Εισαγωγή στην Τεχνητή Νοημοσύνη

(ΜΠΔ 306)

Εργαστήριο

Εργαστηριακός βοηθός: Κατερίνα Γιαννακάκη

e-mail: agiannakaki1@isc.tuc.gr

Εισαγωγή στο MATLAB

Πράξεις με διανύσματα/πίνακες

ΑΝΑΣΤΡΟΦΟΣ

transpose

Αν θέλουμε να μετατρέψουμε ένα διάνυσμα-γραμμή σε διάνυσμα-στήλη ή το αντίστροφο, χρησιμοποιούμε την εντολή **transpose** :

```
>> a1 = [1 2 3 4 5]
a1 =
     1 2 3 4 5

>> b1 = transpose(a1)
b1 =
     1
     2
     3
     4
     5
```

Πράξεις με διανύσματα/πίνακες

ΑΝΑΣΤΡΟΦΟΣ

transpose

Αν θέλουμε να μετατρέψουμε ένα διάνυσμα-γραμμή σε διάνυσμα-στήλη ή το αντίστροφο, χρησιμοποιούμε την εντολή **transpose** :

```
>> a2 = [1;2;3;4;5]
a2 =
     1
     2
     3
     4
     5

>> b2 = transpose(a2)
b2 =
     1  2  3  4  5
```

Πράξεις με διανύσματα/πίνακες

ΑΝΑΣΤΡΟΦΟΣ

`transpose`

Αν θέλουμε να μετατρέψουμε ένα διάνυσμα-γραμμή σε διάνυσμα-στήλη ή το αντίστροφο, χρησιμοποιούμε την εντολή **`transpose`**

Εναλλακτικά, για συντομία, μπορούμε να χρησιμοποιήσουμε την **απόστροφο** (single quotation mark)

```
>> a3 = [1 2 3 4 5];
>> b3 = a3'
b3 =
     1
     2
     3
     4
     5
```

Πράξεις με διανύσματα/πίνακες

ΑΝΑΣΤΡΟΦΟΣ

`transpose`

Με τον ίδιο τρόπο μπορούμε να βρούμε τον ανάστροφο ενός **πίνακα**, δηλαδή έναν νέο πίνακα που θα έχει γραμμές τις στήλες του αρχικού και στήλες τις γραμμές του αρχικού.

```
>> A = [10   3.8   -50;-6.45   0   22];
>> transpose(A)
ans =
   10.00 -6.45
    3.80  0.00
  -50.00 22.00

>> A'
ans =
   10.00 -6.45
    3.80  0.00
  -50.00 22.00
```

Πράξεις με διανύσματα/πίνακες

ΠΡΟΣΘΕΣΗ & ΑΦΑΙΡΕΣΗ

Πρόσθεση/Αφαίρεση μεταξύ **διανυσμάτων/πινάκων**:

Γίνονται στοιχείο προς στοιχείο
Οι **διαστάσεις** των διανυσμάτων/πινάκων πρέπει να **ταιριάζουν**!

Για πρόσθεση/αφαίρεση μεταξύ διανυσμάτων, πρέπει **όλα** να είναι διανύσματα-γραμμής ή **όλα** να είναι διανύσματα στήλης και φυσικά **όλα** να έχουν ίδιο μήκος.

```
>> a = [10  3.8  -50  -6.45  0  22]; b = [2.25  0.11  8  0  -1  2];
>> S = a + b
S =
   12.25    3.91  -42   -6.45   -1   24
```

Πράξεις με διανύσματα/πίνακες

ΠΡΟΣΘΕΣΗ & ΑΦΑΙΡΕΣΗ

Πρόσθεση/Αφαίρεση μεταξύ **διανυσμάτων/πινάκων**:

Γίνονται στοιχείο προς στοιχείο
Οι **διαστάσεις** των διανυσμάτων/πινάκων πρέπει να **ταιριάζουν**!

Για πρόσθεση/αφαίρεση μεταξύ πινάκων, πρέπει **όλοι** να έχουν τις ίδιες διαστάσεις.

```
>> A = [10   3.8   -50;-6.45   0   22];          Διαστάσεις: 2x3
>> R = [8   7   9;10   1   10];                  Διαστάσεις: 2x3
ans =
       2.00   -3.20  -59.00
     -16.45   -1.00   12.00                      Διαστάσεις: 2x3
```

Πράξεις με διανύσματα/πίνακες

ΠΡΟΣΘΕΣΗ & ΑΦΑΙΡΕΣΗ

Πρόσθεση/Αφαίρεση μεταξύ **διανύσματος** και **αριθμού**:

Προσθέτει/Αφαιρεί τον αριθμό σε/από **κάθε** στοιχείο του διανύσματος

```
>> a = [10   3.8   -50   -6.45   0   22];
>> a - 5
ans =
    5    -1.2   -55   -11.45    -5    17
```

Πράξεις με διανύσματα/πίνακες

ΠΡΟΣΘΕΣΗ & ΑΦΑΙΡΕΣΗ

Πρόσθεση/Αφαίρεση μεταξύ **πίνακα** και **αριθμού**:

Προσθέτει/Αφαιρεί τον αριθμό σε/από **κάθε** στοιχείο του πίνακα

```
>> A = [10  3.8  -50;-6.45  0  22];
>> A - 5
ans =
      5.00  -1.20 -55.00
    -11.45  -5.00  17.00
```

Πράξεις με διανύσματα/πίνακες

ΠΡΟΣΘΕΣΗ & ΑΦΑΙΡΕΣΗ

Πρόσθεση των στοιχείων **διανύσματος/πίνακα** μεταξύ τους:

sum()

```
>> a = [10   3.8   -50   -6.45   1   22];
>> sum(a)
ans =
   -19.65
```

Πράξεις με διανύσματα/πίνακες

ΠΡΟΣΘΕΣΗ & ΑΦΑΙΡΕΣΗ

Πρόσθεση των στοιχείων **διανύσματος/πίνακα** μεταξύ τους:

sum()

```
>> A = [1   3   -5;-6   1   2];
>> sum(A)
ans =
    -5      4    -3
```

αθροίζει τα στοιχεία κάθε στήλης

Για να αθροίσει ανά γραμμή:

```
>> sum(A,2)
ans =
    -1
    -3
```

εναλλακτικά: `>> sum(A')`

Πράξεις με διανύσματα/πίνακες

ΠΟΛΛΑΠΛΑΣΙΑΣΜΟΣ

Πολλαπλασιασμός μεταξύ **διανυσμάτων**:

Μπορεί να γίνει μόνο πολλαπλασιασμός στοιχείο προς στοιχείο και μόνο αν τα διανύσματα έχουν ίδιο μήκος. Αυτό γίνεται αν μαζί με το σύμβολο του πολλαπλασιασμού χρησιμοποιήσουμε και την τελεία:

```
>> a = [10  3.8  -50  -6.45  0  22]; b = [2.25  0.11  8  0  -1  2];
>> P = a .* b
P =
   22.5  0.418 -400  0  0   44
```

Πράξεις με διανύσματα/πίνακες

ΠΟΛΛΑΠΛΑΣΙΑΣΜΟΣ

Πολλαπλασιασμός μεταξύ **διανύσματος** και **αριθμού** :

Πολλαπλασιασμός του αριθμού με **κάθε** στοιχείο του διανύσματος:

```
>> a = [10   3.8   -50   -6.45   0   22];
>> a * 2
ans =
    20     7.6      -100     -12.9        0       44
```

Πράξεις με διανύσματα/πίνακες

ΠΟΛΛΑΠΛΑΣΙΑΣΜΟΣ

Πολλαπλασιασμός μεταξύ **πινάκων**:

Ισχύουν οι κανόνες της Γραμμικής Άλγεβρας. Για είναι δυνατός πολλαπλασιασμός μεταξύ δυο πινάκων θα πρέπει ο αριθμός στηλών του ενός να είναι **ίσος** με τον αριθμό γραμμών του άλλου. Οι διαστάσεις τους δηλαδή θα πρέπει να είναι:

[Mx**N**] * [**N**xK]

```
>> A = [10   3.8   -50;-6.45   0   22];
>> B = [2.25   0   0.11  3;-1   8   0   9.2;5   8   -1   2];
>> C = A*B
C =
   -231.3000   -369.6000   51.1000   -35.0400
     95.4875    176.0000  -22.7095    24.6500
```

Διαστάσεις: 2x**3** (Mx**N**)

Διαστάσεις: **3**x4 (**N**xK)

