25 %calculates the distance between two points
26
27 x = sqrt((v1(1,1)-v2(1,1))^2+(v1(1,2)-v2(1,2))^2);
28
29 end
```

```
>> x=[-1,2,2;3,3,-6];
>> ar=triangleArea(x);
>> ar
ar =
   13.5000
```

Αναδρομή – Recursion

Το MATLAB επιτρέπει στις συναρτήσεις να καλούν τον εαυτό τους. Παράδειγμα τέτοιας συνάρτησης είναι ο υπολογισμός του $n!$

```
1  function [result] = nfactorial(n)
2  %returns the n!
3  if round(n) ~= n || n < 0
4      disp([num2str(n),' is not a natural number'])
5      result = NaN;
6      return
7  end
8  if n < 3
9      result = n;
10 else
11     result = n*nfactorial(n-1);
12 end
13 end
```

```
-----------------------------------
>> nfactorial(11)
ans =
    39916800
```

Μετατροπή script σε function .m αρχείο

Στη συνέχεια παρουσιάζεται ένα script αρχείο για τον προσδιορισμό του

1. μέγιστου μέτρου όλων των ριζών
2. μέγιστου μέτρου όλων των πραγματικών ριζών

ενός πολυωνύμου.

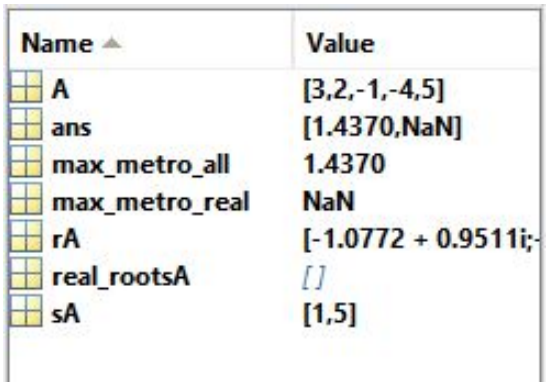
```
1          A=input('dwse tous syntelestes tou polyonumou toulaxiston 2ou bathou:');
2
3  sA=size(A);
4  while sA(1)~=1 ||sA(2)<3||A(1)==0
5      disp('input error, try again');
6      A=input('dwse tous syntelestes tou polyonumou toulaxiston 2ou bathou');
7      sA=size(A);
8  end
9
10 rA=roots(A);
11 max_metro_all=max(abs(rA));
12 real_rootsA=find(imag(rA)==0);
13
14 if ~isempty(real_rootsA)
15     max_metro_real=max(abs(rA(real_rootsA)));
16 else
17     max_metro_real=NaN;
18 end
19
20 [max_metro_all,max_metro_real]
```

Μετατροπή script σε function .m αρχείο

```
1  function [ max_metro_all,  max_metro_real] = absofroots( A )
2  %type help.....
3
4      sA = size(A);
5      if sA(1) ~= 1 ||sA(2) < 3||A(1)==0
6          disp('input error, try again');
7          max_metro_all = NaN;
8          max_metro_real = NaN;
9      end
10
11     rA = roots(A);
12     max_metro_all = max(abs(rA));
13     real_rootsA = find(imag(rA)==0);
14
15     if ~isempty(real_rootsA)
16         max_metro_real = max(abs(rA(real_rootsA)));
17     else
18         max_metro_real = NaN;
19     end
20 end
```

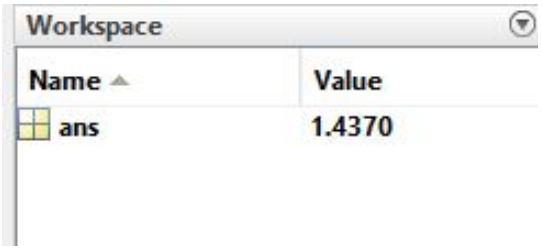
Σύγκριση script και function .m αρχείων

