Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ MATLAB

Η βασική δομή δεδομένων στο MATLAB είναι οι πίνακες. Ετσι θα ξεκινήσουμε παρουσιάζοντας πως δημιουργούμε πίνακες και πως υπολογίζονται οι βασικές πράξεις των πινάκων στο MATLAB. Θα αναφερθούμε στις σημαντικότερες συναρτήσεις που αφορούν τη διαχείριση πινάκων.
Σε επόμενη ενότητα θα παρουσιάσουμε τη δυνατότητα δημιουργίας προγραμμάτων και συναρτήσεων. Στην συνέχεια θα παρουσιάσουμε βασικές μαθηματικές συναρτήσεις όπως οι abs, exp, log, imag, real, sum, τριγονωμετρικές συναρτήσεις καθώς και συναρτήσεις για την ανάλυση δεδομένων όπως οι max, min, sort, mean, median και άλλες.
Βλέποντας το MATLAB ως γλώσσα προγραμματισμού θα αναφερθούμε στις βασικές δομές ελέγχου και επανάληψης (if, if else, for, while). Θα δούμε ακόμη συναρτήσεις που παρέχονται από το περιβάλλον για το άνοιγμα και κλείσιμο αρχείων, όπως οι load, save, fopen, fread και fwrite.
Ενα ιδιαίτερα σημαντικό εργαλείο του MATLAB είναι η δυνατότητα που δίνει για τη δημιουργία γραφημάτων και τον εύκολο χειρισμό τους. Παρέχει πλήθος συναρτήσεων για δισδιάστατα και τρισδιάστατα γραφικά. Θα παρουσιάσουμε συναρτήσεις για τον έλεγχο των παραθύρων γραφικών για το χειρισμό των γραφικών αντικείμενων καθώς και τις σημαντικότερες συναρτήσεις για τη δημιουργία γραφημάτων.

Α.1 Βασικές γνώσεις

Η βασική δομή για το MATLAB είναι οι πίνακες. Το MATLAB χειρίζεται τα βαθμωτά μεγέθη ως 1 x 1 πίνακες και τα διανύσματα ως πίνακες 1 x n, όπου n το μήκος του διανύσματος.

Α.1.1 Δήλωση πίνακα

Τα στοιχεία ενός πίνακα μπορεί να είναι αριθμοί ή ολόκληρες αλγεβρικές παραστάσεις. Για παράδειγμα, αν στο prompt του MATLAB πληκτρολογήσουμε

```
» A=[1 2 3 5]
```

και μετά πατήσουμε Enter το MATLAB απαντά ως εξής:

```
A=
   1 2 3 5
```

Παρατηρήστε οτι με το κενό ανάμεσα στα στοιχεία του διανύσματος ξεχωρίζουμε τις στήλες του διανύσματος. Με την παραπάνω δήλωση έχει δεσμευθεί μνήμη για τη μεταβλητή Α και έχει γίνει σε αυτήν ανάθεση τιμής. Αν γράφαμε

```
» A=[1 2 3 5];
```

προσθέταμε δηλαδή και ένα ερωτηματικό στο τέλος της εντολής δεν θα βλέπαμε το αποτέλεσμα της εντολής όπως πριν, παρ'όλο που η μεταβλητή A έχει κρατηθεί και πάλι στη μνήμη. Αυτή η δυνατότητα είναι ιδιαίτερα χρήσιμη σε περιπτώσεις προγραμμάτων όπου υπάρχουν ενδιάμεσα αποτελέσματα που δεν μας ενδιαφέρουν.
Εστω τώρα οτι θέλουμε να ορίσουμε ένα πίνακα B, 3 x 3

```
» B=[2 0 4;5*(sqrt(7)/6) -5 0;1 0 -9]

B =
    2.0000         0    4.0000
    2.2048   -5.0000         0
    1.0000         0   -9.0000
```

Οι στήλες του πίνακα ξεχωρίζουν με το κενό, ενώ οι γραμμές χωρίζονται με το ερωτηματικό (`;`) . Ενας άλλος τρόπος για να ορίσουμε τον πίνακα B είναι

```
» B=[2 0 4
     5*(sqrt(7)/6) -5 0
     1 0 -9]
```

Αντί δηλαδή να χωρίζουμε τις γραμμές με ; αλλάζουμε γραμμή πατώντας Enter.
Οπως έχουμε ήδη αναφέρει τα στοιχεία ενός πίνακα μπορούν να είναι οποιαδήποτε έκφραση του MATLAB. Μπορούμε για παράδειγμα να έχουμε πίνακα με μιγαδικά στοιχεία. Οι μεταβλητές i και j παριστούν την τετραγωνική ρίζα του -1, εκτός και αν τις ορίσουμε διαφορετικά. Ετσι, ένας πίνακας με μιγαδικά στοιχεία μπορεί να δηλωθεί ως εξής:

```
» A = [5-4*i 4+7*i; 2-6*i  7-9*i]
```

ή

```
» A = [5 4;2 7] + i*[-4 7;-6 -9]
```

A.1.2 Πράξεις πινάκων

Πρόσθεση/Αφαίρεση
Η πρόσθεση και αφαίρεση πινάκων είναι πραγματοποιήσιμες εφόσον οι πίνακες έχουν τις ίδιες διαστάσεις. Έστω για παράδειγμα ότι έχουμε δηλώσει κατάλληλα δύο πίνακες A και B, για να υπολογίσουμε το άθροισμα θα γράφαμε:

```
» C = A+B
```

ή αντίστοιχα για τη διαφορά

```
» D = A-B
```

Πολλαπλασιασμός
Ο πολλαπλασιασμός πινάκων, συμβολίζεται με (*), είναι εφικτός εαν το πλήθος των στηλών του πρώτου πίνακα είναι ίσο με το πλήθος των γραμμών του δεύτερου. Ετσι, αν δηλώσουμε δύο πίνακες a και b ως εξής:

```
» a = [1 2 3; 4 5 6];
» b = [0 1; 2 3; 4 5];
```

το γινόμενό τους υπολογίζεται

```
» c = a * b
c =
    16    22
    34    49
```

Διαίρεση
Στο Matlab υπάρχουν δύο σύμβολα που δηλώνουν διαίρεση πινάκων \ και /. Τα δύο αυτά σύμβολα αναφέρονται στην επίλυση των παρακάτω εξισώσεων για κατάλληλα ορισμένους πίνακες A και B:

X = A\B λύση της A*X = B
X = B/A λύση της X*A = B

Για παράδειγμα εαν δηλώσουμε

```
» A=[1 2 3;3 4 0;5 6 0];
» B=[1 4 7;2 4 6;3 5 7];
```

η αριστερή διαίρεση θα δώσει

```
» X=A\B
X =
    0.0000   -2.0000   -4.0000
    0.5000    2.5000    4.5000
    0.0000    0.3333    0.6667
```

Πίνακας σε δύναμη
Εαν ο πίνακας A είναι τετραγωνικός και p είναι ένας αριθμός, με την έκφραση A^p υπολογίζεται η p δύναμη του A. Ειδικότερα εάν ο p είναι ακέραιος και μεγαλύτερος της μονάδας η έκφραση A^p υπολογίζεται με p διαδοχικούς πολλαπλασιασμούς του πίνακα A. Γαι τις άλλες τιμές του p η έκφραση αυτή υπολογίζεται ως εξής

A^p = V*D.^p/V

όπου ο πίνακας D είναι διαγώνιος και τα στοιχεία της διαγωνίου είναι οι ιδιοτιμές του πίνακα A, ενώ οι στήλες του πίνακα V είναι τα αντίστοιχα ιδιοδιανύσματα. Οι δύο αυτοί πίνακες μπορούν να υπολογιστούν με την συνάρτηση eig() ως εξής:

[V,D] = eig(A)

Ανάστροφος πίνακας
Ο ανάστροφος ενός πίνακα X, είναι ο πίνακας που προκύπτει αν οι γραμμές του X γίνουν στήλες. Ο υπολογισμός του αναστρόφου γίνεται χρησιμοποιώντας την απόστροφο (').

```
» X = [1 2 3;4 5 6;7 8 9];
» D = X';
```

έχουμε δημιουργήσει τους εξής πίνακες

```
X =                    D =
     1 2 3                  1 4 7
     4 5 6                  2 5 8
     7 8 9                  3 6 9
```

Αντίστροφος πίνακα
Στο MATLAB μπορούμε εύκολα να υπολογίσουμε τον αντίστροφο ενός πίνακα με την συνάρτηση **inv(A)** όπου ο A είναι ένας τετραγωνικός πίνακας.
Ετσι αν ο A είναι ένας 3x3 πίνακας, ο B είναι ένας 3x1 και ο C ένας 3x1 πίνακας η λύση του συστήματος AX = C υπολογίζεται στο MATLAB με μια εντολή

X = inv(A)*C

Ορίζουσα πίνακα
Ο υπολογισμός της ορίζουσας ενός πίνακα στο MATLAB γίνεται με την συνάρτηση **det(A)** όπου ο A είναι ένας τετραγωνικός πίνακας.

Κατά σημείο πράξεις
Στο Matlab ορίζονται τελεστές που επιτρέπουν πράξεις στα αντίστοιχα στοιχεία των πινάκων στους οποίους εφαρμόζονται. Οι τελεστές είναι:
`.*` κατά σημείο πολλαπλασιασμός
`./` κατά σημείο διαίρεση
`.^` κατά σημείο ύψωση σε δύναμη

Για παράδειγμα αν δηλώσουμε δύο πίνακες

```
» A=[1 2 3; 4 5 6];
» B=[-1 1 2; 3 4 5];
```

και εφαρμόσουμε τον κατά σημείο πολλαπλασιασμό, παίρνουμε

```
» C = A.*B
C =
    -1     2     6
    12    20    30
```

Η πράξη εφαρμόζεται στα αντίστοιχα στοιχεία των πινάκων. Σημειώνουμε ότι οι πίνακες θα πρέπει να έχουν την ίδια διάσταση, εκτός και αν ένας από τους δύο είναι αριθμός.

Α.2 Χειρισμός διανυσμάτων και πινάκων

Στην ενότητα αυτή θα δούμε κάποιες δυνατότητες του MATLAB για τη δημιουργία διανυσμάτων, τον χειρισμό και την επεξεργασία των γραμμών, των στηλών και των υποτμημάτων ενός πίνακα.

Α.2.1 Δημιουργώντας διανύσματα

Μέχρι τώρα έχουμε δημιουργήσει διανύσματα με δηλώσεις της μορφής

```
» x = [0 1 2 -2 4]
```

Ενας άλλος τρόπος για τη δημιουργία διανυσμάτων είναι με τη χρήση του στίγματος (colon), **:**,το οποίο χρησιμοποιείται ως εξής:

x = first**:**step**:**last

όπου first είναι η τιμή του πρώτου στοιχείου του διανύσματος, η μεταβλητή step είναι το βήμα αύξησης ή ελάττωσης των επόμενων στοιχείων του x. Τα στοιχεία του διανύσματος μεταβάλλονται κατά step έως την τιμή last. Ας δούμε μερικά παραδείγματα:

```
» b = 20:-5:-10
b =
   20   15   10   5   0   -5   -10

» c = pi/4:0.5:pi
c =
   0.7854   1.2854   1.7854   2.2854   2.7854
```

Παρατηρήστε ότι η αρχική, η τελική τιμή και το βήμα μπορεί να είναι ακέραιοι ή πραγματικοί αριθμοί. Αν η τιμή του βήματος μεταβολής παραληφθεί, τότε λαμβάνεται εξ'όρισμού ως μονάδα.
Ενας άλλος τρόπος για να δημιουργούμε διανύσματα είναι με την συνάρτηση **linspace**(first,last,N). Η συνάρτηση δέχεται τρία ορίσματα. Τα δύο πρώτα, first και last, προσδιορίζουν την τιμή του πρώτου και του τελευταίου στοιχείου του διανύσματος. Το τελευταίο όρισμα, N, καθορίζει το πλήθος τον στοιχείων του διανύσματος. Η συνάρτηση θα δημιουργήσει ένα διάνυσμα με N στοιχεία στο διάστημα [first,last] που θα ισαπέχουν μεταξύ τους. Αν η μεταβλητή N παραληφθεί η τιμή της λαμβάνεται ίση με 100.
Στο παράδειγμα που ακολουθεί, η linspace δημιούργησε ένα διάνυσμα με 10 στοιχεία από το -2 έως το 2 σε γραμμική κλίμακα:

```
» a = linspace(-2,2,10)
a =
   Columns 1 through 7
   -2.0000 -1.5556 -1.1111 -0.6667 -0.2222 0.2222 0.6667
   Columns 8 through 10
    1.1111  1.5556  2.0000
```

Μια ακόμη συνάρτηση για την δημιουργία διανυσμάτων είναι η **logspace**(v1,v2,N). Η logspace δημιουργεί ένα διάνυσμα με N το πλήθος στοιχεία στο διάστημα $[10^{v1},10^{v2}]$ με λογαριθμική κλίμακα. Στην περίπτωση που δεν δηλώσουμε τιμή για τη μεταβλητή N, θα θεωρηθεί ίση με 50.