Το αποτέλεσμα είναι πίνακας διαστάσεων 2x4 (MxK)

Πράξεις με διανύσματα/πίνακες

ΠΟΛΛΑΠΛΑΣΙΑΣΜΟΣ

Πολλαπλασιασμός μεταξύ **πίνακα** και **αριθμού** :

Πολλαπλασιασμός του αριθμού με **κάθε** στοιχείο του πίνακα:

```
>> A = [10  3.8  -50;-6.45  0  22];
>> A * 2
ans =
   20.0  7.6     -100.0
  -12.9   0.0      44.0
```

Πράξεις με διανύσματα/πίνακες

ΠΟΛΛΑΠΛΑΣΙΑΣΜΟΣ

Πολλαπλασιασμός των στοιχείων **διανύσματος/πίνακα** μεταξύ τους:

prod()

```
>> a = [10   3.8   -50   -6.45   1   22];
>> prod(a)
ans =
   269610
```

Πράξεις με διανύσματα/πίνακες

ΠΟΛΛΑΠΛΑΣΙΑΣΜΟΣ

Πολλαπλασιασμός των στοιχείων **διανύσματος/πίνακα** μεταξύ τους:

prod()

```
>> A = [1   3   -5;-6   1   2];
>> prod(A)
ans =
   -6      3    -10
```

πολλαπλασιάζει τα στοιχεία κάθε στήλης

Για να πολλαπλασιάσει ανά γραμμή:

```
>> prod(A,2)
ans =
   -15
   -12
```

Πράξεις με διανύσματα/πίνακες

ΠΡΑΞΕΙΣ ΣΤΟΙΧΕΙΟ ΠΡΟΣ ΣΤΟΙΧΕΙΟ

ΓΕΝΙΚΑ για πράξεις στοιχείο προς στοιχείο χρησιμοποιούμε την **τελεία** πριν από το σύμβολο της πράξης:

Πολλαπλασιασμός των διανυσμάτων a και b στοιχείο προς στοιχείο: **a.*b**
Διαίρεση του διανύσματος a με το διάνυσμα b στοιχείο προς στοιχείο: **a./b**
Ύψωση σε δύναμη του διανύσματος a με το b στοιχείο προς στοιχείο: **a.^b**

```
>> a=[2 4 6], b=[3 2 1]
>> a.*b
ans =
     6     8     6
>> a./b
ans =
    0.6667    2.0000    6.0000
>> a.^b
ans =
     8    16     6
```

Παραδείγματα ενεργειών στο Command Window

Θεωρείστε ότι οι ενέργειες πραγματοποι-ούνται με τη σειρά, από πάνω προς τα κάτω

Τι γράφουμε στο Command Window	Τι εμφανίζεται στο Command Window	Τι καταχωρείται στο Workspace
`a = 2`	`a = 2`	Matrix 1x1 (με τιμή: 2)
`a = 2;`	τίποτα	Matrix 1x1 (με τιμή: 2)
`b = 'low'`	`b = 'low'`	Char array 1x3 (με τιμές: `'l' 'o' 'w'`)
`b = 5;`	τίποτα	Matrix 1x1 (με τιμή: 5)
`c = a + b`	`c = 7`	Matrix 1x1 (με τιμή: 7)
`c = a + b;`	τίποτα	Matrix 1x1 (με τιμή: 7)
`clear a`	τίποτα	Διαγραφή της μεταβλητής a
`a`	Μήνυμα σφάλματος: `Undefined function or variable 'a'` (δεν υπάρχει πια)	
`clear`	τίποτα	Διαγραφή όλων των μεταβλητών
`a = 2, b = 5`	`a = 2` `b = 5`	Matrix 1x1 (με τιμή: 2) Matrix 1x1 (με τιμή: 5)
`a = 2, b = 5;`	`a = 2`	Matrix 1x1 (με τιμή: 2) Matrix 1x1 (με τιμή: 5)
`% a = 2`	τίποτα (είναι σχόλιο)	τίποτα

Τι γράφουμε στο Command Window	Αποτέλεσμα
`a = [1 4 10]` ή `a = [1,4,10]`	Δημιουργείται διάνυσμα-γραμμή 1x3: a = 1 4 10
`a(2)`	`ans = 4`
`a = [1;4;10]`	Δημιουργείται διάνυσμα-στήλη 3x1: a = 1 4 10
`a(3)`	`ans = 10`
`a = [1 2 3;4 5 6;7 8 9]`	Δημιουργείται πίνακας 3x3: a = 1 2 3 4 5 6 7 8 9
`a(2,3)`	`ans = 6`
`a(3,:)`	`ans = 7 8 9`

και καταργείται το **διάνυσμα** a

Δημιουργία διανυσμάτων & πινάκων

Τι γράφουμε στο Command Window	Αποτέλεσμα
`b = [4 5 6], z = [1 2 3]`	Δημιουργούνται δύο διανύσματα 1x3: b = 4 5 6 z = 1 2 3
`c = b + z`	Αθροίζονται τα δύο διανύσματα, στοιχείο προς στοιχείο: 4+1 5+2 6+3 c = 5 7 9
`e = b * a`	Πολλαπλασιάζονται οι δυο πίνακες e = 66 81 96
`y = b.*z` (για πίνακες **ίδιων** διαστάσεων)	Πολλαπλασιάζονται τα στοιχεία των πινάκων 1 προς 1 y = 4 10 18

Κατά το πολλαπλασιασμό δύο πινάκων BxA θα πρέπει ο αριθμός στηλών του πίνακα Β να ισούται με τον αριθμό γραμμών του πίνακα Α

εδώ έχουμε:
[1x**3**]*[**3**x3]

Αρχικοποίηση

ΑΥΤΟΜΑΤΗ ΑΡΧΙΚΟΠΟΙΗΣΗ

`d = linspace(x1,x2,n)`

Αρχικοποιεί ένα γραμμικό διάνυσμα:

Δημιουργεί ένα διάνυσμα με `n` τιμές, ξεκινώντας από την τιμή `x1` και καταλήγοντας στην τιμή `x2`, με βήμα $\frac{x2-x1}{n-1}$

```
>> a = linspace(0,10,5)
a =
    0.0    2.5    5.0    7.5   10.0
```

Αρχικοποίηση

εναλλακτικά:

```
d = [x1:step:x2]
```

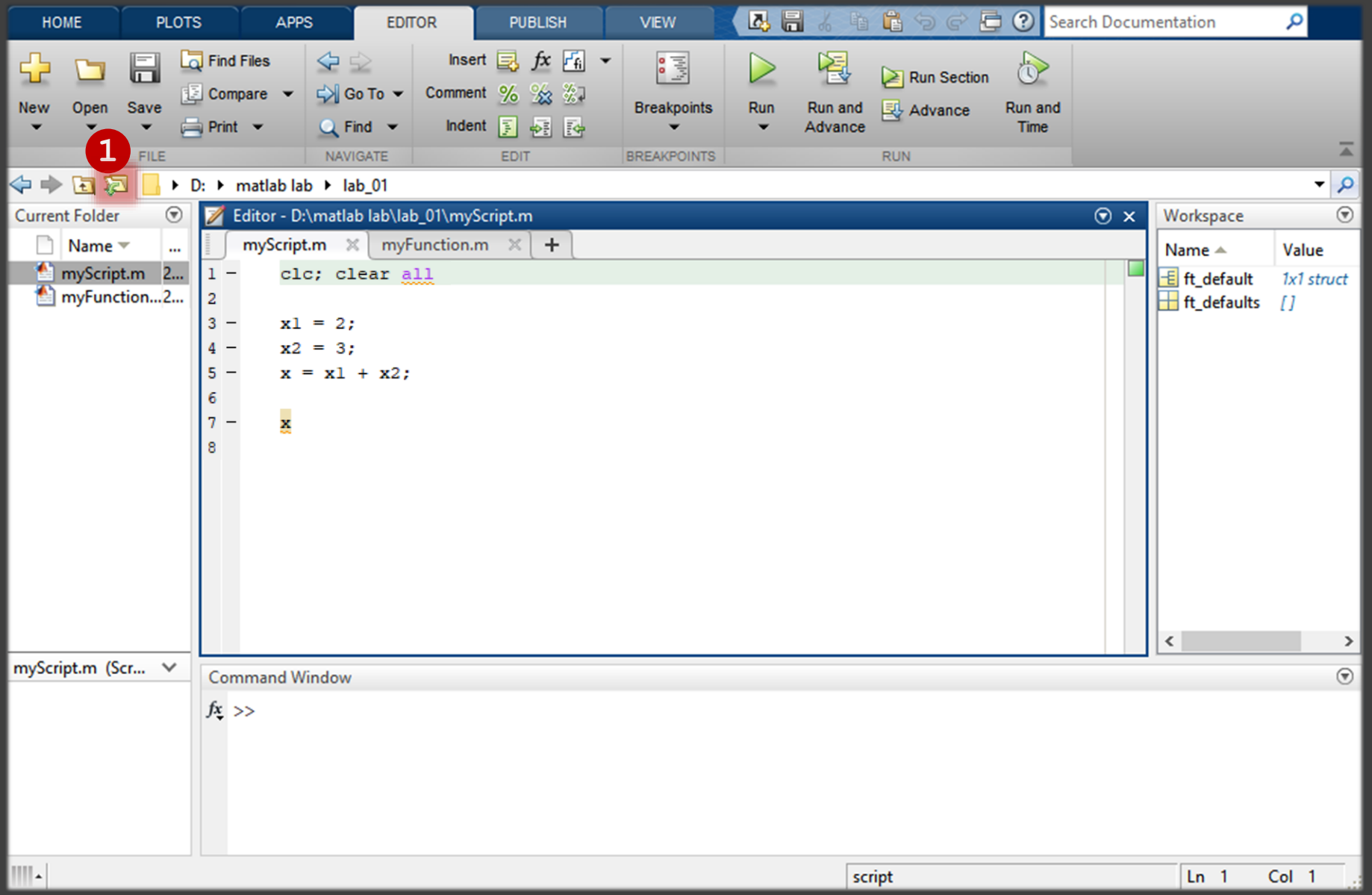
Δημιουργεί ένα διάνυσμα ξεκινώντας από την τιμή `x1` και καταλήγοντας στην τιμή `x2`, με βήμα `step`

```
>> a = [0:2.5:10]
a =
    0.0    2.5    5.0    7.5   10.0
```

