Εισαγωγή

Το MATLAB (Math Works Inc.) παρέχει ένα δυναμικό, εύχρηστο και ανοικτό υπολογιστικό περιβάλλον για υλοποίηση επιστημονικών εφαρμογών σε ένα μεγάλο φάσμα πεδίων, όπως στη Γραμμική Άλγεβρα, Στατιστική, Εφαρμοσμένα Μαθηματικά, Αριθμητική Ανάλυση και Επιστημονικό Υπολογισμό, Επεξεργασία Σημάτων και Εικόνας, Θεωρία Ελέγχου, Θεωρία Βελτιστοποίησης και Γραφικά. Έχει υλοποιηθεί σε πολλές λειτουργικές πλατφόρμες (όπως Windows, Macintosh OS και Unix) και δύο βασικές εκδόσεις, την επαγγελματική και την εκπαιδευτική (student edition). Βασικές και χρήσιμες αναφορές για το Matlab είναι οι [17] και [18].

Το περιβάλλον του Matlab υποστηρίζει ένα μεγάλο αριθμό ενδογενών λειτουργιών και συναρτήσεων καθώς και εξωτερικές βιβλιοθήκες (Toolboxes) για εξειδικευμένες περιοχές εφαρμογών. Υποστηρίζει επίσης μια ευέλικτη, απλή και δομημένη γλώσσα προγραμματισμού (script language) με πολλές ομοιότητες με την Pascal και παρέχει δυνατότητες εύκολης δημιουργίας, διασύνδεσης και χρήσης βιβλιοθηκών σε κώδικα γραμμένο στη γλώσσα αυτή (M files).

Το Matlab εκτελεί από απλούς μαθηματικούς υπολογισμούς μέχρι και προγράμματα με εντολές παρόμοιες με αυτές που υποστηρίζει μια γλώσσα υψηλού επιπέδου. Συγκεκριμένα εκτελεί απλές μαθηματικές πράξεις, αλλά εξίσου εύκολα χειρίζεται μιγαδικούς αριθμούς, δυνάμεις, ειδικές μαθηματικές συναρτήσεις, πίνακες, διανύσματα και πολυώνυμα. Μπορεί επίσης να αποθηκεύει και να ανακαλεί δεδομένα, να δημιουργεί και να εκτελεί ακολουθίες εντολών που αυτοματοποιούν διάφορους υπολογισμούς και να σχεδιάζει γραφικά.

Οι λειτουργίες του Matlab διακρίνονται στις *τυποποιημένες,* δηλαδή *σε* αυτές που χειρίζονται αριθμητικά δεδομένα και εξάγουν αριθμητικά αποτελέσματα, και στις συναρτήσεις του *Symbolic Toolbox,* οι οποίες χειρίζονται και υπολογίζουν *συμβολικές* εκφράσεις, δηλαδή επεξεργάζονται μαθηματικά σύμβολα.

Η ενότητα αυτή είναι μια σύνοψη των βασικότερων χαρακτηριστικών του Matlab και αποτελείται από δύο μέρη. Στο πρώτο μέρος παρουσιάζονται οι βασικές εντολές, λειτουργίες και χαρακτηριστικά του Matlab. Συγκεκριμένα περιγράφονται οι υποστηριζόμενοι τελεστές και πράξεις, οι στοιχειώδεις μαθηματικές συναρτήσεις και οι εντολές που περιλαμβάνει η ενσωματωμένη γλώσσα προγραμματισμού. Ιδιαίτερη έμφαση δίνεται στο λογισμό και στις πράξεις πινάκων. Τέλος δίνονται οι τρόποι σχεδίασης γραφικών παραστάσεων και κατασκευής συναρτήσεων .

Στο δεύτερο μέρος παρουσιάζονται ενδεικτικά μερικές αντιπροσωπευτικές και απλές εφαρμογές του Matlab στη Γραμμική Άλγεβρα και στις Αριθμητικές Μεθόδους. Επίσης δίνονται μερικές αριθμητικές μέθοδοι. υλοποιημένες στη γλώσσα script. Τα προγράμματα αυτά έχουν ληφθεί κατά ένα μέρος από τη βασική βιβλιοθήκη του Matlab ή από το διεθνές δίκτυο και έχουν υποστεί μερικές προσθήκες και βελτιώσεις. Για την πληρέστερη κατανόησή τους ο αναγνώστης παραπέμπεται στις συναφείς παραγράφους του βιβλίου αυτού καθώς και στις αναφορές [17] και [18].

