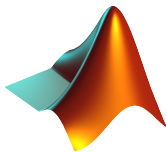


## Εισαγωγή στο MATLAB



Β. Γεωργίου, Λ. Κολώνιας, Ε. Γαλλόπουλος

Τμήμα Μηχανικών Η/Υ & Πληροφορικής  
Πανεπιστήμιο Πατρών

Επιστημονικός Υπολογισμός Ι  
6/10/2017

- 1 Εισαγωγικά
- 2 Συστατικά Μέρη του MATLAB
- 3 Κατασκευές, Απλές Εντολές & Πράξεις
- 4 Αριθμητική Κινητής Υποδιαστολής
- 5 Sparse Matrix Technology
- 6 Μέτρηση Χρόνου Εκτέλεσης
- 7 Οπτικοποίηση Δεδομένων
- 8 Πηγές

## Τι είναι το MATLAB???

- 🚀 MATrix LABoratory
- 🚀 Περιβάλλον Επίλυσης Προβλημάτων
- 🚀 Περιβάλλον Αριθμητικών Υπολογισμών
- 🚀 Βιβλιοθήκες συναρτήσεων (Toolboxes)
- 🚀 Απλή Γλώσσα Προγραμματισμού (4ης γενιάς)
- 🚀 Ανεξάρτητο Πλατφόρμας

## Τι είναι το MATLAB???

- ▶ MATrix LABoratory
- ▶ Περιβάλλον Επίλυσης Προβλημάτων
- ▶ Περιβάλλον Αριθμητικών Υπολογισμών
- ▶ Βιβλιοθήκες συναρτήσεων (Toolboxes)
- ▶ Απλή Γλώσσα Προγραμματισμού (4ης γενιάς)
- ▶ Ανεξάρτητο Πλατφόρμας

### Γιατί MATLAB?

- ▶ Επιτρέπει την εύκολη διαχείριση μητρώων
- ▶ Εύκολη γραφική αναπαράσταση συναρτήσεων και δεδομένων
- ▶ Εύκολη υλοποίηση αλγορίθμων
- ▶ Εύκολη δημιουργία γραφικών διεπαφών
- ▶ Εύκολη διασύνδεση με άλλες γλώσσες προγραμματισμού χαμηλότερου επιπέδου όπως C, C++, Fortran, Java

## Εφαρμογές του MATLAB

Το MATLAB είναι ιδιαίτερα χρήσιμο σε διάφορες εφαρμογές των οποίων τα δεδομένα μπορούν να αναπαρασταθούν με τη μορφή μητρώων.

### Μερικά μαθήματα . . .

- ✚ Αριθμητική Ανάλυση και Περιβάλλοντα Υλοποίησης
- ✚ Εισαγωγή στη Θεωρία Σημάτων και Συστημάτων
- ✚ Επιστημονικός Υπολογισμός I
- ✚ Ψηφιακή Επεξεργασία Σημάτων
- ✚ Ψηφιακές Τηλεπικοινωνίες

### . . . και κάποια πιο προχωρημένα

- ✚ Επιστημονικός Υπολογισμός II
- ✚ Αριθμητική Επίλυση Διαφορικών Εξισώσεων
- ✚ Υπολογιστική Νοημοσύνη
- ✚ Στοχαστικά Σήματα και Εφαρμογές
- ✚ Επεξεργασία και Ανάλυση Εικόνας
- ✚ Προχωρημένα Θέματα Τηλεπικοινωνιών
- ✚ Ανάκτηση Πληροφορίας

## Μερικά Ιστορικά Στοιχεία

- 1970** άρχισε να αναπτύσσεται το MATLAB από τον Cleve Moler έτσι ώστε οι φοιτητές του στο Τμήμα Επιστήμης των Υπολογισμών του Πανεπιστημίου του Νέου Μεξικού να έχουν πρόσβαση σε βιβλιοθήκες λογισμικού αριθμητικής γραμμικής άλγεβρας όπως η LINPACK και η EISPACK χωρίς να χρειάζεται να μάθουν FORTRAN
- 1983** ήταν ήδη γνωστό σε αρκετά Πανεπιστήμια στην Αμερική
- 1984** ξαναγράφηκε σε C και άρχισε να αναπτύσσεται με ραγδαίους ρυθμούς. Την ίδια περίοδο ιδρύθηκε και η MathWorks η εταιρία που διανέμει και αναπτύσσει το εργαλείο
- 2000** ξαναγράφηκε εξ ολοκλήρου για να χρησιμοποιεί τη βιβλιοθήκη LAPACK
- 2017** βρίσκεται στην έκδοση 9.3 ή R2017b και είναι γραμμένο σε C και Java

# Πώς μπορώ να βρω το MATLAB?

Δε διατίθεται δωρεάν!!!! ☹️☹️

Που μπορώ να το βρω;

Το πανεπιστήμιο διαθέτει άδειες που χρησιμοποιεί το υπολογιστικό κέντρο 😊😊😊

Εναλλακτικά Εργαλεία



OpenFOAM

The Open Source CFD Toolbox



Linear Algebra Libraries: BLAS,  
LAPACK, ScaLAPACK, PLASMA,  
MAGMA

Shirley Hoang  
shoang@stanford.edu  
©2010-2011, 2013  
shoang@mathworks.com  
November 12, 2012

Που μπορώ να το βρω ;

Το Πανεπιστήμιο Πατρών διαθέτει το λογισμικό της MATLAB δωρεάν για τα μέλη του

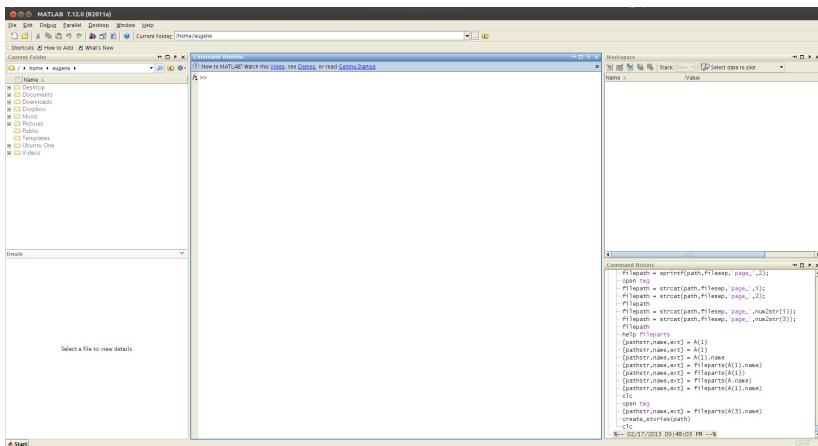
### Οδηγίες

- 1 Κατεβάζετε το λογισμικό της MATLAB από το Software Repository του ΠΠ (<http://www.upnet.gr/software/>)
- 2 Κατεβάζετε την άδεια χρήσης από το MUSSA (<https://mussa.upnet.gr/user/> )
- 3 Περισσότερες οδηγίες εγκατάστασης στο <http://www.upnet.gr/software/matlab/>
- 4 Απαιτεί IP από το ΠΠ για να λειτουργεί