- Το βήμα μπορεί να είναι ακέραιος, δεκαδικός ή και αρνητικός αριθμός
- Αν δεν δηλωθεί βήμα, χρησιμοποιείται ως βήμα το 1

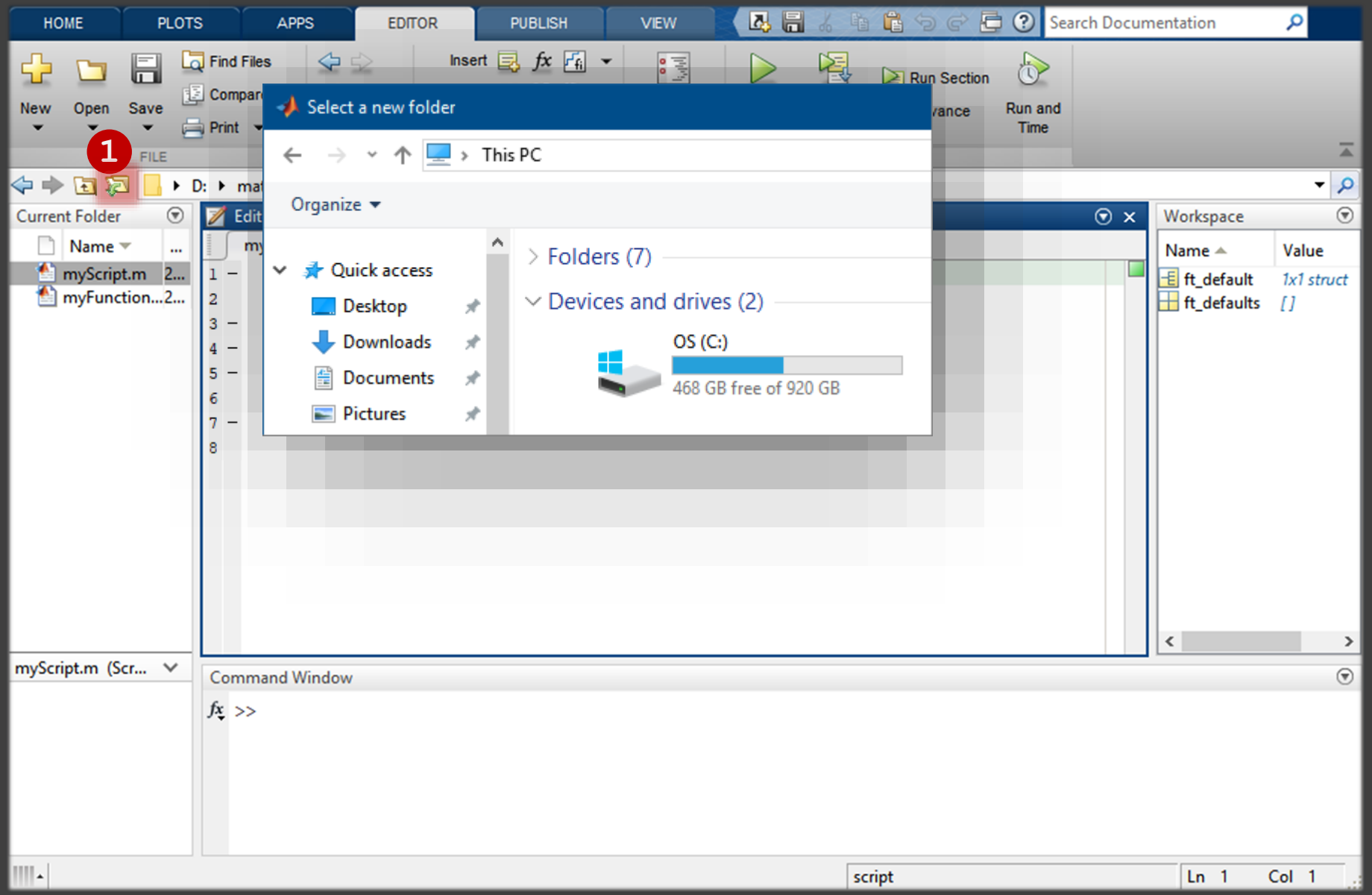
```
>> a = [1:5]
a =
    1   2   3   4   5
```

Scripts



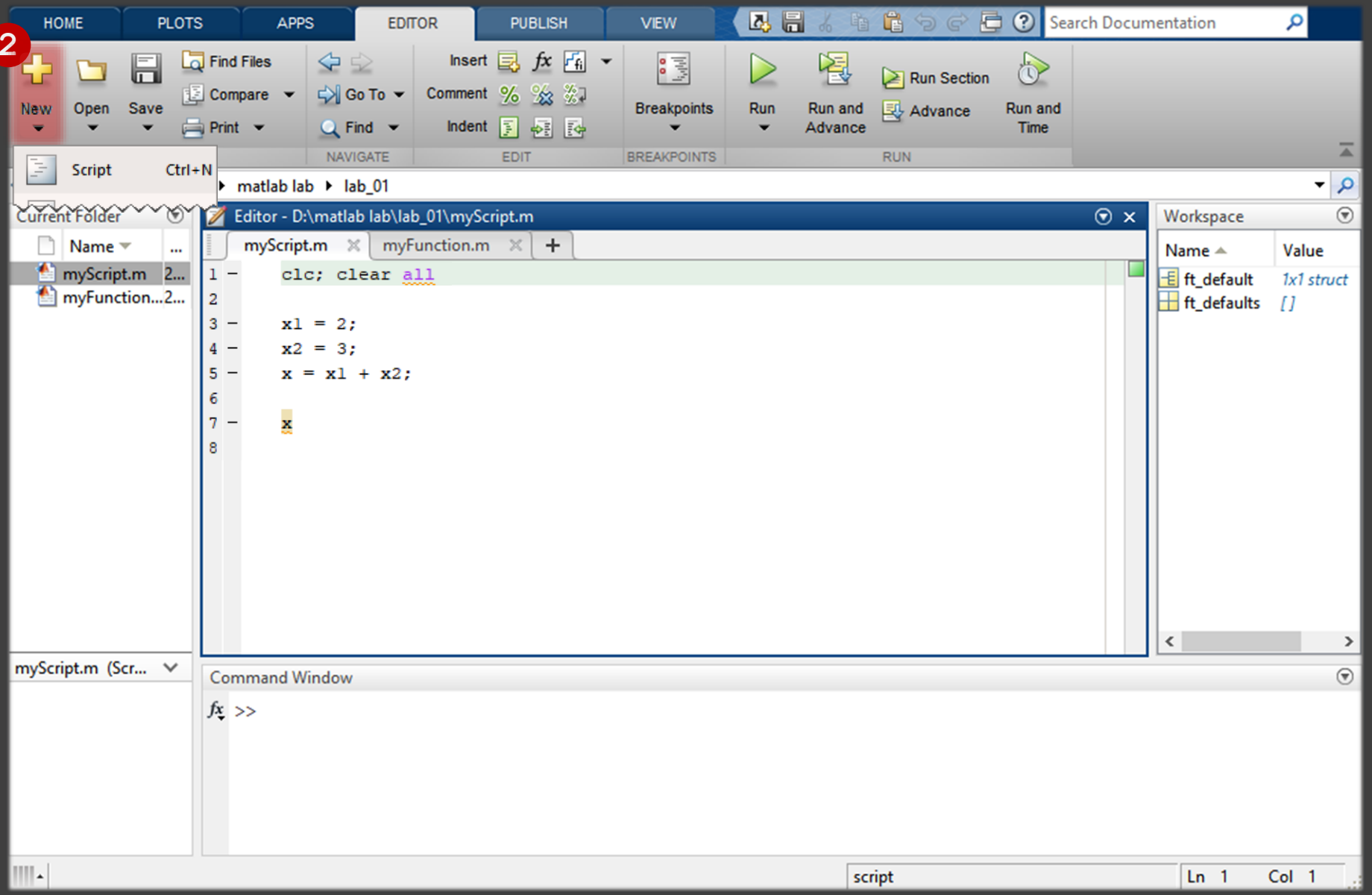
1. Ορίζουμε τον **φάκελο** στον οποίο θα δουλεύουμε
2. Δημιουργούμε ένα **νέο αρχείο** (New ▸ Script) Εμφανίζεται μια νέα καρτέλα στον Editor
3. Αποθηκεύουμε το αρχείο μας ως m-file (.m)