Στο επόμενο παράδειγμα δημιουργούμε ένα διάνυσμα a 100 θέσεων, οι τιμές του οποίου βρίσκονται στο διάστημα [-pi,pi].

```
» a = linspace(-pi,pi,100);
» s = sin(a);
```

Η μεταβλητή s είναι ένα διάνυσμα. Το πρώτο στοιχείο του διανύσματος, s(1), είναι το ημίτονο του πρώτου στοιχείου του διανύσματος a, a(1). Αντίστοιχα το s(2) υπολογίζεται από τη σχέση sin(a(2)) κ.ο.κ. Δηλαδή το διάνυσμα s παράγεται ως εξής:

s = (sin(a(1)) sin(a(2)) ... sin(a(100)))

Με τις εντολές που ακολουθούν δημιουργούμε ένα πίνακα. Η πρώτη στήλη του είναι ένα διάνυσμα στήλη x και η δεύτερη στήλη του είναι ένα διάνυσμα y το οποίο έχει υπολογιστεί από μια συνάρτηση του x.

```
» x = (0:0.2:1)';
» y = exp(-x).*sin(x);
» z = [x y]
z =
    0         0
    0.2000    0.1627
    0.4000    0.2610
    0.6000    0.3099
    0.8000    0.3223
    1.0000    0.3096
```

Δύο ακόμα βασικές συναρτήσεις του MATLAB για τη δημιουργία διανυσμάτων είναι οι **ones(), zeros()** και **eye()**. Η συναρτήσεις αυτές καλούνται με ένα ή δύο ορίσματα ως εξής:

- ones(N) δημιουργεί έναν NxN πίνακα όλο μονάδες
- ones(N,M) ή ones([N,M]) δημιουργεί έναν NxM πίνακα όλο μονάδες

Η συνάρτηση zeros() καλείται με τον ίδιο τρόπο με την ones() και δημιουργεί πίνακα με όλα τα στοιχεία του μηδενικά.
Αν για παράδειγμα θέλουμε να δημιουργήσουμε ένα διάνυσμα όπου κάθε στοιχείο του θα έχει τιμή -3 γράφουμε

```
» c = -3*ones(1,10)
c =
   -3   -3   -3   -3   -3   -3   -3   -3   -3   -3
```

Η συνάρτηση **eye**(N) δημιουργεί έναν NxN ταυτοτικό πίνακα. Αν κληθεί με δύο ορίσματα eye(N,M) δημιουργεί έναν NxM πίνακα με όλα τα στοιχεία του μηδενικά εκτός από τα διαγώνια στοιχεία που είναι μονάδες. Για παράδειγμα

```
» X = eye(4,3)
X =
   1   0   0
   0   1   0
   0   0   1
   0   0   0
```

Α.2.2 Αναφορά στα στοιχεία ενός πίνακα

Εστω ότι έχουμε δηλώσει έναν πίνακα Α ως εξής

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

Μπορούμε να αναφερόμαστε στα στοιχεία του πίνακα δίνοντας το όνομα του πίνακα και μέσα σε παρένθεση τη θέση του στοιχείου που μας ενδιαφέρει. Στο MATLAB η αρίθμηση των στοιχείων του πίνακα ξεκινάει από το ένα.
Ετσι αν θέλουμε την τιμή του στοιχείου που βρίσκεται στη δεύτερη γραμμή και στην τρίτη στήλη του πίνακα θα γράφαμε

```
» A(2,3)
ans =
     6
```

Το MATLAB μας δίνει τη δυνατότητα να αναφερθούμε σε τμήματα του πίνακα. Για παράδειγμα

```
» b = A(1:2,3)
b =
    3
    6
```

Το b είναι ένα διάνυσμα στήλη το οποίο αποτελείται από τα δύο πρώτα στοιχεία της τρίτης στήλης του Α.
Ομοίως αν δώσουμε

```
» B = A(2:3,1:2)
B =
```

```
    4   5
   -1   8
```

ο Β είναι ένας 2x2 πίνακας που προκύπτει από την δεύτερη και τρίτη γραμμή και από την πρώτη και δεύτερη στήλη του Α.
Με τη δήλωση

```
A(:,3)
```

λαμβάνουμε την τρίτη στήλη του Α, ενώ αν δώσουμε

```
A(1:2, :)
```

αναφερόμαστε στις δύο πρώτες γραμμές του πίνακα.
Γενικά αν τα v και w είναι διανύσματα με ακέραια στοιχεία, τότε ο

```
A(v,w)
```

είναι ο πίνακας που λαμβάνεται παίρνοντας από τον Α τα στοιχεία γραμμής που προσδιορίζονται από το v και τα στοιχεία στήλης που προσδιορίζονται από το διάνυσμα w. Ετσι με την εντολή

```
» B = A(:,3:-1:1)
B =
    3   2   1
    6   5   4
    9   8  -1
```

έχουμε αντιστρέψει τη σειρά των στηλών του Α.
Στο παράδειγμα που ακολουθεί κατασκευάζουμε ένα διάνυσμα στήλη που αποτελείται από όλα τα στοιχεία του πίνακα C.

```
» C = [1 2;3 4;5 6]
C =
    1   2
    3   4
    5   6

» b = C(:)
b =
    1
    3
    5
    2
    4
    6
```

Με την προϋπόθεση ότι ο πίνακας C έχει ήδη οριστεί, μπορούμε να τον ξανασχηματίσουμε με μια δήλωση της μορφής

```
» C(:) = 10:15
C =
   10  13
   11  14
   12  15
```

Ο πίνακας C(:) υποδηλώνει ένα πίνακα με διαστάσεις ίδιες με αυτές του C αλλά με νέα στοιχεία.
Μπορούμε επίσης να προσθέσουμε γραμμές η στήλες σε ένα πίνακα. Εστω ένας 3x3 πίνακας

```
» A = [0 1 2;3 4 5;6 7 8]
A =
   0  1  2
   3  4  5
   6  7  8
```

μπορούμε να προσθέσουμε μια γραμμή στον πίνακα ως εξής

```
» A = [A;0 0 0]
A =
   0  1  2
   3  4  5
   6  7  8
   0  0  0
```

Για να προσθέσουμε μια στήλη γράφουμε

```
» A(:,4) = (1:4)'
A =
   0  1  2  1
   3  4  5  2
   6  7  8  3
   0  0  0  4
```

Παρατηρήστε ότι αν θέλουμε να προσθέσουμε στον πίνακα A το διάνυσμα (1 2 3 4)΄ στην έβδομη στήλη η πέμπτη και έκτη στήλη θα γεμίσει με μηδενικά.

```
» A(:,7) = (1:4)'
A =
   0  1  2  1  0  0  1
   3  4  5  2  0  0  2
   6  7  8  3  0  0  3
   0  0  0  4  0  0  4
```

Μπορούμε ακόμη να αντικαταστήσουμε τις στήλες ενός πίνακα με άλλες τιμές. Για παράδειγμα αν γράψουμε

```
» B = ones(4,3);
» A(:,[2 4 6]) = ones(4,3)
```

```
A =
   0   1   2   1   0   1   1
   3   1   5   1   0   1   2
   6   1   8   1   0   1   3
   0   1   0   1   0   1   4
```

έχουμε αντικαταστήσει τη δεύτερη, τέταρτη και έκτη στήλη του Α με τα στοιχεία του πίνακα Β.
Η επιλογή ενός τμήματος από ένα πίνακα μπορεί να γίνει με ένα διάνυσμα που αποτελείται από μηδενικά και μονάδες. Αν ο Α είναι ένας MxN πίνακας και L είναι ένα διάνυσμα μήκους M αποτελούμενο από 0 και 1, τότε ο πίνακας που προκύπτει από τη δήλωση

A(L,:)

αποτελείται από τις εκείνες τις γραμμές του Α για τις οποίες τα αντίστοιχα στοιχεία του L δεν είναι μηδενικά. Στο παράδειγμα που ακολουθεί έχουμε αντικαταστήσει τον πίνακα Α με εκείνες τις γραμμές του πίνακα για τις οποίες η δεύτερη στήλη είναι μεγαλύτερη από 4.5.

```
» A = [1 2 3; 4 5 6;7 8 9]
A =
   1   2   3
   4   5   6
   7   8   9
» L = A(:,2) > 4.5
L =
   0
   1
   1
» A = A(L,:)
A =
   4   5   6
   7   8   9
```

Μέχρι τώρα έχουμε δει πως προσθέτουμε γραμμές και στήλες σε ένα πίνακα. Ας δούμε τώρα πως μπορούμε να διαγράψουμε γραμμές και στήλες από ένα πίνακα.
Στο παράδειγμα που ακολουθεί διαγράφουμε την δεύτερη και τέταρτη γραμμή του πίνακα Α.

```
» A = [1 2 3 ; 4 5 6; 7 8 9; 10 11 12; 13 14 15]
A =
    1   2   3
    4   5   6
    7   8   9
   10  11  12
   13  14  15

» A([2 4],:) = []
A =
```

```
     1   2   3
     7   8   9
    13  14  15
```

Παρατηρήστε ότι η διαγραφή έγινε αντιστοιχώντας σε κάποιες γραμμές και στήλες το σύμβολο []. Γενικά με μια δήλωση της μορφής

x = []

δημιουργούμε έναν κενό πίνακα με διαστάσεις 0x0. Παρατηρήστε ότι η παραπάνω δήλωση είναι διαφορετική από τη δήλωση

clear x

η οποία διαγράφει τη μεταβλητή x. Αν δώσουμε

```
» whos
      Name    Size    Elements  Bytes  Density  Complex
       x     0 by 0       0       0      Full      No
      Grand total is 0 elements using 0 bytes
```

Δηλαδή η μεταβλητή x υπάρχει στη λίστα των μεταβλητών.
Θα δούμε αργότερα πως ένας κενός πίνακας μπορεί να χρησιμοποιηθεί ως αρχική τιμή σε έναν πίνακα, ο οποίος θα γεμίσει μέσα σε ένα βρόγχο επανάληψης.

Α.2.3 Βασικές συναρτήσεις για τη διαχείριση πινάκων

Ολοκληρώνοντας αυτή την ενότητα θα παρουσιάσουμε ακόμη κάποιες βασικές συναρτήσεις του MATLAB για τη διαχείριση των στοιχείων ενός πίνακα.
Η συνάρτηση **fliplr**(X) μεταθέτει τις στήλες του πίνακα X. Ας δουμε ένα παράδειγμα.

```
» X = [1 2 3 4;5 6 7 8]
X =
   1   2   3   4
   5   6   7   8

» Y = fliplr(X)
Y =
   4   3   2   1
   8   7   6   5
```

Παρατηρήστε ότι οι στήλες μετακινήθηκαν κατάλληλα ώστε η πρώτη στήλη να γίνει η τελευταία και η δεύτερη στήλη να γίνει η τρίτη. Όμοια η συνάρτηση **flipud**(X) αναστρέφει τις γραμμές του πίνακα X.

Αν ο X είναι ένας mxn πίνακας η συνάρτηση **rot90**(X) θα δημιουργήσει έναν nxm πίνακα ο οποίος προκύπτει από την περισττροφή του X κατά 90 μοίρες. Η συνάρτηση

rot90 αν κληθεί με δύο ορίσματα, rot90(X,k), θα δημιουργήσει έναν nxm πίνακα με περιστροφή του X κατά k*90 μοίρες με k = +-1, +-2, ... Για παράδειγμα

```
» X = [1 2;3 4;5 6]
X =
    1   2
    3   4
    5   6

» Y = rot90(X)
Y =
    2   4   6
    1   3   5
```

Μια συνάρτηση η οποία επιτρέπει να αλλάξουμε τις διαστάσεις ενός πίνακα είναι η **reshape**(X,M,N). Η συνάρτηση δημιουργεί έναν MxN πίνακα λαμβάνοντας στοιχεία από τον πίνακα X κατά στήλες. Ο πίνακας X θα πρέπει να έχει M*N το πλήθος στοιχεία. Για παράδειγμα:

```
» X = [1 2;3  4;5  6]
X =
    1   2
    3   4
    5   6

» Y = reshape(X,2,3)
Y =
    1   5   4
    3   2   6

» C = reshape(X,1,6)

C =
    1   3   5   2   4   6
```

Με τη συνάρτηση **diag**(A,k) μπορούμε να δημιουργήσουμε διαγώνιους πίνακες ή να λάβουμε τα διαγώνια στοιχεία ενός πίνακα. Αν το A είναι είναι ένα διάνυσμα μήκους n και k ένας αριθμός η κλήση της συνάρτησης, diag(A,k), δημιουργεί ένα τετραγωνικό πίνακα τάξης n+abs(k) με τα στοιχεία του A στην k διαγώνιο. Αν k=0 τα στοιχεία του A θα βρίσκονται στην κύρια διαγώνιο του πίνακα, αν k>0 θα βίσκονται πάνω απο την κύρια διαγώνιο ενώ αν k<0 θα βρίσκονται κάτω από την κύρια διαγώνιο. Η κλήση της συνάρτησης χωρίς την παράμετρο k είναι ισοδύναμη με την περίπτωση όπου k=0. Ας δούμε ένα παράδειγμα δημιουργίας διαγώνιων πινάκων.

```
» D = diag(-1:2,-2)
D =
     0   0   0   0   0   0
     0   0   0   0   0   0
    -1   0   0   0   0   0
     0   0   0   0   0   0
     0   0   1   0   0   0
```

```
     0   0   0   2   0   0

» D = diag(-1:2,1)
D =
     0  -1   0   0   0
     0   0   0   0   0
     0   0   0   1   0
     0   0   0   0   2
     0   0   0   0   0
```

Η ίδια συνάρτηση μορεί να χρησιμοποιηθεί και στην περίπτωση που θέλουμε να εξάγουμε τα διαγώνια στοιχεία ενός πίνακα. Ετσι αν ο X είναι ένας πίνακας τότε η diag(X,k) επιστρέφει τα στοιχεία της k διαγωνίου του πίνακα X σε ένα διάνυσμα στήλη. Για παράδειγμα

```
» X = [1 2 3;4 5 6; 7 8 9]
X =
    1   2   3
    4   5   6
    7   8   9

» v = diag(X)
v =
    1
    5
    9

» v = diag(X,-1)
v =
    4
    8
```

Παρατηρήστε ότι με τη δήλωση diag(diag(X)) δημιουργούμε ένα διαγώνιο πίνακα με τα στοιχεία της κυρίας διαγωνίου του X.

```
» diag(diag(X))
ans =
     1   0   0
     0   5   0
     0   0   9
```

Η συνάρτηση **size**(X) επιστρέφει τις διαστάσεις του πίνακα X. Αν καλέσουμε τη συνάρτηση με μια επιπλέον παράμετρο size(X,1) θα επιστέψει το πλήθος των γραμμών του πίνακα, ενώ αν καλέσουμε τη συνάρτηση με δεύτερη παράμετρο τον αριθμό 2, size(X,2), θα επιστρέψει το πλήθος των στηλών του X. Ετσι για τον παράνω πίνακα X έχουμε

```
» size(X)
ans =
     3   3
```

Η συνάρτηση **length**(X) επιστρέφει το μήκος ενός διανύσματος X. Αν της δοθεί ως παράμετρος ένας πίνακας τότε επιστρέφει τη μέγιστη διάσταση του πίνακα.
Η συνάρτηση **isempty** χρησιμοποιείται για να ελέγξουμε αν ένας πίνακας είναι κενός ή όχι. Δέχεται ως παράμετρο το όνομα μια μεταβλητής, isempty(A), επιστρέφει 1 αν ο A είναι κενός πίνακας και 0 στην αντίθετη περίπτωση. Ενας πίνακας θεωρείται κενός αν τουλάχιστον μια διαστασή του είναι μηδέν.
Η συνάρτηση **exist**('A') ελέγχει την ύπαρξη ή όχι μιας μεταβλητής. Επιστρέφει 0 αν η μεταβλητή A δεν υπάρχει και 1 αν η μεταβλητή υπάρχει στο χώρο εργασίας. Η συνάρτηση μπορεί να χρησιμοποιηθεί και για τον έλεγχο ύπαρξης μιας συνάρτησης.