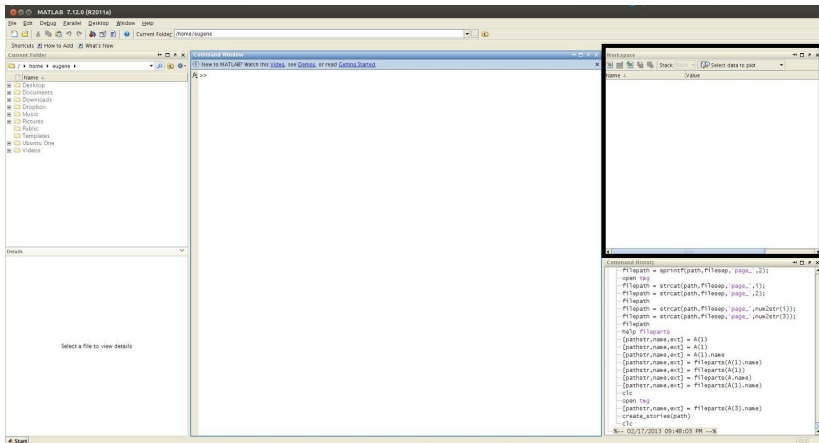


- 1 Εισαγωγικά
- 2 Συστατικά Μέρη του MATLAB
- 3 Κατασκευές, Απλές Εντολές & Πράξεις
- 4 Αριθμητική Κινητής Υποδιαστολής
- 5 Sparse Matrix Technology
- 6 Μέτρηση Χρόνου Εκτέλεσης
- 7 Οπτικοποίηση Δεδομένων
- 8 Πηγές

# Γραφικό Περιβάλλον

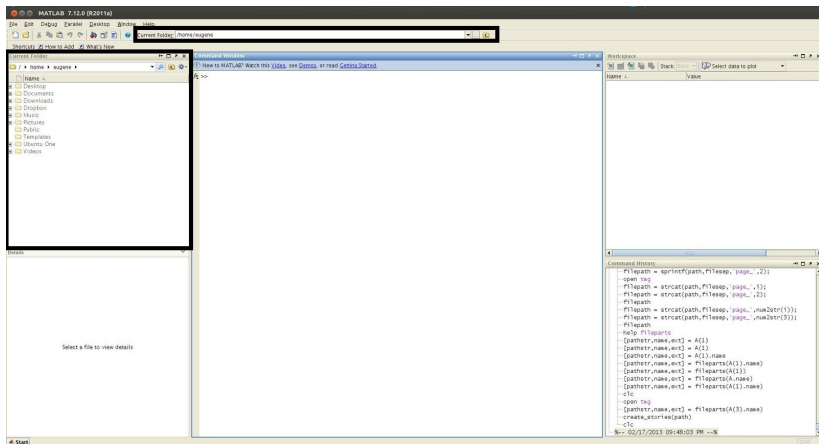


# Workspace I

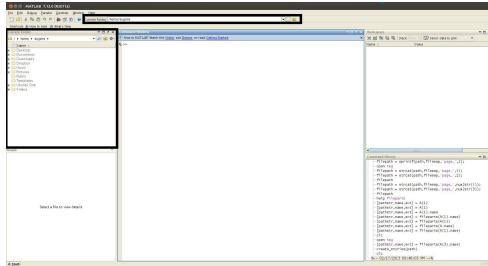




## Current Folder I

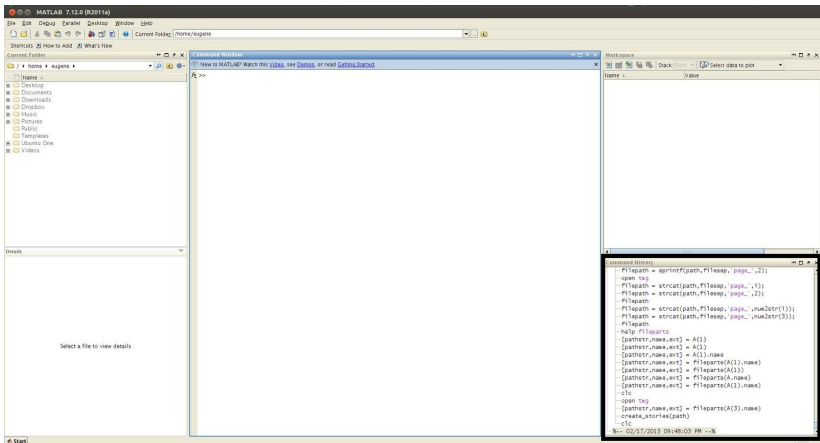


## Current Folder II

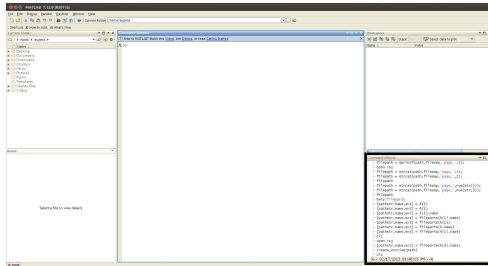


- ✦ Το Current Folder υποδηλώνει το directory από το οποίο το MATLAB θα μπορεί να εκτελεί συναρτήσεις και scripts.
- ✦ Συναρτήσεις και scripts εκτός του Current Folder μπορούν να εκτελεστούν μόνο αν βρίσκονται στο **matlabpath**.

# Command History



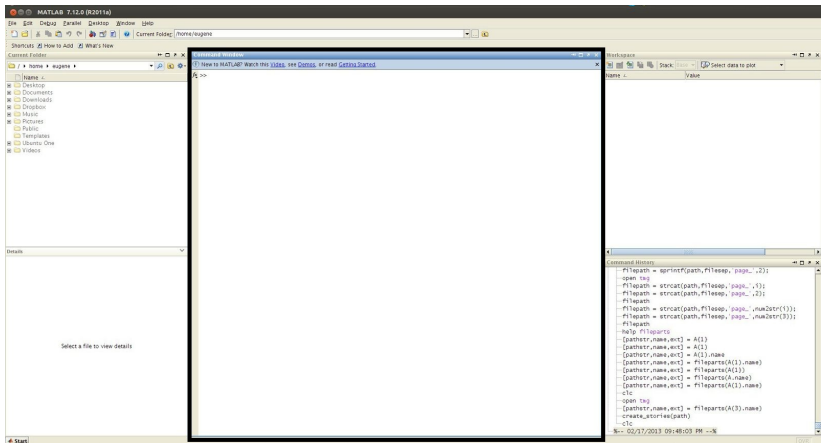
## Command History II



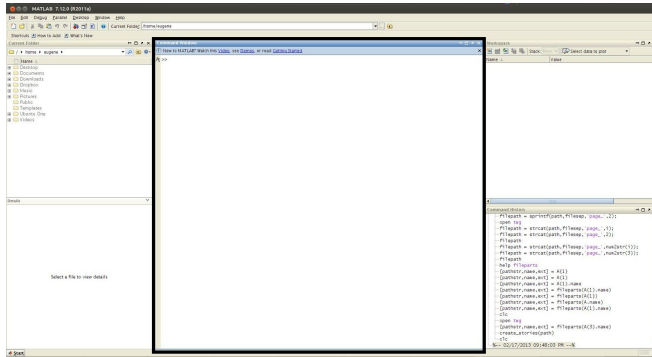
Στο Command History καταγράφονται όλες οι εντολές τις οποίες έχουμε εκτελέσει και ομαδοποιούνται με βάση την χρονική στιγμή που ξεκίνησε τη λειτουργία του το εργαλείο.



