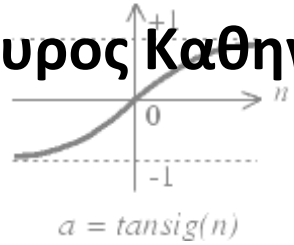
ΣΥΝΤΟΜΟ ΕΓΧΕΙΡΙΔΙΟ MATLAB

ΚΩΝΣΤΑΝΤΙΝΟΣ ΔΙΑΜΑΝΤΑΡΑΣ
Καθηγητής

ΚΩΝΣΤΑΝΤΙΝΟΣ ΓΟΥΛΙΑΝΑΣ
Επίκουρος Καθηγητής



ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε.

ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ

Θεσσαλονίκη 2015

Περιεχόμενα

ΚΕΦΑΛΑΙΟ 1

ΤΙ ΕΙΝΑΙ ΤΟ MATLAB

1.1 Δυνατότητες (Help: MATLAB->Getting Started ->Introduction->What Is MATLAB)

Το MATLAB (MATrix LABoratory) είναι ένα εργαλείο το οποίο αρχικά σχεδιάστηκε για μαθηματικό προγραμματισμό και το οποίο χρησιμοποιείται σήμερα ευρύτατα στην επιστημονική κοινότητα. Ωστόσο το MATLAB προσφέρει πάρα πολλές δυνατότητες στον προγραμματιστή, ώστε να μπορεί να θεωρηθεί πλέον ένα πολύ ισχυρό εργαλείο γενικού προγραμματισμού. Μερικά από τα κυριότερα χαρακτηριστικά του είναι :

- Προγραμματισμός σε γλώσσα *scripting* που μοιάζει πολύ με τη γλώσσα *C*. Τα αρχεία script έχουν κατάληξη .m.
- Εκτέλεση εντολών σε command line από interpreter για γρήγορη δοκιμή ενός προγράμματος, μιας εφαρμογής, ή μιας ιδέας.
- Δυνατότητα εκτέλεσης ενός script από το command line γράφοντας απλώς το όνομα του αρχείου script με ή χωρίς την κατάληξη .m.
- Δυνατότητα δημιουργίας αρχείων exe με χρήση compiler.
- Εύκολη διαχείριση πινάκων (matrices) και διανυσμάτων (vectors).
- Ολοκληρωμένο περιβάλλον editor/debugger (medit.exe). Καλείται από το κεντρικό παράθυρο του MATLAB όταν ζητήσουμε να ανοίξουμε ή να δημιουργήσουμε ένα νέο αρχείο, μπορεί όμως να εκτελεστεί και ως ανεξάρτητο πρόγραμμα.
- Εξαιρετικές δυνατότητες δημιουργίας γραφικών παραστάσεων εύκολα και γρήγορα. Γραφικές παραστάσεις 2, και 3 διαστάσεων. Προχωρημένες δυνατότητες όπως 3-D φωτισμός, αλλαγή οπτικής γωνίας, δημιουργία εικονοσειρών, κ.α..
- Γραφικός προγραμματισμός. Δυνατότητα σχεδιασμού παραθύρων, κουμπιών, γραφικών μενού, κ.λ.π.. Πλήρης γκάμα δυνατοτήτων για σχεδιασμό Graphical User Interfaces (GUI).
- Εξαιρετικό εργαλείο βοήθειας Help. Με τόσες δυνατότητες που προσφέρει το MATLAB είναι πολύ εύκολο να ξεχάσεις τα ακριβή ορίσματα μιας συνάρτησης ή το όνομα της συνάρτησης που εκτελεί μια συγκεκριμένη εργασία. Το εργαλείο Help του MATLAB είναι πλήρες, λεπτομερές, εύχρηστο και εύκολο στην αναζήτηση της πληροφορίας που χρειάζεται ο χρήστης, ενώ περιέχει παραδείγματα και demos.

1.2 Βιβλιοθήκες (Help: MATLAB->Functions -- Categorical List)

Το MATLAB περιέχει μεγάλο πλήθος έτοιμων βιβλιοθηκών. Οι standard βιβλιοθήκες προσφέρουν έτοιμες συναρτήσεις για επεξεργασία αριθμών, διανυσμάτων, πινάκων, για δημιουργία plots, κ.λ.π.. Υπάρχει επίσης μεγάλη σειρά ειδικών βιβλιοθηκών που λέγονται *εργαλειοθήκες* (*toolboxes*) και ειδικεύονται σε μια επιστημονική περιοχή. Μερικά από τα πιο σημαντικά toolboxes είναι:

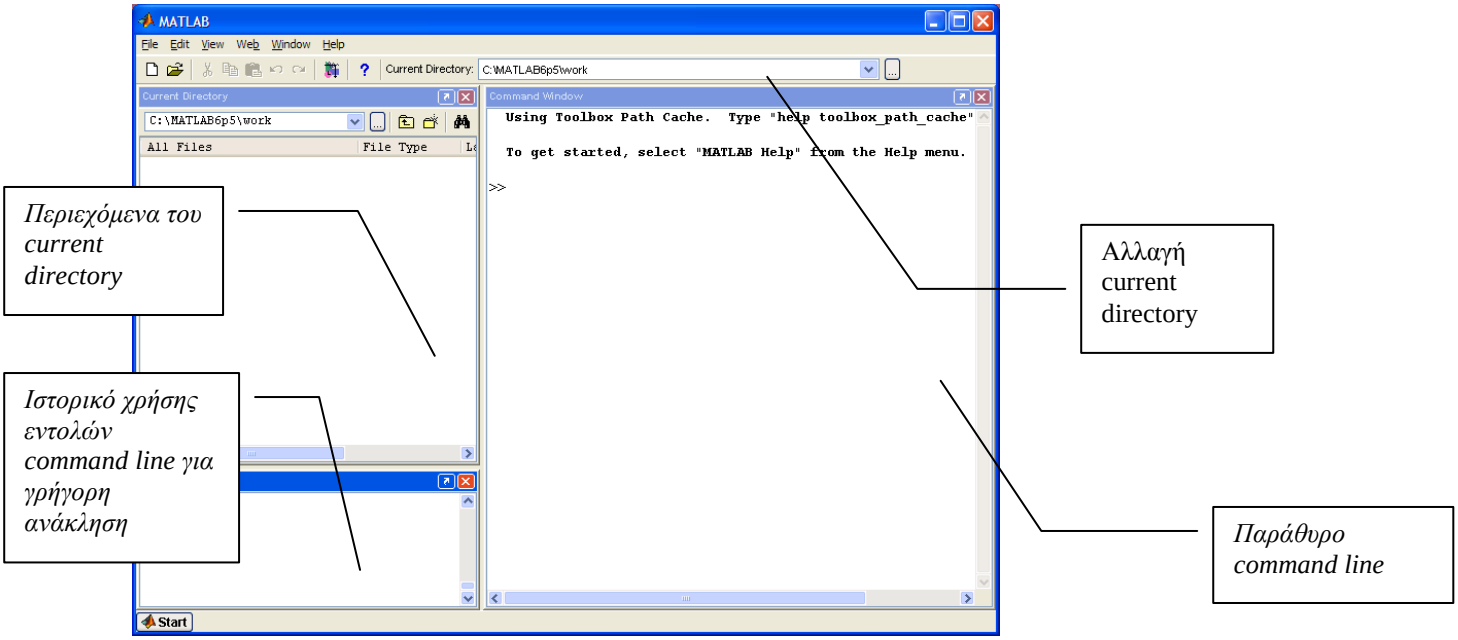
- Signal processing toolbox
- Image processing toolbox
- Neural networks toolbox
- Fuzzy Logic toolbox
- Statistics toolbox
- Optimization toolbox
- Communications toolbox
- Virtual Reality toolbox
- Database toolbox
- Control toolbox
- Symbolic math toolbox
- Financial toolbox
- Mapping toolbox
- Wavelet toolbox

1.3 Περιβάλλον Εργασίας (Help: MATLAB->Getting Started ->Desktop Tools and Development Environment)

Η Εικόνα 1.1 δείχνει το βασικό παράθυρο εργασίας του MATLAB, το οποίο περιλαμβάνει :

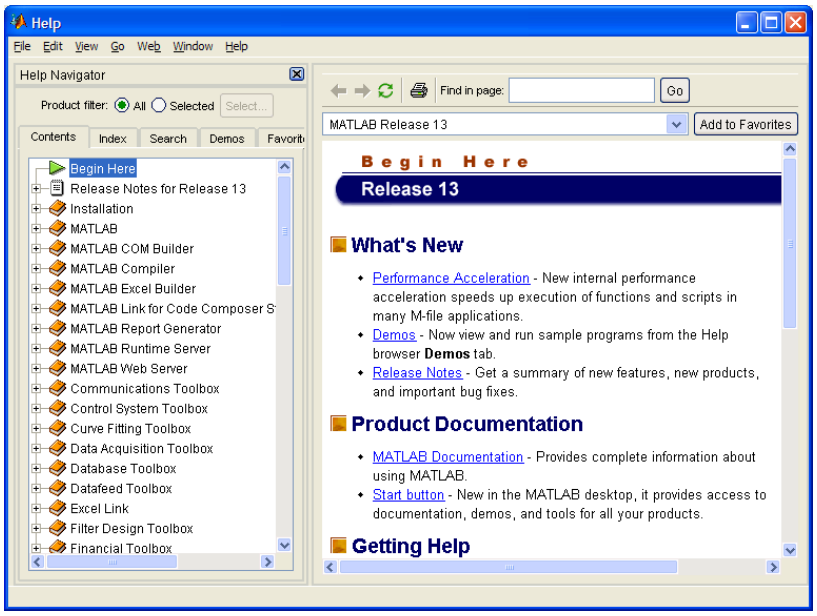
- Το ***Command Wndow***, στο οποίο μπορούμε να πληκτρολογήσουμε το όνομα ενός εκτελέσιμου αρχείου, μια εντολή ή να δούμε τα αποτελέσματα των παραπάνω.
- Το **Ιστορικό** των εντολών που έχουμε δώσει στο Command Line, τις οποίες μπορούμε επίσης να ανακαλέσουμε απ' το Command Window με τα βελάκια πάνω-κάτω.
- Το **Current Directory**, το οποίο μπορούμε να το αλλάξουμε, αντί για το default C:\MATLAB\work.
- Τα περιεχόμενα του **Current Directory**, από όπου μπορούμε να ανοίξουμε με τον Editor ένα οποιοδήποτε αρχείο.
- Τα μενού **File** και **Edit** με τις αντίστοιχες λειτουργίες και δυνατότητες.
- Το μενού **View**, με το οποίο επιλέγουμε τα παράθυρα που θα εμφανίζονται. Έτσι, μπορούμε να ανοίξουμε το παράθυρο ***Workspace***, στο οποίο βλέπουμε τις μεταβλητές που έχουμε ορίσει και το χώρο που καταλαμβάνουν στη μνήμη καθώς και το παράθυρο ***Profiler*** που είναι ένα χρήσιμο εργαλείο για τη βελτιστοποίηση του κώδικά μας. Για ένα συγκεκριμένο πρόγραμμα (script) μας δείχνει πόσος χρόνος CPU σπαταλήθηκε από κάθε γραμμή του κώδικα, πόσες φορές εκτελέστηκε κάθε loop, κ.λ.π..

.



Εικόνα 1.1

- Το μενού-παράθυρο ***Help***, ένα ολοκληρωμένο περιβάλλον αναζήτησης πληροφορίας μέσα στο πλήθος των εργαλειοθηκών, των εντολών, και των συναρτήσεων του MATLAB (Εικόνα 1.2) :



Εικόνα 1.2

Στο παράθυρο αυτό μπορεί κανείς να αναζητήσει πληροφορίες :

- Κάνοντας πλοήγηση στον πίνακα περιεχομένων (*Contents*)
- Από τον ονομαστικό κατάλογο των εντολών/συναρτήσεων (*Index*)
- Με αναζήτηση με λέξεις κλειδιά (*Search*)
- Από προγράμματα επίδειξης (*Demos*)

ΚΕΦΑΛΑΙΟ 2

ΕΙΣΑΓΩΓΗ ΣΤΟ MATLAB

2.1 Βασικά Χαρακτηριστικά (Help: MATLAB->Programming)

Το MATLAB είναι εξοπλισμένο με μια γλώσσα προγραμματισμού που μοιάζει πολύ με τη γλώσσα C. Η γλώσσα αυτή μπορεί να χρησιμοποιηθεί είτε στο command line για να γράφουμε μια-μια τις εντολές που θέλουμε να εκτελέσουμε είτε σε ένα αρχείο script με την κατάληξη .m. Με το που ξεκινάμε το MATLAB στο παράθυρο command line εμφανίζεται το prompt

>>

έτοιμο να δεχτεί τις εντολές που θα πληκτρολογήσουμε.

Γραμμές εντολών

Αν βρισκόμαστε στο command line μια γραμμή εντολών τελειώνει πατώντας <ENTER>. Αν γράφουμε κώδικα σε ένα αρχείο script μια γραμμή εντολών τελειώνει με new line. Αν μια εντολή είναι μεγάλη σε μήκος ή θέλουμε για οποιοδήποτε λόγο να την σπάσουμε σε πολλές γραμμές χρησιμοποιούμε τρεις τελείες. Οι τρεις τελείες σημαίνουν ότι η εντολή συνεχίζεται στην επόμενη γραμμή. Για παράδειγμα, ο παρακάτω κώδικας :

```
fprintf('Τιμές: Δευτέρα=%d, Τρίτη=%d, Τετάρτη=%d\n', ...
p1, ...
p2, ...
p3);
```

αντιστοιχεί στην εντολή

```
fprintf('Τιμές: Δευτέρα=%d, Τρίτη=%d, Τετάρτη=%d\n', p1, p2, p3);
```

Όπως το UNIX και η C, το MATLAB είναι ευαίσθητο στα μικρά και στα κεφαλαία γράμματα (case sensitive). Έτσι η μεταβλητή A δεν είναι ίδια με την μεταβλητή a.

Σχόλια

Τα σχόλια ξεκινούν με το σύμβολο **%** «επί τοις εκατό» και τελειώνουν στο τέλος της γραμμής. Μια ολόκληρη γραμμή είναι σχόλιο αν ξεκινάει με **%**

Η χρήση του ; (semicolon) *(Help: MATLAB->Programming->Basic Program Components -> Symbol Reference)*.

Το σύμβολο **;** στο τέλος μιας εντολής αποτρέπει την εμφάνιση του περιεχομένου μιας μεταβλητής στην οθόνη. Για παράδειγμα, η εντολή

```
y = 3
```

θα κάνει την ανάθεση της τιμής 3 στο y και θα εμφανίσει στην οθόνη το εξής μήνυμα:

```
y =
     3
```

ενώ η εντολή

```
y = 3;
```

θα κάνει την ανάθεση της τιμής 3 στο y χωρίς να εμφανίσει τίποτα στην οθόνη.

Βασικές πράξεις - Τελεστές *(Help: MATLAB->Programming->Basic Program Components->Operators)*

Το MATLAB μπορεί να χρησιμοποιηθεί σαν μια απλή αριθμομηχανή χρησιμοποιώντας τις παρακάτω βασικές πράξεις (Πίνακας 1).

Σύμβολο	Πράξη
+	Πρόσθεση
-	Αφαίρεση
*****	Πολλαπλασιασμός
/	Διαίρεση
^	Ύψωση σε δύναμη

Πίνακας 1

Παράδειγμα 1

```
>> 1 + 25.6

ans  =
     26.6
```

Παράδειγμα 2

```
>>  8 - 15
ans  =
         -7
```

Παράδειγμα 3

```
>> 15 * 2
ans  =
         30
```

Παράδειγμα 4

```
>> 5 / 2
ans =

   2.5000          *** Δε γίνεται ακέραια διαίρεση!!!
```

Παράδειγμα 5

```
>> 2 ^ 3
ans  =
         8
```

Παρατήρηση

- Για την ακέραια διαίρεση μπορεί να χρησιμοποιηθεί ο τελεστής mod με παραμέτρους το Διαιρετέο και το Διαιρέτη, ο οποίος επιστρέφει το υπόλοιπο της διαίρεσης των 2 ακεραίων ή πραγματικών αριθμών.

Παράδειγμα 1

```
>>  mod(7,2)

ans =
            1
```

Παράδειγμα 2

```
>>  mod(6.5,2.5)

ans =
          1.5000
```

Προτεραιότητα πράξεων *(Help: MATLAB->Programming->Basic Program Components->Operators->Operator Precedence)*

Η προτεραιότητα των πράξεων στο MATLAB είναι ανάλογη με τους κανόνες προτεραιότητας των πράξεων της άλγεβρα.

1. Πρώτα εκτελούνται οι πράξεις στις παρενθέσεις από μέσα προς τα έξω.
2. Μετά εκτελούνται οι υψώσεις σε δύναμη.
3. Μετά εκτελούνται οι πολλαπλασιασμοί και οι διαιρέσεις από δεξιά προς τα αριστερά.
4. Τέλος εκτελούνται οι προσθέσεις και οι αφαιρέσεις από δεξιά προς τα αριστερά.

Τελεστές Συσχέτισης *(Help: MATLAB->Programming->Basic Program Components->Operators)*

Υπάρχουν οι παρακάτω τελεστές συσχέτισης, οι οποίοι σε απλές μεταβλητές επιστρέφουν την τιμή 1, αν η συσχέτιση είναι αληθής και το 0, αν δεν ισχύει : (Πίνακας 2).

Σύμβολο	Σύγκριση
==	Ίσον
~=	Διάφορο
>	Μεγαλύτερο
>=	Μεγαλύτερο ή Ίσον
<	Μικρότερο
<=	Μικρότερο ή Ίσον

Πίνακας 2

Παράδειγμα 1

```
>>  a = 5

a =
        5


>>  a > 0

ans =
            1
```

Παράδειγμα 2

```
>>  a < 0

ans =
            0
```

Παράδειγμα 3

```
>>  a == 5

ans =
            1
```

2.2 Μεταβλητές (Help: MATLAB->Programming-> Data Types)

Όπως και στη C, οι μεταβλητές μπορούν να έχουν διάφορους τύπους, για παράδειγμα double, char, string, logical, κ.λ.π.. Αντίθετα με την C, ο τύπος των μεταβλητών του MATLAB που πρόκειται να χρησιμοποιηθούν παρακάτω στο πρόγραμμα δεν χρειάζεται να έχει δηλωθεί εκ των προτέρων. Έτσι π.χ., η εντολή

```
>> x = 3;
```

δημιουργεί αυτομάτως μια μεταβλητή x στο χώρο της μνήμης του προγράμματος (workspace) που έχει τον τύπο double. Αντίστοιχα η εντολή

```
>> s = 'Hello!';
```

δημιουργεί αυτομάτως μια μεταβλητή τύπου string. Αν θέλουμε, μπορούμε να αλλάξουμε τον τύπο μιας μεταβλητής δίνοντάς της μια τιμή άλλου τύπου, ασχέτως ποιος ήταν ο προηγούμενος τύπος της μεταβλητής. Το MATLAB αλλάζει δυναμικά τον τύπο της καθώς τρέχει το πρόγραμμα ανάλογα με τις εντολές μας. Για παράδειγμα, με την εντολή

```
>> a = 5.46;
```

η μεταβλητή a έχει την double τιμή 5.46, ενώ με την εντολή

```
>> a = 'King Kong';
```

η μεταβλητή a γίνεται τύπου string με τιμή 'King Kong' και η παλιά της τιμή τύπου double χάνεται.