Β.1 Το Υπολογιστικό Περιβάλλον του Matlab

Β.1.1 Το περιβάλλον εργασίας

Με την ενεργοποίηση του το Matlab εμφανίζει ένα διαλογικό *παράθυρο εντολών*. Κατά τη διάρκεια της εργασίας το περιβάλλον «θυμάται» τις εντολές που έχουν δοθεί, όπως και τις μεταβλητές που έχουν δημιουργηθεί. Οι εντολές και οι μεταβλητές αυτές αποτελούν το χώρο εργασίας (*workspace*) του Matlab. Τα ονόματα των μεταβλητών που έχουν δημιουργηθεί εμφανίζονται με την εντολή **who**:

```
» who
Your variables are:
a       step    x       y
```

Εκτός από τις μεταβλητές που δημιουργεί ο χρήστης, υπάρχουν *προκαθορισμένες* μεταβλητές και σταθερές με ειδική σημασία οι οποίες φαίνονται στον παρακάτω πίνακα:

Μεταβλητή	Τιμή
ans	Το αποτέλεσμα κάθε αριθμητικής πράξης
pi	Ο αριθμός π
eps	Ο κοντινότερος αριθμός στο 0
inf	Άπειρο
NaN	Όχι αριθμός (π.χ. 0/0)
i και j	$i = j = \sqrt{-1}$
realmin	Ο μικρότερος θετικός πραγματικός αριθμός
realmax	Ο μεγαλύτερος θετικός πραγματικός αριθμός

Ο πιο γρήγορος τρόπος για να μάθουμε τη λειτουργία μιας εντολής του Matlab είναι η εντολή **help**:

```
» help inv
 INV Matrix inverse.
 INV(X) is the inverse of the square matrix X.
 A warning message is printed if X is badly scaled or nearly singular.
```

Η εντολή help χωρίς όρισμα εμφανίζει τα θέματα βοήθειας:

```
» help
HELP topics:
toolbox\local       - Local function library.
matlab\datafun      - Data analysis and Fourier transform functions.
matlab\elfun        - Elementary math functions.
matlab\elmat        - Elementary matrices and matrix manipulation.
matlab\funfun       - Function functions - nonlinear numerical methods.
matlab\general      - General purpose commands.
matlab\color        - Color control and lighting model functions.
matlab\graphics     - General purpose graphics functions.
matlab\iofun        - Low-level file I/O functions.
matlab\lang         - Language constructs and debugging.
matlab\matfun       - Matrix functions - numerical linear algebra.
matlab\ops          - Operators and special characters.
matlab\plotxy       - Two dimensional graphics.
matlab\plotxyz      - Three dimensional graphics.
matlab\polyfun      - Polynomial and interpolation functions.
matlab\sounds       - Sound processing functions.
matlab\sparfun      - Sparse matrix functions.
matlab\specfun      - Specialized math functions.
matlab\specmat      - Specialized matrices.
matlab\strfun       - Character string functions.
matlab\dde          - DDE Toolbox.
matlab\demos        - The MATLAB Expo and other demonstrations.
For more help on directory/topic, type "help topic".
```

Η εντολή **lookfor** δίνει βοήθεια ψάχνοντας την πρώτη γραμμή όλων των κειμένων βοήθειας που είναι διαθέσιμα για τις εντολές του Matlab και επιστρέφει τις γραμμές που περιέχουν την λέξη που αναζητείται:

» lookfor interpolation

```
contents.m: % Polynomial and interpolation functions.
ICUBIC Cubic Interpolation of a 1-D function.
INTERP1 1-D data interpolation (table lookup).
INTERP2 2-D data interpolation (table lookup).
INTERP3 2-D biharmonic data interpolation and gridding.
INTERP4 2-D bilinear data interpolation.
INTERP5 2-D bicubic data interpolation.
INTERP6 2-D Nearest neighbor interpolation.
INTERPFT 1-D interpolation using a FFT method.
SPLINE          Cubic spline data interpolation.
SPAPIDM2 Demonstrate spline interpolation.
```

Το path είναι το μονοπάτι εύρεσης στο οποίο ψάχνει το Matlab το όνομα μιας εντολής που εισάγεται ή το όνομα ενός εκτελέσιμου αρχείου. Η εντολή **path** εμφανίζει τη λίστα των ορισθέντων μονοπατιών :

```
» path
        MATLABPATH

    e:\matlab\toolbox\local
    e:\matlab\toolbox\matlab\datafun
    e:\matlab\toolbox\matlab\elfun
                . . . . . .
```

Για να προσθέσουμε ένα υποκατάλογο P στο τρέχον μονοπάτι, δίνουμε την εντολή **path(PATH,P)**. Για παράδειγμα δίνοντας path(path, 'e:\matlab\myfiles'), το μονοπάτι γίνεται:

```
            MATLABPATH
    e:\matlab\toolbox\local
    e:\matlab\toolbox\matlab\datafun
    e:\matlab\toolbox\matlab\elfun
    . . . . . .
   e:\matlab\myfiles
```

Η εντολή **dir** εμφανίζει τα περιεχόμενα του τρέχοντος καταλόγου, ενώ η εντολή **cd** θέτει τον κατάλογο εργασίας.