# Command Window I

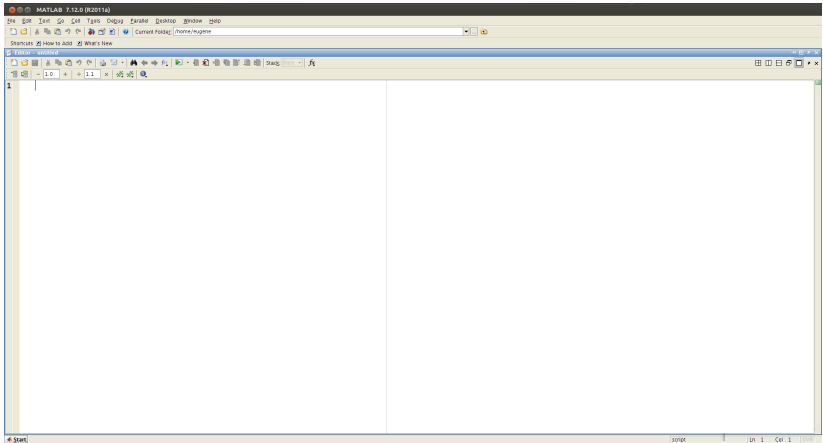


## Command Window II





- Στο Command Window αναγράφουμε όλες τις εντολές, συναρτήσεις ή πράξεις που θα εκτελέσουμε.
- Αν στο τέλος μίας εντολής βάλουμε το σύμβολο `;` τότε το αποτέλεσμα της δε θα παρουσιαστεί στο Command Window, στην αντίθετη περίπτωση το αποτέλεσμα εμφανίζεται ακριβώς κάτω από την εντολή.

# Editor I



Ο Editor του MATLAB μας βοηθάει να γράφουμε καλύτερα μπλοκ κώδικα:

-  Άμεση αποσφαλμάτωση του κώδικα
-  Προτάσεις για βελτιστοποίηση του κώδικα

Στο MATLAB τα μπλοκ κώδικα χωρίζονται σε δύο κατηγορίες :

### Functions (Συναρτήσεις)

Δουλεύουν όπως ακριβώς και οι υπάρχουσες συναρτήσεις του MATLAB με τη διαφορά ότι τις έχουμε δημιουργήσει εμείς. Ορίζονται ως :

```
function [varargout] = name(varargin)
```





Δέχονται σαν είσοδο ένα σύνολο μεταβλητών (varargin) και δίνουν σαν έξοδο ένα δεύτερο σύνολο μεταβλητών (varargout). Οι μεταβλητές των δύο συνόλων πρέπει να εμφανίζονται στο σώμα της συνάρτησης ως μέρος εντολών. Οι μόνες μεταβλητές του Workspace που βλέπουν είναι αυτές του συνόλου (varargin) ενώ αποθηκεύουν σε αυτό μόνο αυτές του συνόλου (varargout).

### Scripts (Σενάρια)





Τα scripts είναι διαδοχικές εντολές MATLAB οι οποίες συνήθως εκτελούν μία πράξη η οποία δεν μπορεί να γίνει με μία μόνο εντολή. Τα script δεν έχουν μεταβλητές εισόδου, δηλώνονται όλα σαν εντολές στο σώμα τους, ενώ μπορούν και βλέπουν όλες τις μεταβλητές του workspace. Τα αποτελέσματα όλων των πράξεων του script αποθηκεύονται επίσης στο workspace.

Τα αρχεία των συναρτήσεων και των scripts πρέπει να έχουν την κατάληξη “.m” για να είναι εκτελέσιμα από το MATLAB

### Έναρξη



-  διπλό κλικ στο εικονίδιο του MATLAB
-  εντολή `matlab` στο command prompt ή στο terminal
-  το εκτελέσιμο του MATLAB να είναι γνωστό στο path του συστήματος
-  να υπάρχει συμβολικός δείκτης ως προς το εκτελέσιμο του MATLAB

### Τερματισμός

-  Στο Command Window γράφουμε:
  -  `exit`
  -  `quit`
-  εικονίδιο τερματισμού




### Μεταβλητές

Στο Command Window γράφουμε:

-  `save όνομα_αρχείου`
-  `load όνομα_αρχείου`



### Μήκος Εξόδου

Στο Command Window γράφουμε:

-  `format long`
-  `format short`
-  `help format` για περισσότερα  
`format`




### Άνοιγμα Αρχείων & Συναρτήσεων

Στο Command Window γράφουμε:

-  `open` όνομα\_συνάρτησης
-  `edit` όνομα\_συνάρτησης



### Καθαρισμός

Στο Command Window γράφουμε:

-  `clear all, clear` (καθαρισμός ολόκληρου workspace)
-  `clear` όνομα\_μεταβλητής (διαγραφή συγκεκριμένης μεταβλητής)
-  `clc` (καθαρισμός οθόνης)

### Επισκόπηση Μεταβλητών

Στο Command Window γράφουμε:

-  `who` (επισκόπηση μεταβλητών)
-  `whos` όνομα\_μεταβλητής  
(πληροφορίες  
μεταβλητών-μεταβλητής)

# Περιεχόμενα

- 1 Εισαγωγικά
- 2 Συστατικά Μέρη του MATLAB
- 3 Κατασκευές, Απλές Εντολές & Πράξεις
- 4 Αριθμητική Κινητής Υποδιαστολής
- 5 Sparse Matrix Technology
- 6 Μέτρηση Χρόνου Εκτέλεσης
- 7 Οπτικοποίηση Δεδομένων
- 8 Πηγές



Οι τελεστές που χρησιμοποιούμε:

Χαρακτήρας	Ιδιότητα	Χαρακτήρας	Ιδιότητα
+	Πρόσθεση	;	Αλλαγή Γραμμής & Τέλος Εντολής
—	Αφαίρεση	%	Σκόλια
*	Πολλαπλασιασμός	%%	Μηλοκ Σχολίων ή Κώδικα
. *	Πολλαπλασιασμός Στοιχείο προς Στοιχείο	'	Αναστροφή
\	Αριστερή Διαίρεση	=	Εκχώρηση Τιμής σε Μεταβλητή
/	Δεξιά Διαίρεση	==	Ισότητα
. \	Αριστερή Διαίρεση Στοιχείο προς Στοιχείο	~=	Διάφορο Ίσο
. /	Δεξιά Διαίρεση Στοιχείο προς Στοιχείο		Λογικό Ή
^	Ύψωση σε Δύναμη	&&	Λογικό ΚΑΙ
. ^	Ύψωση σε Δύναμη Στοιχείο προς Στοιχείο	<=	Μικρότερο Ίσο
.	Κινητή Υποδιαστολή	>=	Μεγαλύτερο Ίσο
,	Διαχωρισμός Στοιχείων		

## Υλοποιημένες Συναρτήσεις


Στο MATLAB υπάρχουν υλοποιημένες συναρτήσεις οι οποίες εκτελούν διάφορες υπολογιστικές πράξεις οι οποίες διακρίνονται σε:


**Ενδογενείς:** συναρτήσεις στις οποίες δεν έχουμε πρόσβαση στον κώδικα τους

**Εξωγενείς:** συναρτήσεις στις οποίες έχουμε πρόσβαση στον κώδικά τους

### Μερικές Συναρτήσεις στο MATLAB:

Ονομασία	Συνάρτηση
<code>sin, cos, tan</code>	Ημίτονο, Συνημίτονο, Εφαπτομένη
<code>log, log2, log10, exp</code>	Λογάριθμος ( $n$ , 2, 10), Εκθετικό
<code>sqrt, abs</code>	Τετραγωνική Ρίζα, Απόλυτη Τιμή
<code>max, min</code>	Μέγιστο, Ελάχιστο
<code>sum, prod</code>	Άθροισμα, Γινόμενο
<code>inv, norm</code>	Αντίστροφος, Νόρμα
<code>lu, svd, qr</code>	Διασπάσεις LU, SVD, QR
<code>eig</code>	Ιδιοτιμές & Ιδιοδιανύσματα
<code>expm, logm, sqrtm</code>	Εκθετικό, Λογάριθμος, Ρίζα Μητρώου

 `type` όνομα\_συνάρτησης:  
ενδογενής ή εξωγενής  
συνάρτηση

 `help` όνομα\_συνάρτησης:  
σύνταξη συνάρτησης

```
>> type norm
'norm' is a built-in function.
>> help norm
NORM Matrix or vector norm.
NORM(X,2) returns the 2-norm of X.
NORM(X) is the same as NORM(X,2).
NORM(X,1) returns the 1-norm of X.
NORM(X,Inf) returns the infinity norm of X.
NORM(X,'fro') returns the Frobenius norm of X.
In addition, for vectors...
NORM(V,p) returns the p-norm of V defined as SUM(ABS(V).^p)^(1/p).
NORM(X,Inf) returns the largest element of ABS(X).
NORM(X,-Inf) returns the smallest element of ABS(X).
By convention, NaN is returned if X or V contains NaNs.
See also cond, rand, condset, normest, hypot.
Overloaded methods:
dplys_norm
```

## Δημιουργία Διανυσμάτων

Έστω τα διανύσματα:

Διάνυσμα Στήλη:

$$a = \begin{bmatrix} 5 \\ 6 \\ 7 \end{bmatrix}$$

Σε MATLAB θα γράφαμε:

```
a = [5 ; 6 ; 7];
```

Διάνυσμα Γραμμή:

$$b = [5 \quad 6 \quad 7]$$

```
b = [5 6 7];
```

```
% or
```

```
b = [5,6,7];
```

Διανύσματα τα οποία έχουν τιμές οι οποίες σχετίζονται με κάποιο βήμα:

$b = [5 \quad 6 \quad 7]$  στοιχεία με βήμα 1

```
b = 5:7;
```

$c = [1 \quad 3 \quad 5]$  στοιχεία με βήμα 2

```
c = 1:2:5;
```

🚩 Για να διαχωρίσουμε τις στήλες χρησιμοποιούμε το **κενό** ή το **“,”**

🚩 Για να διαχωρίσουμε τις γραμμές χρησιμοποιούμε το **“;”**

## Δημιουργία Μητρώων

Τα μητρώα αποτελούνται από στήλες και γραμμές άρα




Σε MATLAB θα γράφαμε:

Έστω το μητρώο:

$$A = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

```
A = [1 4 7 ; 2 5 8 ; 3 6 9 ];  
% or  
A = [1, 4, 7 ; 2, 5, 8 ; 3, 6, 9];  
% or  
A = [1 4 7  
     2 5 8  
     3 6 9];
```

Υπάρχουν έτοιμες συναρτήσεις οι οποίες κατασκευάζουν ειδικά μητρώα όπως :

-  `zeros` κατασκευάζει μητρώα με όλο μηδενικά
-  `ones` κατασκευάζει μητρώα τα οποία είναι όλο μονάδες
-  `eye` κατασκευάζει μητρώα τα οποία έχουν στην κύρια διαγώνιο μονάδες και σε όλες τις άλλες θέσεις μηδενικά (ταυτοτικό)

και άλλα όπως `vander`, `toeplitz`, `hankel`...

## Επιλογές Στοιχείων

Το MATLAB δίνει τη δυνατότητα επιλογής συγκεκριμένων στοιχείων από τα μητρώα και τα διανύσματα.

Έστω το μητρώο:

$$A = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

```
% Select the first element  
A(1,1);  
% Select the first column  
A(:,1);  
% Select the first row  
A(1,:);  
% Select first and third row  
A([1 3],:);  
% Select first and second columns  
% and first and second row  
A(1:2,1:2);
```

Παρατηρούμε ότι:

- ένα μητρώο ορίζεται από δύο οντότητες οι οποίες αντιπροσωπεύουν τις γραμμές και τις στήλες του αντίστοιχα **A(rows,cols)**
- ο τελεστής **':'** μόνος του σημαίνει ότι λαμβάνουμε υπόψιν μας όλα τα στοιχεία της διάστασης
- για να διαλέξουμε διαστάσεις οι οποίες δεν είναι συνεχόμενες τις γράφουμε σε μορφή διανύσματος γραμμή στην αντίστοιχη οντότητα π.χ. [2 5]
- για να διαλέξουμε συνεχόμενες διαστάσεις γράφουμε τη μικρότερη και τη μεγαλύτερη διάσταση διαχωρισμένες με τον τελεστή **':'** π.χ. 3:6

Όλα στο MATLAB είναι  $n \times m$  μητρώα !!!!!

🚀 **Βαθμωτοί:**  $1 \times 1$  μητρώο

🚀 **Διάνυσμα στήλη:**  $n \times 1$  μητρώο

🚀 **Διάνυσμα γραμμή:**  $1 \times n$  μητρώο

Συνεπώς πρέπει να είμαστε προσεκτικοί να εκτελούμε πράξεις οι οποίες γίνονται !!!!!