Για τη μετατροπή πραγματικών αριθμών σε ακέραιους (όπως και στοιχείων πινάκων), μπορούν να χρησιμοποιηθούν οι παρακάτω συναρτήσεις :

Συνάρτηση	**Αποτέλεσμα**
fix(x)	Αποκοπή του δεκαδικού μέρους
round(x)	Στρογγύλευση στον πλησιέστερο ακέραιο
ceil(x)	Στρογγύλευση στον πλησιέστερο μεγαλύτερο ακέραιο του x
floor(x)	Στρογγύλευση στον πλησιέστερο μικρότερο ακέραιο του x

Παραδείγματα

Εντολή	Αποτέλεσμα
>> b = fix(3.4)	b = 3
>> b = fix(3.9)	b = 3
>> b = fix(-3.4)	b = -3
>> b = fix(-3.9)	b = -3
>> b = round(3.4)	b = 3
>> b = round(3.9)	b = 4
>> b = round(-3.4)	b = -3
>> b = round(-3.9)	b = -4
>> b = ceil(3.4)	b = 4
>> b = ceil(3.9)	b = 4
>> b = ceil(-3.4)	b = -3
>> b = ceil(-3.9)	b = -3
>> b = floor(3.4)	b = 3
>> b = floor(3.9)	b = 3
>> b = floor(-3.4)	b = -4
>> b = floor(-3.9)	b = -4

2.3 Μαθηματικές Συναρτήσεις

Το Matlab διαθέτει μια πλούσια βιβλιοθήκη με μαθηματικές συναρτήσεις, μερικές απ' τις οποίες φαίνονται στον πίνακα που ακολουθεί :

Συνάρτηση	Αποτέλεσμα
abs(x)	Απόλυτη Τιμή
sqrt(x)	Τετραγωνική Ρίζα
exp(x)	e^x

log(x)	ln(x)
log10(x)	$log_{10}x$
sin(x)	Ημίτονο
cos(x)	Συνημίτονο
tan(x)	Εφαπτομένη
tanh(x)	Υπερβολική Εφαπτομένη

Πίνακας 3

Παράδειγμα 1

```
>> 1/(1+exp(0))

ans =

   0.5
```

Παράδειγμα 2

```
>> (exp(0)-exp(0))/(exp(0)+exp(0))

ans =

   0
```

Παράδειγμα 3

```
>> tanh(0)

ans =

   0
```

ΚΕΦΑΛΑΙΟ 3

ΔΙΑΝΥΣΜΑΤΑ - ΠΙΝΑΚΕΣ ΣΤΟ MATLAB

3.1 Δημιουργία Διανυσμάτων (Help: MATLAB->Programming->Data Structures)

Τα διανύσματα και οι πίνακες μπορούν πολύ εύκολα να αναπαρασταθούν στο MATLAB χωρίς να χρειάζεται να δηλωθούν εκ των προτέρων. Ένα διάνυσμα μπορεί να πάρει τιμές με τους παρακάτω τρόπους :

1ος Τρόπος

Με ανάθεση τιμών, όπως στη C.

Παράδειγμα 1

```
>>      a = [1, -3, 2]
```

Το αποτέλεσμα της προηγούμενης εντολής είναι η δημιουργία ενός διανύσματος γραμμής ή ενός πίνακα 1x3 (1 γραμμή, 3 στήλες) :

```
a =
        1   -3   2
```

Το ίδιο αποτέλεσμα θα είχαμε με την εντολή :

```
>> a = [1 -3 2]
```

Παρατήρηση

- Μπορούμε να προσπελάσουμε το στοιχείο 1 του διανύσματος a γράφοντας a(1) (με τιμή 1) και γενικά το στοιχείο i του διανύσματος a γράφοντας a(i).

Παράδειγμα 2

```
>>      b = [1; -3; 2]
```

Το αποτέλεσμα της προηγούμενης εντολής είναι η δημιουργία ενός διανύσματος στήλης ή ενός πίνακα 3x1 (3 γραμμές, 1 στήλη) :

```
b =
        1
        -3
        2
```

Το ίδιο αποτέλεσμα θα είχαμε με την εντολή :

```
>>      b = [1
-3
2]
```

Παρατηρήσεις

- Το «,» διαχωρίζει τα στοιχεία ενός διανύσματος ή πίνακα που βρίσκονται στην ίδια γραμμή ενώ το *";"* (*ελληνικό ερωτηματικό*) δείχνει αλλαγή γραμμής.
- Το αν ένα διάνυσμα έχει δηλωθεί σαν γραμμή ή στήλη παίζει σημαντικό ρόλο στο γινόμενο διανύσματος επί διάνυσμα ή διανύσματος επί πίνακα.

2^{ος} Τρόπος

Δημιουργώντας αριθμούς στη σειρά δίνοντας αρχή, τέλος, και προαιρετικά το βήμα. Αν παραβλέψουμε το βήμα τότε αυτό θεωρείται ίσο με 1. Στα παραδείγματα που ακολουθούν η αρχή, το τέλος και το βήμα μπορούν να είναι μεταβλητές ή σταθερές :

Παράδειγμα 1

```
>> a = 0; step = 2; b = 6;
>> x = a : step : b

x  =
        0    2    4    6
```

Το ίδιο αποτέλεσμα θα είχαμε με την εντολή :

```
>> x = 0 : 2 : 6

x  =
        0    2    4    6
```

Παράδειγμα 2

```
>> x = a : b

x =
          0    1    2    3    4    5    6
```

Το ίδιο αποτέλεσμα θα είχαμε με την εντολή :

```
>> x = 0 : 6

x =
          0    1    2    3    4    5    6
```

Παράδειγμα 3

```
>> y = 35 : -5 : 10

y  =
        35   30   25   20   15   10
```

3ος Τρόπος

Με τις συναρτήσεις **rand**, **randn**, **zeros**, **ones, linspace, logspace**.

- Οι συναρτήσεις **rand** και **randn**

Οι συναρτήσεις **rand** και **randn** δίνουν την δυνατότητα δημιουργίας τυχαίων αριθμών και απαιτούν δύο ορίσματα, το πλήθος των γραμμών και των στηλών του πίνακα που δημιουργείται. Η συνάρτηση **rand** παράγει τυχαίους αριθμούς με **ομοιόμορφη** κατανομή μεταξύ 0 και 1. Η συνάρτηση **randn** παράγει τυχαίους αριθμούς με **Γκαουσσιανή** κατανομή με μέση τιμή 0 και διασπορά 1 (γνωστή σαν «*κανονική κατανομή*»).

Παράδειγμα 1

```
>>  x = rand(1,3);      % Δημιουργεί τον πίνακα x με 1 γραμμή και 3 στήλες και τυχαίες τιμές στο (0,1).
```

x =

0.8491 0.9340 0.6787

Παράδειγμα 2

```
>> x = rand(3,1);      % Δημιουργεί τον πίνακα x με 3 γραμμές και 1 στήλη και τυχαίες τιμές στο (0,1).
x =
   0.7577
   0.7431
   0.3922
```

Παράδειγμα 3

```
>> x = rand(1,1);      % Δημιουργεί τον πίνακα x με 1 γραμμή και 1 στήλη ( απλή μεταβλητή ) και τυχαία τιμή στο (0,1).
```

Παράδειγμα 4

```
>> x = randn(1,3);     % Δημιουργεί τον πίνακα x με 1 γραμμή και 3 στήλες και τυχαίες τιμές με Γκαουσσιανή κατανομή.
x =
    0.2939   -0.7873    0.8884
```

- Οι συναρτήσεις **zeros, ones**

Οι συναρτήσεις **zeros**, **ones** γεμίζουν έναν πίνακα με μηδενικά ή μονάδες.

Παράδειγμα 1

```
>> x = zeros (1,3);    % Δημιουργεί τον πίνακα x με 1 γραμμή και 3 στήλες και τιμές 0
```

Παράδειγμα 2

```
>> x = ones (3,1);     % Δημιουργεί τον πίνακα x με 3 γραμμές και 1 στήλη και τιμές 1
```

- Οι συναρτήσεις **linspace, logspace**

Η συνάρτηση **linspace(<αρχή>, <τέλος>, <αριθμός στοιχείων>)** δημιουργεί ένα διάνυσμα με στοιχεία όσα και ο **<αριθμός στοιχείων>** με τιμές από την **<αρχή>** μέχρι το **<τέλος>**.

Η συνάρτηση **logspace(<αρχή>, <τέλος>, <αριθμός στοιχείων>)** δημιουργεί ένα διάνυσμα με στοιχεία όσα και ο **<αριθμός στοιχείων>** με τιμές το 10 υψωμένο στη δύναμη από την **<αρχή>** μέχρι το **<τέλος>**.

Παράδειγμα 1

```
>> a = linspace(1,4,7)

a =

   1.0000    1.5000    2.0000    2.5000    3.0000    3.5000    4.0000
```

Παράδειγμα 2

```
>> a = logspace(0,4,5)

a =
        1        10       100      1000     10000
```

3.2 Επεξεργασία Διανυσμάτων (Help: MATLAB->Functions by category->Programming and Data Type->Operators and Operations)

Το MATLAB προσφέρει βασικές λειτουργίες δημιουργίας και επεξεργασίας πινάκων. Τα διανύσματα είναι απλώς ειδικές περιπτώσεις πινάκων όπου η μια διάσταση έχει μήκος 1. Οι πράξεις μεταξύ διανυσμάτων και αριθμών είναι απλούστατες και χρησιμοποιούν τους τελεστές των βασικών πράξεων :

- Πρόσθεση αριθμού σε διάνυσμα :

```
>>  a = [1 2 3]

a =
            1     2     3

>>  a +2

ans =
          3     4     5
```

- Αφαίρεση αριθμού από διάνυσμα :

```
>>  a-2

ans =
         -1     0     1
```

- Πολλαπλασιασμός διανύσματος με σταθερά :

```
>> a*2

ans =
     2     4     6
```

- Διαίρεση διανύσματος με σταθερά :

```
>> a/2

ans =
    0.5000    1.0000    1.5000
```

- Ύψωση διανύσματος σε δύναμη :

```
>> a^2
??? Error using ==> mpower
Matrix must be square.
```

Παρατηρήσεις

❖ Ό τελεστής '^' όταν χρησιμοποιείται σε διανύσματα ή πίνακες ισοδυναμεί με τον πολλαπλασιασμό του διανύσματος ή πίνακα με τον εαυτό του, τόσες φορές, όσες είναι η τιμή του εκθέτη. Στο προηγούμενο παράδειγμα, η εντολή a^2 ισοδυναμεί με το a*a, το οποίο δεν μπορεί να γίνει, γιατί το διάνυσμα έχει διάσταση 1x3.

❖ Για να υψωθεί κάθε στοιχείο του διανύσματος a στο τετράγωνο, θα πρέπει ο τελεστής '^' να χρησιμοποιηθεί με την τελεία '.', δηλαδή :

Παράδειγμα

```
>> a.^2

ans =
     1     4     9
```

3.2.1 Τελεστής του Ανάστροφου (*Help:Search For: Transpose)*

Στα μαθηματικά ένα διάνυσμα συμβολίζεται με ένα μικρό γράμμα boldface, πχ., **a**, **b**, **c**, **d**, ... και αποτελείται από *n* στοιχεία γραμμένα ανάμεσα σε τετράγωνες αγκύλες [ ]. Κατά πάγια σύμβαση των μαθηματικών θεωρείται ότι ένα διάνυσμα είναι μια **στήλη** και όχι μια γραμμή. Φυσικά μπορούμε να μετατρέψουμε τη στήλη σε γραμμή και τη γραμμή σε στήλη χρησιμοποιώντας τον τελεστή του **αναστρόφου** (**transpose**) που συμβολίζεται στα μαθηματικά με το γράμμα *T* υψωμένο σε εκθέτη. Στο MATLAB για την αναστροφή χρησιμοποιείται το σύμβολο " ' " :

Παράδειγμα 1

```
>> a = [1 2 3]

a =
     1    2    3

>> a'

ans =
     1
     2
     3
```

3.2.2 Πρόσθεση - Αφαίρεση Διανυσμάτων

Για να γίνει η Πρόσθεση ή η Αφαίρεση 2 διανυσμάτων θα πρέπει να έχουν δηλωθεί και τα 2 σαν διανύσματα-γραμμές ή στήλες, διαφορετικά θα πρέπει να χρησιμοποιηθεί στο ένα απ' αυτά ο τελεστής του αναστρόφου και το νέο διάνυσμα που προκύπτει περιέχει το άθροισμα ή τη διαφορά των αντίστοιχων στοιχείων των 2 διανυσμάτων.

Παράδειγμα 1

```
>> a = [1 2 3]

a =
     1    2    3

>> b = [4 5 6]

b =
     4    5    6

>> c= [7;8;9]
```

```
c =
           7
           8
           9

>> a+b

ans =
           5    7    9
```

Παράδειγμα 2

```
>> a-b

ans =

  -3   -3   -3
```

Παράδειγμα 3

```
>> a+c'

ans =

   8   10   12
```

3.2.3 Διαίρεση δύο Διανυσμάτων

Για να γίνει η Διαίρεση 2 διανυσμάτων χρησιμοποιείται ο τελεστής «./». Τα διανύσματα θα πρέπει να έχουν δηλωθεί και τα 2 σαν διανύσματα-γραμμές ή στήλες, διαφορετικά θα πρέπει να χρησιμοποιηθεί στο ένα απ' αυτά ο τελεστής του αναστρόφου και το νέο διάνυσμα που προκύπτει περιέχει το πηλίκο των αντίστοιχων στοιχείων των 2 διανυσμάτων.

Παράδειγμα

```
>> a./b

ans =

  0.5000   1.3333   1.3333
```

3.2.4 Το Εσωτερικό Γινόμενο δύο Διανυσμάτων

Η έννοια του εσωτερικού γινομένου είναι θεμελιώδης στη γραμμική άλγεβρα και θα μας φανεί ιδιαίτερα χρήσιμη. Σύμφωνα με τον ορισμό του το **εσωτερικό γινόμενο** δύο διανυσμάτων **a** και **b** είναι το άθροισμα του γινομένου των στοιχείων τους. Αν τα στοιχεία του διανύσματος **a** είναι **a(1), a(2), …, a(n)** και τα στοιχεία του διανύσματος **b** είναι **b(1), b(2), …, b(n)** το εσωτερικό τους γινόμενο θα είναι το **a(1)*b(1) + a(2) *b(2) + … + a(n) *b(n)**. Για να γίνει ο πολλαπλασιασμός στο Matlab θα πρέπει οι στήλες του αριστερού διανύσματος να είναι όσες και οι γραμμές του δεξιού. Με άλλα λόγια το αριστερό διάνυσμα πρέπει να είναι μια γραμμή *n* στοιχείων ενώ το δεξί πρέπει να είναι μια στήλη *n* στοιχείων. Προφανώς τα δύο διανύσματα πρέπει να έχουν το ίδιο πλήθος στοιχείων, δηλαδή το ίδιο μήκος.

Παρατηρήσεις

- Το εσωτερικό γινόμενο είναι ένας *απλός αριθμός*, (όχι διάνυσμα).
- Αν τα 2 διανύσματα **a, b** είναι γραμμές, το δεξί (**b**) πρέπει να αναστραφεί για να γίνει στήλη, ενώ το αριστερό (**a**) μένει ως έχει.
- Αν τα 2 διανύσματα **a, b** είναι στήλες, το αριστερό (**a**) πρέπει να αναστραφεί για να γίνει γραμμή ενώ το δεξί (**b**) μένει ως έχει.

Παράδειγμα 1

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

a1 =
     1     2     3


>> a2=[4;5;6]

a2 =
   4
   5
   6


>> b1=[7 8 9]

b1 =
   7     8     9
```

```
>> b2=[10;11;12]

b2 =
   10
   11
   12


>> a1*b1

??? Error using ==> mtimes
Inner matrix dimensions must agree.
```

Παρατήρηση

❖ Επειδή τα 2 διανύσματα a1, b1 είναι γραμμές, θα πρέπει να αντιστραφεί το b1 :

Παράδειγμα 2

```
>> a1*b1'

ans =
          50
```

❖ Το εσωτερικό τους γινόμενο είναι 1*7 + 2*8 + 3*9 = 7 + 16 + 27 = 50

Παράδειγμα 3

```
>> a1*b2

ans =
          68
```

Παράδειγμα 4

```
>> b2*a2

??? Error using ==> mtimes
Inner matrix dimensions must agree.
```

Παρατήρηση

❖ Επειδή τα 2 διανύσματα b2, a2 είναι στήλες, θα πρέπει να αντιστραφεί το b2 :

Παράδειγμα 5

```
>> b2'*a2

ans =
          167
```

Παρατήρηση

- Μπορούμε να χρησιμοποιήσουμε και τη συνάρτηση **dot** για τον υπολογισμό του εσωτερικού γινομένου, όπου δεν έχει σημασία αν τα διανύσματα είναι γραμμές ή στήλες :

Παράδειγμα 6

```
>> dot(a1,a2)

ans =
          32
```

Παράδειγμα 7

```
>> dot(a1,b1)

ans =
          50
```

Παράδειγμα 8

```
>> dot(a2,b2)

ans =
          167
```

3.2.5 Συναρτήσεις Διανυσμάτων

Το Matlab διαθέτει τις παρακάτω συναρτήσεις, οι οποίες έχουν σαν όρισμα ένα διάνυσμα :

- Τη συνάρτηση **length**, η οποία επιστρέφει το μήκος (αριθμό στοιχείων) του διανύσματος.

Παράδειγμα

```
>> a = [1 4 8];
```

```
>> n = length(a)

n =
          3
```

- Τη συνάρτηση **max**, η οποία επιστρέφει την τιμή του μεγίστου στοιχείου του διανύσματος.

Παράδειγμα

```
>>  max_el = max(a)

max_el =
          8
```

Παρατήρηση

- Αν θέλουμε η συνάρτηση **max** να μας επιστρέψει και τη θέση του μεγίστου στοιχείου, χρησιμοποιούμε μια ακόμη μεταβλητή για τη θέση και τις βάζουμε σε αγκύλες, όπως φαίνεται στο παράδειγμα που ακολουθεί :

Παράδειγμα

```
>> [max_el thesi_max] = max(a)

max_el =
                8

thesi_max =
                3
```

- Τη συνάρτηση **min**, η οποία επιστρέφει την τιμή του ελαχίστου στοιχείου του διανύσματος.

Παράδειγμα

```
>> min_el = min(a)

min_el =
                1
```

Παρατήρηση

- Αν θέλουμε η συνάρτηση **min** να μας επιστρέψει και τη θέση του ελαχίστου στοιχείου, χρησιμοποιούμε μια ακόμη μεταβλητή για τη θέση και τις βάζουμε σε αγκύλες, όπως φαίνεται στο παράδειγμα που ακολουθεί :

Παράδειγμα

```
>> [min_el thesi_min] = min(a)

min_el =
     1

thesi_min =
          1
```

- Τη συνάρτηση **sum**, η οποία επιστρέφει το άθροισμα των στοιχείων του διανύσματος.

Παράδειγμα

```
>>  a = [1 4 8];


>> sum(a)

ans =
    13
```

- Τη συνάρτηση **prod**, η οποία επιστρέφει το γινόμενο των στοιχείων του διανύσματος.

Παράδειγμα

```
>> prod(a)

ans =
    32
```

- Τη συνάρτηση **norm**, η οποία επιστρέφει την ευκλείδια νόρμα του διανύσματος, δηλαδή την τετραγωνική ρίζα του αθροίσματος των στοιχείων του διανύσματος υψωμένα στο τετράγωνο.

Παράδειγμα

```
>> n = norm(a)

n =
    9 ( = η ρίζα του 1^2 + 4^2 + 8^2 = 1 + 16 + 64 = 81  )
```

- Τη συνάρτηση **sort**, η οποία επιστρέφει τα στοιχεία του διανύσματος ταξινομημένα.

Παράδειγμα

```
>>  a = [1 3 -2]

a =
          1    3   -2

>> sort(a)

ans =
         -2    1    3
```

- Τη συνάρτηση **find**, η οποία επιστρέφει τις θέσεις των στοιχείων του διανύσματος τα οποία ικανοποιούν μια συνθήκη.

Παράδειγμα 1

```
>>  a = [1 3 -2];


>>  find(a>0)

ans =
          1    2
```

επειδή τα στοιχεία a(1) = 1 και a(2) = 3 είναι μεγαλύτερα απ' το μηδέν.

Παράδειγμα 2

```
>>  find(a==-2)

ans =

   3
```

επειδή το στοιχείο a(3) = -2.

Παρατήρηση

- Οι βασικές μαθηματικές συναρτήσεις του Πίνακα 3, αν χρησιμοποιηθούν σε διανύσματα, έχουν σαν αποτέλεσμα τη δημιουργία ενός νέου διανύσματος με στοιχεία το αποτέλεσμα της εφαρμογής της συνάρτησης σε κάθε στοιχείο του αρχικού διανύσματος.

Παράδειγμα

```
>>  a = [1 2 3]

a =
            1    2    3

>> sqrt(a)

ans =

   1.0000    1.4142    1.7321
```

3.3 Δημιουργία Πινάκων (Help: MATLAB->Programming->Data Structures)

Ένας πίνακας 2 διαστάσεων μπορεί να πάρει τιμές με τους παρακάτω τρόπους :

1ος Τρόπος

Με ανάθεση τιμών, όπως στη C.

Παράδειγμα

```
>> A = [1, 2, 3; 4, 5, 6]
```

Το αποτέλεσμα της προηγούμενης εντολής είναι η δημιουργία ενός πίνακα 2x3 (2 γραμμές, 3 στήλες) :

```
A =

   1    2    3
   4    5    6
```

Παρατηρήσεις

- Το ίδιο αποτέλεσμα θα είχαμε με την εντολή :

```
>> A = [1 2 3
 4 5 6]
```

ή με την εντολή :