Scripts



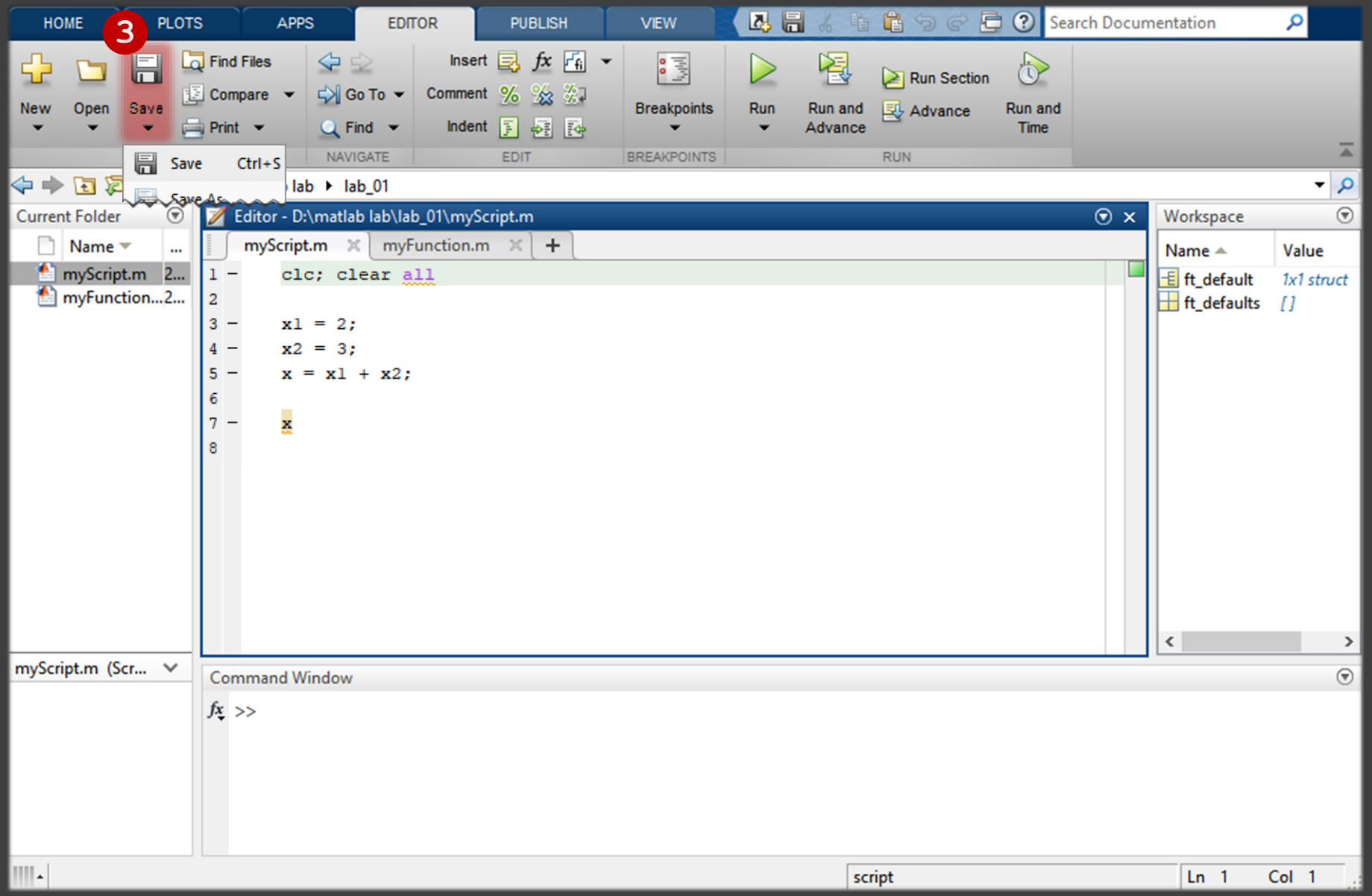
1. Ορίζουμε τον φάκελο στον οποίο θα δουλεύουμε
2. Δημιουργούμε ένα νέο αρχείο (New ▸ Script) Εμφανίζεται μια νέα καρτέλα στον Editor
3. Αποθηκεύουμε το αρχείο μας ως m-file (.m)

Scripts



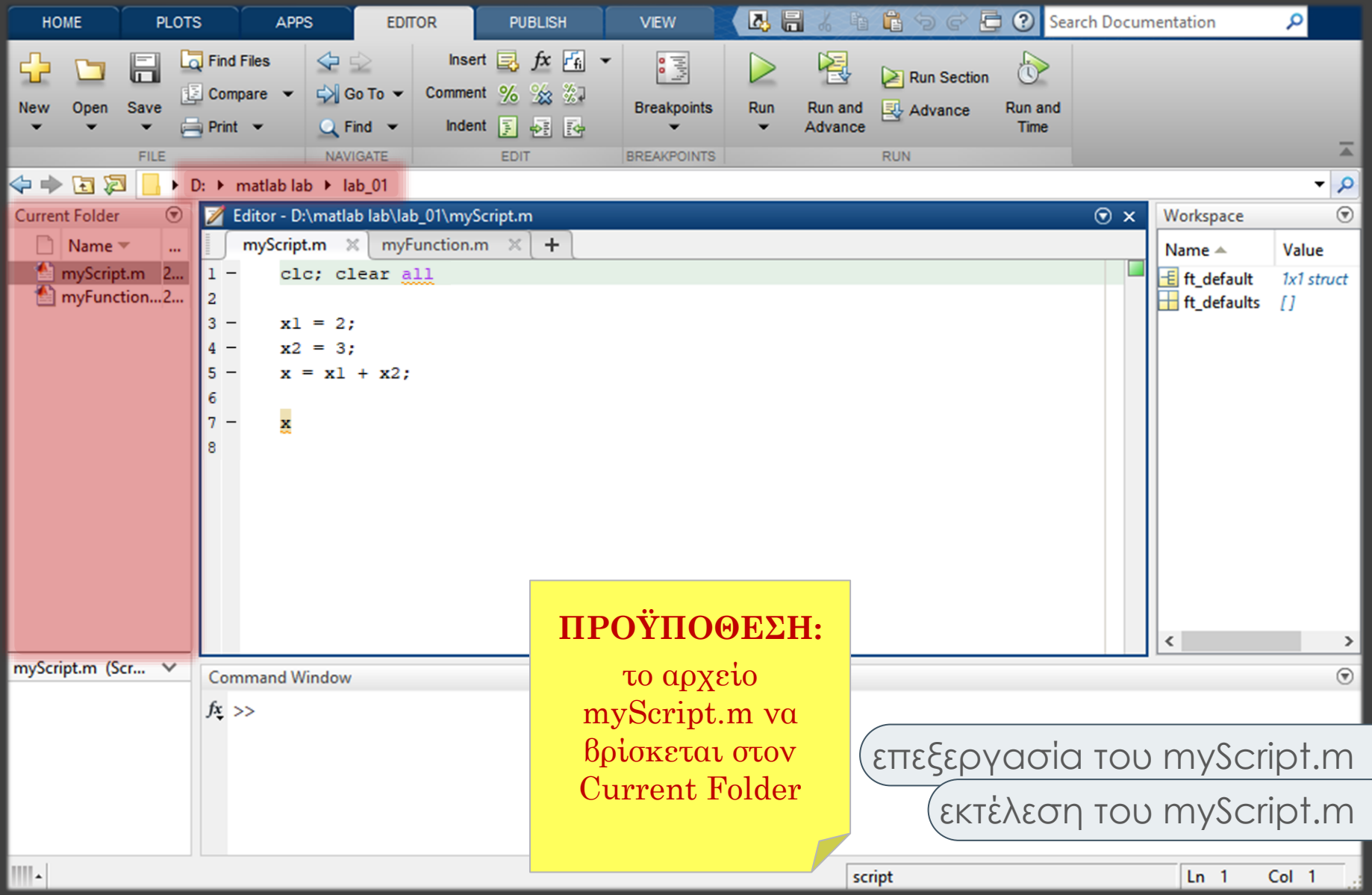
1. Ορίζουμε τον **φάκελο** στον οποίο θα δουλεύουμε
2. Δημιουργούμε ένα **νέο αρχείο** (New ▸ Script) Εμφανίζεται μια νέα καρτέλα στον Editor
3. **Αποθηκεύουμε** το αρχείο μας ως m-file (.m)