Α.3 Δημιουργία προγραμμάτων και συναρτήσεων

Μέχρι τώρα δίνουμε όλες τις εντολές στο Command Window του MATLAB. Μπορούμε να γράψουμε αυτές τις εντολές σε ένα αρχείο κάνοντας έτσι τον κώδικά μας επαναχρησιμοποιήσιμο. Ετσι αντί να δίνουμε τις εντολές στο command prompt τις γράφουμε σε ένα ξεχωριστό text αρχείο. Το όνομα αυτού του αρχείου θα πρέπει να έχει επέκταση .m . Κατά την εκτέλεση ενός τέτοιου αρχείου το MATLAB εκτελεί τις εντολές με τη σειρά που τις βρίσκει στο αρχείο με τον ίδιο τρόπο που θα τις εκτελούσε αν είχαν δοθεί στο command prompt. Απο το menu File επιλέγουμε New και κατόπιν M-file. Στο παράθυρο που εμφανίζεται γράφουμε τις εντολές και σώζουμε το αρχείο μας. Ας δούμε ένα παράδειγμα.

```
% res.m
% Υπολογισμός ολικής αντίστασης
% Το κύκλωμα αποτελείται από τρείς αντιστάσεις
% συνδεδεμένες παράλληλα
r = [10 20 5];
r = 1./r;
R = 1/(r(1)+r(2)+r(3))
```

Για να εκτελέσουμε το παραπάνω πρόγραμμα δίνουμε στο prompt του MATLAB το όνομα του αρχείου χωρίς την επέκταση .m.

```
» res
R =
   2.8571
```

Παρατηρήστε ότι η τιμή της μεταβλητής R εμφανίζεται στην οθόνη. Αυτό γίνεται γιατί δεν έχουμε χρησιμοποιήσει το ερωτηματικό μετά τον υπολογισμό της μεταβλητής. Μετά την εκτέλεση του προγράμματος στο χώρο εργασίας υπάρχουν οι μεταβλητές που χρησιμοποιήθηκαν από το πρόγραμμα

```
» who
Your variables are:
R   r
```

Το σύμβολο % δηλώνει πως ο,τι ακολουθεί μέχρι το τέλος της γραμμής είναι σχόλιο.

Πολλές φορές είναι χρήσιμο σε ένα πρόγραμμα να δίνονται τα δεδομένα από τον χρήστη. Αυτό μπορεί να γίνει με χρήση της **input**. Ετσι το πρόγραμμα res.m μπορεί να γραφεί ως εξής

```
% res.m
% Υπολογισμός ολικής αντίστασης
% Το κύκλωμα αποτελείται από τρείς αντιστάσεις
% συνδεδεμένες παράλληλα
r = input('enter values for r in brackets: ');
r = 1./r;
R = 1/(r(1)+r(2)+r(3))
```

Μόλις τρέξουμε το πρόγραμμα η πρώτη εντολή που συναντα το MATLAB είναι

```
r = input('enter values for r in brackets: ');
```

Ετσι θα δούμε στην οθόνη το μήνυμα

```
» res
enter values for r in brackets:
```

και το πρόγραμμα θα περιμένει να πάρει τιμές από τον χρήστη. Παρατηρήστε ότι αφού δώσουμε τιμές το περιχόμενο του διανύσματος r δεν εμφανίζεται στην οθόνη και αυτό γιατί η εντολή με την οποία λαμβάνει τιμή η μεταβλητή r τελειώνει με ερωτηματικό.

```
» res
enter values for r in brackets: [3 10 7]
R =
   1.7355
```

Μια άλλη συνάρτηση που αφορά την έξοδο των δεδομένων είναι η **disp**(X).Αν η μεταβλητή X είναι ένας πίνακας η συνάρτηση εμφανίζει στην οθόνη το περιεχόμενό του χωρίς το όνομα να εμφανίζει το όνομα της μεταβλητής. Σε περίπτωση που η μεταβλητή είναι κάποιο string το τυπώνει στην οθόνη. Ετσι το παραπάνω πρόγραμμα θα μπορούσε να τυπώνει και ένα μήνυμα ως εξής

```
% res.m
r = input('enter values for r in brackets: ');
r = 1./r;
R = 1/(r(1)+r(2)+r(3));
disp('Total resistance'); disp(R);
```

Αν τρέξουμε και πάλι το πρόγραμμα res.m θα έχουμε

```
» res
enter values for r in brackets: [20 10 5]
Total resistence
    2.8571
```

Εως τώρα έχουμε χρησιμοποιήσει αρκετές συναρτήσεις του MATLAB. Για παράδειγμα τη συνάρτηση ones(m,n). Η συνάρτηση αυτή δέχεται δύο εισόδους και επιστρέφει έναν πίνακα. Μπορούμε να ορίσουμε δικές μας συναρτήσεις. Ας δούμε πως θα γράφαμε το προηγούμενο πρόγραμμα ως μια συνάρτηση η οποία θα δέχεται ως είσοδο ένα διάνυσμα με τις τιμές των αντιστάσεων και θα επιστρέφει την ολική αντίσταση του κυκλώματος.

```
function  R = resistance(r)
% Υπολογισμός ολικής αντίστασης
% Το κύκλωμα αποτελείται από τρείς αντιστάσεις
% συνδεδεμένες παράλληλα
r = 1./r;
R = 1/(r(1)+r(2)+r(3));
```

Παρατηρήστε τη δεσμευμένη λέξη function στην αρχή του προγράμματος. Η χρήση της είναι υποχρεωτική κάθε φορά που θέλουμε να δηλώσουμε μια συνάρτηση. Η συνάρτηση resistance έχει μία είσοδο και μια έξοδο. Θα μπορούσαμε να έχουμε όσες εισόδους και εξόδους θέλουμε. Θα πρέπει μόνο να προσέχουμε κάθε φορά να καλούμε την συνάρτηση κατάλληλα. Το παραπάνω πρόγραμμα το σώζουμε σε ένα αρχείο με όνομα resistance.m.
Για το παραπάνω παράδειγμα η κλήση της συνάρτησης γίνεται ως εξής

```
» t = [20 10 5];
» T = resistance(t)
T =
    2.8571
```

Παρατηρήστε ακόμη ότι οι μεταβλητές r και R που χρησιμοποιούνται μέσα στη συνάρτηση είναι τοπικές, έχουν ισχύ μόνο μέσα στη συνάρτηση. Ετσι μετά την κλήση της συνάρτησης οι μεταβλητές που υπάρχουν στο χώρο εργασίας είναι οι T και t.

```
» who
Your variables are:
 T     t
```

Θα μπορούσαμε να είχαμε ονομάσει τις μεταβλητές R και r αντι T και t, όπως δηλαδή εμφανίζονται και στον ορισμό της συνάρτησης.
Γενικά ένα αρχείο στο οποίο θέλουμε να ορίσουμε μια συνάρτηση θα πρέπει να ξεκινάει με μια δήλωση της μορφής

function return_value = function_name(input)

Τα σχόλια ειδικά στην περίπτωση των συναρτήσεων είναι σημαντικό να υπάρχουν. Δίνοντας help και το όνομα της συνάρτησης το MATLAB μας δίνει τα σχόλια που υπάρχουν στο αρχείο. Είναι σημαντικό να παρατηρήσουμε πως η μόνη πληροφορία που επιστρέφει η συνάρτηση περιέχεται στη μεταβλητή return_value. Ετσι πρέπει να προσέχουμε να λαμβάνει πάντα τιμή η μεταβλητή αυτή.

Αν θέλουμε η συνάρτηση να έχει πολλές επιστρεφόμενες τιμές θα πρέπει να δηλωθεί ως εξής

function [out1,out2,...,outM] = function_name(input)

αν θέλουμε να έχει πολλές εισόδους θα πρέπει αυτές να αναφέρονται στη δήλωση της

function output = function_name(in1, in2,...,inN)

Είναι χρήσιμο να χρησιμοποιούμε τις συναρτήσεις **nargin** και **nargout** για τον έλεγχο του πλήθους των ορισμάτων εισόδου και εξόδου αντίστοιχα.

Σημειώνουμε ότι σε ότι αφορά στα M-Files που εμείς φτιάχνουμε θα πρέπει να προσέχουμε να τα τρέχουμε από τον κατάλογο στον οποίο είναι αποθηκευμένα. Η συνάρτηση **pwd** επιτρέφει το path στο οποίο βρισκόμαστε. Η συνάρτηση **what** επιστρέφει τα M-files που υπάρχουν στον τρέχοντα κατάλογο ενώ η συνάρτηση **which** ακολουθούμενη από το όνομα ενός αρχείου μας επιστρέφει το path στο οποίο βρίσκεται το συγκεκριμένο M-file.

A.4 Βασικές μαθηματικές συναρτήσεις

Στην ενότητα αυτή θα παρουσιάσουμε κάποιες στοιχειώδεις μαθηματικές συναρτήσεις όπως οι abs() και sqrt(), καθώς και τις τριγωνομετρικές συναρτήσεις που παρέχονται από το MATLAB. Θα δούμε ακόμη πως παράγουμε τυχαίους αριθμούς με τις συναρτήσεις rand και randn.

Οι πιο συνηθισμένες συναρτήσεις που χρησιμοποιούμε σε απλούς μαθηματικούς υπολογισμούς είναι οι ακόλουθες:

- **abs**(X)

Υπολογίζει την απόλυτη τιμή των στοιχείων του X. Αν ο X αποτελείται από μιγαδικούς αριθμούς τότε η συνάρτηση επιστρέφει το μέτρο (magnitude) των στοιχείων του X. Αν ο X είναι ένα string, τότε η συναρτηση επιστρέφει τις αντίστοιχες αριθμητικές τιμές των ASCII χαρακτήρων του αλφαριθμητικού.Για παράδειγμα

```
» X = [0 -1 -2 ;3 -4 -5]
X =
   0  -1  -2
   3  -4  -5

» abs(X)
ans =
     0  1  2
     3  4  5
```

στην περίπτωση που ο X περιέχει μιγαδικές τιμές

```
» X = [-1 1]+i*[-2 3]
X =
   -1.0000 - 2.0000i   1.0000 + 3.0000i

» abs(X)
ans =
     2.2361    3.1623
```

στην περίπτωση του αλφαριθμητικού έχουμε

```
» abs('abc')
ans =
     97  98  99
```

- **sqrt**(X)

Η συνάρτηση αυτή υπολογίζει την τετραγωνική ρίζα των στοιχείων του πίνακα X. Σε περίπτωση που κάποιο από τα στοιχεία του πίνακα είναι αρνητικό θα επιστρέψει μιγαδικές τιμές. Γαι παράδειγμα

```
» X = [-1 4; -4 1]
X =
   -1   4
   -4   1

» sqrt(X)
ans =
     0 + 1.0000i   2.0000
     0 + 2.0000i   1.0000
```

- **round**(X)

Η συνάρτηση round(X) κάνει στρογγύλευση των στοιχείων του πίνακα X προς τον πλησιέστερο ακέραιο.

- **fix**(X)

Η συνάρτηση αυτή κάνει στρογγύλευση (αποκοπή) των στοιχείων του X προς τον πλησιέστερο ακέραιο προς το μηδέν.

- **floor**(X)

Η συνάρτηση floor κάνει στρογγύλευση των στοιχείων του X προς τον πλησιέστερο ακέραιο προς το $-\infty$.

- **ceil**(X)

Η συνάρτηση ceil κάνει στρογγύλευση των στοιχείων του X προς τον πλησιέστερο ακέραιο προς το ∞.

Ας δούμε κάποια απλά παραδείγματα για τις παραπάνω συναρτήσεις στρογγύλευσης

```
» round(-2.6)
ans =
     -3
```

```
» fix(-2.6)
ans =
     -2

» floor(-2.6)
ans =
     -3

» ceil(-2.6)
ans =
     -2

» floor(ceil(10.8))
ans =
     11
```

- **sign**(X)

Για κάθε στοιχείο του πίνακα X η συνάρτηση sign επιστρέφει 1 αν το στοιχείο είναι θετικός αριθμός, 0 αν το στοιχείο έχει τιμή μηδέν και -1 αν είναι αρνητικός. Στην περίπτωση που ο πίνακας έχει μιγαδικά στοιχεία η συνάρτηση επιστρέφει το αποτέλεσμα της πράξεις X./abs(X). Ας δούμε ένα απλό παράδειγμα

```
» X = [-1 2 0;3 -4 0]
X =
   -1    2    0
    3   -4    0

» sign(X)
ans =
     -1    1    0
      1   -1    0
```

- **rem**(X,Y)

Η συνάρτηση αυτή υπολογίζει το υπόλοιπο της διαίρεσης των αντίστοιχων στοιχείων των πινάκων X και Y. Το υπόλοιπο υπολογίζεται από τη σχέση X-n.*Y όπου n=fix(X./Y). Για παράδειγμα rem(100,21)=16.