```
>> A = [1 2 3;4 5 6]
```

❖ Μπορούμε να προσπελάσουμε το στοιχείο (i,j) του πίνακα A γράφοντας A(i,j). Οι πίνακες πρέπει να αποτελούνται από γραμμές του ιδίου μήκους, αλλιώς έχουμε σφάλμα.

2ος Τρόπος

Δημιουργώντας αριθμούς στη σειρά δίνοντας αρχή, τέλος, και προαιρετικά το βήμα, όπως και στα διανύσματα :

Παράδειγμα

```
>>  A = [1:3;4:6]

A =

    1    2    3
    4    5    6
```

3ος Τρόπος

Με τις συναρτήσεις **rand**, **randn**, **zeros**, **ones** :

Παράδειγμα 1

```
>>  A = rand(2,3)  % Δημιουργεί τον πίνακα A με 2 γραμμές και 3 στήλες και τυχαίες τιμές
                    στο (0,1).
A =

    0.8147    0.1270    0.6324
    0.9058    0.9134    0.0975
```

Παράδειγμα 2

```
>>  A = rand(2,2)      % Δημιουργεί τον πίνακα A με 2 γραμμές και 2 στήλες και τυχαίες
                        τιμές στο (0,1).
A =

    0.1576    0.9572
    0.9706    0.4854
```

Παρατήρηση

❖ Το ίδιο αποτέλεσμα θα είχαμε με την εντολή :

>> A = rand(2)

Παράδειγμα 3

>> A = randn(2) % Δημιουργεί τον πίνακα A με 2 γραμμές και 2 στήλες και τυχαίες τιμές με Γκαουσσιανή κατανομή.

A =

1.4090 0.6715
1.4172 -1.2075

Παράδειγμα 4

>> A = zeros (3,3) % Δημιουργεί τον πίνακα A με 3 γραμμές και 3 στήλες και τιμές 0

Παρατήρηση

❖ Το ίδιο αποτέλεσμα θα είχαμε με την εντολή :

>> A = zeros (3);

Παράδειγμα 5

>> x = ones (3,4); % Δημιουργεί τον πίνακα A με 3 γραμμές και 4 στήλες και τιμές 1

Παρατήρηση

❖ Σημειώστε ότι το MATLAB επιτρέπει επίσης την δημιουργία 3- διάστατων, 4- διάστατων πινάκων, κ.λ.π..

3.4 Επεξεργασία Πινάκων (Help: MATLAB->Functions by category->Programming and Data Type->Operators and Operations)

Το MATLAB προσφέρει βασικές λειτουργίες δημιουργίας και επεξεργασίας πινάκων. Οι πράξεις μεταξύ πινάκων και αριθμών είναι απλούστατες και χρησιμοποιούν τους τελεστές των βασικών πράξεων :

- Πρόσθεση αριθμού σε Πίνακα :

```
>> A = [1 2 3;4 5 6]

A =
     1    2    3
     4    5    6

>> A+2

ans =
     3    4    5
     6    7    8
```

- Αφαίρεση αριθμού από Πίνακα :

```
>> A-2

ans =
    -1    0    1
     2    3    4
```

- Πολλαπλασιασμός σταθεράς με Πίνακα :

```
>> A*2

ans =
     2    4    6
     8   10   12
```

- Διαίρεση Πίνακα με σταθερά :

```
>> A/2

ans =
    0.5000    1.0000    1.5000
    2.0000    2.5000    3.0000
```

- Ύψωση Πίνακα σε δύναμη :

```
>> A^2
   ??? Error using ==> mpower
   Matrix must be square.
```

Παρατηρήσεις

❖ Η εντολή A^2 ισοδυναμεί με το A*A, το οποίο δεν μπορεί να γίνει, γιατί ο Πίνακας έχει διάσταση 2x3.

❖ Για να υψωθεί κάθε στοιχείο του Πίνακα A στο τετράγωνο, θα πρέπει ο τελεστής '^' να χρησιμοποιηθεί με την τελεία '.', δηλαδή :

```
>> A.^2

ans =
     1     4     9
    16    25    36
```

- Ύψωση Τετραγωνικού Πίνακα σε δύναμη :

Αν ο Πίνακας είναι Τετραγωνικός (ίδιος αριθμός γραμμών-στηλών), ο τελεστής «^» μπορεί να χρησιμοποιηθεί χωρίς την τελεία :

Παράδειγμα

```
>> B = [1:2;3:4]

   B =
         1     2
         3     4

>> B^2

   ans =
         7    10
        15    22
```

3.4.1 Ανάστροφος Πίνακα 2 Διαστάσεων (*Help:Search For: Transpose)*

Όπως και στα διανύσματα, με τον τελεστή του αναστρόφου οι γραμμές του Πίνακα γίνονται στήλες και οι στήλες του γραμμές :

Παράδειγμα

```
>>  A = [1 2 3;4 5 6]

A =

   1    2    3
   4    5    6


>>  A'

ans =

   1    4
   2    5
   3    6
```

3.4.2 Πράξεις μεταξύ Πινάκων και Διανυσμάτων

Για να γίνει ο πολλαπλασιασμός ενός Πίνακα με ένα διάνυσμα θα πρέπει οι γραμμές του διανύσματος να είναι όσες και οι στήλες του Πίνακα. Αν δημιουργήσουμε τους πίνακες A και x :

```
>>  A = [1 2 3;4 5 6]

A =

   1    2    3
   4    5    6

>> x = [7 8 9]

x =

   7    8    9
```

και χρησιμοποιήσουμε την εντολή :

```
>>  A*x
```

θα πάρουμε το μήνυμα λάθους

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

γιατί ο Πίνακας A είναι 2x3 και το διάνυσμα x είναι 1x3, οπότε δεν μπορεί να γίνει ο πολλαπλασιασμός. Για να γίνει ο πολλαπλασιασμός θα πρέπει οι στήλες του πίνακα A να είναι όσες και οι γραμμές του διανύσματος x, οπότε πρέπει να χρησιμοποιήσουμε την εντολή :

```
>> A*x'
```

και το αποτέλεσμα που παίρνουμε είναι ο πίνακας :

```
ans =

   50
  122
```

Ο Πίνακας A είναι 2x3 και το διάνυσμα x είναι 3x1, οπότε μπορεί να γίνει ο πολλαπλασιασμός και ο πίνακας που προκύπτει είναι 2x1.

3.4.3 Πράξεις μεταξύ Πινάκων

3.4.3.1 Πρόσθεση - Αφαίρεση Πινάκων

Για να γίνει ο πρόσθεση ή αφαίρεση ενός Πίνακα με ή από ένα άλλον Πίνακα θα πρέπει οι 2 Πίνακες να έχουν τις ίδιες διαστάσεις. Αν δημιουργήσουμε τους πίνακες A, B και C :

```
>> A = [1 2 3;4 5 6]

A =
    1    2    3
    4    5    6

>> B = [7 8 9;10 11 12]

B =
    7    8    9
   10   11   12


>> C = [1 2;3 4]

C =
    1    2
    3    4
```

και χρησιμοποιήσουμε την εντολή :

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

παίρνουμε το παραπάνω μήνυμα λάθους, γιατί ο Πίνακας Α είναι 2x3, ενώ ο Πίνακας Β είναι 2x2. Η πρόσθεση μπορεί να γίνει μεταξύ των Πινάκων Α και Β που είναι 2x3, αν χρησιμοποιήσουμε την εντολή :

```
>> A + B

ans =

    8    10    12
   14    16    18
```

ενώ η αφαίρεση Πινάκων γίνεται με την εντολή :

```
>>  A - B

ans =

   -6    -6    -6
   -6    -6    -6
```

3.4.3.2 Πολλαπλασιασμός Πινάκων

Για να γίνει ο πολλαπλασιασμός ενός Πίνακα Α με ένα άλλον Πίνακα Β θα πρέπει οι γραμμές του Πίνακα Β να είναι όσες και οι στήλες του Πίνακα Α. Αν π.χ. χρησιμοποιήσουμε την εντολή :

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

παίρνουμε το παραπάνω μήνυμα λάθους, γιατί ο Πίνακας Α είναι 2x3 και ο Πίνακας Β είναι 2x3. Ο πολλαπλασιασμός μπορεί να γίνει μεταξύ των Πινάκων Α και Β’ που είναι 2x3 και 3x2, αν χρησιμοποιήσουμε την εντολή :

```
>>  A*B'

ans =

    50    68
   122   167
```

Παρατήρηση

- Αν στον τελεστή «*» χρησιμοποιήσουμε την τελεία, θα προκύψει ένας νέος πίνακας με στοιχεία το αποτέλεσμα του γινομένου των αντίστοιχων στοιχείων των 2 Πινάκων. Οι 2 Πίνακες όμως πρέπει να έχουν τις ίδιες διαστάσεις.

Παράδειγμα

```
>> A.*B

ans =

    7   16   27
   40   55   72
```

3.4.3.3 Διαίρεση Πινάκων

Αν χρησιμοποιήσουμε τον τελεστή «/» στους Πίνακες Α, Β

```
>> X = A/B

    X =
       3.0000   -2.0000
       2.0000   -1.0000
```

θα πάρουμε τον Πίνακα Χ που είναι η λύση του συστήματος εξισώσεων Χ*Β = Α. Με την «.» όμως πριν τον τελεστή «/», θα πάρουμε έναν Πίνακα ίδιων διαστάσεων που τα στοιχεία του είναι το πηλίκο του κάθε στοιχείου του Πίνακα Α με το αντίστοιχο στοιχείο του Πίνακα Β :

```
>> A./B

ans =
   0.1429   0.2500   0.3333
   0.4000   0.4545   0.5000
```

3.4.3.4 Επιλογή Τμήματος Πίνακα

Έστω ότι δημιουργούμε τον πίνακα A με την εντολή :

```
>>  A = [1, 2, 3, 4; 5, 6, 7, 8; 9, 10, 11, 12]
```

με αποτέλεσμα την εμφάνιση του πίνακα A(3x4) :

```
A  =
        1    2    3    4
        5    6    7    8
        9   10   11   12
```

Μπορούμε να επιλέξουμε τμήματα του πίνακα όπως φαίνεται στα παραδείγματα που ακολουθούν :

Παράδειγμα 1

```
>> A1 = A(2,1:3)        % Επιλέγουμε μόνο τη γραμμή 2 και τις στήλες 1,2,3

A1  =
        5    6    7
```

Παράδειγμα 2

```
>> A2 = A(2,1:4)        % Επιλέγουμε μόνο τη γραμμή 2 και όλες τις στήλες

A2  =
        5    6    7    8
```

Παράδειγμα 3

```
>> A3 = A(2,:)          % Επιλέγουμε μόνο τη γραμμή 2 και όλες τις στήλες

A3  =
        5    6    7    8
```

Παράδειγμα 4

```
>> A4 = A(:,1:2)        % Επιλέγουμε όλες τις γραμμές και τις στήλες 1,2

A4  =
        1    2
        5    6
        9   10
```

Παράδειγμα 5

```
>> A5 = A(:,1)          % Επιλέγουμε όλες τις γραμμές και μόνο τη στήλη 1

A5  =
        1
        5
        9
```

Παράδειγμα 6

```
>> A6 = A(1:2,1:2)      % Επιλέγουμε τις γραμμές 1 έως 2 και τις στήλες 1 έως 2

A6  =
        1    2
        5    6
```

3.4.3.5 Συνένωση Πινάκων

Μπορούμε αντίστοιχα να συνενώσουμε 2 πίνακες, αρκεί να συμφωνεί ο αριθμός των γραμμών ή των στηλών τους.

Παράδειγμα 1

```
>>  A = [1 2 3;4 5 6]

A =

   1    2    3
   4    5    6


>> B = [7 8 9;10 11 12;13 14 15]

B =

    7    8    9
   10   11   12
   13   14   15


>>  [A;B]
```

```
ans =

    1    2    3
    4    5    6
    7    8    9
   10   11   12
   13   14   15
```

Οι Πίνακες Α, Β έχουν τον ίδιο αριθμό στηλών, οπότε ο νέος Πίνακας περιέχει τις 2 γραμμές του Πίνακα Α και τις 3 γραμμές του Πίνακα Β.

Παράδειγμα 2

```
>>  [A,B]

??? Error using ==> horzcat
CAT arguments dimensions are not consistent.
```

Δεν μπορεί να γίνει η συνένωση, γιατί οι Πίνακες Α, Β έχουν διαφορετικό αριθμό γραμμών.

Παράδειγμα 3

```
>> [A,B(1:2,:)]

ans =

    1    2    3    7    8    9
    4    5    6   10   11   12
```

Οι Πίνακες Α, Β(1:2,:) έχουν τον ίδιο αριθμό γραμμών, οπότε ο νέος Πίνακας περιέχει τις 3 στήλες του Πίνακα Α και τις 3 στήλες του Πίνακα Β(1:2,:).

3.4.4 Πίνακες Χαρακτήρων (strings) - cells

Τα στοιχεία ενός διανύσματος ή ενός πίνακα δεν είναι απαραίτητα αριθμοί. Μπορεί να είναι χαρακτήρες αφού οι χαρακτήρες αναπαριστώνται εσωτερικά με τους ASCII κωδικούς τους, δηλαδή με αριθμούς. Έτσι τα strings δεν είναι τίποτα άλλο παρά διανύσματα από χαρακτήρες. Μπορεί να δηλωθεί δίνοντας τον κάθε χαρακτήρα ή όλους τους χαρακτήρες μαζί, όπως φαίνεται στα παραδείγματα που ακολουθούν :

Παράδειγμα 1

```
>>  a = ['M','a','t','l','a','b']
```

a =
Matlab

Παράδειγμα 2

>> b = ['Matlab']
b =
Matlab

Παρατήρηση 1

❖ Μπορούμε να προσπελάσουμε κάποιον χαρακτήρα με το όνομα του πίνακα και τη θέση του χαρακτήρα στο string, όπως φαίνεται στο παράδειγμα που ακολουθεί :

>> b(1)

ans =
M

Παρατήρηση 2

❖ Με τον ίδιο τρόπο μπορούμε να δημιουργήσουμε και έναν πίνακα χαρακτήρων 2 διαστάσεων, αρκεί το κάθε string να περιέχει τον ίδιο αριθμό χαρακτήρων :

Παράδειγμα

>> A = ['Hello';'There']

A =
Hello
There

❖ Στο παραπάνω παράδειγμα, ο πίνακας A έχει διαστάσεις 2x5.

Παρατήρηση 3

❖ Δεν μπορούμε να δημιουργήσουμε έναν πίνακα με strings διαφορετικού μήκους το καθένα. Στην περίπτωση αυτή χρησιμοποιούμε τα «κελιά» - “cells” όπως στο παράδειγμα που ακολουθεί :

>> B = {'Good', 'Morning'}

B =
'Good' 'Morning'

❖ Η μεταβλητή B είναι ένας πίνακας με διαστάσεις 1x4 όπου τα στοιχεία του είναι τύπου cell. Τα cells δεν είναι strings. Αν θέλουμε να χρησιμοποιήσουμε, π.χ, το string 'Good' που αντιστοιχεί στο B(1) γράφουμε B{1} (προσέξτε τη διαφορά στις αγκύλες).

```
>> B{1}

ans =
        'Good'
```

❖ Αν θέλουμε να χρησιμοποιήσουμε, π.χ, τον πρώτο χαρακτήρα του string 'Good' γράφουμε B{1} (1) :

```
>> B{1}(1)

ans =
        G
```

3.5 Συναρτήσεις σε Πίνακες

Μερικές από τις συναρτήσεις που μπορούν να χρησιμοποιηθούν σε Πίνακες 2 διαστάσεων είναι οι **size**, **max**, **min**, **sum**, **prod**.

- Η συνάρτηση **size** επιστρέφει τον αριθμό γραμμών και στηλών του Πίνακα :

```
>> A = [1 2 3;4 5 6]

>> [lin col] = size(A)

lin =

   2

col =

   3
```

- Η συνάρτηση **max** επιστρέφει ένα διάνυσμα γραμμή που το κάθε στοιχείο του είναι το μέγιστο στοιχείο κάθε στήλης του αρχικού Πίνακα :

```
>> max(A)
```

ans =

4 5 6

- Η συνάρτηση **min** επιστρέφει ένα διάνυσμα γραμμή που το κάθε στοιχείο του είναι το ελάχιστο στοιχείο κάθε στήλης του αρχικού Πίνακα :

>> min(A)

ans =

1 2 3

- Η συνάρτηση **sum** επιστρέφει το άθροισμα των στοιχείων της κάθε στήλης του Πίνακα.

Παράδειγμα

>> sum(A)

ans =

5 7 9

- Η συνάρτηση **prod** επιστρέφει το γινόμενο των στοιχείων της κάθε στήλης του Πίνακα.

Παράδειγμα

>> prod(A)

ans =

4 10 18

3.6 Δομές (Structures) (Help: MATLAB->External Interfaces->MATLAB Interface to Generic DLL->Data Conversion->Structures)

Το **struct** είναι ένας τύπος δεδομένων που μπορεί να δημιουργηθεί από τον χρήστη. Η struct() παίρνει ως ορίσματα το όνομα του πεδίου και δίπλα την τιμή του πεδίου. Τα πεδία

είναι απλά ονόματα για τις τιμές. Οι τιμές μπορεί να είναι και πίνακες. Η παρακάτω εντολή είναι ένα παράδειγμα δημιουργίας ενός struct το οποίο περιέχει τρεις μεταβλητές την Course, την color και την numbers.

```
>> s = struct('Course', {'Machine','Learning'}, 'color', {'red'}, 'numbers', {3 4})

s =

1x2 struct array with fields:
   Course
   color
   numbers
```

Για να προσπελάσουμε μία τιμή από ένα struct γράφουμε το εξής:

```
>>  s.Course

ans =

Machine
ans =

Learning

>>  s.color

ans =

red

ans =

red

>>  s.numbers

ans =

        3.00

ans =

        4.00
```

ΚΕΦΑΛΑΙΟ 4

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΣΤΟ MATLAB

4.1 Scripts (Help: Search for: script)

Ένα script είναι ένα εξωτερικό αρχείο που περιέχει μία σειρά από εντολές MATLAB. Με την πληκτρολόγηση του ονόματος του αρχείου στο command line εκτελούνται όλες οι εντολές του script με την σειρά. Τα scripts έχουν επέκταση ονόματος αρχείου .m και συνήθως ονομάζονται και M-Files. Τα script μπορούν να χρησιμοποιήσουν υπάρχοντα δεδομένα από το workspace ή να δημιουργήσουν νέα. Αν και τα script δεν επιστρέφουν τιμές, όποια μεταβλητή δημιούργησαν παραμένει στο workspace.

4.2 Matlab Search Path (Help: MATLAB->Programming->Programming Tips->MATLAB Path)

Το Matlab search path είναι τα directories στα οποία ψάχνει το MATLAB για υπάρχουσες μεταβλητές, functions, scripts, M-Files, αρχεία και άλλες δομές του MATLAB. Η σειρά με την οποία ψάχνει είναι η παρακάτω :

1. Μεταβλητές
2. Subfunctions
3. Private Functions
4. Class Constructors
5. Overloaded Methods
6. M-File που βρίσκεται στον παρόντα κατάλογο
7. Τέλος, M-File που βρίσκεται στο path ή build-in functions του MATLAB

Παράδειγμα

Έστω ότι καλούμε ένα M-File με το όνομα trainnet.m.

>> trainnet

Αφού εκτελέσουμε αυτή την εντολή το MATLAB αρχικά ψάχνει στο search path για μία μεταβλητή με το όνομα trainnet, αν δεν βρει τότε ψάχνει για μία subfunction κ.λπ..

Το path είναι ο παρών φάκελος (current directory), στον οποίο ψάχνει το MATLAB. Στην Εικόνα 1.1 φαίνεται η θέση του current directory.

Για να εισάγουμε ένα φάκελο στο search path μπορούμε να το κάνουμε με δύο τρόπους, είτε από το toolbar File->Set Path είτε από το command window με την εντολή addpath.

```
>> addpath('c:\matlab_projects')
```

Τέλος για να δούμε όλο το search path του MATLAB πληκτρολογούμε την εντολή **matlabpath** στο command window.

4.3 Είσοδος Τιμών από το Πληκτρολόγιο (*Help:Search For: Input)*

Για να εισάγουμε μία τιμή σε μία μεταβλητή από το πληκτρολόγιο χρησιμοποιούμε την εντολή input. Σαν όρισμα της input είναι το μήνυμα που θέλουμε να εμφανιστεί πριν την είσοδο.

Παράδειγμα

```
>> n = input('Δώσε αριθμό : ')
```

Αν στην εμφάνιση του μηνύματος εισάγουμε τον αριθμό 5, θα αποθηκευθεί στη μεταβλητή n.