### Πρόσθεση/Αφαίρεση

Κατά την πρόσθεση/αφαίρεση τα μητρώα πρέπει να είναι ιδίων διαστάσεων εκτός από τους βαθμωτούς οι οποίοι προσθέτονται/αφαιρούνται κανονικά με μητρώα και διανύσματα.

### Πολλαπλασιασμός/Διαίρεση

Κατά τον πολλαπλασιασμό/διαίρεση τα μητρώα πρέπει να συμφωνούν στις εσωτερικές διαστάσεις δηλαδή οι στήλες του πρώτου παράγοντα με τις γραμμές του δεύτερου. Εξαίρεση πάλι αποτελούν οι βαθμωτοί.

### Παραδείγματα:

$$A = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}, B = \begin{bmatrix} 11 & 14 & 17 \\ 12 & 15 & 18 \\ 13 & 16 & 19 \end{bmatrix}, a = \begin{bmatrix} 5 \\ 6 \\ 7 \end{bmatrix}, b = [5 \quad 6 \quad 7], \text{alpha} = 6$$

## Πρόσθεση:

```

Command Window
File Edit Debug Desktop Window Help
New to MATLAB? Watch this Video, see Details, or read Getting Started.

>> A+B

ans =

    12    18    24
    14    20    26
    16    22    28

>> A+a
??? Error using ==> plus
Matrix dimensions must agree.

>> a+b
??? Error using ==> plus
Matrix dimensions must agree.

>> a*alpha

ans =

    11    12    13

>> b*alpha

ans =

    11
    12
    13

>> A*alpha

ans =

     7     10     13
     8     11     14
     9     12     15

```

## Πολλαπλασιασμός:

```

Command Window
File Edit Debug Desktop Window Help
New to MATLAB? Watch this Video, see Details, or read Getting Started.

>> a*b

ans =

    25    30    35
    30    36    42
    35    42    49

>> b*a

ans =

    110

>> alpha*A

ans =

     6     24     42
    12     30     48
    18     36     54

>> A*b
??? Error using ==> mtimes
Inner matrix dimensions must agree.

>> A*B

ans =

    150    186    222
    186    231    276
    222    276    330

```



## Κοινές Προγραμματιστικές Δομές I

### Δομές Επιλογής

```
if / elseif / else / end
```

```
% Preallocate a matrix
nrows = 10;
ncols = 10;
myData = ones(nrows, ncols);
```

```
% Loop through the matrix
for r = 1:nrows
    for c = 1:ncols

        if r == c
            myData(r,c) = 2;
        elseif abs(r - c) == 1
            myData(r,c) = -1;
        else
            myData(r,c) = 0;
        end
    end
end
```

```
end
end
```

```
switch / case / otherwise / end
```

```
mynumber = input('Enter a ...
                 number:');
```

```
switch mynumber
    case -1
        disp('negative one');
    case 0
        disp('zero');
    case 1
        disp('positive one');
    otherwise
        disp('other value');
end
```

### Άλλες Δομές

```
try ... catch ... end
```

## Κοινές Προγραμματιστικές Δομές II

### Δομές Επανάληψης

`for / end`

```
x = ones(1,10); % ...  
    Preallocation!!!!  
for n = 2:6  
    x(n) = 2 * x(n - 1);  
end
```

`while / end`

```
n = 1;  
nFactorial = 1;  
while nFactorial < 1e100  
    n = n + 1;  
    nFactorial = nFactorial * n;  
end
```

### Εντολές Ροής Επανάληψεων

`break, continue`

### Εντολές Ροής Συναρτήσεων/Προγράμματος

`pause, return`

## Πράξη Μητρώο $\times$ Διάνυσμα I

Η πράξη μητρώο διάνυσμα μπορεί να γίνει θεωρητικά με δύο τρόπους:

### Κατά Στήλες

$$c = A \times b = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ a_{3,1} & a_{3,2} & \cdots & a_{3,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix} \times \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \vdots \\ \beta_n \end{bmatrix}$$

Οπότε η πράξη θα γίνει:

$$\begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_3 \end{bmatrix} = \beta_1 \times \begin{bmatrix} a_{1,1} \\ a_{2,1} \\ \vdots \\ a_{n,1} \end{bmatrix} + \beta_2 \times \begin{bmatrix} a_{1,2} \\ a_{2,2} \\ \vdots \\ a_{n,2} \end{bmatrix} + \cdots + \beta_n \times \begin{bmatrix} a_{1,n} \\ a_{2,n} \\ \vdots \\ a_{n,n} \end{bmatrix}$$

Αν θα θέλαμε να το προγραμματίσουμε σε MATLAB θα γράφαμε:

```
% Matrix x Vector - ...  
columnwise  
  
% initialization of c ...  
with zeros  
c = zeros(size(A,1),1);  
  
% obtain each column of A  
for i=1:size(A,2)  
    c = c + b(i)*A(:,i);  
end
```

## Πράξη Μητρώο $\times$ Διάνυσμα II

### Κατά Γραμμές

$$c = A \times b = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ a_{3,1} & a_{3,2} & \cdots & a_{3,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix} \times \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \vdots \\ \beta_n \end{bmatrix}$$

Οπότε η πράξη θα γίνει:

$$c = \begin{bmatrix} a_{1,1}\beta_1 + a_{1,2}\beta_2 + \cdots + a_{1,n}\beta_n \\ a_{2,1}\beta_1 + a_{2,2}\beta_2 + \cdots + a_{2,n}\beta_n \\ \vdots \\ a_{n,1}\beta_1 + a_{n,2}\beta_2 + \cdots + a_{n,n}\beta_n \end{bmatrix}$$

Αν θα θέλαμε να το προγραμματίσουμε σε MATLAB θα γράφαμε:

```
% Matrix x Vector -rowwise
% initialization of c ...
% with zeros
c = zeros(size(A,1),1);

for i=1:size(A,1),
    for j=1:size(A,2),
        c(i) = c(i) + ...
            A(i,j)*b(j);
    end
end
```

Όπως είδαμε η ίδια πράξη μπορεί να γίνει πολύ πιο απλά με μία εντολή:

```
c = A*b;
```

# Περιεχόμενα

- 1 Εισαγωγικά
- 2 Συστατικά Μέρη του MATLAB
- 3 Κατασκευές, Απλές Εντολές & Πράξεις
- 4 Αριθμητική Κινητής Υποδιαστολής**
- 5 Sparse Matrix Technology
- 6 Μέτρηση Χρόνου Εκτέλεσης
- 7 Οπτικοποίηση Δεδομένων
- 8 Πηγές

## Αριθμητική Κινητής Υποδιαστολής στο MATLAB

Το MATLAB χρησιμοποιεί το μοντέλο Αριθμητικής Κινητής Υποδιαστολής : **IEEE-754**

Υποδομή για :

- 🚩 Διπλή Ακρίβεια (64bits - default) `double`
- 🚩 Μονή Ακρίβεια (32bits) `single`

```
Command Window
>> a = [1 2 3];
>> b = single(a);
>> whos
  Name      Size      Bytes  Class  Attributes
   a         1x3         24  double
   b         1x3         12  single

>> b = double(b);
>> whos
  Name      Size      Bytes  Class  Attributes
   a         1x3         24  double
   b         1x3         24  double
fx >> |
```

Άλλες εντολές: `realmin`, `realmax`, `eps`

Θα εξηγήσουμε σε επόμενο εργαστήριο περισσότερα!!