- **exp**(X)

Αν X είναι ένας πίνακας η συνάρτηση exp(X) υπολογίζει για κάθε στοιχείο z του πίνακα την τιμή e^z. Για πράδειγμα αν το X είναι ένα διάνυσμα της μορφής

```
» X = -2:2
X =
   -2   -1    0    1    2
```

τοτε η συνάρτηση exp(X) θα επιστρέψει ένα διάνυσμα τα στοιχεία του οποίου θα είναι

$$[\ e^{-2}\ \ e^{-1}\ \ e^{0}\ \ e^{1}\ \ e^{2}\ ]$$

οπότε στο MATLAB θα έχουμε

```
» exp(X)
ans =
     0.1353    0.3679    1.0000    2.7183    7.3891
```

- **log**(X)

Η συνάρτηση αυτή υπολογίζει το λογάριθμο με βάση e, για κάθε στοιχείο του πίνακα X.

- **log10**(X)

Η συνάρτηση log10(X) υπολογίζει το λογάριθμο με βάση το 10, κάθε στοιχείου του πίνακα X.

- **real**(X)

Αν ο X περιέχει μιγαδικά στοιχεία τότε η real(X) επιστρέφει το πραγματικό μέρος κάθε στοιχείου του X.

- **imag**(X)

Αν ο X περιέχει μιγαδικά στοιχεία τότε η imag(X) επιστρέφει το φανταστικό μέρος κάθε στοιχείου του X. Ας δούμε ένα απλό παράδειγμα

```
» X = [-1 2 ; -2 8]+i*[-2 3;-5 -7]
X =
   -1.0000 - 2.0000i   2.0000 + 3.0000i
   -2.0000 - 5.0000i   8.0000 - 7.0000i

» R=real(X)
R =
   -1    2
   -2    8

» I = imag(X)
I =
   -2    3
   -5   -7
```

Αν υποθέσουμε οτι ο πίνακας X είναι ένα μιγαδικό εκθετικό διάνυσμα το οποίο δημιουργείται ως εξής

```
» n = 0:25;
» X = exp(j*n/3);
```

το X θα είναι ίσο με:

[exp(j*0) exp(j*1/3) exp(j*2/3) exp(j*3/3) ... exp(j*25/3)]

Οι βασικότερες τριγωνομετρικές συναρτήσεις παρουσιάζονται στον ακόλουθο πίνακα. Οι συναρτήσεις αυτές υπολογίζουν τον τριγωνομετρικό αριθμό ή τον αντίστροφο τριγωνομετρικό αριθμό σε κάθε στοιχείο του πίνακα που δέχονται ως όρισμα.

Τριγωνομετρικές	*Συναρτήσεις*
sin(X)	sine
cos(X)	cosine
tan(X)	tangent
asin(X)	arcsine
acos(X)	arccosine
atan(X)	arctangent
sinh(X)	hyperbolic sine
cosh(X)	hyperbolic cosine
tanh(X)	hyperbolic tangent
asinh(X)	hyperbolic arcsine
acosh(X)	hyperbolic arccosine
atanh(X)	hyperbolic arctangent

Στη συνέχεια θα αναφερθούμε σε συναρτήσεις οι οποίες είναι ιδιαίτερα χρήσιμες για την ανάλυση των δεδομένων ενός πίνακα. Οι σημαντικότερες από αυτές είναι

Μέγιστο και ελάχιστο

- **max**(X)

Αν το X είναι ένα διάνυσμα η συνάρτηση max(X) επιστρέφει το μέγιστο στοιχείο του διανύσματος X. Στην περίπτωση που ο X είναι πίνακας η συνάρτηση επιστρέφει ένα διάνυσμα το οποίο περιέχει το μέγιστο στοιχείο κάθε στήλης. Αν η συνάρτηση κληθεί με δύο εξόδους, [Y,I] = max(X), τότε το Y περιέχει το μέγιστο στοιχείο κάθε στήλης και το I είναι ένα διάνυσμα που περιέχει τη θέση κάθε στοιχείου του Y στον πίνακα X. Αν ο X περιέχει μιγαδικά δεδομένα τότε καλούμε τη συνάρτηση ως εξής: max(abs(X)).

- **min**(X)

Η συνάρτηση min(X) λειτουργεί όπως ακριβώς και η max μόνο που υπολογίζει το ελάχιστο στοιχείο. Ας δούμε ένα παράδειγμα.

```
» X = [-1 9 2;10 0 -3; 5 7 8]
X =
   -1    9    2
   10    0   -3
    5    7    8

» [Y,I] = max(X)
Y =
   10    9    8
I =
    2    1    3
```

```
» [Y,I] = min(X)
Y =
   -1    0   -3
I =
    1    2    2
```

Αν θέλουμε να βρούμε το μέγιστο στοιχείο του X θα καλέσουμε τη συνάρτηση max, Y=max(X), για να βρούμε το μέγιστο στοιχείο κάθε στήλης. Κατόπιν θα καλέσουμε ξανά τη max για να βρούμε το μέγιστο στοιχείο του διανύσματος Y, το οποίο είναι και το μέγιστο στοιχείο του X. Παρατηρήστε ότι στο παράδειγμα που ακολουθεί η συνάρτηση δέχεται ως όρισμα το διάνυσμα max(X).

```
» M = max(max(X))
M =
   10
» m = min(min(X))
m =
   -3
```

- **max**(X,Y)

Στην περίπτωση που η max κληθεί με δύο ορίσματα η συνάρτηση επιστρέφει έναν πίνακα ιδίων διαστάσεων με τους X και Y. Κάθε στοιχείο του πίνακα περιέχει τη μέγιστη τιμή των αντίστοιχων στοιχείων των X και Y.

- **min**(X,Y)

Η συνάρτηση min(X,Y) επιστρέφει ένα πίνακα ιδίων διαστάσεων με τους X και Y όπου κάθε στοιχείο του είναι το ελάχιστο των αντίστοιχων στοιχείων των X και Y. Για παράδειγμα αν X και Y είναι δύο διανύσματα

```
» X = [-2  3  4  0];
» Y = [ 0  9  4  7];
```

η min(X,Y) θα δώσει

```
» min(X,Y)
ans =
      -2    3    4    0
```

Μέση και Ενδιάμεση τιμή

- **mean**(X)

Εστω N μεταβλητές, $x_1, x_2, ..., x_N$. Η μέση τιμή τους (mean value), μ, δίνεται από τη σχέση

$$\mu = (1/N) * (x_1+x_2 + ... + x_N)$$

Αν το X είναι ένα διάνυσμα η συνάρτηση mean(X) επιστρέφει τη μέση τιμή των στοιχείων του X. Αν η μεταβλητή X είναι πίνακας τότε η συνάρτηση επιστρέφει ένα διάνυσμα γραμμή όπου κάθε στοιχείο του είναι η μέση τιμή της αντίστοιχης στήλης του X.

- **median**(X)

Εστω N μεταβλητές, x_1, x_2 , ..., x_N . Αν το N είναι περιττός η ενδιάμεση τιμή (median) το στοιχείο που βρίσκεται στην N/2+1 θέση. Αν το N είναι άρτιος η ενδιάμεση τιμή υπολογίζεται απο το μέσο όρο των στοιχείων που βρίσκονται στις θέσεις N/2 και N/2+1.
Αν το X είναι ένα διάνυσμα η συνάρτηση median(X) επιστρέφει την ενδιάμεση τιμή των στοιχείων του X. Αν η μεταβλητή X είναι πίνακας τότε η συνάρτηση επιστρέφει ένα διάνυσμα γραμμή όπου κάθε στοιχείο του είναι η ενδιάμεση τιμή της αντίστοιχης στήλης του X. Για παράδειγμα

```
» a = [0  10  3  2  -3  0];
» mean(a)
ans =
     2
» median(a)
ans =
     1
```

Στην περίπτωση που το N είναι περιττό έχουμε

```
» a = [0  10  3  2  -3  0  100];
» mean(a)
ans =
     16
» median(a)
ans =
     2
```

Άθροισμα και Γινόμενο

- **sum**(X)

Αν το X είναι ένα διάνυσμα η συνάρτηση sum(X) υπολογίζει το άθροισμα των στοιχείων του X. Αν το X είναι πίνακας τότε η συνάρτηση επιστρέφει ένα διάνυσμα όπου κάθε στοιχείο του είναι το άθροισμα των στοιχείων των αντίστοιχων στηλών του X.

- **prod**(X)

Αν το X είναι ένα διάνυσμα η συνάρτηση prod(X) υπολογίζει το γινόμενο των στοιχείων του X. Αν το X είναι πίνακας τότε η συνάρτηση επιστρέφει ένα διάνυσμα όπου κάθε στοιχείο του είναι το γινόμενο των στοιχείων των αντίστοιχων στηλών του X.

- **cumsum**(X)

Αν το X είναι διάνυσμα η συνάρτηση επιστρέφει ένα διάνυσμα ιδίου μεγέθους τα στοιχεία του οποίου είναι τα συσσωρευτικά αθροίσματα (cumulative sums) των στοιχείων του X. Για παράδειγμα

```
» x = [-1  2  3  4];

» sum(x)
```

```
ans =
     8

» cumsum(x)
ans =
    -1   1   4   8
```

- **cumprod**(X)

Αν το X είναι διάνυσμα η συνάρτηση επιστρέφει ένα διάνυσμα ιδίου μεγέθους τα στοιχεία του οποίου είναι τα συσσωρευτικά γινόμενα (cumulative products) των στοιχείων του X. Για παράδειγμα

```
» x = [-1   2   3   4];

» prod(x)
ans =
    -24

» cumprod(x)
ans =
    -1    -2    -6    -24
```

Ταξινόμηση

- **sort**(X)

Αν το X είναι διάνυσμα η συνάρτηση ταξινομεί τα στοιχεία του κατά αύξουσα σειρά. Αν η μεταβλητή X είναι πίνακας η συνάρτηση επιστρέφει τον πίνακα X έχοντας ταξινομήσει τις στήλες του. Αν ο X είναι μιγαδικός η ταξινόμηση γίνεται ως προς το μέτρο του X, abs(X). Για τον παραπάνω πίνακα X έχουμε

```
» sort(X)
ans =
    -1   -1    1
    -1    0    2
     1    4    3
```

Διασπορά και Τυπική απόκλιση

- **std**(X)

Η διασπορά (variance) σ^2 και η τυπική απόκλιση (standard deviation) σ, είναι δύο σημαντικά στατιστικά μέτρα. Στο MATLAB η συνάρτηση std(X) υπολογίζει την τυπική απόκλιση των τιμών του διανύσματος X. Αν το X είναι πίνακας η συνάρτηση επιστρέφει ένα διάνυσμα, κάθε στοιχείο του οποίου είναι η τυπική απόκλιση των στοιχείων της αντίστοιχης στήλης του X. Για τον υπολογισμό της διασποράς απλώς υψώνουμε στο τετράγωνο την τυπική απόκλιση.

Τυχαίοι Αριθμοί

Το MATLAB, όπως και όλες οι γλώσσες προγραμματισμού, δίνει τη δυνατότητα δημιουργίας τυχαίων αριθμών. Πολλές φορές χρησιμοποιούμε τυχαίους αριθμούς για την προσομοίωση σύνθετων προβλημάτων ή για να αναπαραστήσουμε ένα σήμα θορύβου. Στο MATLAB ορίζονται δύο συναρτήσεις που παράγουν τυχαίους, η rand() και η randn().

- **rand**

Η συνάρτηση αυτή επιστρέφει τυχαίους αριθμούς ομοιόμορφα κατανεμημένους στο διάστημα [0,1]. Αν η συνάρτηση κληθεί με ένα όρισμα, **rand(N)**, επιστρέφει έναν NxN πίνακα όπου κάθε στοιχείο του είναι τυχαίος στο [0,1]. Αν κληθεί με δύο ορίσματα , **rand(N,M)**, επιστρέφει έναν NxM πίνακα όπου κάθε στοιχείο του είναι τυχαίος στο [0,1].
Το MATLAB χρησιμοποιεί μια μεταβλητή, **seed**, με την οποία αρχικοποιεί την ακολουθία των τυχαίων αριθμών που παράγει η rand. Μπορούμε να δούμε την τρέχουσα τιμή της seed δίνοντας **rand('seed')**. Μπορούμε επίσης να αλλάξουμε τιμή στη μεταβλητή seed δίνοντας **rand('seed',s)** όπου s ένας αριθμός. Ενας τρόπος για παράδειγμα αρχικοποίησης της μεταβλητής seed είναι

```
rand('seed',sum(100*clock))
```

Η συνάρτηση clock επιστρέφει ένα διάνυσμα της μορφής

```
[year month day hour minute seconds]
```

έτσι η τιμή που αναθέτουμε στη μεταβλητή seed αλλάζει κάθε φορά.
Στο παράδειγμα που ακολουθεί δημιουργούμε μια ακολουθία από 8 αριθμούς ομοιόμορφα κατανεμημένους στο διάστημα [-5 , 5].

```
» y = 10*rand(1,6)-5
y =
  4.0921   -4.3944    4.0465    0.0452    0.1629   -1.8097
```

- **randn**

Οταν η ακολουθία των τυχαίων που παράγουμε ακολουθεί την ομοιόμορφη κατανομή, όλες οι τιμές έχουν την ίδια πιθανότητα να εμφανιστούν. Το MATLAB δίνει επίσης τη δυνατότητα δημιουργίας τυχαίων αριθμών που ακολοθούν Gaussian κατανομή με μέση τιμή 0 και διασπορά 1 με τη συναρτηση randn. Αν η συνάρτηση κληθεί με ένα όρισμα, **randn(N)**, επιστρέφει έναν NxN πίνακα, αν κληθεί με δύο ορίσματα , **randn(N,M)**, επιστρέφει έναν NxM πίνακα όπου κάθε στοιχείο του ακολουθεί Gaussian κατανομή με μέση τιμή 0 και διασπορά 1.
Αν θέλουμε να παράγουμε τυχαίους με μέση τιμή μ και διασπορά σ^2, παράγουμε πρώτα με την randn. Κατόπιν πολλαπλασιάζουμε κάθε έναν από αυτούς με την επιθυμητή τυπική απόκλιση σ και προσθέτουμε τη μέση τιμή μ.
Για παράδειγμα, για να παράγουμε μια ακολουθία τυχαίων με μέση τιμή 3 και διασπορά 1 γράφουμε

```
» data = randn(1,500)+3;
```

Α.5 Δομές ελέγχου -Λογικές συναρτήσεις - Δομές επανάληψης

Οπως σε όλες οι γλώσσες προγραμματισμού έτσι και το MATLAB υποστηρίζει δομές ελέγχου και δομές επανάληψης. Στην ενότητα αυτή θα δούμε τη δυνατότητα χρήσης αυτών των δομών καθώς και κάποιες λογικές συναρτήσεις που παρέχονται από το MATLAB.