```
Δώσε αριθμό : 5

n =

    5
```

Αν θέλουμε να αποθηκευθεί η είσοδος σε μεταβλητή τύπου string, θα πρέπει να συμπεριλάβουμε μετά το μήνυμα και το όρισμα ‘s’.

Παράδειγμα

```
>> chr = input('Θα συνεχίσεις Y/N : ','s')
Θα συνεχίσεις Y/N : Y

chr =

Y
```

4.4 Εμφάνιση Τιμών Μεταβλητών (*Help:Search For: fprintf, disp)*

Για να εμφανίσουμε την τιμή μίας μεταβλητής, διανύσματος ή Πίνακα χρησιμοποιούμε τις εντολές **disp** και **fprintf**.

- Με την εντολή **disp** περνάμε σαν όρισμα τη μεταβλητή. Αν πρόκειται για πραγματικούς αριθμούς, εμφανίζονται στρογγυλοποιημένοι σε 4 δεκαδικά ψηφία.

Παράδειγμα 1

```
>>  n = 5

n =

    5


>> disp(n)
    5
```

Παράδειγμα 2

```
>>  p = 3.14159265

p =
          3.14159265


>> disp(p)
   3.1416
```

Παράδειγμα 3

```
>>  a = [1 3 -2]

a =
    1     3    -2


>> disp(a)
    1     3    -2
```

Παράδειγμα 4

```
>> A = [1 2 3;4 5 6]

A =
   1   2   3
   4   5   6

>> disp(A)
   1   2   3
   4   5   6
```

Παρατήρηση 1

- Μπορούμε να εμφανίσουμε τις τιμές των μεταβλητών με 15 δεκαδικά ψηφία, αν χρησιμοποιήσουμε την εντολή **format long** (αν τα ψηφία είναι λιγότερα, γεμίζει τις υπόλοιπες θέσεις με μηδενικά, διαφορετικά, κάνει στρογγύλευση).

Παράδειγμα 5

```
>> format long


>> disp(pi)
  3.141592653589793
```

Παρατήρηση 2

- Οι τιμές των μεταβλητών εμφανίζονται με 4 δεκαδικά ψηφία, γιατί by default ενεργοποιείται η εντολή **format short**.

Παράδειγμα 6

```
>> format short


>> disp(pi)
   3.1416
```

Παρατήρηση 3

- Το **pi** και το **eps** είναι δεσμευμένες λέξεις για τις τιμές του **π** και του **αριθμού μηχανής** ή της σταθεράς σχετικής ακρίβειας που είναι 2^-52 για πραγματικούς αριθμούς διπλής ακρίβειας.

Παράδειγμα 7

```
>>  format long

>> disp(eps)
   2.220446049250313e-016          % εμφάνιση τιμής σε επιστημονική μορφή
```

Παρατήρηση 4

- Με την εντολή **disp** μπορούμε να εμφανίζουμε και μηνύματα, τα οποία περικλείονται σε απλές αποστρόφους.

Παράδειγμα 8

```
>>  disp('Pinakas')
Pinakas
```

- Με την εντολή **fprintf** περνάμε σαν όρισμα τη μεταβλητή και τον αντίστοιχο προσδιοριστή, όπως και στη Java και C, με τον οποίο καθορίζουμε και το πλήθος των δεκαδικών ψηφίων που θα εμφανιστούν.

Παράδειγμα 1

```
>>  fprintf('%12.8f\n', pi)
 3.14159265
```

Παράδειγμα 2

```
>>  a = [1 3 -2]

a =
    1    3   -2

>> fprintf('%12.8f', a)
 1.00000000  3.00000000 -2.00000000>>
```

Παρατήρηση 1

- Επειδή υπάρχει μόνο ένας προσδιοριστής %12.8f στην εντολή, χρησιμοποιείται τόσες φορές όσα και τα στοιχεία του διανύσματος. Αν χρησιμοποιήσουμε και αλλαγή γραμμής, τα στοιχεία θα εμφανιστούν σε στήλη :

Παράδειγμα 3

```
>> fprintf('%12.8f\n',a)
 1.00000000
 3.00000000
-2.00000000
```

Παράδειγμα 4

```
>>  A = [1 2 3;4 5 6]

A =

   1   2   3
   4   5   6

>> fprintf('%12.8f', A)
 1.00000000  4.00000000  2.00000000  5.00000000  3.00000000  6.00000000>>
```

Παράδειγμα 5

```
fprintf('%12.8f\n', A)
 1.00000000
 4.00000000
 2.00000000
 5.00000000
 3.00000000
 6.00000000
```

Παρατήρηση 2

- Τα στοιχεία του Πίνακα Α και στις 2 περιπτώσεις εμφανίζονται σε μια γραμμή ή σε μια στήλη αντίστοιχα διατρέχοντας τα στοιχεία του πίνακα με τη σειρά και ανά στήλες. Αν θέλουμε να εμφανιστούν όλα τα στοιχεία ανά γραμμές και μετά να αλλάξουμε γραμμή θα πρέπει να χρησιμοποιήσουμε 3 προσδιοριστές και τον ανάστροφο Πίνακα Α'.

Παράδειγμα 6

```
>> fprintf('%12.8f %12.8f %12.8f\n', A')
 1.00000000   2.00000000   3.00000000
 4.00000000   5.00000000   6.00000000
```

Παρατήρηση 3

- Μπορούμε να εμφανίσουμε και μηνύματα στην ίδια εντολή fprintf.

Παράδειγμα 7

```
>>  fprintf('π = %12.8f\n', pi)
π =   3.14159265
```

Παρατήρηση 4

- Με την εντολή fprinf μπορούμε να εμφανίσουμε και τις τιμές ακεραίων αριθμών με τον προσδιοριστή %d .

Παράδειγμα 8

```
>> a = 12;

>> fprintf('%5d\n',a)
  12
```

Παρατήρηση 5

- Με την εντολή fprinf μπορούμε να εμφανίσουμε και τις τιμές αλφαριθμητικών χαρακτήρων, string με τον προσδιοριστή %s και χαρακτήρων με τον προσδιοριστή %c.

Παράδειγμα 9

```
>> onoma = 'kostas';

>> fprintf('%s\n',onoma)
kostas
```

Παράδειγμα 10

```
>> fprintf('%c\n',onoma(1))
K
```

4.5 Η Εντολή if-else (Help: MATLAB->Programming->Basic Program Components->Program Control Statements)

Η σύνταξη της εντολής **if** είναι παρόμοια με αυτή της C, Java με την προσθήκη ενός end στο τέλος της εντολής.

Παράδειγμα

```
>> age = input('Δώσε Ηλικία : ')
>> if (age<=18)
        fprintf('Νέος\n');
elseif (age<=65)
        fprintf('Ενήλικας\n');
else
        fprintf('Ηλικιωμένος\n');
end
```

- Παρατηρήστε ότι στην εντολή if-then-else αν έχουμε πάνω από δύο περιπτώσεις μπορούμε να χρησιμοποιήσουμε το elseif.

4.6 Η Εντολή switch (Help: MATLAB->Programming->Basic Program Components->Program Control Statements)

Η σύνταξη της εντολής **switch - case** είναι παρόμοια με αυτή της C, Java με την προσθήκη ενός end στο τέλος της εντολής, ενώ οι τιμές που θα παίρνει η μεταβλητή για κάθε περίπτωση, θα πρέπει να είναι ακέραιες ή χαρακτήρες.

Παράδειγμα

```
>> tekna = input('Δώσε Αριθμό Παιδιών >= 1 : ')

    switch (tekna)
    case 1
        fprintf('Απαλλαγή 2000 ΕΥΡΩ\n');
    case 2
        fprintf('Απαλλαγή 4000 ΕΥΡΩ\n');
    case 3
        fprintf('Απαλλαγή 6000 ΕΥΡΩ\n');
    otherwise
        fprintf('Απαλλαγή 10000 ΕΥΡΩ\n');
    end
```

Παρατήρηση

- Μπορούμε να έχουμε πολλαπλές επιλογές, οι οποίες πρέπει να βρίσκονται σε άγγιστρα. Π.χ. case {1,2,3}

4.7 Η Εντολή while (Help: MATLAB->Programming->Basic Program Components->Program Control Statements)

Η σύνταξη της εντολής **while** είναι παρόμοια με αυτή της C, Java με την προσθήκη ενός end στο τέλος της εντολής.

Παράδειγμα

```
i = 0;
a = 5;
while (i <= a)
        i = i + 1;
        fprintf( 'i =  %12.0f\n',  i);
end
```

Το παραπάνω τμήμα κώδικα θα έχει το παρακάτω αποτέλεσμα :

```
i =          1
i =          2
i =          3
i =          4
i =          5
```

4.8 Η Εντολή for (Help: MATLAB->Programming->Basic Program Components->Program Control Statements)

Η σύνταξη της εντολής **for** είναι παρόμοια με αυτή της C, Java με την προσθήκη ενός end στο τέλος της εντολής.

Παράδειγμα

```
for i = 1:5
        fprintf( 'i =  %12.0f\n',  i);
end
```

- Το παραπάνω τμήμα κώδικα θα έχει το ίδιο αποτέλεσμα με το προηγούμενο παράδειγμα.

Παρατήρηση 1

- Με την εντολή **continue** μπορούμε να αγνοήσουμε την εκτέλεση του κώδικα για κάποια τιμή στην επανάληψη.

Παράδειγμα

```
for i = 1:5
        if ( i == 3 )
          continue
        end;
        fprintf( 'i =  %12.0f\n',  i);
end;
```

❖ Το παραπάνω τμήμα κώδικα θα έχει το παρακάτω αποτέλεσμα :

```
i =     1
i =     2
i =     4
i =     5
```

Παρατήρηση 2

❖ Η διακοπή από ένα βρόχο μπορεί να γίνει με την εντολή **break**.

Παράδειγμα

```
sum = 0;
for i = 1:
        if ( sum > 20 )
          break
        end;
        sum = sum + i;
end;
fprintf( 'i =  %12.0f   sum = %12.0f\n',  i, sum);
```

❖ Το παραπάνω τμήμα κώδικα θα έχει το παρακάτω αποτέλεσμα :

```
i =     5   sum =       15
```

4.9 Λογικοί Τελεστές (Help: MATLAB->Programming->Basic Program Components->Operators)

Για την ένωση 2 λογικών εκφράσεων χρησιμοποιούμε τους παρακάτω λογικούς τελεστές :

Σύμβολο	Σύγκριση
|	Λογικό OR
&	Λογικό AND
||	Βραχυκυκλωμένο (Short-circuit) OR
&&	Βραχυκυκλωμένο (Short-circuit) AND

Πίνακας 4

Οι τελεστές τύπου *Short-circuit* χρησιμοποιούνται για να ενώσουν δύο λογικές εκφράσεις, π.χ. (a==2) || (b(2)>8). Η διαφορά από το κοινό OR | (ή AND &) είναι ότι η δεύτερη λογική έκφραση (b(2)>8) δεν υπολογίζεται αν δεν χρειάζεται, αν δηλαδή το αποτέλεσμα είναι γνωστό από την πρώτη κιόλας έκφραση. Π.χ. αν η πρώτη έκφραση είναι *true* τότε το λογικό OR θα δώσει *true* άσχετα από την τιμή της δεύτερης έκφρασης ενώ, αντίστοιχα, αν η πρώτη έκφραση είναι *false* τότε το λογικό AND θα δώσει *false* άσχετα από την τιμή της δεύτερης έκφρασης.

4.10 Συναρτήσεις (Functions) (Help: MATLAB->Getting Started->Programming->Scripts and Functions-> Functions)

Οι functions είναι M-Files πού δέχονται δεδομένα ως είσοδο και επιστρέφουν δεδομένα ως έξοδο. **Το όνομα του M-File και της function πρέπει να είναι ίδιο**. Οι functions πρέπει να βρίσκονται στο ίδιιο workspace με το πρόγραμμα που τις καλεί.

Η πρώτη γραμμή του M-File που περιέχει τον κώδικα της συνάρτησης ξεκινά με την λέξη function. Έπειτα δίνουμε το όρισμα ή τα ορίσματα (παραμέτρους εξόδου) στις οποίες θα επιστρέψει το αποτέλεσμα '=' το όνομα της συνάρτησης και σε παρένθεση το όρισμα ή τα ορίσματα με τα οποία θα περάσουν οι τιμές στη συνάρτηση (παραμέτρους εισόδου).

Παράδειγμα 1

Ας υποθέσουμε πως γράψαμε ένα M-File με όνομα **testfun1.m**, το οποίο διαβάζει την τιμή μιας μεταβλητής **n** και δημιουργεί με τη συνάρτηση **randn n** τυχαίες τιμές. Το πρόγραμμα θα πρέπει να εφαρμόσει τη **Βηματική Συνάρτηση** σε κάθε στοιχείο του πίνακα **u** και να επιστρέψει το αποτέλεσμα στο πρόγραμμα μέσω του πίνακα **v** με την κλήση της function **step01.m**.

Ο κώδικας του προγράμματος **testfun1.m** θα μπορούσε να περιέχει τα εξής :

```
clear;
clc;
n = input ( 'Δώσε n : ');
u = randn(1,n);
disp('u = ');
disp(u);
v = step01(u, n);
disp('v = ');
disp(v);
```

Ο κώδικας της συνάρτησης **step01.m** θα μπορούσε να περιέχει τα εξής :

```
function v = step01(u, n);
for i = 1 : n
    if ( u(i) > 0 )
       v(i) = 1;
    else
       v(i) = 0;
    end %if
end;  %for
```

Με το τρέξιμο του προγράμματος θα δούμε τα παρακάτω :

```
Δώσε n : 5
u =
  1.4384   0.3252  -0.7549   1.3703  -1.7115
```

```
v =
   1   1   0   1   0
```

Παρατήρηση 1

- Αν θέλουμε η **Βηματική Συνάρτηση** να εφαρμόζεται κάθε φορά στο στοιχείο **u(i)** του πίνακα **u** και και να επιστρέψει το αποτέλεσμα στο πρόγραμμα στο στοιχείο **v(i)** του πίνακα **v** με την κλήση της function **step01.m**, θα πρέπει να τροποποιήσουμε τον κώδικα του προγράμματος και της συνάρτησης.

Ο κώδικας του προγράμματος **testfun2.m** θα πρέπει να περιέχει τα εξής :

```
clear;
clc;
n = input ( 'Δώσε n : ');
u = randn(1,n);
disp('u = ');
disp(u);
for i = 1 : n
    v1 = step01(u, i);
    v(i) = v1;
end %for
disp('v = ');
disp(v);
```

Ο κώδικας της συνάρτησης **step01.m** θα μπορούσε να περιέχει τα εξής :

```
function v = step01(u, i);
if ( u(i) > 0 )
       v = 1;
else
       v = 0;
end %if
```

Με το τρέξιμο του προγράμματος θα δούμε τα παρακάτω :

```
Δώσε n : 5
u =
  0.0859  -1.4916  -0.7423  -1.0616   2.3505

v =
   1   0   0   0   1
```

Παρατήρηση 2

Για να δηλώσουμε μία function με παραπάνω από δύο ορίσματα εξόδου θα πρέπει να τα βάλουμε σε αγγύλες. Θα μπορούσαμε να τροποποιήσουμε τον κώδικα του προγράμματος **testfun1.m** και της συνάρτησης **step01.m** ώστε να επιστρέφει εκτός από τον πίνακα **v** και έναν πίνακα με τις θέσεις των στοιχείων του πίνακα **v** που έχουν την τιμή 1.

Ο κώδικας του προγράμματος **testfun3.m** θα μπορούσε να περιέχει τα εξής :

```
clear;
clc;
n = input ( 'Δώσε n : ');
u = randn(1,n);
disp('u = ');
disp(u);
[v, one] = step01(u, n);
disp('v = ');
disp(v);
disp('ones = ');
disp(one);
```

Ο κώδικας της συνάρτησης **step01.m** θα μπορούσε να περιέχει τα εξής :

```
function [v, one] = step01(u, n);
for i = 1 : n
    if ( u(i) > 0 )
       v(i) = 1;
    else
       v(i) = 0;
    end %if
end %for
one = find(v==1);
```

Με το τρέξιμο του προγράμματος θα δούμε τα παρακάτω :

```
Δώσε n : 5
u =
  -0.6156   0.7481  -0.1924   0.8886  -0.7648

v =
   0   1   0   1   0

ones =
   2   4
```

ΚΕΦΑΛΑΙΟ 5

ΑΡΧΕΙΑ ΚΕΙΜΕΝΟΥ ΣΤΟ MATLAB

5.1 Επεξεργασία Αρχείων Κειμένου (*Help:Search For: Writing Text Data, Reading Text Data, fscanf, fprintf)*

Το MATLAB δίνει την δυνατότητα εγγραφής και ανάγνωσης αρχείων κειμένου.

- Για να γίνει εγγραφή σε ένα αρχείο κειμένου θα πρέπει να ανοίξουμε το αρχείο, όπως φαίνεται στο παρακάτω παράδειγμα :

```
outf = fopen('test.txt','w');
```

Με την εντολή **fopen** δημιουργούμε το αρχείο με όνομα 'test.txt', στο οποίο δίνουμε το λογικό όνομα outfile. Ο προσδιοριστής 'w' χρησιμοποιείται για να δηλώσουμε ότι το αρχείο είναι για γράψιμο (write). Για να γράψουμε το περιεχόμενο μιας μεταβλητής στο αρχείο χρησιμοποιούμε την εντολή **fprintf** προσδιορίζοντας και το λογικό όνομα του αρχείου.

Παράδειγμα 1

```
>> onoma = 'kostas'

onoma =

kostas

>> fprintf(outf,'%s\n', onoma)

ans =

    7
```

Με την προηγούμενη εντολή το περιεχόμενο της μεταβλητής onoma γράφεται στην πρώτη γραμμή του αρχείου 'test.txt'. Το 6 σημαίνει ότι έχουν γραφεί 6 χαρακτήρες και μια αλλαγή γραμμής. Αν ανοίξουμε το αρχείο θα δούμε την παρακάτω εικόνα :

```
1  kostas
2
```

Με παρόμοιο τρόπο μπορούμε να γράψουμε και τις τιμές κάποιων ακέραιων ή πραγματικών μεταβλητών ή Πινάκων στο αρχείο, για όσο το έχουμε ανοιχτό.

Παράδειγμα 2

```
>> a = 5

a =

    5


>> b = 3.14

b =

   3.1400

>> fprintf(outf,'%d %f\n', a, b)

ans =

   11
```

Με την προηγούμενη εντολή το περιεχόμενο των μεταβλητών a και b γράφονται στη δεύτερη γραμμή του αρχείου 'test.txt'. Το 11 σημαίνει ότι έχουν γραφεί 10 χαρακτήρες και μια αλλαγή γραμμής. Αν ανοίξουμε το αρχείο θα δούμε την παρακάτω εικόνα :

```
1  kostas
2  5   3.140000
3
```

Παράδειγμα 3

```
>> pina = [1 2 3]

pina =

    1    2    3

>> fprintf(outf,'%f %f %f\n', pina);
```

Με την προηγούμενη εντολή το περιεχόμενο του διανύσματος pina γράφεται στην τρίτη γραμμή του αρχείου 'test.txt'. Αν ανοίξουμε το αρχείο θα δούμε την παρακάτω εικόνα :

```
1  kostas
2  5   3.140000
3  1.000000 2.000000 3.000000
4
```

Όταν τελειώσουμε με την εγγραφή δεδομένων στο αρχείο, θα πρέπει να το κλείσουμε με την εντολή :

fclose(outf);

- Για να γίνει ανάγνωση από ένα αρχείο κειμένου θα πρέπει να ανοίξουμε το αρχείο, όπως φαίνεται στο παρακάτω παράδειγμα :

```
inf = fopen('test.txt','r');
```

Με την εντολή **fopen** δίνουμε στο αρχείο με όνομα 'test.txt' το λογικό όνομα inf. Ο προσδιοριστής 'r' χρησιμοποιείται για να δηλώσουμε ότι το αρχείο είναι για ανάγνωση (read). Για να διαβάσουμε το περιεχόμενο του αρχείου σε μια μεταβλητή χρησιμοποιούμε την εντολή **fscanf** προσδιορίζοντας και το λογικό όνομα του αρχείου. Αν το αρχείο δεν υπάρχει, η εντολή **fopen** μας επιστρέφει το -1. Αν π.χ. χρησιμοποιήσουμε την εντολή :

```
>>  inf = fopen('aa.txt','r')

inf =

   -1
```

θα μας εμφανίσει το -1, γιατί το αρχείο 'aa.txt' δεν υπάρχει. Για να διαβάσουμε απ' το αρχείο 'test.txt' στο οποίο γράψαμε κάποια δεδομένα με τα προηγούμενα παραδείγματα, θα πρέπει να χρησιμοποιήσουμε την εντολή :