Η εμφάνιση των αριθμών διέπεται από ορισμένους κανόνες. Εκτός και αν ορισθεί διαφορετικά, αν ένα αποτέλεσμα είναι ακέραιος, εμφανίζεται σαν ακέραιος. Όμοια, αν το αποτέλεσμα είναι πραγματικός, εμφανίζεται με 4 δεκαδικά ψηφία. Αν τα σημαντικά ψηφία του αποτελέσματος είναι έξω από αυτό το όριο, το αποτέλεσμα εμφανίζεται σε μορφή κινητής υποδιαστολής. Ο χρήστης όμως μπορεί να ορίσει τη μορφή του αποτελέσματος με την εντολή **format**. Ο παρακάτω πίνακας συνοψίζει τις μορφές της:

Εντολή	Παράδειγμα Τιμής	Σχόλια
format long	3.14159265358979	16 ψηφία
format short e	3.1416e+000	5 ψηφία και εκθέτης
format long e	3.141592653589793e+000	16 ψηφία και εκθέτης
format hex	400921fb54442d18	Δεκαεξαδικό
format bank	3.14	2 δεκαδικά ψηφία
format +	+	πρόσημο ή 0
format rat	355/113	κλασματική προσέγγιση
format short	3.1416	4 δεκαδικά ψηφία

Β.1.2 Τελεστές και ειδικά σύμβολα

Οι υποστηριζόμενοι τελεστές και η σημασία τους φαίνονται στον παρακάτω πίνακα:

Χαρακτήρας	Λειτουργία

+	Πρόσθεση
-	Αφαίρεση
*	Πολλαπλασιασμός αριθμών ή πινάκων
.*	Πολλαπλασιασμός πινάκων κατά στοιχείο
^	Ύψωση σε δύναμη αριθμού ή πίνακα
.^	Ύψωση σε δύναμη πίνακα κατά στοιχείο
\	Αριστερή Διαίρεση
/	Δεξιά Διαίρεση
./	Διαίρεση πινάκων κατά στοιχείο
:	Άνω κάτω τελεία (σύμβολο περιοχής)
( )	Παρενθέσεις
[ ]	Αγκύλες
.	Δεκαδική υποδιαστολή
,	Διαχωριστής στοιχείων
;	Διαχωριστής γραμμών κατά τη δημιουργία πίνακα
%	Εισαγωγή σχολίων
!	Εισαγωγή εντολών στο λειτουργικό σύστημα
‘	Μετατροπή αριθμού ή στοιχείων πίνακα σε συζυγείς
=	Απόδοση τιμής
==	Έλεγχος ισότητας
< , >, >=, <=	Τελεστές συσχέτισης
~	Λογικό ‘ΟΧΙ’
|	Λογικό ‘Η’
&	Λογικό ‘ΚΑΙ’
xor	Αποκλειστικό ‘Η’

Αριθμητικές Πράξεις

Για την εκτέλεση απλών αριθμητικών πράξεων μεταξύ πραγματικών και ακεραίων χρησιμοποιούνται οι οι συνήθεις τελεστές (όπως σε ένα απλό υπολογιστή τσέπης). Π.χ. :

```
» 3+5                » 8 / 3
 ans =                ans =
      8                    2.6667
```

Η προκαθορισμένη μεταβλητή ***ans*** δέχεται αυτόματα το αποτέλεσμα ενός υπολογισμού, όταν δεν χρησιμοποιείται συγκεκριμένη μεταβλητή για την εκχώρηση τιμής.

Οι βασικές πράξεις μεταξύ πραγματικών επεκτείνονται και στους μιγαδικούς αριθμούς. Οι προσδιοριστές **i** και **j** έχουν την προκαθορισμένη σημασία της φανταστικής μονάδας.

Για να μην εμφανίζεται το αποτέλεσμα μιας εντολής, αρκεί να τεθεί στο τέλος της το σύμβολο «;».