Α.5.1 Δομές ελέγχου

Γενικά μια απλή if δήλωση συντάσσεται ως εξής

if λογική έκφραση
  λίστα εντολών
end

Στην περίπτωση που η λογική έκφραση είναι αληθής εκτελούνται οι εντολές που βρίσκονται μεταξύ του **if** και του **end**. Στην αντίθετη περίπτωση το πρόγραμμα εκτελεί την εντολή που συναντά μετά το **end**. Είναι σημαντικό στο σημείο αυτό να αναφερθούμε στους σχεσιακούς και λογικούς τελεστές. Οι τελεστές αυτοί χρησιμοποιούνται στις λογικές εκφράσεις που ελέγχουμε σε μια if δήλωση.
Το MATLAB έχει έξι σχεσιακούς τελεστές για τη σύγκριση πινάκων με ίσες διαστάσεις. Η σύγκριση δύο πινάκων με ένα τέτοιο τελεστή δίνει έναν πίνακα με την ίδια διάσταση. Ο πίνακας αυτός περιέχει 1 στις θέσεις εκείνες όπου η σύγκριση των αντίστοιχων στοιχείων των δύο πινάκων είναι αληθής και 0 αλλού. Μια έκφραση που περιέχει σχεσιακό τελεστή είναι μια λογική έκφραση γιατί το αποτέλεσμα είναι ένας πίνακας του οποίου οι τιμές είναι μηδενικά ή μονάδες. Οι τιμές αυτές ερμηνεύονται ως ψευδείς και αληθείς αντίστοιχα. Στον πίνακα που ακολουθεί παρουσιάζονται οι τελεστές αυτοί.

Σχεσιακοί Τελεστές	
<	μικρότερο
<=	μικρότερο ή ίσο
>	μεγαλύτερο
>=	μεγαλύτερο ή ίσο
==	ίσο
~=	διάφορο

Μπορούμε να συνδυάσουμε δύο λογικές εκφράσεις, χρησιμοποιώντας λογικούς τελεστές. Στον παρακάτω πίνακα παρουσιάζουμε τα σύμβολα που χρησιμοποιεί το MATLAB για τους λογικούς τελεστές and, not και or.

Λογικοί Τελεστές	**Σύμβολο**

not	~
and	**&**
or	|

Οι τελεστές αυτοί ενεργούν πάνω σε 0-1 πίνακες.

Ας δούμε τώρα αναλυτικά ένα παράδειγμα με πολλαπλές if δηλώσεις

```
if g<50
   count =count +1;
   disp(g);
    if b>g
      b=0;
    end
end
```

Ας υποθέσουμε κατ' αρχήν ότι οι g και b είναι αριθμοί. Τότε, αν g<50, αυξάνουμε την μεταβλητή count κατά ένα και εμφανίζουμε το g. Επιπλέον, αν b>g, θέτουμε το b ίσο με το μηδέν. Αν g>50 τότε το πρόγραμμα θα εκτελέσει την εντολή που βρίσκεται μετά το δεύτερο end. Αν η μεταβλητή g είναι πίνακας η συνθήκη είναι αληθής μόνο όταν κάθε στοιχείο του g μικρότερο από 50. Αν επιπλέον και η μεταβλητή b είναι πίνακας τότε η συνθήκη b>g είναι αληθής αν κάθε στοιχείο του b είναι μεγαλύτερο από το αντίστοιχο στοιχείο του g.

Δομή ελέγχου if else if
Μια απλή δομή if else έχει τη μορφή

```
if  λογική έκφραση
       λίστα εντολών
else
       λίστα εντολών
end
```

Σε περίπτωση που θέλουμε να χρησιμοποιήσουμε πολλαπλές if else δηλώσεις, το πρόγραμμά μας μπορεί να γίνει αρκετά πολύπλοκο. Στην περίπτωση συτή είναι καλύτερα να χρησιμοποιήσουμε δήλωση **elseif**. Στο παράδειγμα που ακολουθεί ελέγχεται η τιμή της μεταβλητής dist.

```
 if  dist > 100
     time = time + 3;
elseif  dist > 90
      time =time +2;
elseif  dist > 50
      time =time +1;
else
      time=time+0.5;
end
```

παρατηρήστε ότι η αύξηση της μεταβλητής time κατά 2 γίνεται όταν η μεταβλητή disp είναι μεγαλύτερη από 90 και μικρότερη από 100. Η αύξηση της time κατά 1 γίνεται όταν η disp είναι μεγαλύτερη από 50 και μικρότερη από 90.

Α.5.2 Λογικές συναρτήσεις

Στο MATLAB ορίζονται οι παρακάτω λογικές συναρτήσεις, οι οποίες χρησιμοποιούνται συχνά σε δομές ελέγχου.

- **any**(X)

Η συνάρτηση αυτή επιστρέφει μονάδα αν κάποιο από τα στοιχεία του διανύσματος X είναι μη μηδενικό και μηδέν στην περίπτωση όπου όλα τα στοιχεία του X είναι μηδενικά. Αν ο X είναι πίνακας, η επιστρεφόμενη τιμή είναι ένα διάνυσμα γραμμή. Ενα στοιχείο του διανύσματος αυτού είναι μονάδα αν τουλάχιστον ένα από τα στοιχεία της αντίστοιχης στήλης του X είναι διάφορο του μηδενός και μηδέν στην περίπτωση όπου όλα τα στοιχεία της αντίστοιχης στήλης του X είναι μηδέν.

- **all**(X)

Η συνάρτηση αυτή επιστρέφει μονάδα αν όλα τα στοιχεία του διανύσματος X είναι μη μηδενικά και μηδέν στην περίπτωση που κάποιο από αυτά είναι διάφορο του μηδενός. Αν ο X είναι πίνακας, η επιστρεφόμενη τιμή είναι ένα διάνυσμα γραμμή. Ενα στοιχείο του διανύσματος αυτού είναι μονάδα αν όλα τα στοιχεία της αντίστοιχης στήλης του X είναι διάφορα του μηδενός και μηδέν στην αντίθετη περίπτωση. Ας δούμε ένα παράδειγμα:

```
» X=[-1 0 3;0 0 6;0 0 8]
X =
   -1    0    3
    0    0    6
    0    0    8

» Y=any(X)
Y =
    1    0    1

» Z=all(X)
Z =
    0    0    1
```

Παρατηρήστε ότι η συνάρτηση any είναι αληθής αν υπάρχει τουλάχιστον ένα μη μηδενικό στοιχείο ενώ η συνάρτηση all είναι αληθής αν όλα τα στοιχεία είναι μη μηδενικά.

- **find**(X)

Η συνάρτηση find επιστρέφει ένα διάνυσμα τα στοιχεία του οποίου είναι οι δείκτες των μη-μηδενικών στοιχείων του διανύσματος X. Αν το X είναι πίνακας, οι δείκτες

επιλέγονται από το διάνυσμα X(**:**). Το διάνυσμα αυτό είναι ένα διάνυσμα στήλη το οποίο αποτελείται από τις στήλες του πίνακα X. Για παράδειγμα

```
» X = [0 12 0 32 -1 0];
» F = find(X)
F =
    2   4   5
```

Παρατηρήστε το διάνυσμα F στην περίπτωση όπου ο X δεν είναι διάνυσμα

```
» X=[0 1 0 -1;3 5 0 8]
X =
   0   1   0   -1
   3   5   0    8
» F = find(X)
F =
    2
    3
    4
    7
    8
```

- **isnan**(X)

Η συνάρτηση isnan επιστρέφει έναν πίνακα με μονάδες στις θέσεις εκείνες όπου το αντίστοιχο στοιχείο του X δεν ορίζεται (Nan) και μηδέν στην αντίθετη περίπτωση.

- **finite**(X)

Η συνάρτηση αυτή επιστρέφει έναν πίνακα με μονάδες όπου το αντίστοιχο στοιχείο του πίνακα X είναι πεπερασμένο και μηδέν στην περίπτωση όπου το στοιχείο είναι inf ή Nan.

- **isempty**(X)

Η συνάρτηση αυτή επιστρέφει μονάδα αν ο X είναι ένας κενός πίνακας και μηδέν στην αντίθετη περίπτωση.

Α.5.3 Δομές επανάληψης

Οταν θέλουμε να επαναλάβουμε μια ή περισσότερες εντολές σε ένα πρόγραμμα χρησιμοποιούμε μια δομή επανάληψης. Γενικά στο MATLAB είναι προτιμότερο να αποφεύγουμε τη χρήση δομών επανάληψης γιατί αυξάνεται ο υπολογιστικός χρόνος του προγράμματος.

Η γενική σύνταξη μιας δομής **for** είναι

```
for   index = exp
  statement
      ...
```

```
    statement
  end
```

όπου exp είναι ένας πίνακας και οι εντολές επαναλαμβάνονται τόσες φορές όσες είναι οι στήλες του πίνακα exp. Κάθε φορά μέσα στο loop, η μεταβλητή index λαμβάνει την τιμή ενός στοιχείου του πίνακα. Οταν γράφουμε μια δομή for πρέπει να προσέχουμε τα ακόλουθα:

1. θα πρέπει η index να είναι μεταβλητή
2. Αν ο πίνακας exp είναι κενός τότε η εντολές δεν θα εκτελεστούν και ο έλεγχος θα μεταφερθεί στην εντολή που ακολουθεί το **end.**
3. Αν ο exp είναι βαθμωτό μέγεθος τότε η εντολές θα εκτελεστούν μία μόνο φορά.
4. Αν ο exp είναι ένα διάνυσμα γραμμή, κάθε φορά που εκτελείται το loop, η μεταβλητή index θα περιέχει την επόμενη τιμή του διανύσματος
5. Αν ο exp είναι ένας πίνακας, κάθε φορά που εκτελείται το loop, η μεταβλητή index θα περιέχει την επόμενη στήλη του πίνακα.
6. Με την έξοδο από το for loop η ματαβλητή index θα περιέχει την τελευταία τιμή που είχε.
7. Η πιο συχνή έκφραση για τη μεταβλητή index είναι
   index = first:step:last

Η μεταβολή της μεταβλητής index μπορεί να είναι ως εξής

```
index = 2:-.5:-1
```

Ας δούμε ένα παράδειγμα χρήσης της for. Εστω το διάνυσμα

```
» d = rand([1 1000]);
```

θέλουμε να προσδιορίσουμε τις τιμές που είναι μικρότερες του 0.5 και να τις αλλάξουμε σε μηδέν. Οσες δε είναι μεγαλύτερες να τις κάνουμε 1.

```
» for k=1:length(d)
    if d(k)<.5
      d(k)=0;
    else
      d(k)=1;
    end
  end
```

Το παραπάνω πρόγραμμα μπορεί να γίνει με την συνάρτηση find. Η υλοποίηση με τη συνάρτηση find είναι σημαντικά γρηγορότερη.

Μπορούμε να έχουμε και for που περιέχουν άλλα for μέσα τους. Για παράδειγμα

```
for I=1:N
  for J=1:N
    A(I,J) = 1/(I+J-1);
  end
```

end

Στην περίπτωση αυτή δεν θα πρέπει να ξεχνάμε το end στο τέλος κάθε for.

Η δομή επανάληψης **while** επαναλαμβάνει ένα σύνολο εντολών όσο μια συγκεκριμένη συνθήκη είναι αληθής. Η γενική της σύνταξη είναι

while expression
  statement
    ...
  statement
end

Αν η έκφραση είναι αληθής, οι εντολές εκτελούνται. Μετά την εκτέλεση των εντολών η συνθήκη ξαναελέγχεται. Αν η συνθήκη είναι και πάλι αληθής, οι εντολές εκτελούνται και πάλι. Αν η συνθήκη είναι ψευδής οι εντολές δεν εκτελούνται και ο έλεγχος μεταβιβάζεται στην εντολή που ακολουθεί το **end**.
Οι μεταβλητές που ελέγχονται στην έκφραση πρέπει να μεταβάλονται από τις εντολές, έτσι ώστε η έκφραση να γίνει κάποτε ψευδής. Στην αντίθετη περίπτωση δεν θα μπορεί να τερματιστεί η δομή επανάληψης.