- 1 Εισαγωγικά
- 2 Συστατικά Μέρη του MATLAB
- 3 Κατασκευές, Απλές Εντολές & Πράξεις
- 4 Αριθμητική Κινητής Υποδιαστολής
- 5 Sparse Matrix Technology**
- 6 Μέτρηση Χρόνου Εκτέλεσης
- 7 Οπτικοποίηση Δεδομένων
- 8 Πηγές

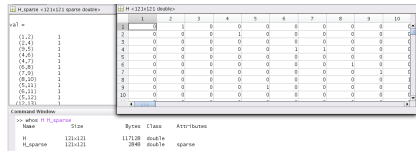
## Αραιά Μητρώα (Sparse Matrices)

Μητρώα τα οποία περιέχουν λίγα μη-μηδενικά στοιχεία (nnz). Είναι σημαντικά γιατί:

- επιδέχονται οικονομική αποθήκευση (με ειδικούς τρόπους που θα γνωρίσουμε)
- μειώνουν το αριθμητικό και χρονικό κόστος διάφορων μεθόδων, ειδικά αν έχουν και περαιτέρω μορφή (π.χ. ταινιακά μητρώα)

Το MATLAB διαθέτει ειδικό περιβάλλον διαχείρισης αραιών μητρώων και όπως θα δούμε πολλές συναρτήσεις που θα συναντήσουμε διαθέτουν εξειδικευμένη έκδοση για αραιά μητρώα.

Για να δηλώσουμε ένα μητρώο αραιό, χρησιμοποιούμε την εντολή **sparse**



Σε επόμενα μαθήματα-εργαστήρια θα μάθουμε περισσότερα για την τεχνολογία αυτή, και για τον τρόπο χρήσης της!



# Περιεχόμενα

- 1 Εισαγωγικά
- 2 Συστατικά Μέρη του MATLAB
- 3 Κατασκευές, Απλές Εντολές & Πράξεις
- 4 Αριθμητική Κινητής Υποδιαστολής
- 5 Sparse Matrix Technology
- 6 Μέτρηση Χρόνου Εκτέλεσης**
- 7 Οπτικοποίηση Δεδομένων
- 8 Πηγές

## Χρονομέτρηση I

Η χρονομέτρηση στο MATLAB μπορεί να γίνει χρησιμοποιώντας κάποια από τις επόμενες εντολές:

### `tic,toc`

```
tic
for i = 1:n,
--code--
end
t = toc/n;
```

### `timeit`

```
t = timeit(f,numOutputs)
```

**Σημείωση:** Για MATLAB > R2013b η `timeit` είναι ενδογενής, για τις υπόλοιπες MATLAB μπορείτε να κατεβάσετε τον κώδικα από [εδώ](#)!

## Χρονομέτρηση II

### Παράδειγμα 1:

Επίλυση Γραμμικού Συστήματος  $Ax = b$

#### Command Window

```
>> solve  
  
classic_time =  
  
    5.1000e-05  
  
Mat_time_tictoc =  
  
    5.2000e-05  
  
Mat_time_timeit =  
  
    1.0002e-05  
  
fx >> |
```

Παρατηρήστε την χρήση του τελεστή  
“\” ή **mldivide**

Σε MATLAB έχουμε τον παρακάτω κώδικα

```
% Solving Linear System Ax=b  
  
A = [1,2,3; 1,1,1; 2,3,1];  
% A = rand(5000);  
  
b = [6;3;6];  
% b = rand(5000,1);  
  
% Linear Algebra way  
tic  
    x = inv(A) * b;  
classic_time = toc  
  
% MATLAB way  
%--tic-toc  
tic  
x = A\b;  
Mat_time_tictoc = toc  
%--timeit  
% handler to the function  
f = @( ) mldivide(A,b);  
Mat_time_timeit = timeit(f)
```

## Χρονομέτρηση III

Σε MATLAB έχουμε τον παρακάτω κώδικα

### Παράδειγμα 2:

Άθροισμα Στοιχείων Διανύσματος

#### Command Window

```
>> summation  
  
classic_time =  
  
    5.3000e-05  
  
LA_time =  
  
    8.4000e-05  
  
Mat_time_tictoc =  
  
    2.0800e-04  
  
Mat_time_timeit =  
  
    8.0950e-06
```

```
% Sum of the elements of a vector
```

```
% we want column vector
```

```
b = [1:5000]';
```

```
% C way
```

```
tic
```

```
s1 = 0;
```

```
for i = 1:length(b),  
    s1 = s1 + b(i);
```

```
end
```

```
classic_time = toc
```

```
% Linear Algebra way
```

```
tic
```

```
e = ones(length(b),1);
```

```
s3 = b'*e;
```

```
LA_time = toc
```

```
% MATLAB way
```

```
%--tic-toc
```

```
tic
```

```
s2 = sum(b);
```

```
Mat_time_tictoc = toc
```

```
%--timeit
```

```
f = @() sum(b);
```

```
Mat_time_timeit = timeit(f)
```

## Χρονομέτρηση IV

Για να είμαστε ακριβείς στη χρονομέτρηση όσο το δυνατόν περισσότερο:

- ✚ Απορρίπτουμε την πρώτη μέτρηση λόγω του ότι εμπεριέχει χρόνο για μεταφορές των δεδομένων στην κύρια μνήμη (**warming up**)
- ✚ Κρατάμε ένα σύνολο μετρήσεων για την ίδια πράξη και επιστρέφουμε το **μέσο όρο**
- ✚ Η `timeit` κάνει όλα τα παραπάνω αυτόματα! Προσοχή στην `tic, toc...`

### Παράδειγμα:

```
% accurate time evaluation
```

```
A = [1,2,3; 1,1,1; 2,3,1];  
b = [6;3;6];
```

```
x = A\b; % first evaluation  
% does not count
```

```
% Preallocation!!!!
```

```
Mat_time = zeros(10,1);
```

```
for t = 1:10,  
    tic
```

```
        x = A\b;
```

```
        Mat_time(t) = toc;
```

```
end
```

```
accurate_time = sum(Mat_time)./t;
```

```
% or
```

```
tic
```

```
for t = 1:10,  
    x = A\b;
```

```
end
```

```
Mat_time = toc./t;
```

```
% timeit
```

```
f = @() mldivide(A,b);
```

```
Mat_time = timeit(f);
```

Μην ξεχνάτε ποτέ να κάνετε **Preallocation** μητρώα και διανύσματα των οποίων το μέγεθος γνωρίζεται ακόμα και αν χρησιμοποιείται αργές προγραμματιστικές δομές (**for**), μπορεί να μειώσει το χρονικό κόστος αισθητά!!!