```
>>  inf = fopen('test.txt'  ,'r')

inf =

    5
```

Το αρχείο 'test.txt' υπάρχει και είναι έτοιμο για διάβασμα της $1^{ης}$ γραμμής. Το διάβασμα των δεδομένων του αρχείου και η αποθήκευσή τους στις αντίστοιχες μεταβλητές γίνεται με την εντολή **fscanf**, η οποία, εκτός απ' τον προσδιοριστή απαιτεί και τον αριθμό των δεδομένων που θα διαβαστούν, όπως φαίνεται στα παραδείγματα που ακολουθούν :

Παράδειγμα 1

```
>>  onoma = fscanf(inf,'%s',1)

onoma =

Kostas
```

Με την προηγούμενη εντολή το περιεχόμενο της πρώτης γραμμής του αρχείου 'test.txt' γράφεται στη μεταβλητή onoma. Με παρόμοιο τρόπο μπορούμε να διαβάσουμε και τις υπόλοιπες γραμμές του αρχείου, για όσο το έχουμε ανοιχτό.

Παράδειγμα 2

```
>>  a = fscanf(inf,'%d',1)

a =

    5


>>  b = fscanf(inf,'%f',1)

b =

   3.1400
```

Με τις 2 προηγούμενες εντολές το περιεχόμενο της δεύτερης γραμμής του αρχείου 'test.txt' αποθηκεύονται στις μεταβλητές a και b.

Παράδειγμα 3

```
>> pina = fscanf(inf,'%f',3)

pina =

    1
    2
    3
```

Με την προηγούμενη εντολή το περιεχόμενο της τρίτης γραμμής του αρχείου 'test.txt' αποθηκεύεται στο διάνυσμα pina.

Παρατήρηση

- Αν αποθηκεύσουμε τα στοιχεία ενός Πίνακα 2 διαστάσεων σε ένα αρχείο, τα στοιχεία του Πίνακα αποθηκεύονται κατά στήλες και όχι κατά γραμμές. Έτσι, με τις παρακάτω εντολές :

```
>> outf = fopen('test1.txt','w');

>> A = [1 2 3;4 5 6]

A =

   1   2   3
   4   5   6

>> fprintf(outf,'%f %f\n', A);

>> fclose(outf)
```

η εικόνα του αρχείου 'test1.txt' θα είναι :

```
1  1.000000 4.000000
2  2.000000 5.000000
3  3.000000 6.000000
4
```

- Αν θέλουμε να διαβάσουμε τα περιεχόμενα του αρχείου στον Πίνακα A, θα χρησιμοποιήσουμε τις εντολές :

```
>> inf = fopen('test1.txt','r');

>> A = fscanf(inf,'%f %f',[2,3])

A =

   1   2   3
   4   5   6

>> fclose(inf);
```

όπου [2,3] είναι ο αριθμός των γραμμών και στηλών του Πίνακα στον οποίο θα αποθηκευθούν οι τιμές που θα διαβαστούν.

Παρατήρηση

- Αν και τα στοιχεία του Πίνακα είναι αποθηκευμένα στο αρχείο κατά στήλες, διαβάζονται στον πίνακα κανονικά κατά γραμμές.

❖ Αν θέλουμε να αποθηκεύσουμε τα στοιχεία του Πίνακα στο αρχείο κατά γραμμές, θα πρέπει να χρησιμοποιήσουμε στην εντολή **fprintf** τον ανάστροφο του Πίνακα, όπως φαίνεται στο παράδειγμα που ακολουθεί :

```
>> outf = fopen('test2.txt','w');

>> fprintf(outf,'%f %f %f\n', A')

>> fclose(outf);
```

και η εικόνα του αρχείου 'test2.txt' θα είναι :

```
1  1.000000 2.000000 3.000000
2  4.000000 5.000000 6.000000
3
```

❖ Για να διαβάσουμε τα περιεχόμενα του αρχείου στον Πίνακα A, θα χρησιμοποιήσουμε τις εντολές :

```
>>  inf = fopen('test2.txt','r');

>>  A = fscanf(inf,'%f %f %f',[3,2])

A =

    1    4
    2    5
    3    6

>> fclose(inf);
```

και θα πρέπει να πάρουμε τον ανάστροφο του Πίνακα A :

```
>> A = A'

A =

    1    2    3
    4    5    6
```

ΚΕΦΑΛΑΙΟ 6

COMPILER & DEBUGGER ΓΙΑ M-FILES

6.1 Interpreter και Compiler (Help: MATLAB Compiler)

Η γραμμή εντολών (command line) λειτουργεί ως interpreter των εντολών που γράφουμε εκεί. Κάθε εντολή που πληκτρολογούμε μεταφράζεται σε γλώσσα μηχανής και εκτελείται μόλις πατήσουμε <ENTER>. Με την ίδια ακριβώς λογική λειτουργούν και τα προγράμματα script, δηλαδή είναι σαν να γράφαμε στο command line του MATLAB κάθε εντολή που είναι γραμμένη στο αρχείο script και πατούσαμε το <ENTER>. Έτσι κάθε εντολή μεταφράζεται και εκτελείται ξανά και ξανά ακόμη και εάν βρίσκεται σε ένα βρόχο. Αυτό δημιουργεί καθυστερήσεις στους βρόχους και είναι λιγότερο αποδοτική μέθοδος σε σχέση με την μετάφραση του κώδικα μια φορά στην αρχή και την δημιουργία binary εκτελέσιμου αρχείου τύπου exe. Η μέθοδος του interpreter είναι βέβαια ιδανική όταν θέλουμε να τεστάρουμε γρήγορα μια ιδέα ή έναν αλγόριθμο χωρίς να χρειάζεται κάθε φορά να μεσολαβεί το χρονοβόρο στάδιο της μετάφρασης και της εκτέλεσης του εκτελέσιμου αρχείου.

Το MATLAB προσφέρει έναν compiler με το όνομα **mcc** οποίος δημιουργεί εκτελέσιμο κώδικα σε μορφή αρχείου **exe**. Για παράδειγμα έστω ότι έχουμε γράψει το παρακάτω πρόγραμμα στο αρχείο **dokimi.m**

```
function main()

for (ii=1:10)
   fprintf('i=%f\n', ii);
end
```

Ο κώδικας του κυρίως προγράμματος πρέπει να είναι μια **function** με το όνομα **main()**. Εκτελούμε τον compiler μέσα από το MATLAB χρησιμοποιώντας την παράμετρο **–m** :

```
>> mcc -m dokimi
```

Με την προηγούμενη εντολή, δημιουργείται στον παρόντα κατάλογο **μετά από αρκετή αναμονή** το αρχείο dokimh_main.c και εμφανίζεται το παρακάτω μήνυμα :

Select a compiler:
[1] Lcc C version 2.4 in C:\PROGRAM FILES\MATLAB704\BIN\WIN32\\..\..\sys\lcc
[2] Microsoft Visual C/C++ version 6.0 in C:\Program Files\Microsoft Visual Studio

[0] None

Compiler: 1

Αφού επιλέξουμε ένα C-Compiler, δημιουργείται στον παρόντα κατάλογο το αρχείο dokimh_main.exe, το οπίο μπορούμε να τρέξουμε, όπως οποιοδήποτε εκτελέσιμο αρχείο.

6.2 Ο Editor/Debugger (*Help: MATLAB -> Desktop Tools and Development Environment -> Editing and Debugging M-Files*)

Ο editor είναι ένα ολοκληρωμένο περιβάλλον επεξεργασίας scripts. Προσφέρει αυτόματη μορφοποίηση του κώδικα που πληκτρολογούμε, ενώ με διαφορετικά χρώματα καταδεικνύονται οι διαφορετικές εντολές, τα σχόλια με πράσινο χρώμα, οι built-in εντολές με μπλε χρώμα και τα strings με κόκκινο χρώμα. Ο editor έχει ενσωματωμένο και το *debugger.*
Ο *debugger* προσφέρει τη δυνατότητα δημιουργίας σημείων στον κώδικα όπου σταματάει προσωρινά η εκτέλεση του script ώστε να μπορεί να γίνει επισκόπηση των μεταβλητών και του συστήματος γενικότερα. Τα σημεία αυτά λέγονται **breakpoints**.

- Η δημιουργία ενός **breakpoint** γίνεται με την επιλογή **Debug-> *S*et/*C*lear Breakpoint** με το δρομέα στην εντολή την οποία έχουμε επιλέξει για έλεγχο ή με αριστερό κλικ με το ποντίκι στην παύλα, δίπλα στον αριθμό της εντολής που έχουμε επιλέξει για έλεγχο.

- Η εκτέλεση του προγράμματος γίνεται μέχρι τη γραμμή στην οποία έχουμε βάλει **breakpoint**, η οποία εμφανίζεται με τον αριθμό της και το prompt αλλάζει σε K>>.

- Στο prompt αυτό μπορούμε να γράψουμε το όνομα μιας μεταβλητής χωρίς να ακολουθεί το σύμβολο ; οπότε εμφανίζεται στην οθόνη το περιεχόμενο της μεταβλητής αυτής.

- Με την εντολή **return** επιστρέφουμε στην κανονική εκτέλεση του προγράμματος (βγαίνουμε από τον debugger), εκτός αν βρισκόμαστε σε εντολή επανάληψης, οπότε θα χρειαστεί και ο αντίστοιχος αριθμός εντολών return για να δούμε τις επόμενες τιμές .

- Το **καθάρισμα ενός *breakpoint*** γίνεται πάλι με την επιλογή **Debug-> *S*et/*C*lear Breakpoint** με το δρομέα στην εντολή την οποία έχουμε επιλέξει για έλεγχο ή με αριστερό κλικ με το ποντίκι στην παύλα, δίπλα στον αριθμό της εντολής που είχαμε επιλέξει για έλεγχο.

- Αφού σταματήσει η εκτέλεση σε κάποιο breakpoint, τότε αυτή μπορεί να συνεχιστεί βηματικά γραμμή-γραμμή χρησιμοποιώντας την επιλογή **Debug-> Step** ή το πλήκτρο **F10**.

- Αν η επόμενη γραμμή καλεί μια συνάρτηση τότε μπορούμε να μπούμε μέσα στη συνάρτηση (φορτώνεται ο κώδικας της συνάρτησης) και να την εκτελέσουμε γραμμή-γραμμή με την επιλογή **Debug-> Step In** ή το πλήκτρο **F11**.

- Μπορούμε να βγούμε από τη συνάρτηση πίσω στο κυρίως πρόγραμμα με την επιλογή **Debug-> Step Out** ή με το συνδιασμό πλήκτρων **Shift+F11**.

- Η επιλογή **Debug-> Continue** συνεχίζει κανονικά (όχι βηματικά) την εκτέλεση μέχρι το επόμενο breakpoint.

- Επίσης προσφέρονται λειτουργίες **Debug-> Stop If Errors/Warnings**..., όπου το πρόγραμμα σταματάει να εκτελείται ακόμα κι αν δεν υπάρχει breakpoint εφόσον εμφανιστεί κάποια συνθήκη σφάλματος. Τέτοιες συνθήκες είναι: Error, Warning, Not a Number (NaN) ή Infinity (Inf). Το πρόγραμμα σταματάει στη γραμμή που εμφανίστηκε το σφάλμα ώστε να μπορούμε να επιθεωρήσουμε τις τιμές των μεταβλητών και να βοηθηθούμε στην ανακάλυψη της αιτίας του σφάλματος.

Παράδειγμα 1

Έστω το script με τις παρακάτω εντολές :

```
 1 - clear;
 2 - clc;
 3 - n = input ( 'Δώσε n : ');
 4 - u = randn(1,n);
 5 - disp('u = ');
 6 - disp(u);
 7 - for i = 1 : n
 8 -     fprintf('i = %d\n',i);
 9 -     v = step01(u, n);
10 - end %for
11 - disp('v = ');
12 - disp(v);
```

Με τη δημιουργία ενός breakpoint στη γραμμή 8 και την εκτέλεση του script θα πάρουμε την παραπάνω εικόνα :

```
Δώσε n : 3
u =
  -1.2075    0.7172    1.6302

8     fprintf('i = %d\n',i);
K>>
```

Αν θέλουμε να παρακολουθήσουμε τις τιμές που παίρνει η μεταβλητή I, γράφουμε στο prompt **K>>** το **i**, οπότε θα εμφανίζονται οι τιμές που παίρνει μέχρι να τελειώσει η εντολή επανάληψης, όπως φαίνεται παρακάτω, αν συνεχίσουμε την εκτέλεση του προγράμματος με το F10 :

```
K>> i

i =
   1
i = 1
9     v = step01(u, n);
10  end %for
8     fprintf('i = %d\n',i);
i = 2
9     v = step01(u, n);
10  end %for
8     fprintf('i = %d\n',i);
i = 3
9     v = step01(u, n);
10  end %for
11  disp('v = ');
v =
12  disp(v);
   0    1    1

End of script testf1.
K>> return
>>
```

Παράδειγμα 2

Έστω το script με τις παρακάτω εντολές :

```
 1 - clear;
 2 - clc;
 3 - n = input ( 'Δώσε n : ');
 4 - u = randn(1,n);
 5 - disp('u = ');
 6 - disp(u);
 7 - for i = 1 : n
 8 -     fprintf('i = %d\n',i);
 9 ●     v = step01(u, n);
10 - end %for
11 - disp('v = ');
12 - disp(v);
```

Με τη δημιουργία ενός breakpoint στη γραμμή 9, η οποία καλεί το function step01 και την εκτέλεση του script θα πάρουμε την παραπάνω εικόνα :

```
Δώσε n : 3
u =
```

```
   0.4889    1.0347    0.7269
8      fprintf('i = %d\n',i);
i = 1
```

Με το πάτημα του πλήκτρου F11, ο έλεγχος πηγαίνει στο function step01 και παίρνουμε την παρακάτω εικόνα :

```
9      v = step01(u, n);
2   for i = 1 : n
```

Και εμφανίζεται ο κώδικας του function, στην οποία ανήκει και η εντολή 2. Με το συνεχές πάτημα του πλήκτρου F11, εκτελούμε μια μια τις εντολές του function, όσες φορές χρειαστεί και παίρνουμε την παρακάτω εικόνα :

```
3      if ( u(i) > 0 )
4        v(i) = 1;
8   end %for
3      if ( u(i) > 0 )
5      else
6        v(i) = 0;
7      end %if
8   end %for
3      if ( u(i) > 0 )
5      else
6        v(i) = 0;
7      end %if
8   end %for
End of function step01.
10  end %for
8      fprintf('i = %d\n',i);
i = 2
9      v = step01(u, n);
2   for i = 1 : n
3      if ( u(i) > 0 )
4        v(i) = 1;
8   end %for
3      if ( u(i) > 0 )
5      else
6        v(i) = 0;
7      end %if
8   end %for
3      if ( u(i) > 0 )
5      else
6        v(i) = 0;
7      end %if
8   end %for
End of function step01.
10  end %for
8      fprintf('i = %d\n',i);
```

```
i = 3
9      v = step01(u, n);
2   for i = 1 : n
3      if ( u(i) > 0 )
4         v(i) = 1;
8   end %for
3      if ( u(i) > 0 )
5      else
6         v(i) = 0;
7      end %if
8   end %for
3      if ( u(i) > 0 )
5      else
6         v(i) = 0;
7      end %if
8   end %for
End of function step01.
10  end %for
11  disp('v = ');
v =
12  disp(v);
    1     0     0

End of script testf1.
K>> return
>>
```

ΚΕΦΑΛΑΙΟ 7

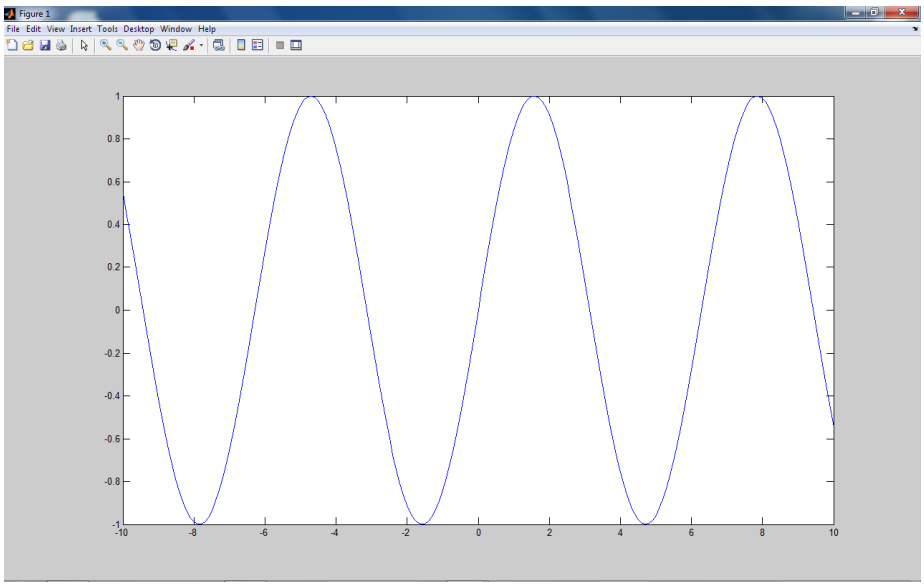
ΓΡΑΦΙΚΕΣ ΠΑΡΑΣΤΑΣΕΙΣ

7.1 Γραφικές Παραστάσεις 2 Διαστάσεων – Εντολή plot

Οι γραφικές παραστάσεις 2 διαστάσεων (στο επίπεδο) γίνονται με την εντολή - συνάρτηση **plot** (Help: Matlab->Functions – Categorical List->Graphics->Basic Plots and Graphs>Plot). Η plot γενικά απαιτεί 2 ορίσματα, τα οποία πρέπει να είναι διανύσματα με τον ίδιο αριθμό στοιχείων και σχεδιάζει τη γραφική παράσταση (Γράφημα 7.1) που ορίζουν τα σημεία των 2 διανυσμάτων, όπως φαίνεται στο παράδειγμα που ακολουθεί :

Παράδειγμα 1

```
>> x = [-10:0.1:10];
>> y = sin(x);
>> plot(x,y)
```



Γράφημα 7.1

Παρατηρήσεις

- Η συνάρτηση plot δέχεται σαν ορίσματα δύο διανύσματα και για το κάθε ζεύγος τιμών απεικονίζει στο επίπεδο το αντίστοιχο σημείο.
- Τα δύο διανύσματα πρέπει να έχουν τον ίδιο αριθμό στοιχείων.
- Δημιουργεί το γράφημα Figure 1, 2, 3,… ανάλογα με τον αριθμό των γραφημάτων που έχουμε δημιουργήσει.
- Εμφανίζει τους 2 άξονες και τις τιμές που έχουμε δώσει στα 2 διανύσματα.
- Εμφανίζει τη συνεχή καμπύλη του γραφήματος με μπλε χρώμα.
- Αν δεν κλείσουμε το γράφημα, μπορούμε να σχεδιάσουμε και κάποιο άλλο γράφημα πάνω στο υπάρχον.

7.1.2 Σύμβολα, Χρώματα και Γραμμές

Στα γραφήματα μπορούμε να παρέμβουμε στα διάφορα χαρακτηριστικά της γραμμής, όπως :

- Το **είδος** της γραμμής
- Το **χρώμα** της γραμμής
- Το **πάχος** της γραμμής
- Τον **τύπο** της κουκίδας
- Το **μέγεθος** της κουκίδας
- Τα **χρώματα** της κουκίδας (περίγραμμα και εσωτερικό)

συμπεριλαμβάνοντας στην εντολή plot ένα string, το οποίο περιέχει τους κωδικούς με τους οποίους επιλέγουμε τα παραπάνω χαρακτηριστικά.

Είδος Γραμμής

Οι κωδικοί που καθορίζουν το είδος της γραμμής φαίνονται στον Πίνακα 7.1 :

Κωδικός	Τύπος γραμμής
-	Μονοκόμματη γραμμή
--	Διακεκομμένη γραμμή
:	Γραμμή με τελίτσες
-.	Γραμμή με παύλα-τελεία

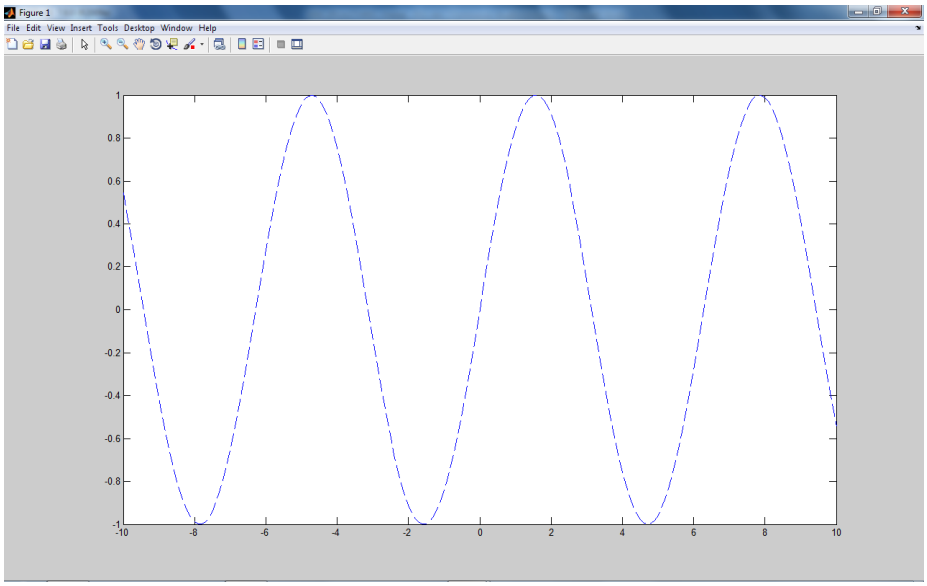
Πίνακας 7.1

Παράδειγμα 2

Αν, στο προηγούμενο Παράδειγμα 1, θέλουμε η γραμμή να εμφανίζεται **διακεκομμένη**, θα χρειαστούν οι παρακάτω εντολές :