A.6 Τα αρχεία στο MATLAB

A.6.1 Εντολές save/load

Εχουμε δει οτι με την εντολή quit ή exit μπορούμε να βγούμε από το περιβάλλον του MATLAB. Η έξοδος αυτή έχει ως συνέπεια τη διαγραφή από τη μνήμη κάθε μεταβλητής που έχει δημιουργηθεί Μπορούμε να αποθηκεύσουμε τις μεταβλητές μας σε ένα αρχείο με την εντολή save. Η αποθήκευση θα γίνει σε ένα αρχείο με κατάληξη .mat. Για παράδειγμα, αν έχουμε δημιουργήσει τους παρακάτω πίνακες:

```
» whos
  Name        Size          Elements    Bytes   Density  Complex
   a         3 by 3                9       72      Full       No
   b         4 by 3               12       96      Full       No
   x       100 by 100          10000    80000      Full       No
```

μπορούμε να τους αποθηκεύσουμε ως εξής

```
» save file
```

Με την εντολή save θα αποθηκευτούν στο αρχείο file.mat όλες οι μεταβλητές. Με την παρακάτω δήλωση μπορούμε να σώσουμε σε ένα αρχείο μόνο τις μεταβλητές a και b.

```
» save file a b
```

Γενικά η εντολή save συντάσσεται ως εξής

save όνομα_αρχείου

ή αν θέλουμε να αποθηκεύσουμε κάποιες από τις μεταβλητές

save όνομα_αρχείου μεταβλητή1 μεταβλητή2 ... μεταβλητήN

Το MATLAB θεωρεί ότι το αρχείο με κατάληξη .mat στο οποίο θα γίνει η αποθήκευση είναι binary. Αν θέλουμε να κρατήσουμε την πληροφορία σε ascii αρχείο τότε οι παραπάνω δηλώσεις θα γράφονταν

save όνομα_αρχείου **-ascii**
save όνομα_αρχείου μεταβλητή1 μεταβλητή2 ... μεταβλητήN **-ascii**

Η ανάκτηση των μεταβλητών από ένα .mat αρχείο γίνεται με την εντολή load. Η εντολή load, για την περίπτωση .mat αρχείου συντάσσεται ως εξής:

load όνομα_αρχείου

ενώ αν αναφερόμαστε σε ascii αρχείο

load όνομα_αρχείου **-ascii**

A.6.2 Εντολές fread/fwrite

Εκτός από τις συναρτήσεις save και load μπορούμε να αποθηκεύσουμε ή αντίστοιχα να διαβάσουμε δεδομένα από ένα αρχείο με τη βοήθεια των συναρτήσεων fwrite και fread. Για να μπορέσουμε να εκτελέσουμε Ε/Ε σε αρχείο, θα πρέπει πρώτα να αποκτήσουμε πρόσβαση στο αρχείο με τη συνάρτηση fopen ως εξής:

fid = **fopen**('όνομα_αρχείου', κατάσταση_ανοίγματος)

όπου το όνομα του αρχείου προσδιορίζεται μέσα σε μονά εισαγωγικά. Θα μπορούσαμε, αν χρειάζεται, να δώσουμε και το πλήρες μονοπάτι στο οποίο βρίσκεται το αρχείο. Η δεύτερη παράμετρος προσδιορίζει την κατάσταση στην οποία θέλουμε να ανοίξουμε το αρχείο. Οι δυνατές καταστάσεις ανοίγματος περιγράφονται στον ακόλουθο πίνακα:

Κατάσταση Ανοίγματος	Περιγραφή
'r'	Ανοιγμα αρχείου με δυνατότητα ανάγνωσης
'w'	Ανοιγμα αρχείου με δυνατότητα εγγραφής (αν χρειάζεται το δημιουργεί)
'a'	Ανοιγμα αρχείου με δυνατότητα προσάρτησης στο τέλος του αρχείου (αν χρειάζεται το δημιουργεί)
'r+'	Ανοιγμα αρχείου με δυνατότητα ανάγνωσης και εγγραφής (δεν το δημιουργεί)
'w+'	Επικαλύπτει τα περιεχόμενα του αρχείου ή το δημιουργεί με δικαίωμα ανάγνωσης και εγγραφής
'a+'	Δυνατότητα ανάγνωσης και προσάρτησης στο τέλος

του αρχείου (αν χρειάζεται το δημιουργεί)

Αν καλέσουμε τη συνάρτηση με μόνη παράμετρο το όνομα του αρχείου η συνάρτηση υποθέτει κατάσταση αρχείου 'r' .
Η συνάρτηση fopen επιστρέφει μια τιμή, στην παραπάνω δήλωση την έχουμε καταχωρήσει στη μεταβλητή fid. Η τιμή αυτή είναι ένας ακέραιος ο οποίος δηλώνει αν άνοιξε επιτυχώς το αρχείο. Σε περίπτωση λάθους η συνάρτηση επιστρέφει την τιμή -1. Μπορούμε επίσης να λαμβάνουμε και ένα μήνυμα λάθους με την παρακάτω δήλωση

[fid,message] = **fopen**('όνομα_αρχείου', κατάσταση_ανοίγματος)

Στη μεταβλητή message καταχωρείται ένα μήνυμα από το σύστημα σε περίπτωση ανεπιτυχούς ανοίγματος του αρχείου. Η μεταβλητή στην οποία καταχωρούμε την επιστρεφόμενη τιμή της fopen δεν χρησιμοποιείται μόνο για τον έλεγχο επιτυχούς ή όχι ανοίγματος του αρχείου αλλά όπως θα δούμε αποτελεί έναν προσδιοριστή του αρχείου για άλλες συναρτήσεις. Το φυσικό όνομα του αρχείου χρησιμοποιείται μόνο στην fopen. Μετά το άνοιγμα του αρχείου μας η αναφορά σε αυτό από άλλες συναρτήσεις θα γίνεται με τη μεταβλητή fid.
Η fopen ανοίγει τα αρχεία σε binary mode. Αν θέλουμε να επεξεργαστούμε ένα text αρχείο τότε θα πρέπει να προσθέσουμε το 't' στις παραπάνω καταστάσεις ανοίγματος. Για παράδειγμα αν θέλαμε να ανοίξουμε ένα text αρχείο με δυνατότητα ανάγνωσης θα γράφαμε:

fid = fopen('c:\users\file.txt', 'rt')

Θα μπορούσαμε επίσης να καλέσουμε τη συνάρτηση ως εξής

[fname,permission] = fopen(fid)

Στην περίπτωση αυτή η συνάρτηση επιστρέφει στη μεταβλητή fname το όνομα του αρχείου και στη μεταβλητή permission την κατάσταση ανοίγματος του αρχείου που σχετίζεται με τη συγκεκριμένη μεταβλητή fid.
Οταν η τιμή της fid είναι 0 απευθυνόμαστε στο standard input, όταν είναι 1 στο standard output και όταν είναι 2 στο standard error. Αν για παράδειγμα δώσουμε την εντολή

```
» [file,permission] = fopen(0)
```

Το MATLAB απαντά

```
file =
      "stdin"
permission =
            r
```

Μέχρι στιγμής έχουμε χρησιμοποιήσει την fopen για να ανοίξουμε το αρχείο μας. Η επεξεργασιά του αρχείου ανάγνωση ή εγγραφή θα γίνει με τις συναρτήσεις fread και fwrite αντίστοιχα. Η συνάρτηση fread συντάσσεται ως εξής:

[A,count] = **fread**(fid,size,precision,skip)

Η συνάρτηση fread διαβάζει δεδομένα από ένα binary αρχείο και τα αποθηκεύει στον πίνακα A. Η μεταβλητή count είναι προαιρετική και προσδιορίζει το πλήθος των δεδομένων που διαβάστηκαν από το αρχείο επιτυχώς. Η συνάρτηση δέχεται τέσσερα ορίσματα. Το πρώτο από αυτά, fid, είναι ένας ακέραιος που προσδιορίζει μοναδικά το προς επεξεργασία αρχείο. Είναι ο ακέραιος που λαμβάνεται από την fopen και πρέπει απαραίτητα να δηλώνεται στην fread. Το δεύτερο όρισμα, size, είναι προαιρετικό. Δηλώνει το πλήθος των στοιχείων που θα διαβαστούν από το αρχείο. Οι τιμές που μπορεί να πάρει είναι:

N: διαβάζει N στοιχεία και γεμίζει με αυτά ένα διάνυσμα στήλη.
inf: διαβάζει έως το τέλος του αρχείου.
[M,N] διαβάζει M*N δεδομένα από το αρχείο ώστε να γεμίσει έναν MxN πίνακα, γεμίζοντας πρώτα τις στήλες του.

Αν η μεταβλητή size δεν προσδιορίζεται τότε θα διαβαστεί ολόκληρο το αρχείο.
Το τρίτο όρισμα, precision, ελέγχει τον αριθμό των bits που διαβάζονται για κάθε τιμή. Οι δυνατές τιμές για τη μεταβλητή precision παρουσιάζονται στον παρακάτω πίνακα.

Σύμβολο	Περιγραφή
Char	χαρακτήρας , 8 bits
Short	ακέραιος, 16 bits
Int	ακέραιος, , 16 ή 32 bits
Long	ακέραιος, 32 ή 64 bits
Float	κινητής υποδιαστολής, 32 bits
Double	κινητής υποδιαστολής, 64 bits
Uchar	μη-προσημασμένος χαρακτήρας, 8 bits
Ushort	μη-προσημασμένος ακέραιος, 16 bits
uint	μη-προσημασμένος ακέραιος, 16 ή 32 bits
Ulong	μη-προσημασμένος ακέραιος, 32 bits
Float32	κινητής υποδιαστολής, 32 bits
Float64	κινητής υποδιαστολής, 64 bits
Int8	ακέραιος, 8 bits
Int16	ακέραιος, 16 bits
Int32	ακέραιος, 32 bits
IntN	ακέραιος, N bits
UintN	μη-προσημασμένος ακέραιος, N bits

όπου για τις δύο τελευταίες περιπτώσεις, intN και uintN, το N δηλώνει έναν ακέραιο από το 1 έως το 32. Αν η μεταβλητή precision δεν προσδιορίζεται τότε το MATLAB θεωρεί ότι επεξεργάζεται uchar τιμές. Τα στοιχεία του πίνακα που θα γίνει η αποθήκευση καταχωρούνται στη μνήμη ώς μεταβλητές κινητής υποδιαστολής 32bits. Το τελευταίο όρισμα της συνάρτησης, skip, είναι προαιρετικό και δηλώνει το πλήθος των bytes που θέλουμε να παραλείπουμε μετά από κάθε ανάγνωση.
Η εγγραφή δεδομένων σε ένα αρχείο θα γίνει με την συνάρτηση fwrite, η κλήση της οποίας γίνεται ως εξής:

count = **fwrite**(fid,A,precision,skip)

Με την fwrite γράφουμε τα δεδομένα του πίνακα A σε ένα binary αρχείο μεταφράζοντας τις τιμές του πίνακα όπως προσδιορίζει η μεταβλητή precision. Τα δεδομένα λαμβάνονται από τον πίνακα κατά στήλες. Η μεταβλητή count δηλώνει τον αριθμό των στοιχείων που γράφτηκαν στο αρχείο επιτυχώς.
Η μεταβλητή fid προσδιορίζει το αρχείο και λαμβάνεται από την fopen. Η μεταβλητή precision δηλώνει τον τύπο με τον οποίο θα γραφούν τα στοιχεία στο αρχείο. Οι τιμές που μπορεί να πάρει είναι αυτές που αναφέραμε και στην fread. Η μεταβλητή skip δηλώνει το πλήθος των bytes που θέλουμε να παραλείπουμε πριν από κάθε εγγραφή.
Οταν τελειώσουμε με την επεξεργασία του αρχείου δεν θα πρέπει να ξεχνάμε να κλείσουμε το αρχείο με την fclose. Η δήλωση της συνάρτησης είναι:

c = **fclose**(fid)

Η συνάρτηση κλείνει το αρχείο που προσδιορίζεται από τον ακέραιο fid και επιστρέφει στη μεταβλητή c το 0 αν η εντολή εκτελέστηκε επιτυχώς και -1 στην αντίθετη περίπτωση. Με την εντολή fclose('all') κλείνουμε όλα τα αρχεία εκτός απο αυτά με fid 0,1 και 2.
Στο παράδειγμα ακολουθεί χειριζόμαστε δύο αρχεία. Από το αρχείο input.dat διαβάζουμε τα δεδομένα και αποθηκεύουμε τις τιμές σε ένα διάνυσμα A. Κατόπιν γράφουμε στο αρχείο output.dat το μήκος του διανύσματος A καθώς και το μέγιστο στοιχείο του A.

```
»  fid1 = fopen('input.dat','r');
»  A = fread(fid1,'int');
»  fid2 = fopen('output.dat','w');
»  fwrite(fid2,length(A),'int');
»  fwrite(fid2,max(A),'int');
»  fclose(fid1);
»  fclose(fid2);
```

A.7 Γραφικά

Η γραφική αναπαράσταση των δεδομένων είναι μια ιδιαίτερα σημαντική δυνατότητα που παρέχεται από το MATLAB. Στην ενότητα αυτή θα παρουσιάσουμε τις σημαντικότερες συναρτήσεις που παρέχονται για δισδιάστατα και τρισδιάστατα γραφικά καθώς και κάποιες συναρτήσεις που αφορούν στην παρουσίαση της γραφικής παράστασης.