Μπορούμε εναλλακτικά να αποθηκεύουμε τιμές σε μεταβλητές και να κάνουμε πράξεις με αυτές. Οι μεταβλητές μπορούν να είναι βαθμωτά μεγέθη, διανύσματα και πίνακες.

Παραδείγματα

```
»a = 3                          » c1=1+2i
   a =                          c1 =
       3                          1.0000 + 2.0000i
»b = 5;                         » c2=2+3i
»c = a+b                        c2 =
   c =                            2.0000 + 3.0000i
       8                        » c1*c2'  {γινόμενο επί συζυγή}
»2*c ^3                         ans =
   ans=                           8.0000 + 1.0000i
       1024
```

Β.1.3 Στοιχειώδεις μαθηματικές συναρτήσεις

Οι υποστηριζόμενες βασικές συναρτήσεις και η σημασία τους φαίνονται στον παρακάτω πίνακα:

Συνάρτηση	Λειτουργία
sin	Ημίτονο
sinh	Υπερβολικό ημίτονο
asin	Αντίστροφο ημίτονο
asinh	Αντίστροφο υπερβολικό ημίτονο
cos	Συνημίτονο
cosh	Υπερβολικό συνημίτονο
acos	Αντίστροφο συνημίτονο
acosh	Αντίστροφο υπερβολικό συνημίτονο
tan	Εφαπτομένη
tanh	Υπερβολική εφαπτομένη
atan	Αντίστροφη εφαπτομένη
atanh	Αντίστροφη Υπερβολική εφαπτομένη
sec	Τέμνουσα
sech	Υπερβολική τέμνουσα
asec	Αντίστροφη τέμνουσα
asech	Αντίστροφη υπερβολική τέμνουσα
csc	Συντέμνουσα
csch	Υπερβολική συντέμνουσα
acsc	Αντίστροφη συντέμνουσα
acsch	Αντίστροφη υπερβολική συντέμνουσα
cot	Συνεφαπτομένη
coth	Υπερβολική συνεφαπτομένη
acot	Αντίστροφη συνεφαπτομένη
acoth	Αντίστροφη υπερβολική συνεφαπτομένη
exp	Εκθετική συνάρτηση e^{-x}
log	Νεπέριος (φυσικός) λογάριθμος
log10	Δεκαδικός λογάριθμος
sqrt	Τετραγωνική ρίζα
abs	Απόλυτη τιμή
angle	Γωνίες φάσης στοιχείων μιγαδικού πίνακα
conj	Συζυγής μιγαδικού
imag	Φανταστικό μέρος μιγαδικού
real	Πραγματικό μέρος μιγαδικού
fix	Ακέραιο μέρος
floor	Κάτω ακέραιο μέρος
ceil	Πάνω ακέραιο μέρος
round	Στρογγυλοποίηση
rem	Υπόλοιπο διαίρεσης
sign	Πρόσημο

Οι βασικές συναρτήσεις καλούνται με το όνομά τους. Π.χ. η κλήση sin(pi) θα δώσει ans = 0.0000.

Β.1.4 Πίνακες και Επεξεργασία τους

Ορισμός πινάκων

Οι πίνακες παίζουν σημαντικό ρόλο στις λειτουργίες του Matlab και υποστηρίζονται από αντίστοιχες μεταβλητές. Ένας πίνακας γράφεται μέσα σε αγκύλες "[ ]", τα στοιχεία μιας γραμμής χωρίζονται με κενά ή «,», ενώ κάθε γραμμή με ερωτηματικό (;) ή "return". Για να προσπελάσουμε έναν πίνακα, δηλώνουμε μέσα σε παρένθεση τους δείκτες του. Τα στοιχεία ενός πίνακα αποθηκεύονται *κατά στήλες*. Ο κενός πίνακας συμβολίζεται με [].

<u>*Παράδειγμα*</u>: Για να εκχωρήσουμε τον πίνακα $A = \begin{pmatrix} 0 & 2 & 1 \\ 1 & -2 & 4 \\ 4 & -1 & 2 \end{pmatrix}$, γράφουμε στο prompt του Matlab:

```
» A=[0,2,1;1,-2,4;4,-1,2]                                   {πίνακας 3 x 3}
A =
    0    2    1
    1   -2    4
    4   -1    2
» A(2,2)
   ans = -2

» firstrow=[0 2 1]                                          {πίνακας 1 x 3}
b =
  1    2    1

» firstcol=[0;1;4]                                          {πίνακας 3 x  1}
B =
  0
  1
  4
```

Κατασκευή και Προσπέλαση πινάκων

Η αναφορά σε *τμήματα στοιχείων* πίνακα γίνεται με χρήση του συμβόλου '**:**'. Έτσι, με την αναφορά **A(i:k, j:h)** προσπελαύνουμε τα στοιχεία ενός πίνακα Α δύο διαστάσεων που ανήκουν στις γραμμές i μέχρι και k και στις στήλες j μέχρι και h. Γενικότερα, με τον τρόπο αυτόν μπορούν να απομονώνονται και να κατασκευάζονται τμήματα πινάκων.