Scripts



1. Ορίζουμε τον **φάκελο** στον οποίο θα δουλεύουμε
2. Δημιουργούμε ένα **νέο αρχείο** (New ▸ Script) Εμφανίζεται μια νέα καρτέλα στον Editor
3. Αποθηκεύουμε το αρχείο μας ως m-file (.m)

Scripts



ΠΡΟΫΠΟΘΕΣΗ:
το αρχείο myScript.m να βρίσκεται στον Current Folder

επεξεργασία του myScript.m ←
εκτέλεση του myScript.m ←

- Αρχεία (με κατάληξη .m) που περιέχουν κώδικα, συνήθως δομημένο
- Ο κώδικας που περιέχουν εκτελείται με τη σειρά, από αριστερά προς τα δεξιά και από πάνω προς τα κάτω
- Command window:

```
>> edit myScript
>> myScript
```

Scripts

Το σύμβολο ‘ ; ’ (semicolon)

- Συνηθίζουμε να το βάζουμε στο τέλος κάθε γραμμής κώδικα ή στο τέλος κάθε εντολής → με αυτόν τον τρόπο δεν εμφανίζεται στο Command Window το αποτέλεσμα της εντολής που προηγείται
- Το χρησιμοποιούμε επίσης όταν δημιουργούμε διανύσματα ή πίνακες, για να δηλώσουμε μια νέα γραμμή (`b=[1 2;3 4;5 6]`)

Σχόλια ‘ % ’ (percent)

Μια καλή πρακτική είναι να βάζουμε σχόλια ώστε να περιγράψουμε το κομμάτι του κώδικα που ακολουθεί ή προηγείται

Χρήσιμο επίσης όταν θέλουμε να “απενεργοποιήσουμε” κομμάτι κώδικα

► Τα σχόλια δεν εκτελούνται κατά το “τρέξιμο” του script

```
% initialize
x1 = 20;    % distance from point 1 (in cm)
x2 = 82;    % distance from point 2 (in cm)

% add the two numbers
x = x1 + x2;

% display the result
x

% display message
fprintf('done!');
```

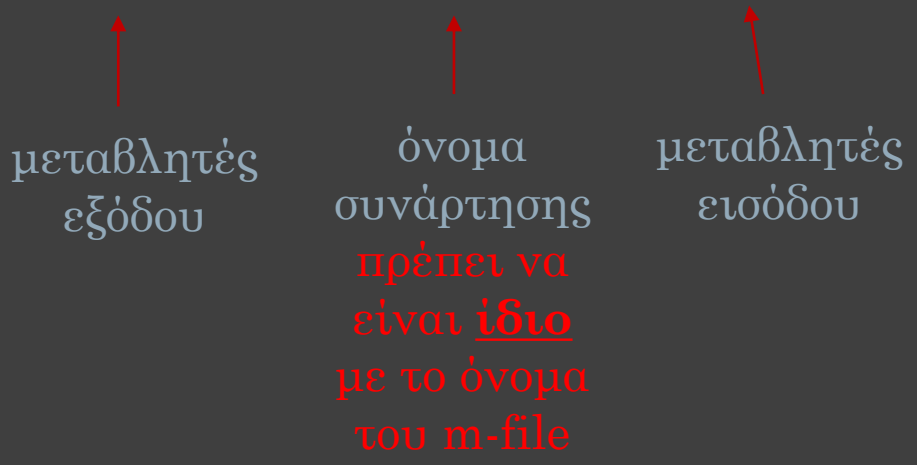
Συναρτήσεις

ΣΥΝΑΡΤΗΣΗ

Μια ολοκληρωμένη ακολουθία εντολών που χρησιμοποιεί τα δεδομένα εισόδου και παράγει ένα αποτέλεσμα, που εκχωρείται στα δεδομένα εξόδου.

- είναι m-file

Γενική μορφή: `[R,S] = myFunction(A,B,C)`



Συναρτήσεις

ΣΥΝΑΡΤΗΣΗ

Μια ολοκληρωμένη ακολουθία εντολών που χρησιμοποιεί τα δεδομένα εισόδου και παράγει ένα αποτέλεσμα, που εκχωρείται στα δεδομένα εξόδου.

- είναι m-file

πρέπει **<u>οπωσδήποτε</u>** η πρώτη εκτελέσιμη "λέξη" του αρχείου να είναι η `function`

Γενική μορφή: **`[R,S] = myFunction(A,B,C)`**

μεταβλητές εξόδου

όνομα συνάρτησης πρέπει να είναι **<u>ίδιο</u>** με το όνομα του m-file

μεταβλητές εισόδου

```
function [result] = mySum(a,b)
% Αυτή η συνάρτηση υπολογίζει το άθροισμα
% δυο αριθμών
%
% ΕΙΣΟΔΟΣ:
%   a,b      : [αριθμός] οι δυο αριθμοί που
%                        έλω να προσθέσω
%
% ΕΞΟΔΟΣ:
%   result   : [αριθμός] το αποτέλεσμα της πρόσθεσης
%                        των αριθμών που δίνει
%                        ο χρήστης ως είσοδο
%

result = a + b;

end
```

Ενσωματωμένες συναρτήσεις

Το MATLAB έχει πάρα πολλές έτοιμες συναρτήσεις που μπορούμε να χρησιμοποιήσουμε

Τις βρίσκουμε εδώ: mathworks.com/matlabcentral/fileexchange

Επίσης πληροφορίες για κάθε συνάρτηση (τι υλοποιεί, πώς συντάσσεται, τι δεδομένα και με τι μορφή πρέπει να μπουν ως είσοδος, τι δεδομένα και με ποια μορφή δίνει ως έξοδο, κ.α.) μπορούμε να βρούμε στην ιστοσελίδα mathworks.com/help/matlab

Για παράδειγμα μπορούμε να βρούμε πληροφορίες για τη συνάρτηση `sin` εδώ: mathworks.com/help/matlab/ref/sin.html

Εύρεση σε διανύσματα/πίνακες

`find()`

- Εμφανίζει **<u>τις θέσεις</u>** των στοιχείων του διανύσματος/πίνακα που ικανοποιούν μια συνθήκη, η οποία ορίζεται από τον χρήστη

```
>> a = [15 0 1;0 59 40;0 0 5]
ans =
     15   0   1
      0  59  40
      0   0   5
>> find(a)
ans =
     1
     3
     5
     6
     9
```

→ επιστρέφει τη θέση όλων των μη μηδενικών στοιχείων του a (ξεκινάει την αρίθμηση από το a(1,1) και προχωρώντας προς τα δεξιά και προς τα κάτω, αυξάνει κατά ένα)

Εύρεση σε διανύσματα/πίνακες

`find()` → www.mathworks.com/help/matlab/ref/find.html

Εύρεση σε διανύσματα/πίνακες

`find()` →

find

Find indices and values of nonzero elements

Syntax

```
k = find(X)
k = find(X,n)
k = find(X,n,direction)

[row,col] = find( ___ )
[row,col,v] = find( ___ )
```

Description

`k = find(X)` returns a vector containing the linear indices of each nonzero element in array `X`.