```
>> scriptfile
dwse tous syntelestes tou polyonumou toulaxiston 2ou bathou:[3,2,-1,-4,5]

ans =

    1.4370       NaN
```

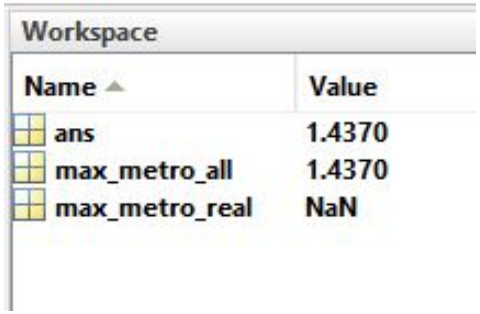


Σύγκριση script και function .m αρχείων

```
>> absofroots( [3,2,-1,-4,5] )
ans =
    1.4370
```




```
>> max_metro_all
Undefined function or variable 'max_metro_all'.

>>[max_metro_all,max_metro_real]=absofroots([3,2,-1,-4,5])

max_metro_all =
    1.4370

max_metro_real =
   NaN
```



Σύγκριση script και function .m αρχείων

Τα script και function .m αρχεία έχουν αρκετές ομοιότητες αλλά και ουσιώδεις διαφορές, όπως:

- Στα function .m αρχεία οι παράμετροι της συνάρτησης, δηλώνονται κατά την κλήση της συνάρτησης ενώ στα sript .m μπορούμε να εισάγουμε τις τιμές που θέλουμε με την εντολή input.
- Οι μεταβλητές που ορίζονται στα function .m αρχεία είναι τοπικές, ενώ στα sript .m αρχεία όχι.
- Στα function .m αρχεία μπορούν να οριστούν και επιπρόσθετες συναρτήσεις (υπό-συναρτήσεις), ενώ στα στα sript .m αρχεία όχι.

Κέντρο βάρους

Κατασκευάστε μια συνάρτηση που να μας δίνει το κέντρο βάρους μιας δοκού (χωρίς βάρος), όταν τοποθετούμε σε αυτήν βάρη σε διάφορα σημεία της. Δοκιμάστε τη συνάρτηση τοποθετώντας τα βάρη

$$weights = [9, 2, 3, 1, 1]$$

στις αποστάσεις

$$locations = [1, 14, 15, 18, 19]$$

από την αρχή της δοκού, η οποία αρχή υποθέτουμε ότι βρίσκεται στο μηδέν.

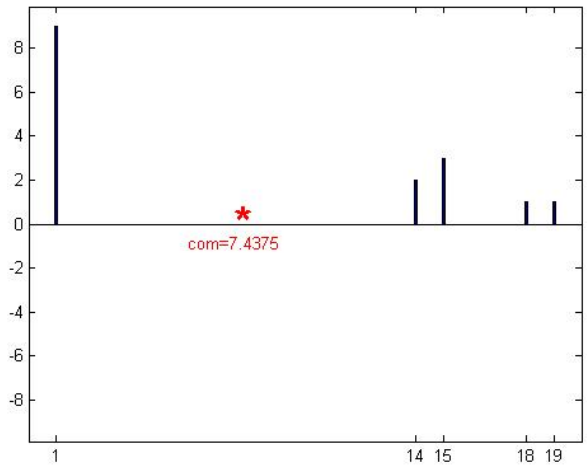
Απάντηση: κέντρο βάρους $= 7.4375$

$\hookrightarrow$ Μπορείτε να παραστήσετε και οπτικά το κέντρο βάρους χρησιμοποιώντας την εντολή $bar(locations, weights, 0.1)$;

```
1   function [ x ] = kentrovarous( w,l )
2   %kentro varous
3   if length(w)~=length(l)
4       x=NaN;
5       disp('input error')
6       return
7   end
8
9   x=sum(w.*l)/sum(w);
10
11  end
```