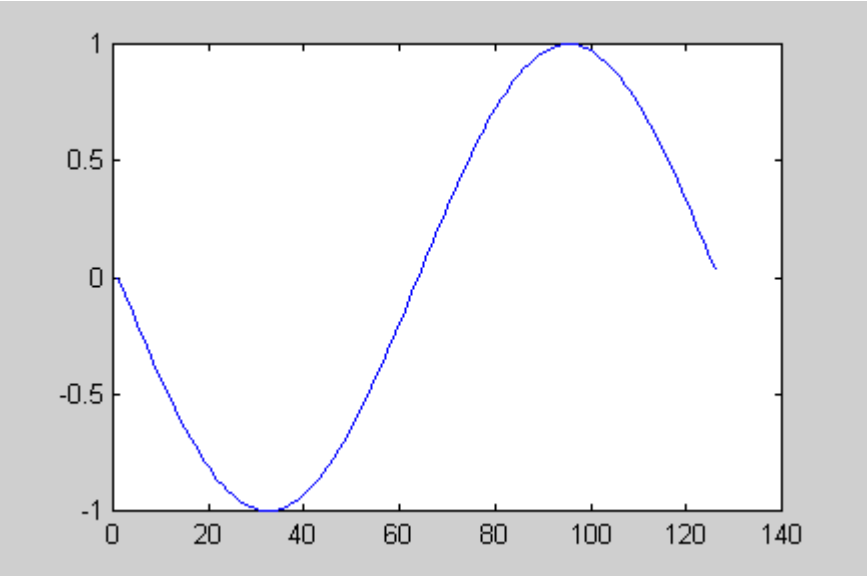
Α.7.1 **Δισδιάστατα Γραφικά**

- **plot**

Η συνάρτηση αυτή είναι από τις πιο βασικές συναρτήσεις για την σχεδίαση δισδιάστατων γραφικών. Με την κλήση της σχεδιάζονται γραμμικά οι άξονες x και y. Εαν το Y είναι διάνυσμα και η συνάρτηση κληθεί με ένα όρισμα, **plot(Y),** λαμβάνουμε τη γραφική παράσταση του Y ως προς τους δείκτες των στοιχείων του. Αν τα X και Y είναι διανύσματα του ιδίου μήκους, η εντολή **plot(X,Y)** σχεδιάζει την γραφική παράσταση των στοιχείων του Y ως προς τα στοιχεία του X. Ας δούμε ένα απλό παράδειγμα.

```
» x = -pi:.05:pi;
» y = sin(x);
» plot(y)
```

Μόλις η εντολή plot(y) εκτελεστεί εμφανίζεται η ακόλουθη γραφική παράσταση.

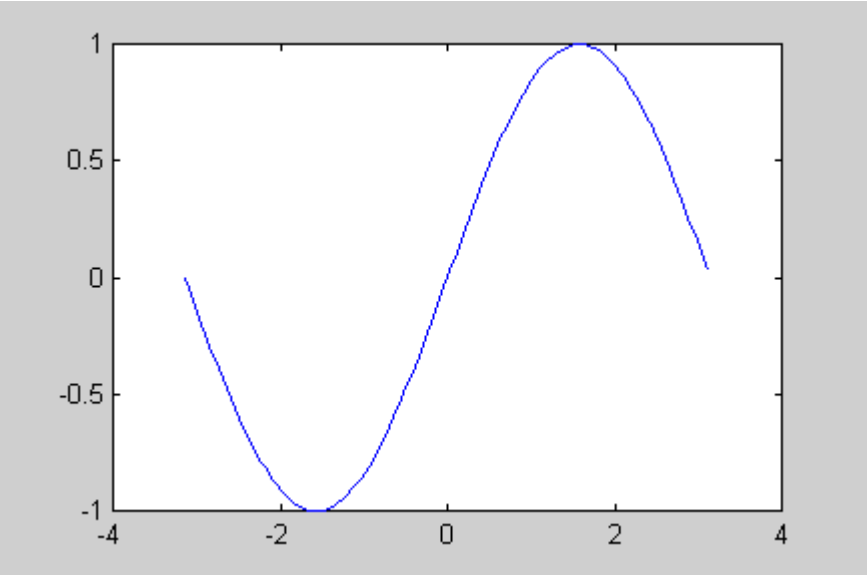


Η γραφική παράσταση θα δημιουργηθεί σε ένα νέο παράθυρο, το Figure Window. Το MATLAB προσφέρει δύο περιβάλλοντα εργασίας. Ενα στο οποίο γράφουμε τις εντολές, Command Window, και ένα στο οποίο παριστά την εκάστοτε γραφική παράσταση. Οπως θα δούμε στη συνέχεια μπορούμε αν θέλουμε να έχουμε πολλά παράθυρα γραφικών ενεργά.

Παρατηρήστε ότι οι άξονες έχουν βαθμολογηθεί αυτόματα. Ο οριζόντιος άξονας στην περίπτωση αυτή αντιστοιχεί στους δείκτες του διανύσματος y. Αν στην εντολή plot προσδιορίσουμε τον οριζόντιο άξονα, δηλαδή δώσουμε

```
» plot(x,y)
```

τότε η γραφική παράσταση θα είναι ως εξής



Παρατηρήστε εδώ την αλλαγή στην βαθμολόγηση του οριζόντιου άξονα. Ο τύπος της γραμμής που χρησιμοποιεί η plot για να ενώσει τα σημεία της μπορεί να αλλάξει καθώς επίσης και ο τύπος του στίγματος που χρησιμοποιεί Για παράδειγμα η εντολή

```
plot(x,y,'*')
```

σχεδίαζει την ίδια γραφική παράσταση χρησιμοποιώντας όμως αντί του στίγματος (**.**) το *. Μπορούμε ακόμα να αλλάξουμε και το χρώμα των γραμμών και των στιγμάτων με τρόπο ανάλογο με αυτόν που αλλάξουμε τον τύπο τους. Για παράδειγμα η εντολή

```
plot(x,y,'+r')
```

θα σχεδίασει την παραπάνω γραφική παράσταση χρησιμοποιώντας κόκκινο + στίγμα. Γενικά η συνάρτηση μπορεί να κληθεί με τρείς παραμέτρους , **plot(x,y,s)**, όπου το s είναι μια ακολουθία χαρακήρων καθένας από τους οποίους μπορεί να επιλεγεί απο τις στήλες του επόμενου πίνακα.

Τύπος Γραμμής	**Τύπος στίγματος**	**Χρώμα**
- solid	**.** point	**y** yellow
: dotted	**o** circle	**m** magenta
-. Dashdot	**x** x-mark	**c** cyan
-- dashed	+ plus	**r** red
	* star	**g** green

s	square	**b**	blue
d	diamond	**w**	white
v	triangle (down)	**k**	black
^	triangle (up)		
<	triangle (left)		
>	triangle (right)		
p	pentagram		
h	hexagram		

Μπορούμε να σχεδιάσουμε πολλές γραμμές σε ένα διάνυσμα με την εντολή plot(X,Y) όπου τα X ,Y είναι ή και τα δύο πίνακες ή το ένα από αυτά είναι πίνακας. Τότε:

1. Εαν το Y είναι πίνακας και το X διάνυσμα, η εντολή plot(X,Y) σχεδιάζει τις γραμμές ή τις στήλες του Y ως προς το X, χρησιμοποιώντας διαφορετικό τύπο γραμμής . Η επιλογή της σχεδιάσης των γραμμών ή στηλών του Y γίνεται έτσι ώστε να υπάρχει ίδιο πλήθος στοιχείων με το διάνυσμα X. Αν ο πίνακας είναι τετραγωνικός τότε σχεδιάζονται οι στήλες του Y ως προς το X.
2. Εαν ο X είναι πίνακας και το Y διάνυσμα, τότε ισχύουν τα παραπάνω με τη διαφορά ότι τώρα οι γραμμές του X σχεδιάζονται σε σχέση με το διάνυσμα Y.
3. Εαν τα X,Y είναι και τα δύο πίνακες με το ίδιο μέγεθος, η εντολή plot(X,Y) σχεδιάζει τις στήλες του X σε σχέση με τις στήλες του Y.
4. Εαν το X δεν προσδιορίζεται, plot(Y), και το Y είναι πίνακας , τότε σχεδιάζεται η κάθε γραμμή του πίνακα σε σχέση με τον δείκτη που έχουν τα στοιχεία σε κάθε γραμμή.

Ενας άλλος τρόπος για να σχεδιάσουμε πολλαπλές γραμμές σε ένα γράφημα είναι να χρησιμοποιήσουμε την εντολή plot με πολλές παραμέτρους, όπως φαίνεται παρακάτω

```
plot(X1,Y1,X2,Y2,...,Xn,Yn)
```

οι μεταβλητές X1,Y1, X2,Y2, ..., Xn,Yn είναι ζευγάρια από διανύσματα. Κάθε ζευγάρι δημιουργεί μια γραφική παράσταση και τελικά έχουμε πολλές γραφικές παραστάσεις στους ίδιους άξονες. Αυτή η δεύτερη μέθοδος έχει το πλεονέκτημα ότι απεικονίζονται στο ίδιο γράφημα διανύσματα μεδιαφορετικό μήκος. Για παράδειγμα η εντολή

```
plot(X1,Y1,':',X2,Y2,'+')
```

σχεδιάζει το πρώτο ζευγάρι χρησιμοποιώντας τελείες αντί συνεχή γραμμή ενώ για το δεύτερο ζευγάρι χρησιμοποιεί σαν στίγμα το σταυρό.
Στην περίπτωση που ο Z είναι μιγαδικός η εντολή plot(Z) είναι ισοδύναμη της plot(real(Z),imag(Z)).

Οι συναρτήσεις polar, loglog, semilogx και semilogy και χρησιμοποιούνται όπως και η plot. Η διαφορά τους είναι στο ότι τα δεδομένα σχεδιάζονται σε διαφορετικά συστήματα συντεταγμένων.

- **polar**

Η συνάρτηση αυτή καλείται με δύο ορίσματα, **polar(theta,rth)**. Σχεδιάζει τις πολικές συντεταγμένεςτης γωνίας theta σε rad, σε σχέση με την ακτίνα rho. Αν μετά από την χρήση της εντολής χρησιμοποιήσουμε την εντολή grid , τότε πετυχαίνουμε τη σχεδίαση πολικών διακεκομένων γραμμών.

- **loglog**

Η συνάρτηση αυτή χρησιμοποιήται όπως και η plot μόνο που κάνει την εκάστοτε σχεδίαση χρησιμοποιώντας λογαριθμική κλίμακα και στους δύο άξονες.

- **semilogx**

Για την semilogx ισχύει ότι ισχύει και για την plot μόνο που η σχεδίαση γίνεται θέτοντας τον άξονα X σε λογαριθμική κλίμακα και τον άξονα Y σε γραμμική.

- **semilogy**

Για την semilogy ισχύει ότι ισχύει και για την plot μόνο που η σχεδίαση γίνεται θέτοντας τον άξονα Y σε λογαριθμική κλίμακα και τον άξονα X σε γραμμική.

- **contour(Z)**

Η συνάρτηση αυτή παρουσιάζει τον πίνακα Z με περιγράμματα, αντιστοιχώντας κάθε τιμή του πίνακα σε ύψος πάνω από το επίπεδο. Μπορούμε να προσδιορίσουμε πόσα περιγράμματα θα χρησιμοποιηθούν με μι αδεύτερη παράμετρο, contour(Z,N).

- **bar(X,Y)**

Η συνάρτηση **bar** χρησιμοποιεί ράβδους για τη γραφική αναπαράσταση των δεδομένων. Αν ο Y είναι ένας MxN πίνακας τότε σχεδιάζει τις στήλες του Y με N ράβδους. Το διάνυσμα X πρέπει να είναι αύξουσα ή φθίνουσα ακολουθία. Αν το X δεν προσδιορίζεται στην κλήση της συνάρτησης, λαμβάνεται ίσο με το διάνυσμα 1**:**M. Αν το Y είναι διάνυσμα τότε η συνάρτηση σχεδιάζει length(Y) ράβδους.
Μπορούμε ακόμη να προσδιορίσουμε το πάχος της ράβδου με μια τρίτη παράμετρο, **bar(X,Y,width)**. Αν η μεταβλητή width έχει τιμή μεγγαλύτερη της μονάδας, έχουμε επικάλυψη των ράβδων.

- **stairs(X,Y)**

Η συνάρτηση αυτή σχεδιάζει το διάνυσμα Y ως προς το X χρησιμοποιώντας σκαλοπάτια.

- **hist(Y)**

Η συνάρτηση αυτή δημιουργεί το ιστόγραμμα των στοιχείων του Y. Αν ο Y είναι πίνακας το ιστόγραμμα δημιουργείται για κάθε στήλη του Y. Αν η συνάρτηση κληθεί ως εξής:

N = hist(Y,M)

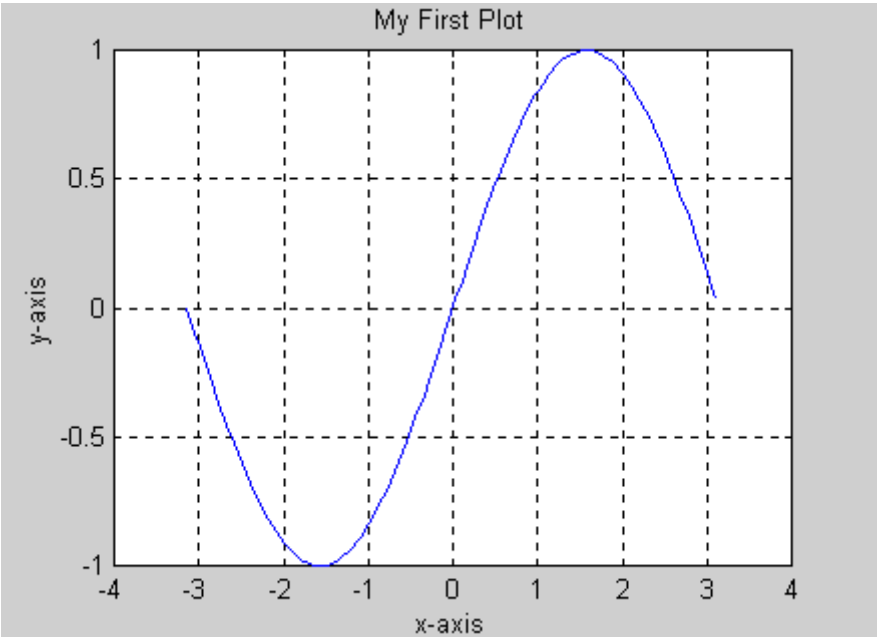
όπου το M είναι ένας αριθμός, η συνάρτηση θα σχεδιάσει M ράβδους. Η συνάρτηση επιστρέφει ένα διάνυσμα N με το πλήθος των στοιχείων του Y σε κάθε ράβδο.