```
>> x = [-10:0.1:10];
>> y = sin(x);
>> plot(x,y,'--')';
```

με αποτέλεσμα να δημιουργηθεί το γράφημα που φαίνεται στο Γράφημα 7.2 :



Γράφημα 7.2

Χρώμα Γραμμής

Οι κωδικοί που καθορίζουν το χρώμα της γραμμής φαίνονται στον Πίνακα 7.2 :

Κωδικός	Χρώμα γραμμής
r	κόκκινο
g	πράσινο
b	μπλε
c	κυανό
m	μωβ
y	κίτρινο
k	μαύρο
w	άσπρο

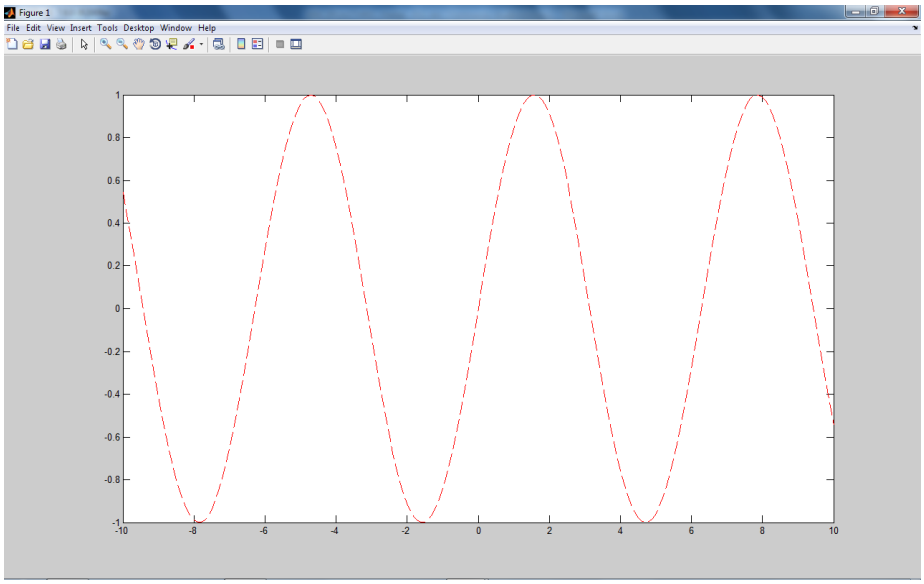
Πίνακας 7.2

Παράδειγμα 3

Αν, στο Παράδειγμα 1, θέλουμε η γραμμή να εμφανίζεται **διακεκομμένη** και **κόκκινη**, θα χρειαστούν οι παρακάτω εντολές :

```
>> x = [-10:0.1:10];
>> y = sin(x);
>> plot(x,y,'r--')';
```

με αποτέλεσμα να δημιουργηθεί το γράφημα που φαίνεται στο Γράφημα 7.3 :



Γράφημα 7.3

Πάχος Γραμμής

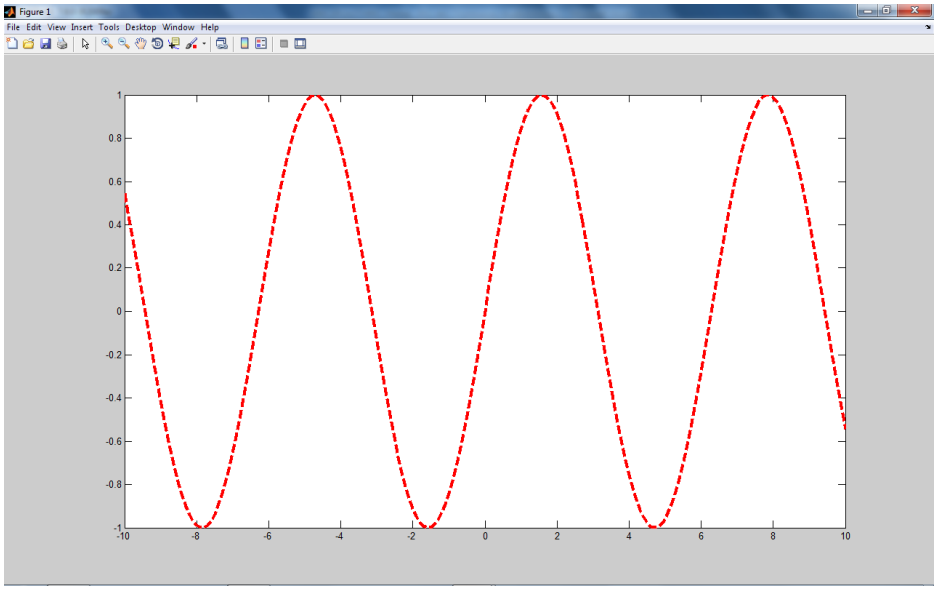
Για να αλλάξουμε το default πάχος της γραμμής του γραφήματος θα πρέπει στην εντολή plot να συμπεριλάβουμε και το string 'LineWidth' και μια τιμή μεταξύ του 0.5 και 30 για το αντίστοιχο πάχος της γραμμής.

Παράδειγμα 4

Αν, στο Παράδειγμα 1, θέλουμε η γραμμή να εμφανίζεται διακεκομμένη και κόκκινη και με πάχος γραμμής 3, θα χρειαστούν οι παρακάτω εντολές :

```
>> x = [-10:0.1:10];
>> y = sin(x);
>> plot(x,y,'r--', 'LineWidth', 3)';
```

με αποτέλεσμα να δημιουργηθεί το γράφημα που φαίνεται στο Γράφημα 7.4 :



Γράφημα 7.4

Τύπος Κουκίδας

Για την επιλογή της κουκίδας που θα αναπαριστά τα σημεία του γραφήματος, θα πρέπει στο string να συμπεριληφθεί ένας κωδικός, απ' αυτούς που παρουσιάζονται στον Πίνακα 7.3 :

Κωδικός	**Τύπος κουκίδας**
+	σύμβολο πρόσθεσης
o	κύκλος
*	αστερίσκος
.	σημείο
x	σταυρός
'square' or s	τετράγωνο
'diamond' or d	Διαμάντι - ρόμβος
^	τρίγωνο με μύτη επάνω
v	τρίγωνο με μύτη κάτω
>	τρίγωνο με μύτη προς τα δεξιά
<	τρίγωνο με μύτη προς τα αριστερά
'pentagram' or p	πεντάγραμμο
'hexagram' or h	εξάγραμμο

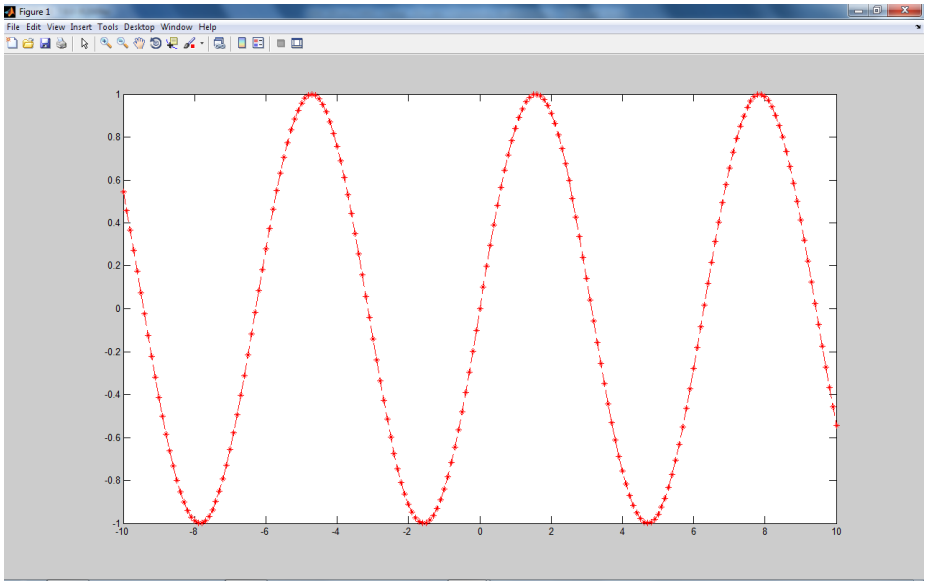
Πίνακας 7.3

Παράδειγμα 5

Αν, στο Παράδειγμα 1, θέλουμε η γραμμή να εμφανίζεται **διακεκομμένη** και **κόκκινη** και με **αστεράκια**, θα χρειαστούν οι παρακάτω εντολές :

```
>> x = [-10:0.1:10];
>> y = sin(x);
>> plot(x,y,'r--*')';
```

με αποτέλεσμα να δημιουργηθεί το γράφημα που φαίνεται στο Γράφημα 7.5 :



Γράφημα 7.5

Μέγεθος Κουκίδας

Για να αλλάξουμε το default μέγεθος της κουκίδας του γραφήματος θα πρέπει στην εντολή plot να συμπεριλάβουμε και το string 'MarkerSize' και μια τιμή μεταξύ του 0.5 και 30 για το αντίστοιχο μέγεθος της κουκίδας.

Χρώμα Περιγράμματος Κουκίδας

Για να αλλάξουμε το default χρώμα του περιγράμματος της κουκίδας του γραφήματος θα πρέπει στην εντολή plot να συμπεριλάβουμε και το string 'MarkerEdgeColor' και ένα ακόμη string με μια τιμή για το χρώμα, σύμφωνα με τον Πίνακα 7.2.

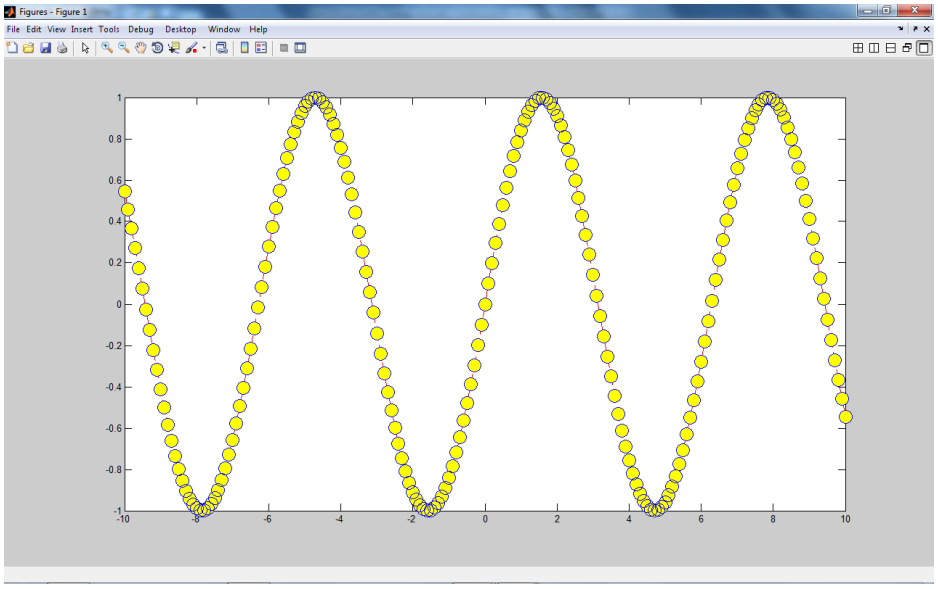
Χρώμα Γεμίσματος Κουκίδας

Για να αλλάξουμε το default χρώμα του γεμίσματος της κουκίδας του γραφήματος θα πρέπει στην εντολή plot να συμπεριλάβουμε και το string 'MarkerFaceColor' και ένα ακόμη string με μια τιμή για το χρώμα, σύμφωνα με τον Πίνακα 7.2.

Παράδειγμα 6

Αν, στο Παράδειγμα 1, θέλουμε η γραμμή να εμφανίζεται **διακεκομμένη** και **κόκκινη** και με **κύκλους** μεγέθους **14**, με χρώμα περιγράμματος **μπλε** και χρώμα γεμίσματος **κίτρινο**, θα χρειαστούν οι παρακάτω εντολές με αποτέλεσμα το γράφημα 7.6 :

```
>> x = [-10:0.1:10];
>> y = sin(x);
>> plot(x,y,'r--o', 'MarkerSize',14, 'MarkerFaceColor','y', 'MarkerEdgeColor','b' );
```

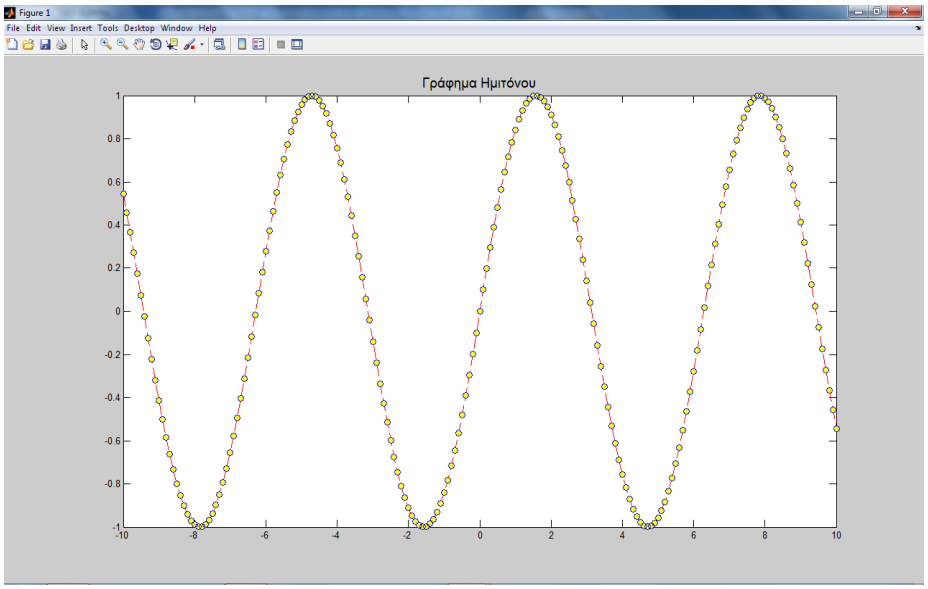


Γράφημα 7.6

Τίτλος Γραφήματος

Μπορούμε να βάλουμε στο γράφημα Τίτλο (Γράφημα 7.7) με την εντολή :

```
>> title( 'Γράφημα Ημιτόνου','FontSize',14 );
```



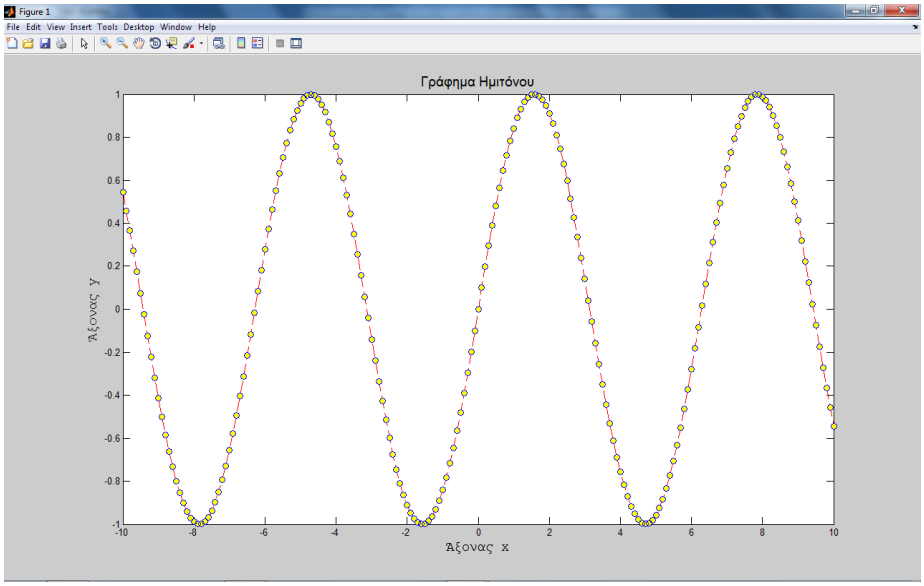
Γράφημα 7.7

Ονόματα Αξόνων Γραφήματος

Αντίστοιχα, μπορούμε να βάλουμε στο γράφημα και ονομασίες για τους Άξονες x και y, επιλέγοντας γραμματοσειρά και μέγεθος γραμμάτων με τις εντολές :

```
>> xlabel ( 'Άξονας x','FontName','Courier New','FontSize',14);
>> ylabel ( 'Άξονας y','FontName','Courier New','FontSize',14);
```

οπότε το προηγούμενο γράφημα γίνεται :

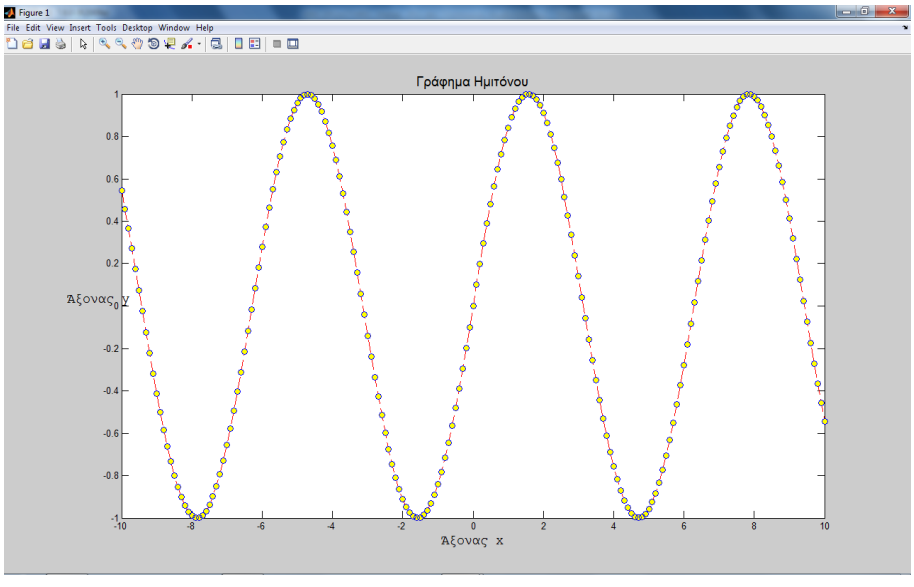


Γράφημα 7.8

Παρατήρηση

- Αν θέλουμε ο Τίτλος του Άξονα y να εμφανίζεται οριζόντια και όχι κάθετα, χρησιμοποιούμε το 'Rotation' στην εντολή ylabel, όπως φαίνεται στο Γράφημα 7.9 :

```
>> ylabel ( 'Άξονας y','FontName','Courier New','FontSize',14, 'Rotation',0);
```



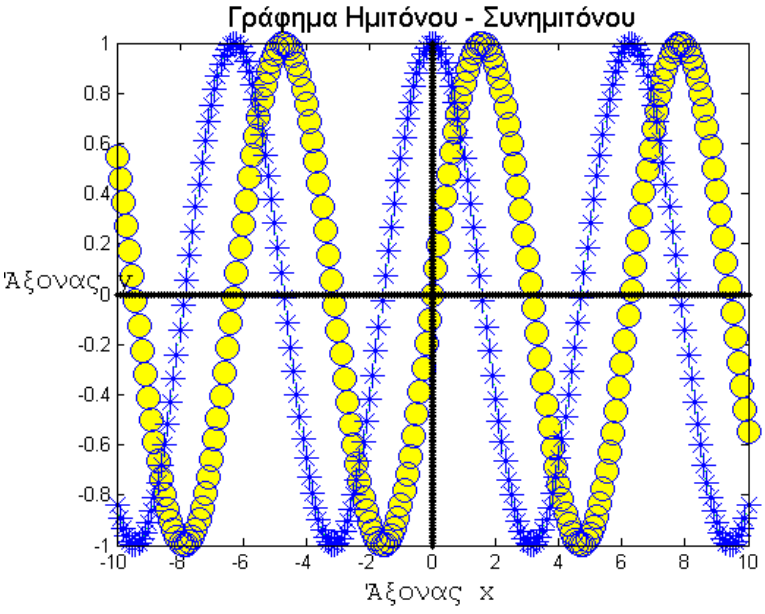
Γράφημα 7.9

7.1.3 Η Εντολή hold on

Η εντολή hold μας δίνει την δυνατότητα να σχεδιάσουμε πολλαπλά γραφήματα σε ένα figure. Η function hold καθορίζει το αν στην επόμενη φορά που θα εισάγουμε μία νέα γραφική παράσταση θα αντικαταστήσει την προηγούμενη ή θα σχεδιαστεί από πάνω. Όταν καλούμε την hold με το όρισμα on να την ακολουθεί τότε η γραφική παράσταση που έχουμε σχεδιάσει διατηρείται. Η δράση της hold on διαρκεί μέχρι να καλέσουμε την hold off. Τότε ότι έχουμε σχεδιάζει θα σβηστεί από την επόμενη γραφική παράσταση που θα σχεδιάσουμε. Αν π.χ. θέλουμε να σχεδιάσουμε και τη γραφική παράσταση του συνημιτόνου στο προηγούμενο γράφημα, θα πρέπει να χρησιμοποιήσουμε τις εντολές :