- If `X` is a vector, then `find` returns a vector with the same orientation as `X`.
- If `X` is a multidimensional array, then `find` returns a column vector of the linear indices of the result.
- If `X` contains no nonzero elements or is empty, then `find` returns an empty array.

`k = find(X,n)` returns the first `n` indices corresponding to the nonzero elements in `X`.

`k = find(X,n,direction)`, where `direction` is `'last'`, finds the last `n` indices corresponding to nonzero elements in `X`. The default for `direction` is `'first'`, which finds the first `n` indices corresponding to nonzero elements.

`[row,col] = find( ___ )` returns the row and column subscripts of each nonzero element in array `X` using any of the input arguments in previous syntaxes.

`[row,col,v] = find( ___ )` also returns vector `v`, which contains the nonzero elements of `X`.

Εύρεση σε διανύσματα/πίνακες

find() → www.mathworks.com/help/matlab/ref/find.html

```
>> a = [15 0 1;0 59 40;0 0 5];
>> [r,c] = find(a)
r =
     1
     2
     1
     2
     3
c =
     1
     2
     3
     3
     3
```

```
a =
    15     0     1
     0    59    40
     0     0     5
```

Εύρεση σε διανύσματα/πίνακες

`find()`

`find(~a)`: επιστρέφει τη θέση που έχουν όλα τα **μηδενικά** στοιχεία του `a`

`find(a>5)`: επιστρέφει τη θέση που έχουν όλα τα στοιχεία του `a` με τιμή **μεγαλύτερη από 5**

`find(a==5)`: επιστρέφει τη θέση που έχουν όλα τα στοιχεία του `a` με τιμή **ίση με 5**

- Αν δεν υπάρχει κανένα στοιχείο που να ικανοποιεί τη συνθήκη, τότε το MATLAB επιστρέφει το μήνυμα:

```
Empty matrix: 1-by-0
```

Εύρεση σε διανύσματα/πίνακες

`min()` / `max()`

- Βρίσκει την **ελάχιστη/μέγιστη** τιμή ενός διανύσματος/πίνακα
  Όπως και η `find`, έχει διάφορες επιπλέον δυνατότητες, π.χ. να βρει και σε ποια θέση μέσα στο διάνυσμα/πίνακα βρίσκεται αυτή η τιμή

```
>> a = [15 0 1 4 -2 8 35 16];
>> max(a)
ans =
    35

>> [maxValue,maxInd] = max(a)
maxValue =
    35
maxInd =
     7
```

Εύρεση σε διανύσματα/πίνακες

`min()` / `max()`

- Βρίσκει την **ελάχιστη/μέγιστη** τιμή ενός διανύσματος/πίνακα
  Όπως και η `find`, έχει διάφορες επιπλέον δυνατότητες, π.χ. να βρει και σε ποια θέση μέσα στο διάνυσμα/πίνακα βρίσκεται αυτή η τιμή

```
>> a = [15 0 1 4 -2 8 35 16];
>> min(a)
ans =
    -2

>> [minValue,minInd] = min(a)
minValue =
    -2
minInd =
     5
```

Διαχείριση πινάκων

`cat()`

- Ενσωματώνει πίνακες/διανύσματα μέσα σε άλλο πίνακα

Γενική μορφή: `A = cat(dim,A1,A2,...)`

σε ποια διάσταση να γίνει η ενσωμάτωση

πίνακες/διανύσματα προς ενσωμάτωση

ΠΡΟΣΟΧΗ στις διαστάσεις

```
>> a = [1 2 3 4;5 6 7 8]; b = [1 1 1 1;2 2 2 2;3 3 3 3];
>> cat(1,a,b)
ans =
     1     2     3     4
     5     6     7     8
     1     1     1     1
     2     2     2     2
     3     3     3     3
```

a → 2x4
b → 3x4
} ενσωμάτωση στη διάσταση 1

Διαχείριση πινάκων

`repmat()`

- Δημιουργεί πίνακα που αποτελείται από πολλαπλά αντίγραφα ενός αρχικού πίνακα

Γενική μορφή: `A = repmat(B,n,m)`

- `B`: πίνακας προς αντιγραφή
- `n`: # αντιγράφων στη διάσταση y
- `m`: # αντιγράφων στη διάσταση x

```
>> b = [1 2;3 4]
ans =
     1     2
     3     4
>> repmat(b,1,3)
ans =
     1     2     1     2     1     2
     3     4     3     4     3     4
```

Έλεγχος ροής

`if()`

- Ελέγχει αν ισχύει η συνθήκη που ορίζει ο χρήστης, και αν ισχύει, εκτελεί μια ομάδα εντολών. Αν δεν ισχύει, υπάρχει η δυνατότητα να εκτελέσει και μια άλλη ομάδα εντολών (χρησιμοποιώντας το `else`) ή να ελέγξει μια άλλη συνθήκη (`elseif`).
  Τερματίζει όταν συναντήσει τη λέξη `end` (απαραίτητη).

Περίπτωση 1:

```
if(condition)
  commands_A
end
```

Παράδειγμα:

```
% Generate a random number
a = randi(100,1);

% If it is greater than 10, divide by 2
if(a>10)
    disp('a is greater than 10')
    b = a/2;
end
```

Έλεγχος ροής

`if()`

- Ελέγχει αν ισχύει η συνθήκη που ορίζει ο χρήστης, και αν ισχύει, εκτελεί μια ομάδα εντολών. Αν δεν ισχύει, υπάρχει η δυνατότητα να εκτελέσει και μια άλλη ομάδα εντολών (χρησιμοποιώντας το `else`) ή να ελέγξει μια άλλη συνθήκη (`elseif`).
  Τερματίζει όταν συναντήσει τη λέξη `end` (απαραίτητη).

Περίπτωση 2:

```
if(condition)
  commands_A
else
  commands_B
end
```

Παράδειγμα:

```
% Generate a random number
a = randi(100,1);

% If it is greater than 10, divide by 2
% if not, add 2
if(a>10)
    disp('a is greater than 10')
    b = a/2;
else
    b = a + 2;
end
```

Έλεγχος ροής

`if()`

- Ελέγχει αν ισχύει η συνθήκη που ορίζει ο χρήστης, και αν ισχύει, εκτελεί μια ομάδα εντολών. Αν δεν ισχύει, υπάρχει η δυνατότητα να εκτελέσει και μια άλλη ομάδα εντολών (χρησιμοποιώντας το `else`) ή να ελέγξει μια άλλη συνθήκη (`elseif`).
  Τερματίζει όταν συναντήσει τη λέξη `end` (απαραίτητη).

Περίπτωση 3:

```
if(condition1)
  commands_A
elseif(condition2)
  commands_B
end
```

Παράδειγμα:

```
% Generate a random number
a = randi(100,1);

if(a<30)
   disp('low')
elseif(a>80)
   disp('high')
end
```

Έλεγχος ροής

`if()`

- Ελέγχει αν ισχύει η συνθήκη που ορίζει ο χρήστης, και αν ισχύει, εκτελεί μια ομάδα εντολών. Αν δεν ισχύει, υπάρχει η δυνατότητα να εκτελέσει και μια άλλη ομάδα εντολών (χρησιμοποιώντας το `else`) ή να ελέγξει μια άλλη συνθήκη (`elseif`).
  Τερματίζει όταν συναντήσει τη λέξη `end` (απαραίτητη).