<u>*Παραδείγματα:*</u>

```
» A(2:5)                    { Τα στοιχεία 2 ως 5 του Α. Τα στοιχεία του Α διατάσσονται κατά στήλες }

ans =
    1    4    2   -2

» A(1:2, 2:3)                       { Οι γραμμές 1 και 2 των στηλών 2 και 3 }
ans =
    2    1
   -2    4

» A(:, 2:3)                         {ο Α χωρίς την 1η στήλη του }
ans =
    2    1
   -2    4
   -1    2
» A(2:3,:)                          {ο Α χωρίς την 1η γραμμή του }
```

```
ans =
    1   -2    4
    4   -1    2

» A(:,:)                          { Όλες οι γραμμές και οι στήλες του A }
A =
    0    2    1
    1   -2    4
    4   -1    2

» A(:);              {Ο A μετατρέπεται σε πίνακα-στήλη ανά στήλη και αποθηκεύεται στον πίνακα ans }
                     { Το ';' αποτρέπει την εμφάνιση των στοιχείων του τελευταίου }
```

Μπορούμε να κατασκευάζουμε μονοδιάστατους πίνακες-γραμμές, χωρίς να καταγράφουμε όλα τα στοιχεία τους, αλλά δηλώνοντας την αρχική και τελική τιμή του καθώς και το βήμα, ή τον αριθμό των στοιχείων του. Έτσι παρέχεται η δυνατότητα κατασκευής γραμμικών και λογαριθμικών διαστημάτων τιμών:

- Η εντολή ***x=(a:s:b)*** (ή ***x=a:s:b*** ή ***x=[a:s:b]***) κατασκευάζει γραμμικά διαστήματα, δηλαδή πίνακες-γραμμές με πρώτο στοιχείο το *a,* τελευταίο το *b*, και βήμα *s*. Η ένδειξη ***a:s:b*** χρησιμοποιείται ευρύτατα σαν μηχανισμός για προσπέλαση πινάκων, αφού μπορεί να προσδιορίζει ευέλικτα περιοχές δεικτών.
- *Γραμμικά διαστήματα* κατασκευάζονται επίσης με την εντολή ***linspace***. Η μορφή της είναι:

linspace(αρχική_τιμή, τελική_τιμή, αριθμός_τιμών)

- *Λογαριθμικά διαστήματα* κατασκευάζονται με την εντολή ***logspace***. Η μορφή της είναι:

logspace(αρχικός_εκθέτης , τελικός_εκθέτης, αριθμός_τιμών)

Παραδείγματα:

```
» x=2:8                           {πίνακας ισαπεχόντων τιμών με βήμα 1στο διάστημα [2, 8]}
x =
     2    3    4    5    6    7    8

x=[0,1,2:8]
x =
    0    1    2    3    4    5    6    7    8

» y=56:3:80                       {πίνακας  ισαπεχόντων τιμών με βήμα 3 στο διάστημα [56, 80] }
y =
   56   59   62   65   68   71   74   77   80

» x=linspace(0,pi,7)              {πίνακας 7 ισαπεχόντων τιμών μεταξύ 0 και π}
x =
       0   0.5236   1.0472   1.5708   2.0944   2.6180   3.1416

» logspace(0,2,7)                 {δημιουργία 7 τιμών σε λογαριθμική κλίμακα μεταξύ 1 και 100}
  ans =
    Columns 1 through 7
     1.0000   1.3895   1.9307   2.6827   3.7276   5.1795   7.1969
```

Μπορούμε επίσης να προσπελάσουμε στοιχεία πίνακα με οποιαδήποτε σειρά βάζοντας τις θέσεις τους εντός [ ]. Ο μηχανισμός αυτός δεικτοδότησης επεκτείνεται και για προσπέλαση γραμμών και στηλών πινάκων. Μπορεί επίσης να συνδυασθεί με τον μηχανισμό προσπέλασης που παρέχει το σύμβολο ':'. Με βάση τα παραπάνω μπορούμε να δημιουργούμε πίνακες από τμήματα άλλων.