```
>> x = [-10:0.1:10];
>> y = sin(x);
>> plot(x,y,'r--o', 'MarkerSize',12, 'MarkerFaceColor','y', 'MarkerEdgeColor','b' );
>> title( 'Γράφημα Ημιτόνου - Συνημιτόνου','FontSize',14 );
>> xlabel ( 'Άξονας x','FontName','Courier New','FontSize',14);
>> ylabel ( 'Άξονας y','FontName','Courier New','FontSize',14, 'Rotation',0);
>> hold on;
>> y = cos(x);
>> plot(x,y,'g:*', 'MarkerSize',12, 'MarkerFaceColor','k', 'MarkerEdgeColor','b' );
>> hold on;
>> plot(x,0,'.k','LineWidth',5);
>> hold on;
>> y1=(-1:0.01:1);
>> plot(0,y1,'.k','LineWidth',5);
```

και η γραφική παράσταση με τους άξονες x,y φαίνεται στο επόμενο Γράφημα 7.10 :



Γράφημα 7.10

Παρατήρηση

Αν θέλουμε να απενεργοποιήσουμε τη δυνατότητα να σχεδιάζουμε στο ίδιο γράφημα, θα πρέπει να χρησιμοποιήσουμε την εντολή **hold off**.

7.1.4 Η Εντολή subplot

Με την εντολή subplot μπορούμε στο ίδιο γράφημα να σχεδιάσουμε περισσότερα από ένα υπογραφήματα, δίνοντας κάθε φορά τη διάταξη που θα έχουν τα υπογραφήματα, δηλαδή πόσες γραμμές και πόσες στήλες, τα οποία μετά αριθμούνται διατρέχοντας το δισδιάστατο πίνακα κατά γραμμές, δηλαδή το πρώτο υπογράφημα στην $1^η$ γραμμή $1^η$ στήλη, το δεύτερο στην $1^η$ γραμμή $2^η$ στήλη κ.λ.π..

Παράδειγμα 7

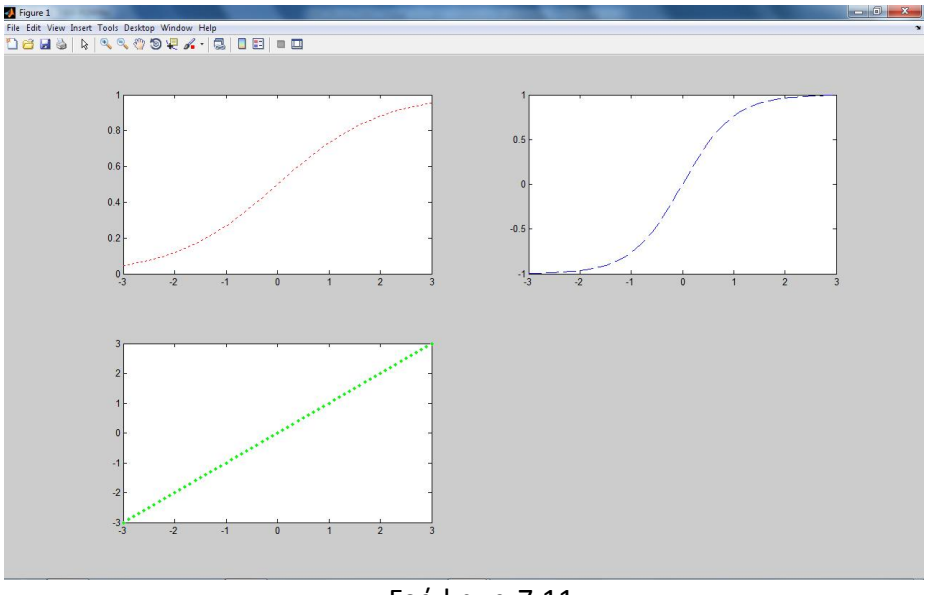
Να γίνει η γραφική παράσταση των 3 συνεχών συναρτήσεων ενεργοποίησης, δηλαδή της Σιγμοειδούς, της Υπερβολικής Εφαπτομένης και της Γραμμικής Συνάρτησης με τη χρήση της εντολής subplot :

```
>> subplot(2,2,1);
>> x=-3:0.1:3;
>> y=1./(1+exp(-x));
>> plot(x,y,'r:');

>> subplot(2,2,2);
>> y=tanh(x);
>> plot(x,y,'b--');

>> subplot(2,2,3);
>> y = x;
>> plot(x,y,'g.');
```

Με τις παραπάνω εντολές γίνονται 2x2 υπογραφήματα, τα οποία αριθμούνται σαν 1 το (1,1), 2 το (1,2) και 3 το (2,1), όπως φαίνεται στο Γράφημα 7.11 :



Γράφημα 7.11

Παράδειγμα 8

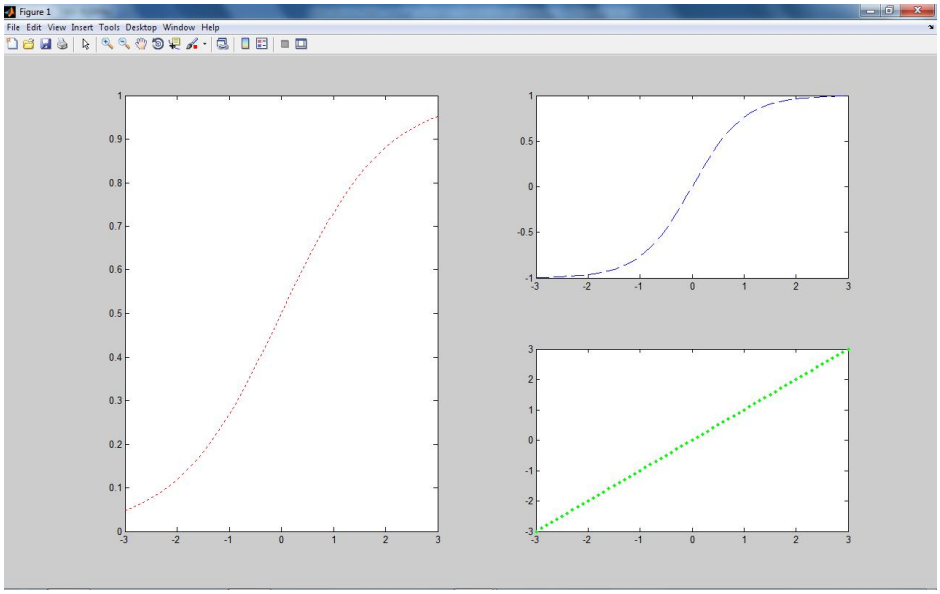
Αν στο προηγούμενο παράδειγμα θέλαμε στη θέση του 1ου και του 3ου υπογραφήματος να εμφανίζεται το πρώτο απ' τα 3 γραφήματα και τα υπόλοιπα δύο στις θέσεις 2 και 4, τροποποιούμε τις προηγούμενες εντολές αριθμώντας το 1ο υπογράφημα σαν 1 και 3 μαζί, όπως φαίνεται στις εντολές που ακολουθούν :

```
>> subplot(2,2,[1 3]);
>> x=-3:0.1:3;
>> y=1./(1+exp(-x));
>> plot(x,y,'r:');

>> subplot(2,2,2);
>> y=tanh(x);
>> plot(x,y,'b--');

>> subplot(2,2,4);
>> y = x;
>> plot(x,y,'g.');
```

Με τις παραπάνω εντολές το [1 3] στην πρώτη εντολή δηλώνει ότι το συγκεκριμένο subplot θα είναι στο χώρο του 1ου και 3ου plot, οπότε γίνονται 3 υπογραφήματα, τα οποία καλύπτουν το χώρο των τεσσάρων (2x2), όπως φαίνεται στο Γράφημα 7.11 :



Γράφημα 7.12

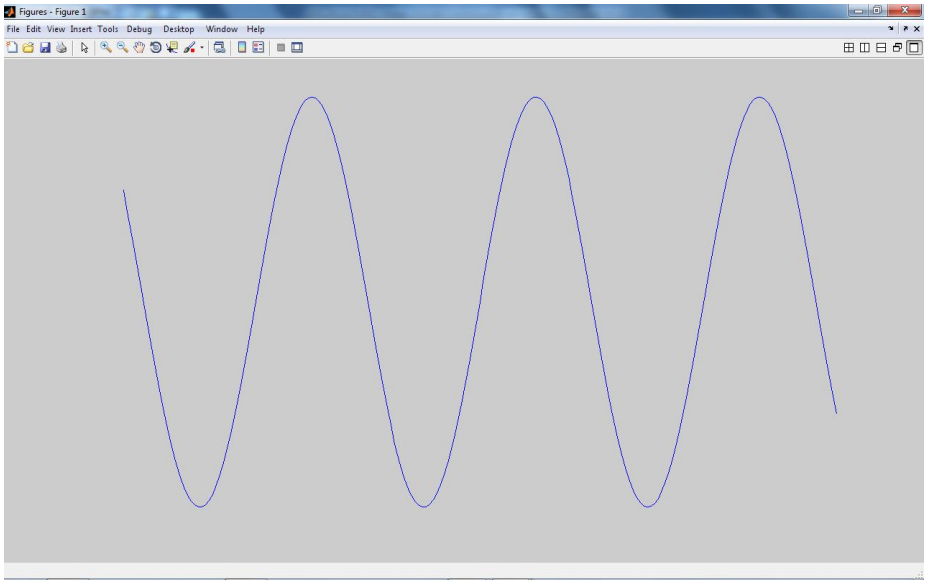
7.1.5 Αφαίρεση Αξόνων από Γράφημα (Εντολή axis off)

Αν θέλουμε να μην εμφανίζονται οι άξονες σε κάποιο γράφημα, χρησιμοποιούμε την εντολή **axis off**.

Παράδειγμα 9

Να αφαιρεθούν οι άξονες απ' το γράφημα του Παραδείγματος 1 :

```
>> x = [-10:0.1:10];
>> y = sin(x);
>> plot(x,y)
>> axis off
```



Γράφημα 7.13

7.1.6 Εισαγωγή Εικόνας σε Γράφημα (Εντολές imread, image)

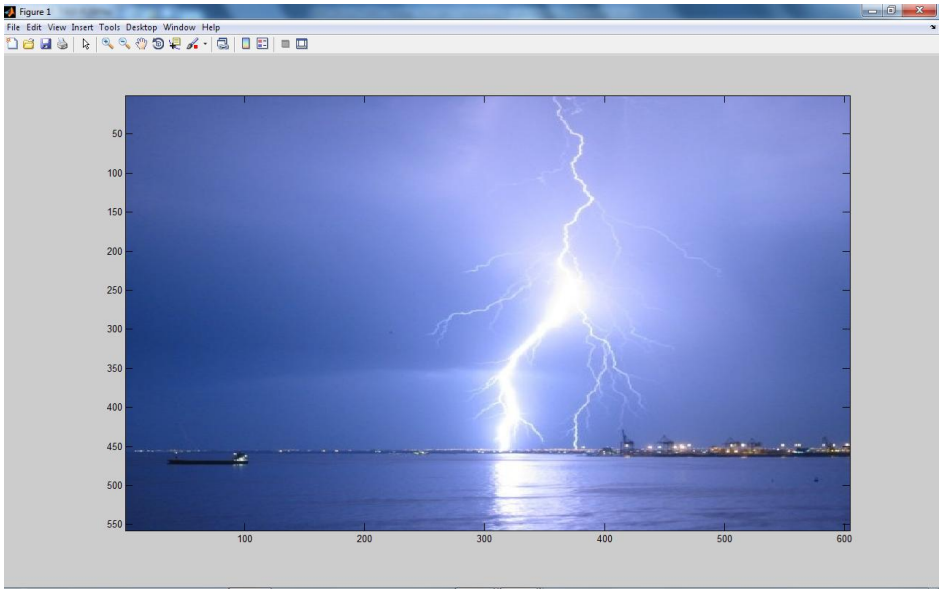
Για να εισάγουμε μια εικόνα jpg, tiff κ.λ.π. σε κάποιο γράφημα, θα πρέπει να διαβάσουμε την εικόνα σε κάποια μεταβλητή με την εντολή **imread** και να την αποθηκεύσουμε στο Figure με την εντολή **image**.

Παράδειγμα 10

Να γραφούν οι εντολές για την αποθήκευση στο figure 1 της εικόνας att00000.jpg που βρίσκεται στον παρόντα φάκελο. Με τις παρακάτω εντολές

```
>> figure(1);
>> x = imread('att00000.jpg');
>>  image(x);
```

το figure 1 θα έχει την παρακάτω εικόνα :



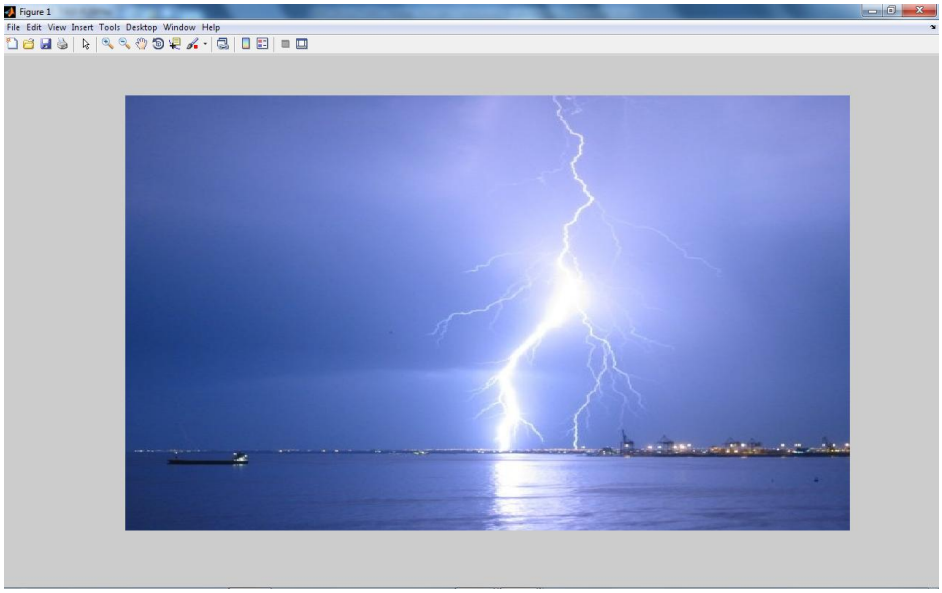
Γράφημα 7.14

Παρατήρηση

Αν χρησιμοποιήσουμε και την εντολή

>> axis off

θα αφαιρεθούν οι άξονες, οπότε θα έχουμε την παρακάτω εικόνα :



Γράφημα 7.15

7.1.7 Εισαγωγή Κειμένου σε Γράφημα (Εντολή gtext)

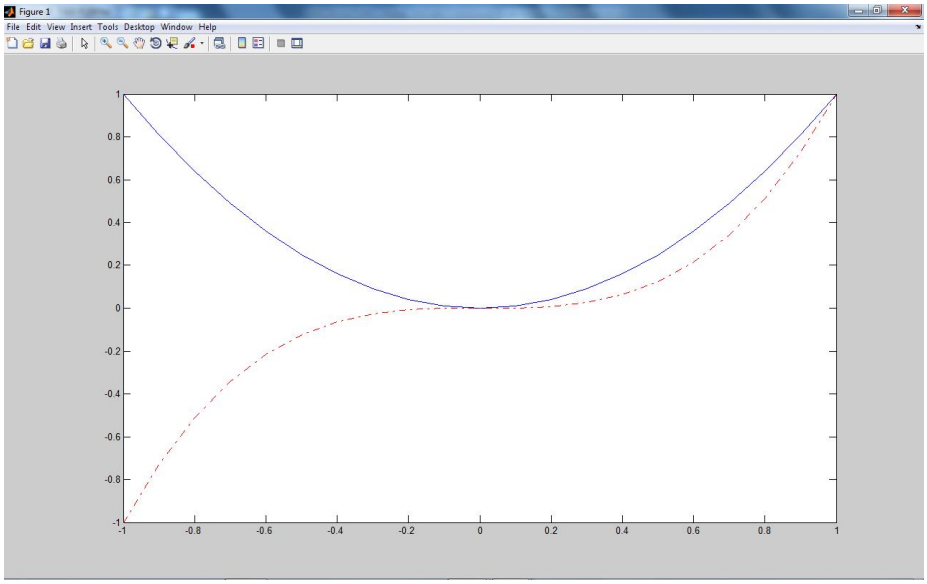
Μπορούμε να εισάγουμε κείμενο σε ένα γράφημα με τις εντολές **gtext** και **text**.

Παράδειγμα 11

Να γραφούν οι εντολές για τη δημιουργία ενός γραφήματος των συναρτήσεων x^2 και x^3 με το αντίστοιχο κείμενο. Με τις παρακάτω εντολές :

```
>> x = -1:0.1:1;
>> y1 = x.^2;
>> y2 = x.^3;
>> plot(x,y1,'b-',x,y2,'r-.');
```

θα δημιουργηθεί το γράφημα :

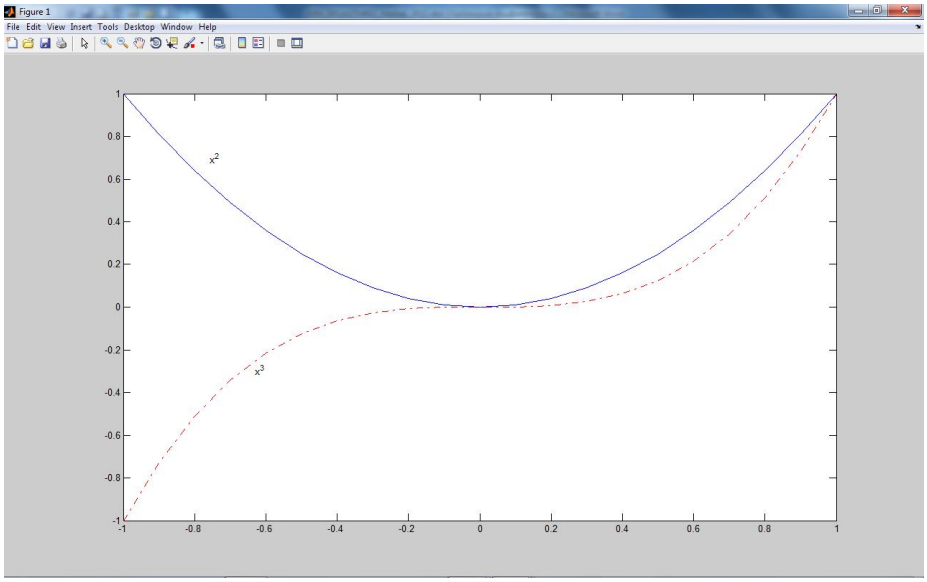


Γράφημα 7.16

Με τις εντολές