Περίπτωση 4:

```
if(condition1)
  commands_A
elseif(condition2)
  commands_B
else
  commands_C
end
```

Παράδειγμα:

```
% Generate a random number
a = randi(100,1);

if(5<a<30)
   disp('low')
elseif(30<=a<=80)
   disp('medium')
else
   disp('high')
end
```

Έλεγχος ροής

for

- Εκτελεί μια ομάδα εντολών επαναλαμβανόμενα, **για όσες επαναλήψεις** έχει ορίσει ο χρήστης. Τερματίζει όταν συναντήσει τη λέξη `end` (απαραίτητη).

```
for i=x1:step:x2
  commands_A
end
```

Παράδειγμα1:

```
for a = 1:-0.2:0
   disp(a)
end
```

Παράδειγμα2:

```
for n = 1:6
    p(n) = 2 * n - 1;
end
```

Έλεγχος ροής

while()

- Εκτελεί μια ομάδα εντολών επαναλαμβανόμενα, **μέχρι να ικανοποιηθεί μια συνθήκη**.
  Απαραίτητη η λέξη `end` στο τέλος

- ο χρήστης δεν ορίζει αριθμό επαναλήψεων
- **ΠΡΟΣΟΧΗ**: υπάρχει ο κίνδυνος ατέρμονων επαναλήψεων (η συνθήκη δεν ικανοποιείται ποτέ)

```
while(condition)
 commands
end
```

Παράδειγμα1:

```
a=1;
while(a<100)
    a = a + 5;
end
```

Παράδειγμα2:

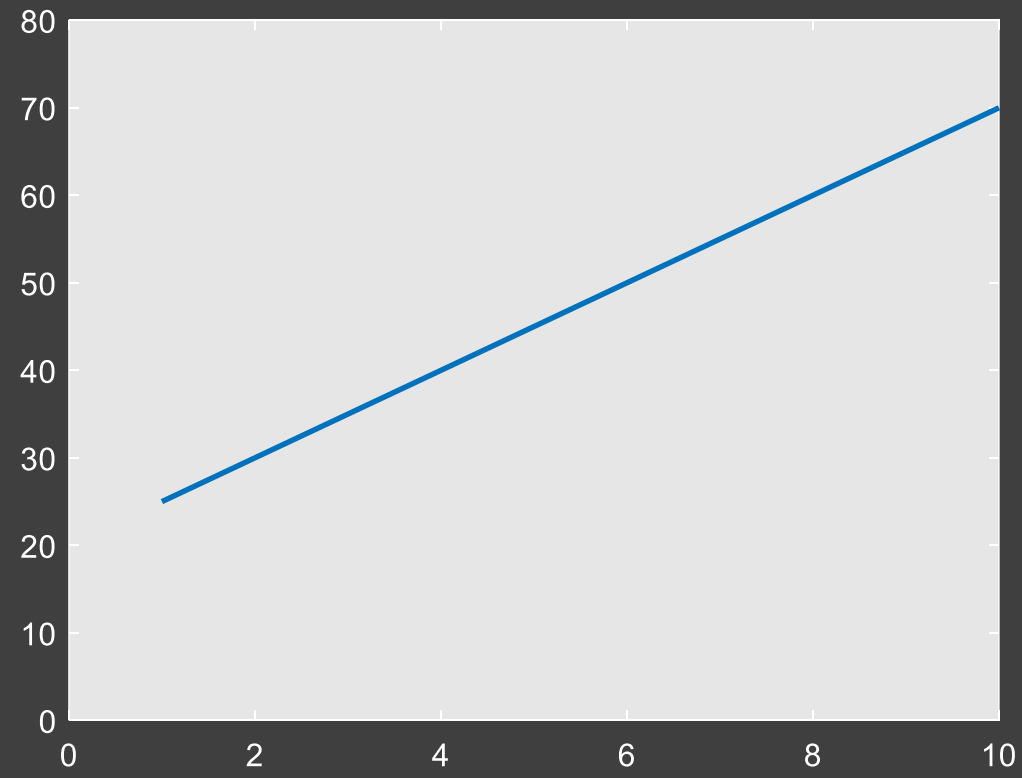
```
x=1;
while(x~=0)
    disp(x);
    x = x + 1;
end
```

Γραφήματα

ΒΑΣΙΚΟΙ ΤΥΠΟΙ ΓΡΑΦΗΜΑΤΩΝ

`plot(x,y)`: τυπώνει το γράφημα του διανύσματος y ως προς το διάνυσμα x

```
>> x = [1:10]; y = 5*x+20;
>> plot(x,y)
```

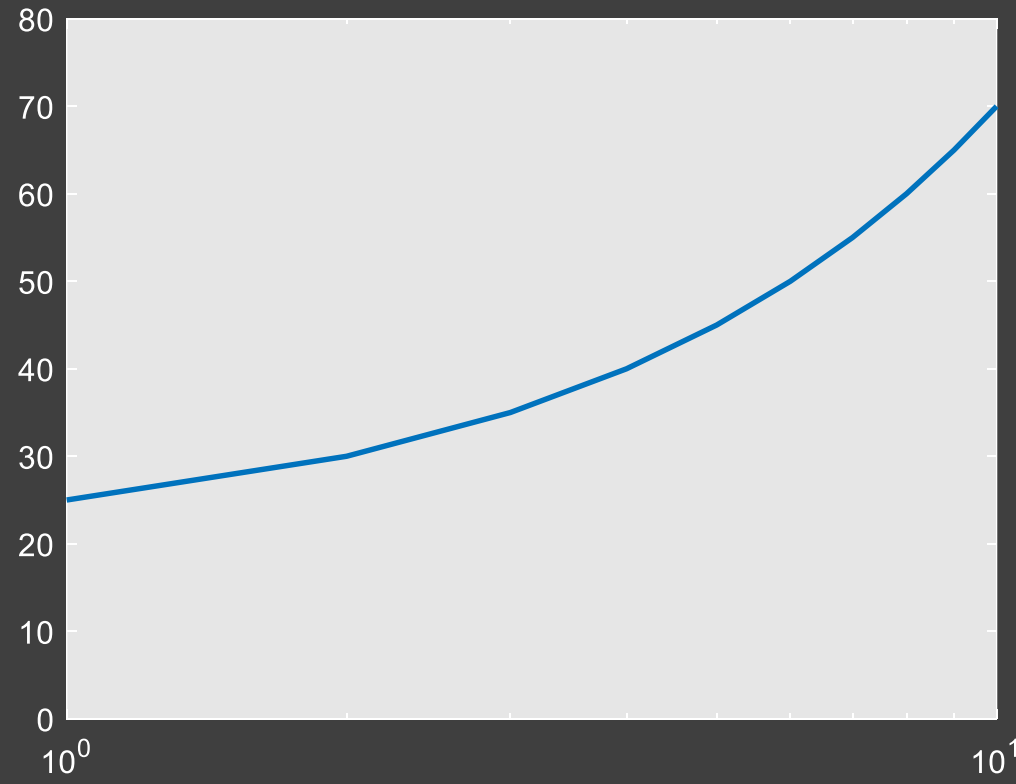


Γραφήματα

ΒΑΣΙΚΟΙ ΤΥΠΟΙ ΓΡΑΦΗΜΑΤΩΝ

`semilogx(x,y)`: τυπώνει το γράφημα του διανύσματος y ως προς το διάνυσμα x ο άξονας x είναι λογαριθμικός, ο άξονας y είναι γραμμικός

```
>> x = [1:10]; y = 5*x+20;
>> semilogx(x,y)
```

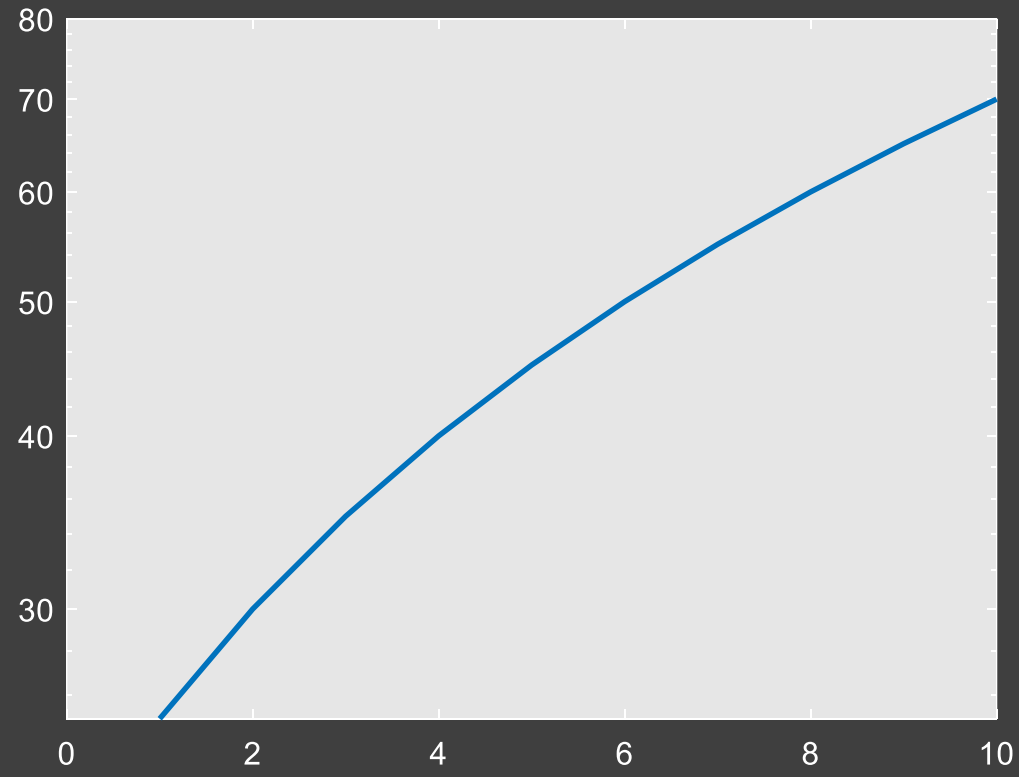


Γραφήματα

ΒΑΣΙΚΟΙ ΤΥΠΟΙ ΓΡΑΦΗΜΑΤΩΝ

`semilogy(x,y)`: τυπώνει το γράφημα του διανύσματος y ως προς το διάνυσμα x ο άξονας x είναι γραμμικός, ο άξονας y είναι λογαριθμικός

```
>> x = [1:10]; y = 5*x+20;
>> semilogy(x,y)
```