Παραδείγματα Για να προσπελάσουμε διακεκριμένα στοιχεία ή περιοχές του πίνακα Α που είδαμε παραπάνω, δίνουμε τις εξής εντολές:

```
» A([6,9,1,2,4])  προσπέλαση τυχαίων στοιχείων του Α (τα στοιχεία προσπελαύνονται κατά στήλες)}
ans =
   -1    2    0    1    2

» A([6,9,1:3])
ans =
   -1    2    0    1    4

» A(:,[1 3])                          {η πρώτη και τρίτη στήλη του Α }
ans =
    0    1
    1    4
    4    2

» A([1,3],:)                          {η πρώτη και τρίτη γραμμή του Α }
ans =
    0    2    1
    4   -1    2

» A(1:2,[1 3])
ans =
    0    1
    1    4

» A(1,[1 3])
ans =
    0    1
```

Είναι δυνατό επίσης να ενώσουμε δύο και παραπάνω πίνακες, αρκεί να έχουν τον ίδιο αριθμό γραμμών.

Παραδείγματα:

```
x=linspace(0,pi,7);                   {συνένωση διαστημάτων }
y=logspace(0,2,7);
» c=[x y]
  Columns 1 through 7
       0    0.5236    1.0472    1.5708    2.0944    2.6180    3.1416
  Columns 8 through 14
    1.0000    2.1544    4.6416   10.0000   21.5443   46.4159  100.0000

» C=[A B];                            {οι πίνακες Α, Β πρέπει να έχουν τον ίδιο αριθμό γραμμών}

» b=[6 2 3];
» Ab=[A b(:)]                         {δημιουργία επαυξημένου πίνακα}
Ab =
    0    2    1    6
    1   -2    4    2
    4   -1    2    3
```

Το βήμα s στον μηχανισμό δεικτοδότησης *a:s:b* μπορεί να είναι και αρνητικό, έτσι ώστε να μπορούμε να προσπελάσουμε ευέλικτα περιοχές πινάκων, διατρέχοντας τις γραμμές ή τις στήλες κατά οποιαδήποτε κατεύθυνση.

Παραδείγματα:

```
»  j=A(3:-1:1)
j =
    4    1    0

» k=A(3:-1:1,1:2)
k =
    4   -1
    1   -2
    0    2

» l=A(1:1:3,1:2)
l =
    0    2
    1   -2
    4   -1

»  j=A(3:-1:1,1:3)
j =
    4   -1    2
    1   -2    4
    0    2    1

» B=[1 1 1 ;2 3 -1 ;4 -2 6]                    {ορισμός νέου πίνακα 3x3}
B =
    1    1    1
    2    3   -1
    4   -2    6

»  j=[A B(:,[1 2])]                              {συνένωση πίνακα με τμήμα ενός άλλου}
j =
    0    2    1    1    1
    1   -2    4    2    3
    4   -1    2    4   -2

»  j=[A B(:,[3])]
j =
    0    2    1    1
    1   -2    4   -1
    4   -1    2    6
```

Τέλος, ο ανάστροφος ενός πίνακα A προσπελαύνεται με χρήση του τελεστή '**.'**'. Ο τελεστής '**'**' χρησιμοποιείται για τον υπολογισμό του ανάστροφου συζυγή, προκειμένου για πίνακες μιγαδικών.

Παραδείγματα:

```
» A.'        {Ανάστροφος πίνακας. Ο συμβολισμός A' είναι ισοδύναμος αν  ο A είναι πίνακας πραγματικών}
ans =
    0    1    4
    2   -2   -1
    1    4    2
» [A,b.']                                       {επαυξημένος πίνακας}
ans =
    0    2    1    6
    1   -2    4    2
    4   -1    2    3
```

```
»d=[ 1-i  2+3*i ]                          { Δημιουργία πίνακα μιγαδικών }
d =
    1.0000 - 1.0000i   2.0000 + 3.0000i
»d '                                       { Μετατροπή σε ανάστροφο με συζυγή στοιχεία }
ans =
          1.0000  + 1.0000i
          2.0000 - 3.0000i
```

Πράξεις Πινάκων

Υποστηρίζονται τρεις κατηγορίες πράξεων σε πίνακες. Οι χρησιμοποιούμενοι τελεστές διακρίνονται και διαφοροποιούνται ανάλογα με την κατηγορία. Οι τελεστές είναι οι

' +', '-', '*', '^' ,'\' ,'/', '.*', './', '.\' και **'.^'.**

- **Πράξεις μεταξύ πινάκων ίδιων διαστάσεων**, ***"στοιχείο προς στοιχείο"***:
  Οι πράξεις πινάκων *"στοιχείο προς στοιχείο"* εκτελείται μία φορά για κάθε στοιχείο των πινάκων-εντέλων και αποθηκεύεται στην αντίστοιχη θέση του πίνακα-αποτελέσματος.