```
>> gtext('x^2');
>> gtext('x^3');
```

ανοίγει το γράφημα και μπορούμε με το ποντίκι να μετακινήσουμε το σταυρό που εμφανίζεται στη θέση που θέλουμε να εμφανιστεί το κείμενο. Μετά τις δύο εισαγωγές κειμένου, η προηγούμενη εικόνα θα πάρει την παρακάτω μορφή :



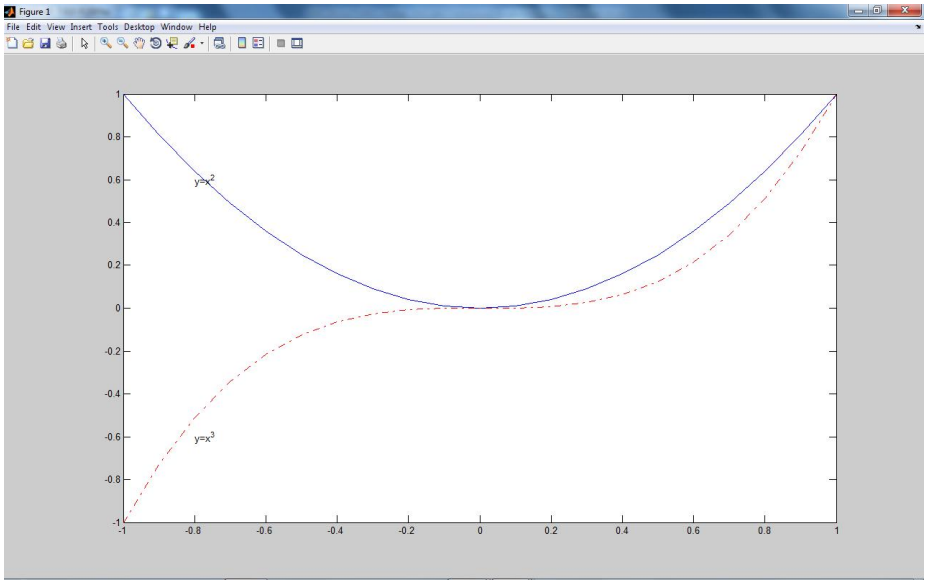
Γράφημα 7.17

Παρατήρηση

Παρόμοιο αποτέλεσμα μπορούμε να έχουμε με τη χρήση της εντολής **text**, αρκεί να δώσουμε τις συντεταγμένες της θέσης στην οποία θα μπει το κείμενο στο γράφημα. Αν στο πρηγούμενο παράδειγμα, αντί των εντολών gtext, δώσουμε τις εντολές :

```
>> text(-0.8,0.6,'y=x^2')
>> text(-0.8,-0.6,'y=x^3')
```

θα πάρουμε το παρακάτω γράφημα :



Γράφημα 7.18

7.1.8 Αποθήκευση Γραφήματος σε μορφή pdf, tiff,…

Για να αποθηκεύσουμε ένα γράφημα σε οποιαδήποτε μορφή – εκτός της default fig – μπορούμε να χρησιμοποιήσουμε την επιλογή Save As στο αντίστοιχο figure ή την εντολή **print** στο Command Prompt. Με την εντολή

```
>> print -dtiff -f1 figsin.tiff
```

Ο προσδιοριστής -dtiff αφορά τη μορφή του αρχείου (tiff στο παράδειγμα), ο -f1 το γράφημα 1, 2 κ.λ.π. στο οποίο αναφερόμαστε (figure 1 στο παράδειγμα) και το figsin.tiff είναι το όνομα του αρχείου στο οποίο θέλουμε να αποθηκευθεί το γράφημα.

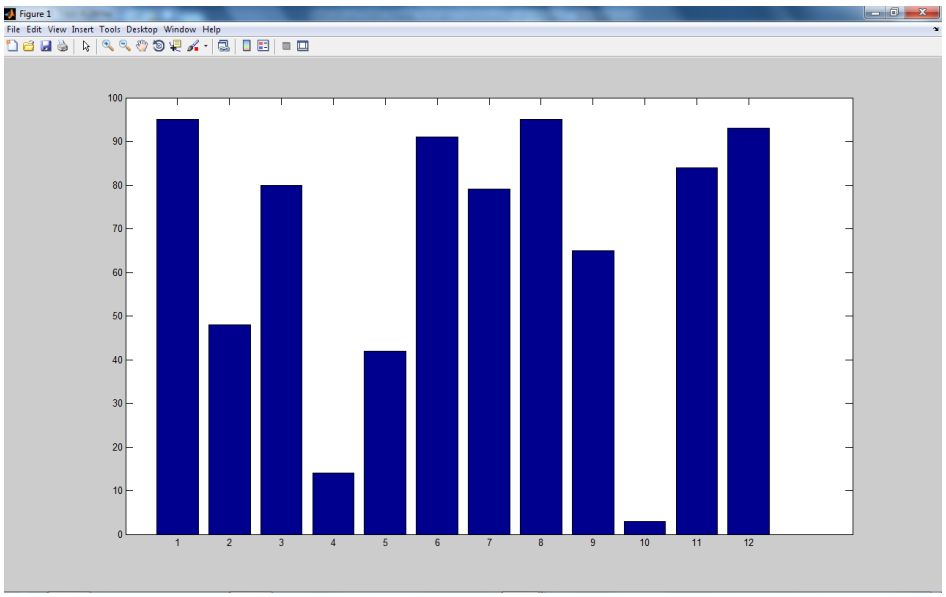
7.2 Η Εντολή bar

Με την εντολή bar, αντί της plot, σχεδιάζουμε ένα ραβδογράφημα.

Παράδειγμα 12

Για να γίνει ένα ραβδογράφημα με τις πωλήσεις κάθε μήνα, όπου οι τιμές των πωλήσεων είναι τυχαίοι ακέραιοι αριθμοί μεταξύ του 1 και του 100 θα χρειαστούν οι παρακάτω εντολές, με αποτέλεσμα το γράφημα 7.19 :

```
>> x = 1:1:12;
>> y = randint(1,12,100);
>> bar(x,y);
```



Γράφημα 7.19

Παρατήρηση

Η εντολή **randint** μετατρέπει τους τυχαίους αριθμούς στο (0, 1) σε ακέραιους αφού τους πολλαπλασιάσει με το **100**.

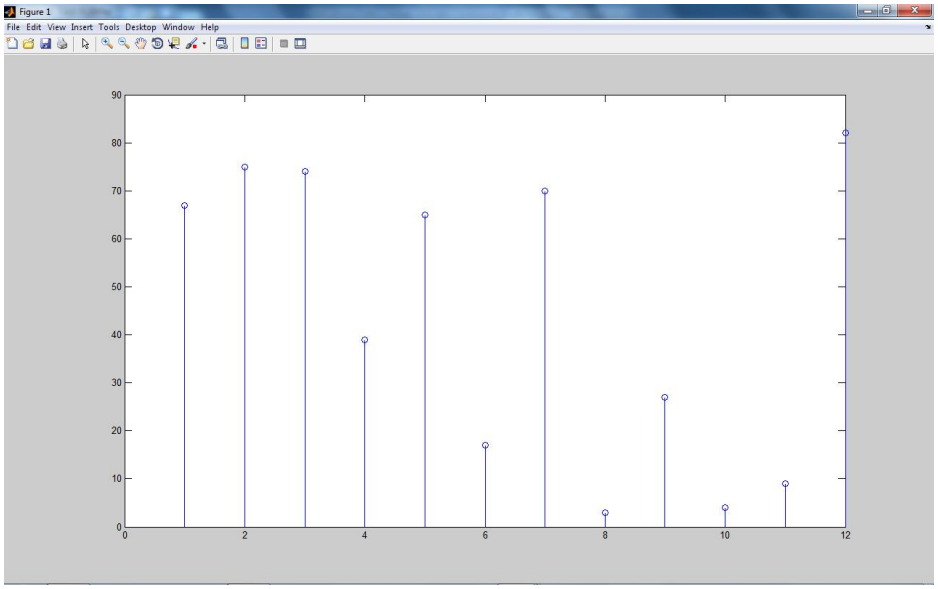
7.3 Η Εντολή stem

Παρόμοιο με την εντολή bar αποτέλεσμα, με πιο λεπτές μπάρες και με κύκλους όμως στην κορυφή κάθε μπάρας έχει και η εντολή stem.

Παράδειγμα 13

Αν στο Παράδειγμα 12 χρησιμοποιήσουμε την εντολή stem αντί της bar, θα χρειαστούν οι παρακάτω εντολές, με αποτέλεσμα το γράφημα 7.20 :

```
>> x = 1:1:12;
>> y = randint(1,12,100);
>> stem(x,y);
```



Γράφημα 7.20

7.3.1 Η Εντολή pause

Η εντολή **pause** (χωρίς όρισμα) σταματά την εκτέλεση του κώδικα μέχρι να πατήσει ο χρήστης το πλήκτρο **Enter** ή για όσο χρονικό διάστημα έδωσε ο χρήστης σαν όρισμα στην παρένθεση. Σαν όρισμα δίνεται ένας αριθμός σε δευτερόλεπτα. Αν για παράδειγμα δώσουμε την εντολή

```
>> pause(1);
```

το πρόγραμμα θα κάνει παύσεις ενός δευτερολέπτου μέχρι να εμφανίσει το επόμενο γράφημα.

7.4 Γραφικές Παραστάσεις 3 Διαστάσεων

7.4.1 Η Εντολή meshgrid

Η συνάρτηση-εντολή meshgrid δημιουργεί Πίνακες που ορίζουν το πλέγμα των γραφημάτων. Η meshgrid μπορεί να χρησιμοποιηθεί μόνο για δισδιάστατο ή τρισδιάστατο καρτεσιανό χώρο. Παίρνει για ορίσματα δύο διανύσματα x, y και επιστρέφει δύο Πίνακες X, Y. Οι γραμμές του X είναι αντίγραφα του x και οι στήλες του Y είναι αντίγραφα του y. Η επόμενη εντολή δίνει ένα παράδειγμα χρήσης της meshgrid.

```
>> [X,Y] = meshgrid(1:1:3, 4:1:7)
```

```
X =

   1   2   3
   1   2   3
   1   2   3
   1   2   3


Y =

   4   4   4
   5   5   5
   6   6   6
   7   7   7
```

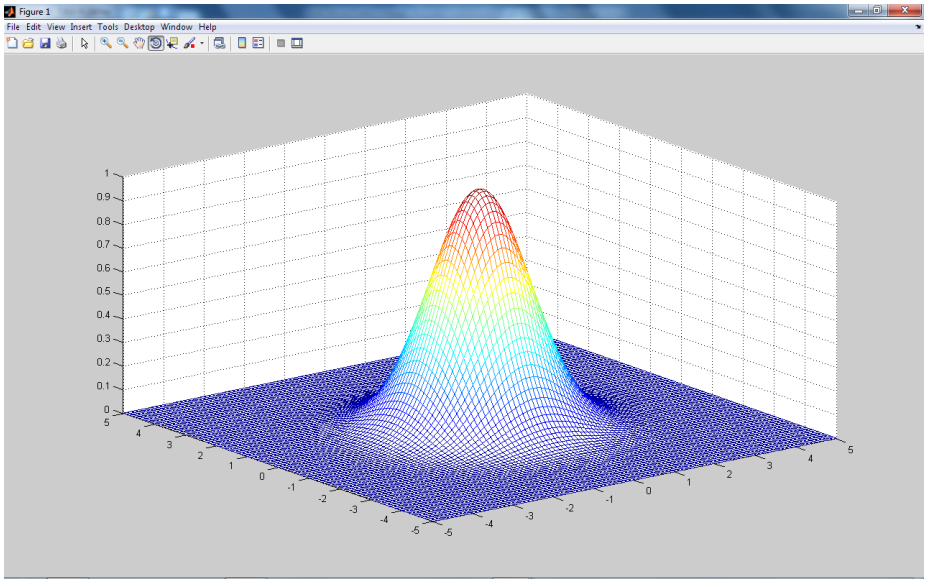
7.4.2 Η Εντολή mesh

Η συνάρτηση mesh σχεδιάζει την γραφική παράσταση μιας επιφάνειας. Παίρνει τρία ορίσματα, δύο Πίνακες που περιέχουν τις συντεταγμένες του οριζόντιου επιπέδου και ένα τρίτο Πίνακα που περιέχει τις συντεταγμένες της επιφάνειας στην τρίτη διάσταση.

Παράδειγμα 14

Για να γίνει το γράφημα της καμπάνας του Gauss θα χρειαστούν οι παρακάτω εντολές, ενώ το αποτέλεσμα φαίνεται στο Γράφημα 7.21.

```
>> [X,Y] = meshgrid(-5:0.1:5, -5:0.1:5);
>> Z = exp(-(X.^2 + Y.^2)/2);
>> mesh(X,Y,Z);
```

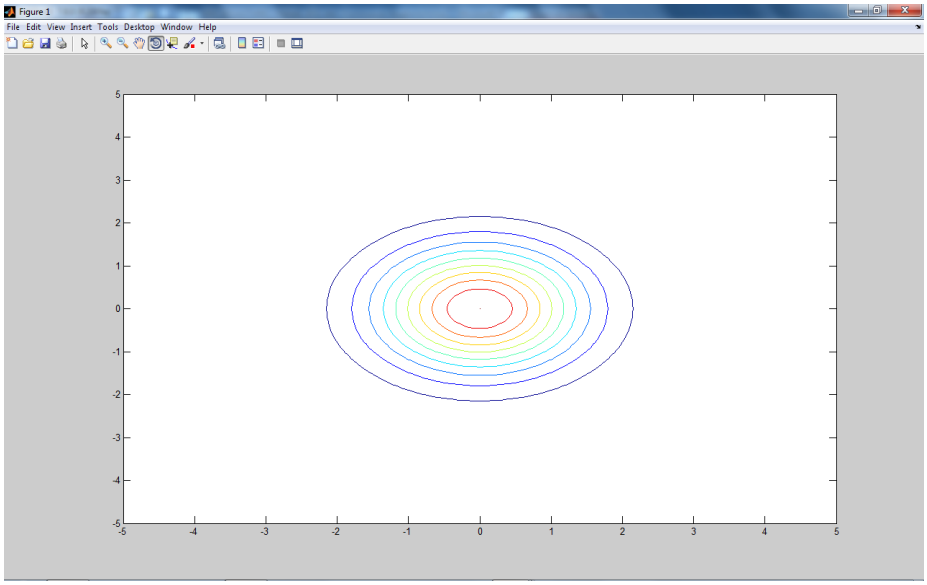


Γράφημα 7.21

7.4.3 Η Εντολή contour

Η συνάρτηση contour σχεδιάζει διαγράμματα ισοϋψών καμπυλών. Αν στο προηγούμενο παράδειγμα χρησιμοποιήσουμε την εντολή **contour** αντί της mesh, θα πάρουμε το παρακάτω γράφημα 7.22 :

```
>> [X,Y] = meshgrid(-5:0.1:5, -5:0.1:5);
>> Z = exp(-(X.^2 + Y.^2)/2);
>> contour(X,Y,Z);
```

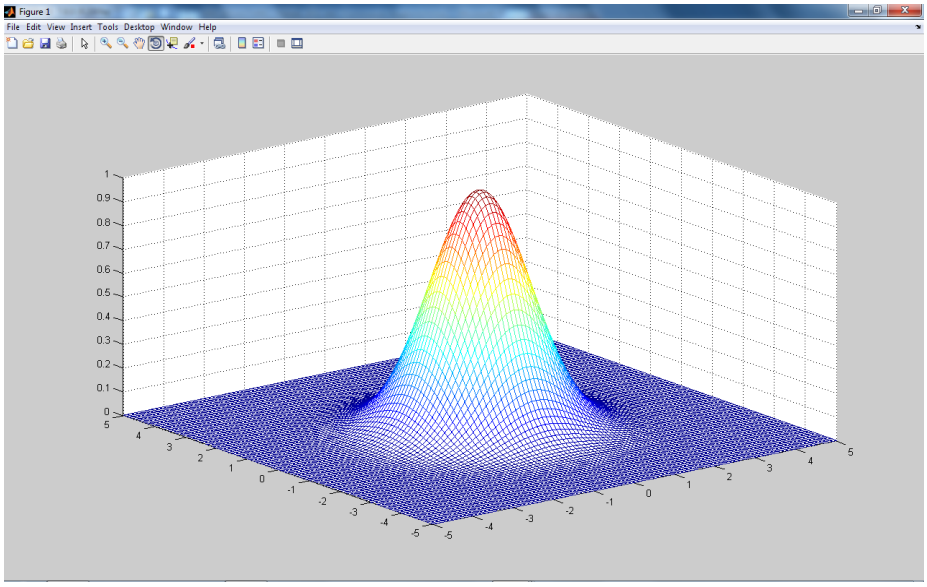


Γράφημα 7.22

7.4.4 Η Εντολή meshc

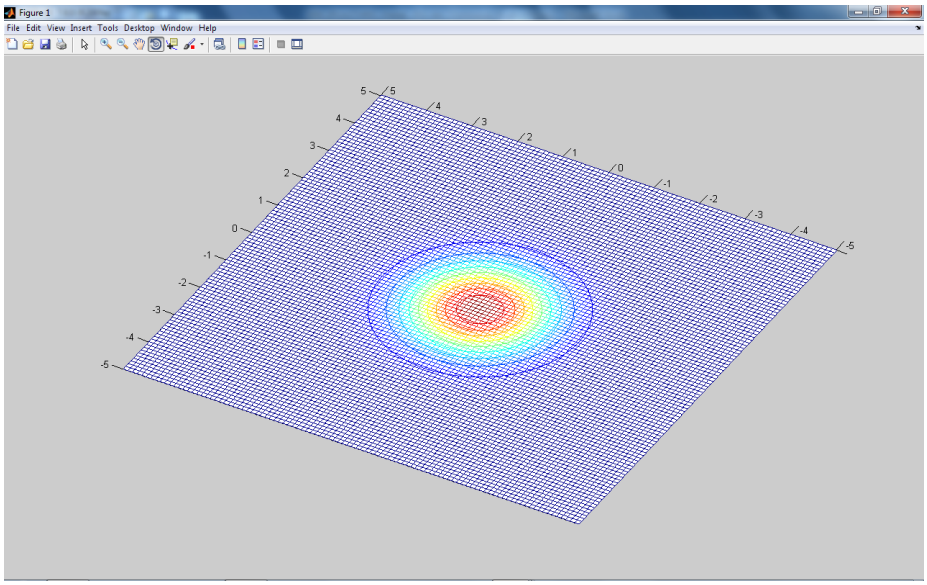
Η meshc είναι απλά ο συνδυασμός της mesh με την contour. Το προηγούμενο παράδειγμα με τη χρήση της **meshc** θα έχει σαν αποτέλεσμα να δημιουργηθεί το γράφημα 7.23.

```
>> [X,Y] = meshgrid(-5:0.1:5, -5:0.1:5);
>> Z = exp(-(X.^2 + Y.^2)/2);
>> meshc(X,Y,Z);
```



Γράφημα 7.23

Αν χρησιμοποιήσουμε την επιλογή Rotate 3D στο Γράφημα, η βάση του γραφήματος θα είναι αυτή που φαίνεται στο 7.24, παρόμοια με αυτή του 7.22.

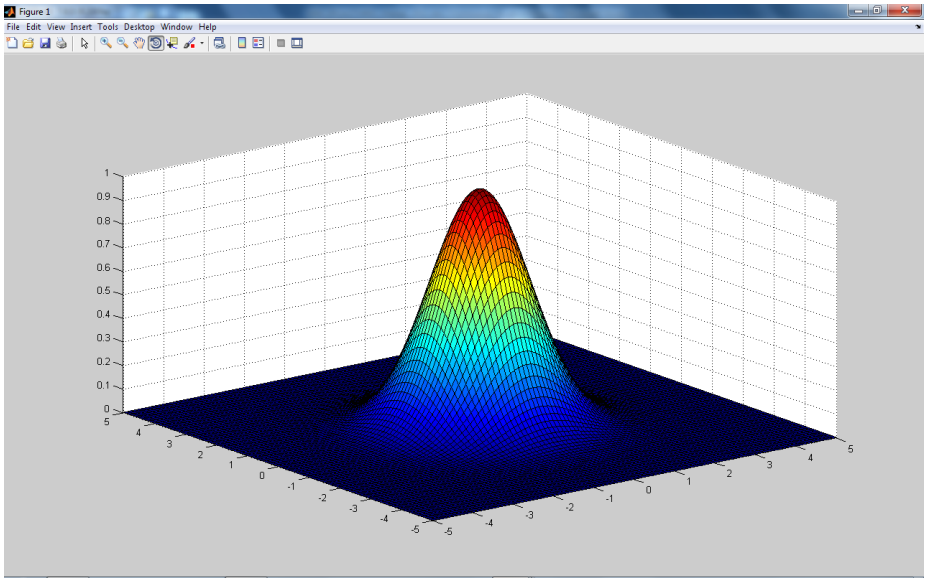


Γράφημα 7.24

7.4.5 Η Εντολή surf

Η surf σχεδιάζει την γραφική παράσταση μιας επιφάνειας με την διαφορά ότι η επιφάνεια θα είναι σκιασμένη. Το προηγούμενο παράδειγμα με τη χρήση της **surf** θα έχει σαν αποτέλεσμα να δημιουργηθεί το γράφημα 7.25.

```
>> [X,Y] = meshgrid(-5:0.1:5, -5:0.1:5);
>> Z = exp(-(X.^2 + Y.^2)/2);
>> surf(X,Y,Z);
```



Γράφημα 7.25