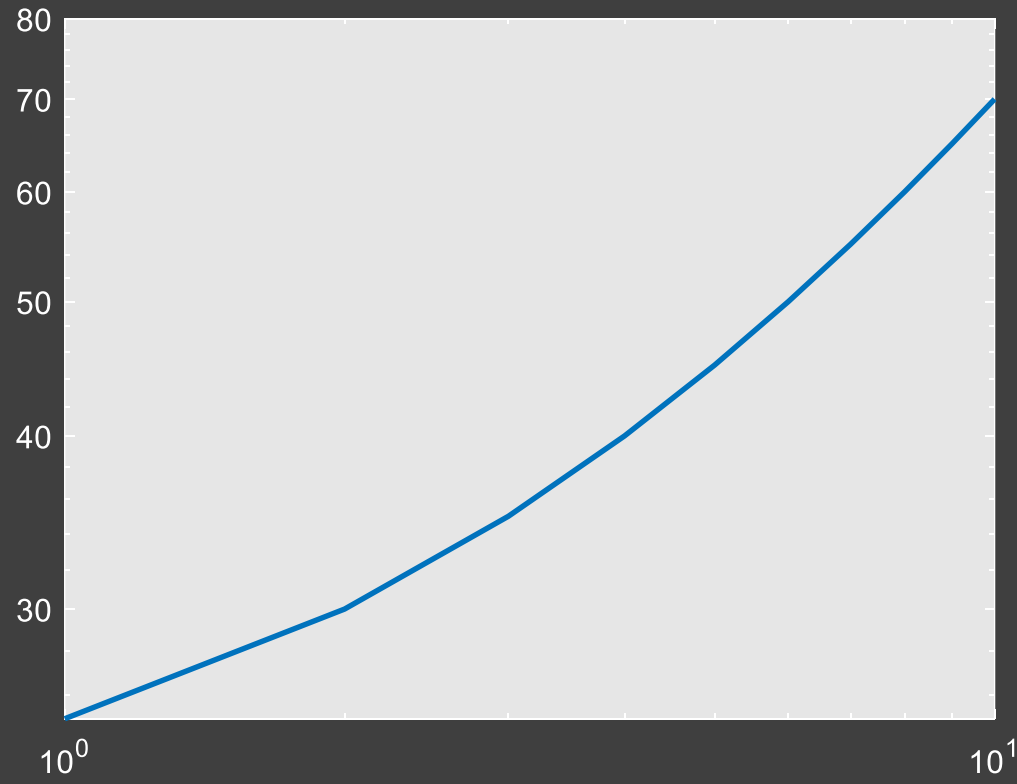


Γραφήματα

ΒΑΣΙΚΟΙ ΤΥΠΟΙ ΓΡΑΦΗΜΑΤΩΝ

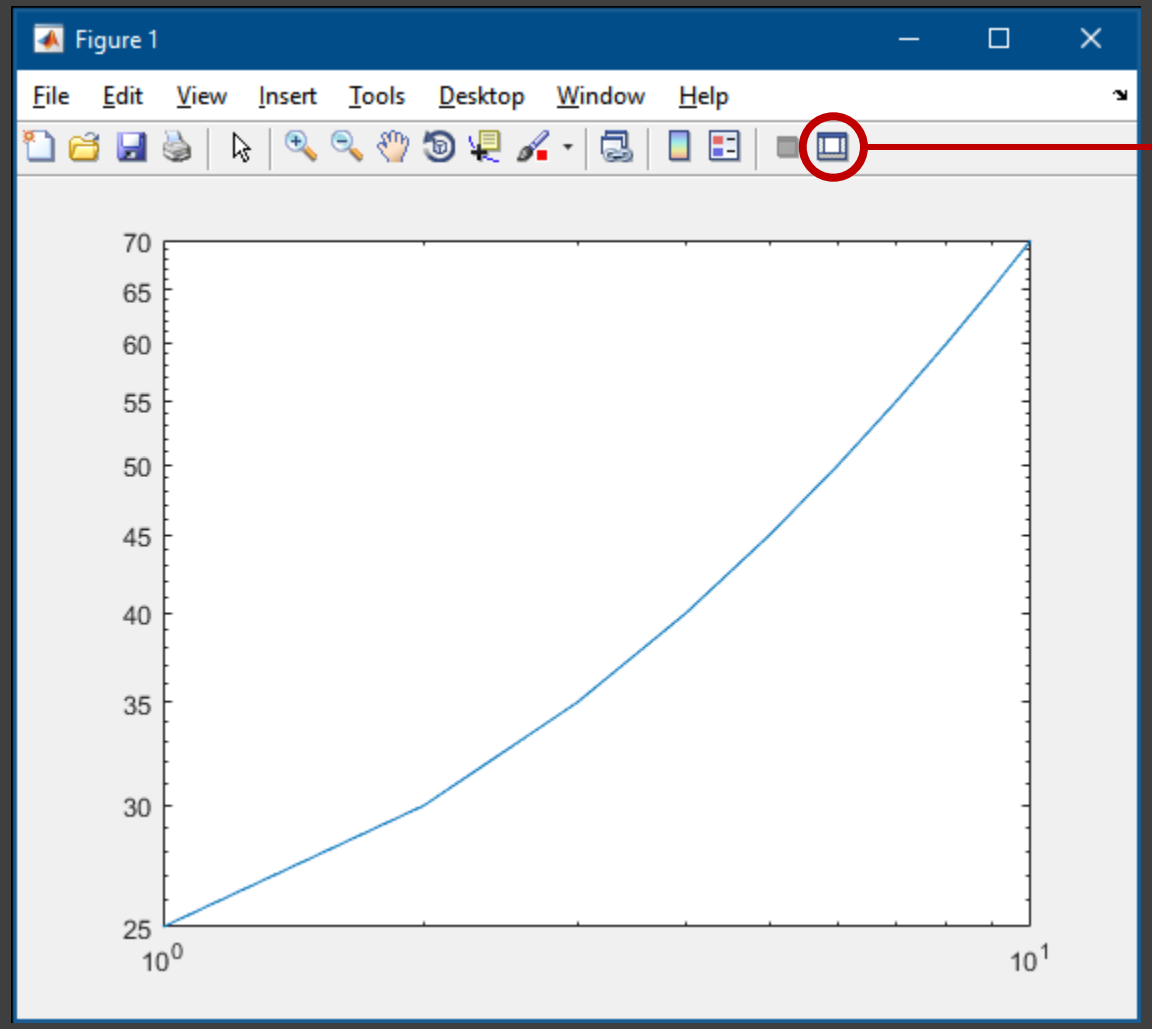
loglog(x,y): τυπώνει το γράφημα του διανύσματος y ως προς το διάνυσμα x και οι δυο άξονες είναι λογαριθμικοί

```
>> x = [1:10]; y = 5*x+20;
>> loglog(x,y)
```



Γραφήματα

ΜΟΡΦΟΠΟΙΗΣΗ ΓΡΑΦΗΜΑΤΩΝ



εμφανίζει όλα τα εργαλεία για την επεξεργασία γραφήματος

Εξάσκηση

Μπορείτε να βρείτε ασκήσεις για εξοικείωση/εξάσκηση στο MATLAB

στη σελίδα του μαθήματος στο eclass:

Έγγραφα → Εργαστήριο → Εισαγωγή στο MATLAB - Ασκήσεις εξάσκησης