```
>>locations=[1,14,15,18,19];
>>weights=[9,2,3,1,1];
>> kentrovarous(weights,locations)
ans =
    7.4375
```

```
>>com(weights,locations);
```



Εφαπτομένη συνάρτησης

Κατασκευάστε μια συνάρτηση που να μας δίνει την εφαπτομένη της συνάρτησης

$$f(x) = log(x)$$

στο σημείο (x_0, y_0).
Δοκιμάστε τη συνάρτηση στο σημείο $(x_0, y_0) = (2, log(2))$.

Απάντηση: $y = 0.5x - 0.30685$

↪ Τροποποιήστε τη συνάρτηση έτσι ώστε να δίνει τη συνάρτηση εφαπτομένης για κάθε συνεχή συνάρτηση, το τύπο της οποίας θα τον εισάγει ο χρήστης. (νέες εντολές: *feval* και *inline*)

↪ Έχετε λάβει υπόψη σας την περίπτωση ο χρήστης να εισάγει μια συνάρτηση και ένα σημείο στο οποίο η συνάρτηση έχει κατακόρυφη εφαπτομένη (π.χ. $f(x) = \sqrt{x},\ x \geq 0$ και $(x_0, y_0) = (0, 0)$);

```
1  function [slope,constant] = tangentlogx(x0,xmin,xmax)
2  %returns of the tangent line of the continous function log(x)
3  %at the point (x0,f(x0)) and plots f(x)=log(x) and the tangent line
4  %of f(x) at (x0,f(x0)) over the intermal (xmin,xmax)
5  %Note that x0, xmin and xmax must satisfy xmin<x0 and x0<xmax
6  if ~(xmin<x0 & x0<xmax)
7      slope=NaN;constant=NaN;
8      disp('input error')
9      return
10 end
11
12 slope=df(x0);
13 constant= f(x0)-df(x0)*x0;
14 disp(['y=',num2str(slope),'x+',num2str(constant)])
15
16 %plots f(x)=log(x) and the tangent line of the
17 %at the point (x0,f(x0))
18 d=xmax-xmin;
19 x=xmin:d/100:xmax;
20 y1=f(x);
21 y2=slope.*x+constant;
22 plot(x,y1,x,y2)
23 end
24
25 function [x]=f(x0)
26 %calculates log(x0)
27     x=log(x0);
28 end
29
30 function [x]=df(x0)
31 %calculates the derivative of f(x) at x0
32     x=1/x0;
33 end
```

```
1  function [slope,constant] = tangent(fin,dfin,x0,xmin,xmax)
2  %returns of the tangent line of a continous function fin at (x0,fin(x0))
3  %call: tangent(fin,dfin,x0,xmin,xmax)
4  %examples
5  % tangent(inline('cos(x)'),inline('-sin(x)'),1,-2,5)
6  % tangent(inline('sqrt(x)'),inline('1/(2*sqrt(x))'),0,-2,5)
7  % tangent(inline('sqrt(x)'),inline('1/(2*sqrt(x))'),2,0,5)
8
9  if ~(xmin<=x0 & x0<=xmax)
10     slope=NaN;constant=NaN;
11     disp('input error')
12     return
13 end
14 slope=df(dfin,x0);
15
16 if slope==Inf | slope==-Inf
17  disp(['x=',num2str(x0)])
18  constant=NaN;
19  return
20 end
21 constant= f(fin,x0)-df(dfin,x0)*x0;
22
23 disp(['y=',num2str(slope),'x+',num2str(constant)])
24
25 %plots f(x) and the tangent line of the at the point (x0,f(x0))
26 d=xmax-xmin;        x=xmin:d/100:xmax;
27 y1=feval(fin,x);    y2=slope.*x+constant;
28 plot(x,y1,x,y2)
29 end
30
31 function [x]=f(fin,x0)
32 %calculates f(x0)
33     x=feval(fin,x0);
34 end
35
36 function [x]=df(dfin,x0)
37 %calculates the derivative of f(x) at x0
38     x=feval(dfin,x0);
39 end
```

Εξετάσεις 6/2014

Ένας μηχανικός είχε φτιάξει παλιότερα μια συνάρτηση (*functionfile*) στο *MATLAB* με το όνομα *converttemp*, με τη βοήθεια της οποίας μετέτρεπε τη θερμοκρασία από βαθμούς *Celsius* σε βαθμούς *Fahrenheit* και βαθμούς *Kelvin*.

Μάλιστα, είχε κρατήσει και την ακόλουθη εκτύπωση του αποτελέσματος της συνάρτησης για θερμοκρασια 10 βαθμών *Celsius*.