  - Πρόσθεση πινάκων: **A+B**
  - Αφαίρεση πινάκων: **A-B**
  - Πολλαπλασιασμός κατά στοιχεία: **A.*B**
  - Διαίρεση πινάκων κατά στοιχεία: **A./B** ή **B.\A**.
  - Ύψωση των στοιχείων πίνακα στις δυνάμεις των αντίστοιχων στοιχείων ενός άλλου: **A.^B**

- **Πράξεις μεταξύ βαθμωτού (πραγματικού ή μιγαδικού) και πίνακα**:

  - Πρόσθεση βαθμωτού c σε (όλα τα στοιχεία) πίνακα: **c+A** ή **A+c**
  - Αφαίρεση πίνακα από βαθμωτό: **c-A**
  - Πολλαπλασιασμός πίνακα με βαθμωτό : **c*A** ή και **c.*A**
  - Διαίρεση πίνακα με βαθμωτό: **A/.c** ή και **c.\A**.
  - Ύψωση των στοιχείων πίνακα σε μια βαθμωτή δύναμη: **A.^x**

- **Σ*ύνθετες πράξεις* πινάκων**, δηλαδή αυτές που προϋποθέτουν πιο σύνθετους υπολογισμούς:

  - Πολλαπλασιασμός πινάκων (με διαστάσεις που συμφωνούν): **A*B**
  - Αντιστροφή πίνακα: δίνεται από τη συνάρτηση ***inv***: **inv(A)**
  - Η *ορίζουσα* πίνακα δίνεται από τη συνάρτηση ***det***: **det(A)**
  - Διαίρεση πινάκων, που έχει το νόημα του πολλαπλασιασμού με τον αντίστροφο πίνακα, αν αυτός υπάρχει: **A/B** $=AB^{-1}$, *αριστερή διαίρεση*, ή **B\A**$= B^{-1}A$, *δεξιά διαίρεση*.
  - Ύψωση τετραγωνικού πίνακα σε μια βαθμωτή δύναμη c: **A^c**

Οι κατασκευαζόμενες αλγεβρικές εκφράσεις πινάκων υπολογίζονται αυτόματα και τα αποτελέσματα αποδίδονται στις χρησιμοποιούμενες μεταβλητές ή στην προκαθορισμένη μεταβλητή *ans*.

Παράδειγμα 1: Μερικές πράξεις πινάκων είναι:

```
»A=[0 2 1;1 -2 4;4 -1 2];              {ορισμός πινάκων}
»B=[1 1 1 ;2 3 -1 ;4 -2 6];
»b=[6 2 3];

» b.*b                                 {ισοδύναμο με b.^2}
ans =
   36    4    9
```

```
» b.*(2*b-1)
ans =
   66     6    15

y=(1.5:0.1:2)*pi                    {πολλαπλασιασμός διανύσματος περιοχής με π}
y =
   4.7124    5.0265    5.3407    5.6549    5.9690    6.2832

» C=A+2*B                           {μια αλγεβρική έκφραση πινάκων}
C =
    2     4     3
    5     4     2
   12    -5    14

» A.*B                              {πολλαπλασιασμός κατά στοιχεία}
ans =
    0     2     1
    2    -6    -4
   16     2    12

» A./B                              {διαίρεση κατά στοιχεία}
ans =
       0    2.0000    1.0000
   0.5000   -0.6667   -4.0000
   1.0000    0.5000    0.3333

» A/B                               {πολλαπλασιασμός με τον αντίστροφο του B}
ans =
   3.0000   -0.6250   -0.4375
   1.0000   -0.7500    0.3750
  -3.0000    1.3750    1.0625

» B./A                              {διαίρεση κατά στοιχεία}
Warning: Divide by zero
ans =                               {το άπειρο συμβολίζεται από τη σταθερά Inf}
      Inf    0.5000    1.0000
   2.0000   -1.5000   -0.2500
   1.0000    2.0000    3.0000

» B.\A                              {διαίρεση κατά στοιχεία}
ans =
       0    2.0000    1.0000
   0.5000   -0.6667   -4.0000
   1.0000    0.5000    0.3333

» A^2                               {τετραγωνική δύναμη πίνακα}
ans =

    6    -5    10
   14     2     1
    7     8     4

» A.^2                              {ύψωση στοιχείων πίνακα σε δύναμη}
ans =
    0     4     1
    1     4    16
   16     1     4
```

```
» A.^(1/2)                          {ύψωση στοιχείων πίνακα σε δύναμη}
ans =
      0          1.4142          1.0000
  1.0000          0.0000 + 1.4142i   2.0000
  2.0000          0.0000 + 1.0000i   1.4142
```