Α.7.2 **Ελεγχος Γραφήματος**

Στο σημείο αυτό θα αναφερθούμε σε κάποιες συναρτήσεις με τις οποίες μπορούμε να επέμβουμε στον τρόπο παρουσίασης της γραφικής παράστασης.
Μπορούμε να τοποθετήσουμε στο γράφημά μας κάποιο τίτλο, να ονομάσουμε τους άξονες καθώς επίσης και να τοποθετήσουμε διακεκομένες γραμμές σε αυτό. Αν γράψουμε στο Command Window:

```
» x = -pi:.05:pi;
» y = sin(x);
» plot(x,y)
» title('My First Plot')
» xlabel('x-axis')
» ylabel('y-axis')
» grid
```

θα πάρουμε την ακόλουθη γραφική παράσταση



Δηλαδή με την εντολή **title**('text') μπορούμε να εισάγουμε ένα τίτλο στο γράφημά μας, με τις **xlabel**('text') και **ylabel**('text') μπορούμε να ονομάσουμε τους άξονες x

και y. Αντίστοιχα ορίζεται και στα τρισδιάστατα γραφικά η **zlabel**('text') για τον τρίτο άξπνα. Με την **grid** δημιουργείται το πλέγμα που βλέπουμε στην παραπάνω γραφική παράσταση.
Μπορούμε ακόμη να ορίσουμε εμείς το εύρος τιμών των αξόνων με τη συνάρτηση **axes**. Η συνάρτηση αυτή καλείται ως εξής

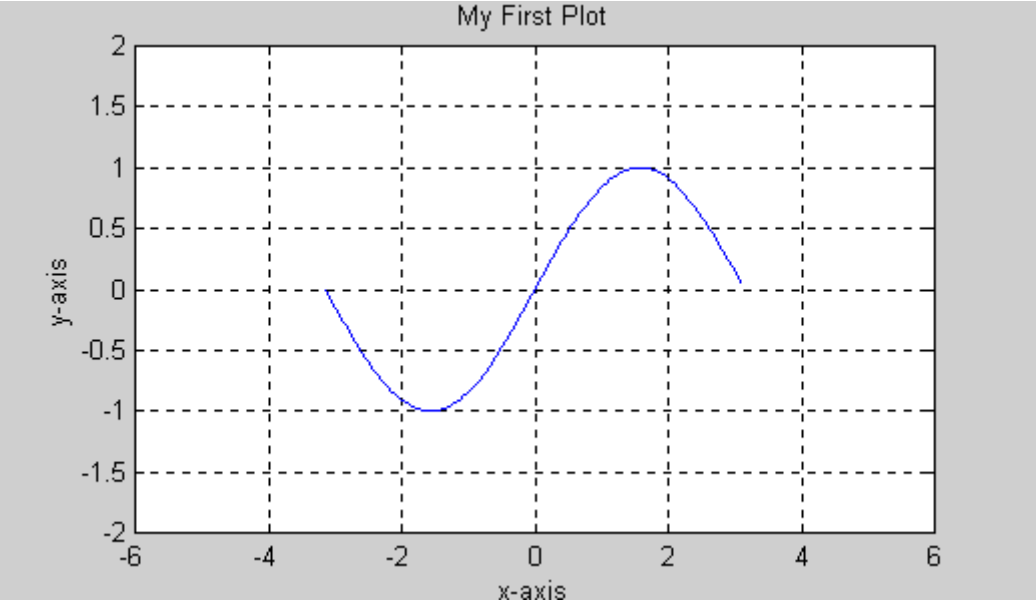
axes([Xmin Xmax Ymin Ymax])

μετά την κλήση της ο άξονας x βαθμολογήται με τιμές από Xmin έως Xmax και ο άξονας y με τιμές από Ymin έως Ymax. Στην περίπτωση τρισδιάστατης γραφικής παράστασης η συνάρτηση μπορεί να κληθεί ως εξής:

axes([Xmin Xmax Ymin Ymax Zmin Zmax])

Παρατηρήστε τις αλλαγές στους άξονες της προηγούμενης γραφικής παράστασης, αν δώσουμε

```
» axis([-6 6 -2 2])
```



Μια άλλη δυνατότητα που παρέχει το σύστημα για την εισαγωγή κειμένου στο γράφημα είναι η συνάρτηση **text**(x,y,'text'). Η συνάρτηση αυτή προσθέτει το κείμενο στη θέση (x,y) του γραφήματος. Αν τα x και y είναι διανύσματα το κείμενο γράφεται στις θέσεις (x(1),y(1)),..., (x(n),y(n)). Αν επιπλέον η τρίτη μεταβλητή είναι πίνακας με πλήθος γραμμών όσο το μήκος των διανυσμάτων x, y , η συνάρτηση text γράφει στις θέσεις (x(1),y(1)),..., (x(n),y(n)) τις αντίστοιχες γραμμές του πίνακα.
Μπορούμε ακόμη να εισάγουμε κείμενο σε οποιοδήποτε σημείο του γραφήματος θέλουμε με την εντολή **gtext**('text'). Η συνάρτηση αυτή δημιουργεί ένα σταυρό στο γράφημά μας, ο οποίος ουσιαστικά είναι ένας δείκτης τον οποίο μετακινούμε όπου

θέλουμε είται με τα βελάκια είται με το ποντίκι, και έπειτα πατώντας κάποιο πλήκτρο εμφανίζεται η λέξη text εκεί που προηγούμενα ήταν ο σταυρός.
Θα μπορούσαμε να γράψουμε ένα κείμενο με πολλές γραμμές στο γράφημα ως εξής

gtext({'This is the first line','This is the second line'})

Μια άλλη σημαντική συνάρτηση είναι η **ginput**(N). Με τη συνάρτηση αυτή μπορούμε να επιλέξουμε N σημεία από το γράφημα. Αν δώσουμε

[X,Y] = ginput(N)

στα διανύσματα X και Y αποθηκεύονται οι συντεταγμένες των σημείων. Ο κέρσορας μπορεί να τοποθετηθεί με το ποντίκι στις διάφορες θέσεις του γραφήματος, τις συντεταγμένες των οποίων επιθυμούμε να λάβουμε. Τα σημεία επιλέγονται με το πάτημα του ποντικού ή κάποιου πλήκτρου από το πληκτρολόγιο, εκτός από το πλήκτρο Enter, με το οποίο τερματίζεται η επιλογή των σημείων πριν εισαχθούν N σημεία. Η συνάρτηση μπορεί να κληθεί και χωρίς την παράμετρο N. Στην περίπτωση αυτή επιλέγουμε όσα σημεία θέλουμε μέχρι να πατήσουμε το πλήκτρο Enter.

Μέχρι τώρα έχουμε δει πως μπορούμε να δημιουργούμε γραφικές παραστάσεις σε ένα παράθυρο γραφικών. Μπορούμε να δημιουργήσουμε ένα νέο παράθυρο γραφικών με την εντολή **figure**. Ας δούμε ένα παράδειγμα

```
» n = 0:99;
» y = sin(2*pi*n/30);
» plot(n,y)
» figure
» x = sin(2*pi*n/50);
» plot(n,x)
```

Οταν εκτελεστεί η εντολή plot(n,y) δημιουργείται ένα παράθυρο γραφικών στο οποίο σχεδιάζεται η γραφική παράσταση του y ως προς το n. Με την εντολή figure δημιουργείται ένα νέο παράθυρο στο οποίο σχεδιάζεται η γραφική παράσταση του x ως προς n. Μπορούμε να αναφερόμαστε στα παράθυρα αυτά δίνοντας **figure(N)** όπου N είναι ο αριθμός του παραθύρου. Στο παραπάνω παράδειγμα με την εντολή figure(1) θα εμφανιστεί το παράθυρο στο οποίο έγινε η γραφική παράσταση του διανύσματος y.
Αν δεν είχαμε δώσει την εντολή figure στο παραπάνω παράδειγμα τότε με την εκτέλεση της δεύτερης εντολή plot, plot(n,x), θα χάναμε το γράφημα της plot(n,y).

Αν θέλαμε να έχουμε στο ίδιο παράθυρο και τις δύο γραφικές παραστάσεις θα γράφαμε

```
» n = 0:99;
» y = sin(2*pi*n/30);
» plot(n,y,'c')
» hold
Current plot held
```

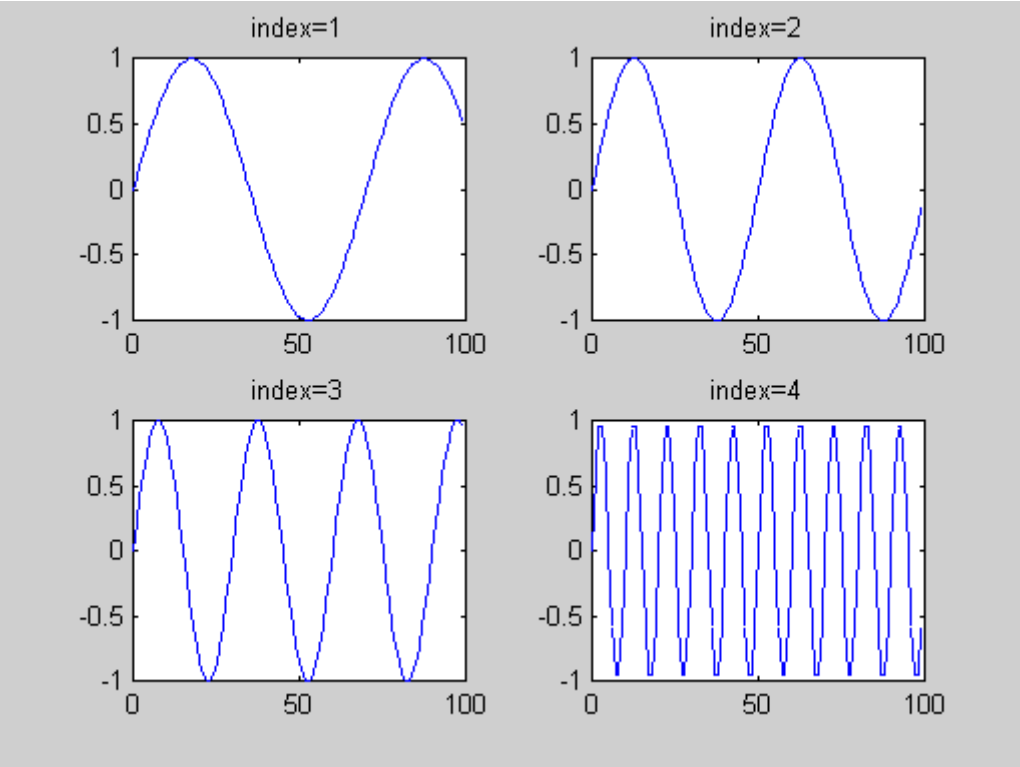
```
» x = sin(2*pi*n/50);
» plot(n,x,'r')
```

οπότε οι γραφικές παραστάσεις θα δημιουργούνταν στο ίδιο παράθυρο.
Οσα plot ακολουθούν την εντολή **hold** θα γίνουν στο ίδιο παράθυρο χωρίς να χαθεί η προηγούμενη γραφική παράσταση. Με την επόμενη εντολή hold αναιρείται η ισχύς της προηγούμενης εντολής hold. Ετσι με την επόμενη plot θα χαθούν τα γραφήματα που υπάρχουν στο παράθυρο.

Με τη συνάρτηση **subplot** μπορούμε να χωρίσουμε το παράθυρο των γραφικών σε πολλές θέσεις εργασίας. Η συνάρτηση καλείται με τρείς ακεραίους, **subplot(n,m,index)**, όπου n είναι το πλήθος των γραμμών που θα έχει το παράθυρό μας m το πλήθος των στηλών και index είναι η τρέχουσα θέση. Ας υποθέσουμε για παράδειγμα ότι θέλουμε να εμφανίζουμε τέσσερις γραφικές παραστάσεις στο ίδιο παράθυρο. Χωρίζουμε το παράθυρο σε δύο γραμμές και δύο στήλες. Σε κάθε μια από αυτές τις θέσεις κάνω και μια γραφική παράσταση ως εξης:

```
» n = 0:99;
» x1 = sin(2*pi*n/70);
» x2 = sin(2*pi*n/50);
» x3 = sin(2*pi*n/30);
» x4 = sin(2*pi*n/10);
» subplot(2,2,1)
» plot(n,x1)
» title('index=1')
» subplot(2,2,2)
» plot(n,x2)
» title('index=2')
» subplot(2,2,3)
» plot(n,x3)
» title('index=3')
» subplot(2,2,4)
» plot(n,x4)
» title('index=4')
```

Παρατηρήστε ότι οι εντολες plot και title που ακολουθούν την subplot(2,2,3) σχεδιάζονται στη θέση όπου η μεταβλητή index έχει τιμή τρία.



Σημειώνουμε ακόμη ότι η κλήση της subplot μπορεί να γίνει είτε γράφοντας subplot(n,m,index) είτε subplot(nmindex).

Α.8 Βιβλιογραφία

1. "Introduction To MATLAB For Engineers And Scientists", Delores M.Etter, Prentice Hall

2. MATLAB Reference Guide
MathWorks

3. The MATLAB EXPO
MathWorks

4. **"**Computer -Based Exercises for Signal Processing Using MATLAB"
Prentice Hall