```
[F,K]=converttemp(-10)
F =
    14
K =
  263.1500
```

Βοηθήστε το μηχανικό να ξαναφτιάξει τη συνάρτηση αυτή. Δίνεται ότι

$Fahrenheit = 9/5 Celsius + 32$ και $Kelvin = Celsius + 273.15$.

Εξετάσεις 9/2014

- Δημιουργήστε μια συνάρτηση (*functionfile*) η οποία θα υπολογίζει και θα επιστρέφει (υπό μορφή διανύσματος) τους συντελεστές του πολυωνύμου με συγκεκριμένες ρίζες. Η συνάρτηση πρέπει να δέχεται σαν όρισμα το διάνυσμα των ριζών του πολυωνύμου.

  Παραθέστε τον τρόπο εκτέλεσης της συνάρτησης καθώς και το αποτέλεσμά της για το πολυώνυμο με ρίζες το 2, το 3 και το 4.

  >>

- Συμπληρώστε την παραπάνω συνάρτηση, έτσι ώστε να επιστρέφει και την τιμή του πολυωνύμου στο σημείο $x = -1$.

  Παραθέστε τον τρόπο εκτέλεσης της συνάρτησης καθώς και το αποτέλεσμά της για το πολυώνυμο με ρίζες το 2, το 3 και το 4.

  >>

Τυχερό παιχνίδι

Κατασκευάστε μια συνάρτηση που να μας δίνει τους τυχερούς αριθμούς και τον νικητή στο ακόλουθο παιχνίδι.

> Κ παίκτες επιλέγουν τυχαία s διαφορετικούς αριθμούς ανάμεσα στους $1, 2, \ldots, n$.
>
> Ο υπολογιστής επιλέγει και αυτός s αριθμούς ανάμεσα στους $1, 2, \ldots, n$. Αν ένας παίκτης έχει βρει και τους s τυχερούς αριθμούς τότε ανακυρήσσεται νικητής.

Τυχερό παιχνίδι

↪ Για την κατασκευή της συνάρτησης χρήσιμη μπορεί να σας φανεί η εντολή *nchoosek*, την οποία θα την καλέσετε ως

```
>> nchoosek(1:n,s)
>> nchoosek(n,s)
```

Στην περίπτωση που χρησιμοποιήσετε την εντολή αυτή το n πρέπει να είναι το πολύ 15.

↪ Για να μην έχετε περιορισμό στο n μπορείτε εναλλακτικά να χρησιμοποιήσετε την εντολή *randperm*.

Θέμα - 2η πρόοδος 1/2017

Απαντήστε στις ερωτήσεις για την παρακάτω συνάρτηση.