## Profiler

Ένα άλλο σημαντικό εργαλείο που βοηθάει στη χρονομέτρηση ενός κώδικα MATLAB είναι ο **Profiler**.

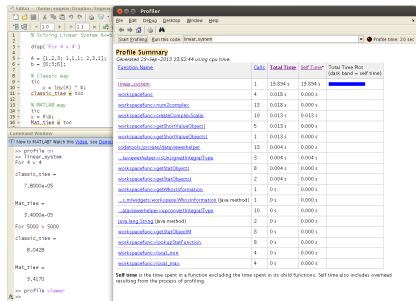
🚀 Ενεργοποίηση: `profile on`

🚀 Προβολή Αποτελεσμάτων:  
`profile viewer`

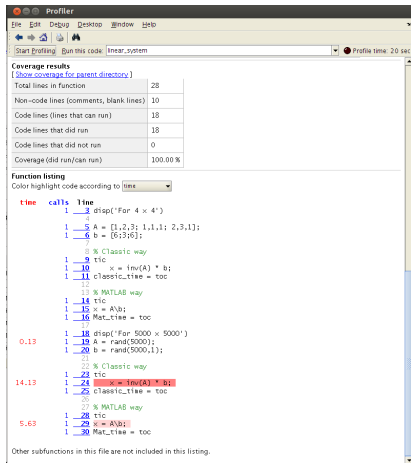
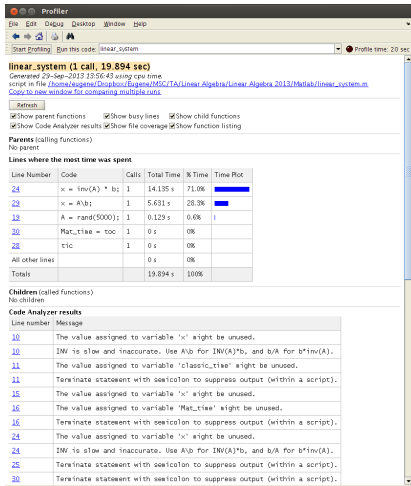
Εναλλακτικά, από το μενού:

🚀 MATLAB < R2012a: Desktop → Profiler

🚀 MATLAB ≥ R2012a: Editor tab → Run and Time



# Χρονομέτρηση VI



# Περιεχόμενα

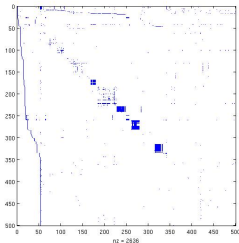
- 1 Εισαγωγικά
- 2 Συστατικά Μέρη του MATLAB
- 3 Κατασκευές, Απλές Εντολές & Πράξεις
- 4 Αριθμητική Κινητής Υποδιαστολής
- 5 Sparse Matrix Technology
- 6 Μέτρηση Χρόνου Εκτέλεσης
- 7 Οπτικοποίηση Δεδομένων
- 8 Πηγές



## Οπτικοποίηση Μητρώων I

Το MATLAB προσφέρει τη δυνατότητα να οπτικοποιήσουμε τη δομή των μητρώων, μέσω της εντολής **spy**

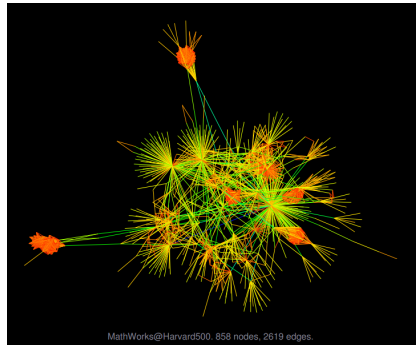
`spy` μητρώου Harvard500



Το Harvard500 είναι το μητρώο γειτνίασης του γραφήματος το οποίο απεικονίζει τις συνδέσεις μεταξύ 500 ιστοσελίδων με αρχική την κεντρική ιστοσελίδα του Harvard.

## Γράφημα Harvard500

The University of Florida Sparse Matrix Collection



## Οπτικοποίηση Συναρτήσεων στις 2 Διαστάσεις I

Το MATLAB προσφέρει τη δυνατότητα οπτικοποίησης συναρτήσεων.

### Παράδειγμα:

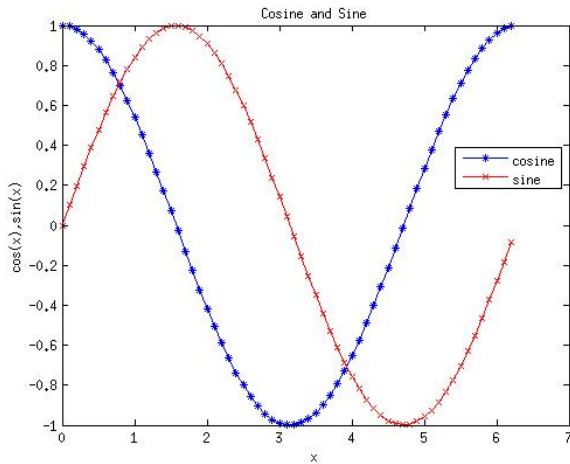
```
x = [0:0.1:2*pi];  
cosine = cos(x);  
sine = sin(x);
```

Αν θέλουμε να οπτικοποιήσουμε τις γραφικές σε μία εικόνα μπορούμε να το κάνουμε με δύο τρόπους χρησιμοποιώντας την `plot`:

```
% 2-d plot  
figure % new figure  
plot(x, cosine, 'b*-', x, sine, 'rx-')  
legend('cosine', 'sine')  
title('Cosine and Sine')  
xlabel('x')  
ylabel('cos(x), sin(x)')
```

```
% 2-d plot  
figure % new figure  
plot(x, cosine, 'b*-')  
hold on  
plot(x, sine, 'rx-')  
hold off  
legend('cosine', 'sine')  
title('Cosine and Sine')  
xlabel('x')  
ylabel('cos(x), sin(x)')
```

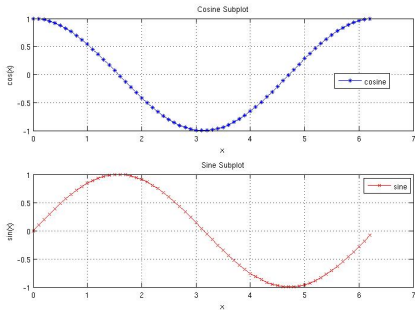
## Οπτικοποίηση Συναρτήσεων στις 2 Διαστάσεις II



## Οπτικοποίηση Συναρτήσεων στις 2 Διαστάσεις III

Αν θέλουμε να οπτικοποιήσουμε τις δύο γραφικές παραστάσεις σε διαφορετικές εικόνες αλλά στο ίδιο παράθυρο τότε μπορούμε να χρησιμοποιήσουμε επιπλέον την εντολή `subplot`:

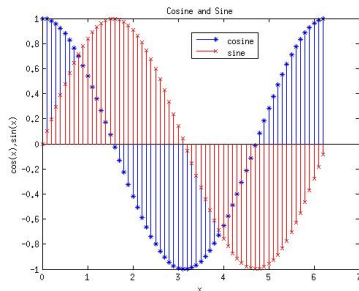
```
% Subplot
figure
subplot(2,1,1)
plot(x,cosine,'b*-')
xlabel('x')
ylabel('cos(x)')
legend('cosine')
title('Cosine Subplot')
grid on
subplot(2,1,2)
plot(x,sine,'rx-')
xlabel('x')
ylabel('sin(x)')
legend('sine')
title('Sine Subplot')
grid on
```



## Οπτικοποίηση Διακριτών Συναρτήσεων στις 2 Διαστάσεις

Μπορούμε επίσης να αναπαραστήσουμε συναρτήσεις με διακριτό τρόπο μέσω της stem:

```
% 2-d stem
figure % new figure
stem(x,cosine,'b*-')
hold on
stem(x,sine,'rx-')
hold off
legend('cosine','sine')
title('Cosine and Sine')
xlabel('x')
ylabel('cos(x),sin(x)')
```



## Οπτικοποίηση Συναρτήσεων στις 2 Διαστάσεις με Λογαριθμικούς Άξονες I

Μερικές φορές τα δεδομένα που θέλουμε να αναπαραστήσουμε είναι λογαριθμικά και όχι γραμμικά κατανομημένα. Αυτό δημιουργεί πρόβλημα στην κλασσική αναπαράσταση των δεδομένων (`plot`) και για αυτό χρησιμοποιούνται άλλες συναρτήσεις οι οποίες υποδηλώνουν λογαριθμική κατανομή ενός εκ των δύο (`semilogy`, `semilogx`) ή και των δύο αξόνων (`loglog`).

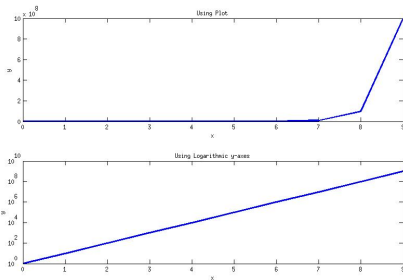
### Παράδειγμα 1:

Λογαριθμική Κατανομή στον Άξονα y

```
% Logarithmic y
x = 0:9;
y = (10.*ones(1,10)).^x;

figure
subplot(2,1,1)
plot(x,y)
title('Using Plot')
xlabel('x')
ylabel('y')

subplot(2,1,2)
semilogy(x,y)
title('Using Logarithmic ...
      y-axes')
xlabel('x')
ylabel('y')
```



## Οπτικοποίηση Συναρτήσεων στις 2 Διαστάσεις με Λογαριθμικούς Άξονες II

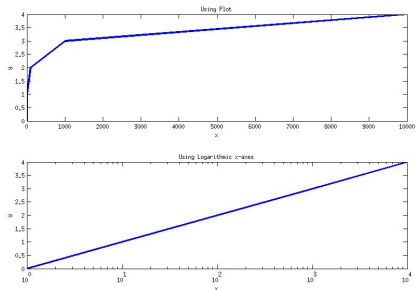
### Παράδειγμα 2:

Λογαριθμική Κατανομή στον Άξονα x

```
% Logarithmic x
x = [1,10,100,1000,10000];
y = log10(x);

figure
subplot(2,1,1)
plot(x,y)
title('Using Plot')
xlabel('x')
ylabel('y')

subplot(2,1,2)
semilogx(x,y)
title('Using Logarithmic ...
      x-axes')
xlabel('x')
ylabel('y')
```



## Οπτικοποίηση Συναρτήσεων στις 2 Διαστάσεις με Λογαριθμικούς Άξονες III

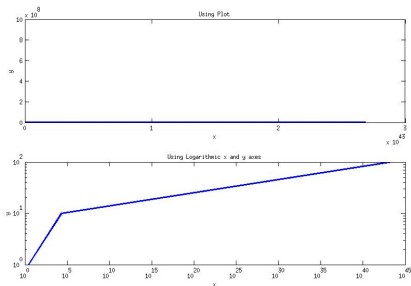
### Παράδειγμα 3:

Λογαριθμική Κατανομή στους Δύο Άξονες

```
% Logarithmic x and y
pow = 0:9;
y = (10.*ones(1,10)).^pow;
x = exp(y);

figure
subplot(2,1,1)
plot(x,y)
title('Using Plot')
xlabel('x')
ylabel('y')

subplot(2,1,2)
loglog(x,y)
title('Using Logarithmic ...
      x and y axes')
xlabel('x')
ylabel('y')
```

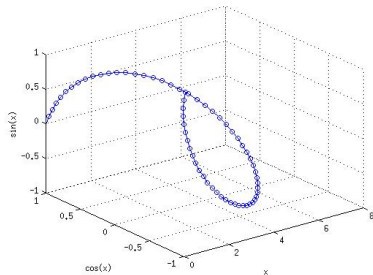




## Οπτικοποίηση Συναρτήσεων στις 3 Διαστάσεις I

Μπορούμε να αναπαραστήσουμε  
δεδομένα σε τρεις διαστάσεις με την  
plot3 ...

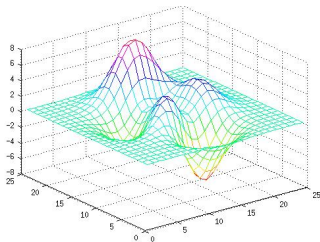
```
% 3-d plot  
figure % new figure  
plot3(x, cosine, sine, 'bo-')  
xlabel('x')  
ylabel('cos(x)')  
zlabel('sin(x)')
```



## Περισσότερες Συναρτήσεις για 3 Διαστάσεις I

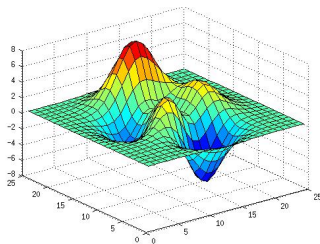
... αλλά και με άλλες συναρτήσεις όπως η  
mesh ...

```
% mesh demo-Matlab  
z=peaks(25);  
mesh(z);  
colormap(hsv)
```



... και η surf

```
% surf demo-Matlab  
z=peaks(25);  
surf(z);  
colormap(jet);
```



# Περιεχόμενα

- 1 Εισαγωγικά
- 2 Συστατικά Μέρη του MATLAB
- 3 Κατασκευές, Απλές Εντολές & Πράξεις
- 4 Αριθμητική Κινητής Υποδιαστολής
- 5 Sparse Matrix Technology
- 6 Μέτρηση Χρόνου Εκτέλεσης
- 7 Οπτικοποίηση Δεδομένων
- 8 Πηγές

## Μερικές Πηγές

### Αριθμητικές Μέθοδοι με το MATLAB

Cleve Moler

Numerical Computing with MATLAB

### MATLAB guide

Desmond Higham

Nick Higham

Matlab Guide

### Mathworks site

site