```
1   function [ w ] = thema1B(x)
2       sx=size(x);
3       if max(sx)~=1
4           w=NaN;
5           disp('input error')
6           return
7       end
8       if x<=0
9           w=2;
10      else
11          w=2+thema1B(x-1);
12      end
13  end
```

i. Εξηγήστε εν συντομία ποιος είναι ο σκοπός της ομάδας εντολών που εμφανίζονται στις γραμμές 3 έως 7.

ii. Παραθέστε τον τρόπο εκτέλεσης της συνάρτησης για x=2.3.

>>

iii. Ποιο είναι το αποτέλεσμα της συνάρτησης για x=2.3;

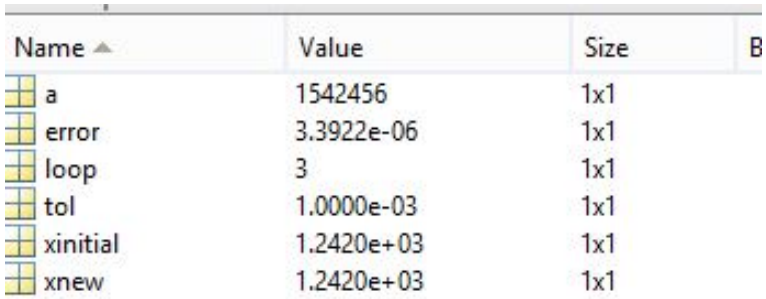
Σύγκριση script και function .m αρχείων

Σε προηγούμενο μάθημα είχαμε δημιουργήσει ένα script αρχείο για την αριθμητική προσέγγιση της τετραγωνικής ρίζας ενός θετικού αριθμού. Στη συνέχεια θα φτιάξουμε μια συνάρτηση που θα κάνει το ίδιο.

```
1  function [xnew] = herosfunction(a,xinitial)
2  %use the Heron's formula to determine
3  %the square root of a positive number a.
4  %User has to provide an initial quess
5  tol=10^(-3);
6  xnew=(xinitial+a/xinitial)/2;
7  error=abs(xinitial-xnew);
8  loop=0;
9  while error>=tol
10         xinitial=xnew;
11         xnew=(xinitial+a/xinitial)/2;
12             loop=loop+1;
13         error=abs(xinitial-xnew);
14 end
15 disp(['You have found the square root of ',num2str(a), ' after ', num2str(loop), ' loops.
       '])
16 disp(['The square root of ',num2str(a), ' is ', num2str(xnew)])
17
18 end
```

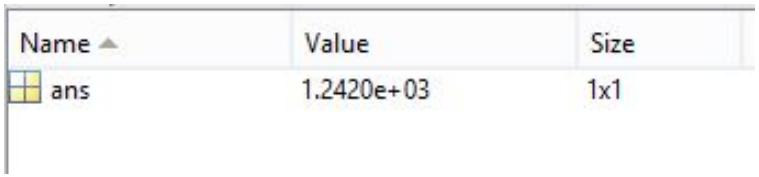
Σύγκριση script και function .m αρχείων

```
>> heros_script
Enter a: 1542456
Enter initial guess: 1452
You have found the square root of 1542456 after 3 loops.
The square root of 1542456 is 1241.9565
```



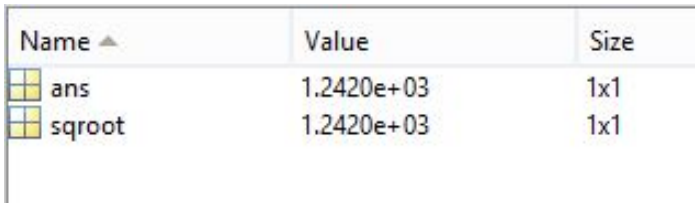
Name	Value	Size	B
a	1542456	1x1	
error	3.3922e-06	1x1	
loop	3	1x1	
tol	1.0000e-03	1x1	
xinitial	1.2420e+03	1x1	
xnew	1.2420e+03	1x1	

```
>> herosfunction(1542456,1452);
You have found the square root of 1542456 after 3 loops.
The square root of 1542456 is 1241.9565
```



Name	Value	Size
ans	1.2420e+03	1x1

```
>> xnew
Undefined function or variable 'xnew'.

>> sqroot=herosfunction(1542456,1452);
You have found the square root of 1542456 after 3 loops.
The square root of 1542456 is 1241.9565
```



Name	Value	Size
ans	1.2420e+03	1x1
sqroot	1.2420e+03	1x1