Παράδειγμα 2: ***Επίλυση γραμμικού συστήματος nxn.***
Σχετικά με την επίλυση ενός συστήματος Ax=b, που δίνεται από την $x=A^{-1}b$, έχουμε τους παρακάτω υπολογισμούς.

```
» A=[0 2 1;1 -2 4;4 -1 2];                { πίνακας Α των συντελεστών των αγνώστων}
» b=[6 2 3].'                             {το διάνυσμα b σε μορφή στήλης}
» det(A)                                  {υπολογίζουμε την ορίζουσα του Α}
ans =
   35                                     {είναι η μηδενική, άρα υπάρχει μια μοναδική λύση}

» inv(A)                                  {Υπολογισμός αντιστρόφου}
ans =
   0.0000   -0.1429    0.2857
   0.4000   -0.1143    0.0286
   0.2000    0.2286   -0.0571

» x=inv(A)*b                              { Υπολογισμός διανύσματος λύσεων}
x =
   0.5714
   2.2571
   1.4857

» x=A\b          {Εναλλακτική και προτιμότερη υπολογιστικά λύση: η χρήση του συμβόλου \ }
                 { χρησιμοποιεί εσωτερικά την μέθοδο LU}
x =
   0.5714
   2.2571
   1.4857
```

Σύγκριση σε πίνακες και Λογικοί Πίνακες

Το Matlab υποστηρίζει λογικές εκφράσεις που εμπλέκουν πίνακες. Έτσι είναι δυνατή η σύγκριση πινάκων, είτε μεταξύ τους είτε με ένα αριθμό, η οποία και παράγει πίνακες λογικών τιμών. Οι λογικές τιμές αναπαρίστανται με 1 (true) και 0 (false). Έτσι , η εντολή

<πίνακας >|<αριθμός> <τελεστής σύγκρισης> <πίνακας >|<αριθμός>

παράγει έναν πίνακα με 1 και 0. Με τη βοήθεια των λογικών πινάκων είναι σε πολλές περιπτώσεις δυνατή η απλούστευση της προσπέλασης και της επεξεργασίας τους.

Παραδείγματα

```
»A=[0,2,1;1,-2,4;4,-1,2];
»B=[1 1 1 ;2 3 -1 ;4 -2 6];
»  F=abs(A)==2
F =
   0    1    0
   0    1    0
   0    0    1

»  F=abs(A)>=2
```

```
F =
    0    1    0
    0    1    1
    1    0    1

» A>2
ans =
    0    0    0
    0    0    1
    1    0    0

» F=abs(A)>B
F =
    0    1    0
    0    0    1
    0    1    0

»F=abs(A)~=abs(B)
F =
    1    1    0
    1    1    1
    0    1    1
```

Ένας πίνακας F λογικών τιμών (1 ή 0) μπορεί να δεικτοδοτήσει έναν πίνακα A ίδιων διαστάσεων. Η αναφορά A(F) δίνει ένα διάνυσμα-στήλη που περιέχει τα στοιχεία A(i,j) για τα οποία είναι F(i,j)=1. Π.χ.:

```
»A=[0,2,1;1,-2,4;4,-1,2];
»L=[1,0,0;1,0,0;0 0 1];
» A(L)
ans =
    0
    1
    2
```

Ο παραπάνω μηχανισμός μας επιτρέπει να πάρουμε τα στοιχεία ενός πίνακα που επαληθεύουν μια σύγκριση. Δεν έχουμε για τον σκοπό αυτό, παρά να δεικτοδοτήσουμε τον πίνακα με την λογική έκφραση που αναπαριστά τη σύγκριση:

```
»y=A(A>2)                    {τα στοιχεία του A που είναι μεγαλύτερα από το 2}
y =
    4
    4

» y=A(A>B);                  {τα στοιχεία του A που είναι μεγαλύτερα από τα αντίστοιχα του B}
                             {αποτέλεσμα: y =(2,-1,4)^t }
```

Η διευθύνσεις των στοιχείων πίνακα που επαληθεύουν μια σύγκριση μπορούν να ανακτηθούν με τη συνάρτηση ***find***. Οι τρόποι κλήσης της είναι:

- **i = *find*(x)** : επιστρέφει τους δείκτες των μη μηδενικών στοιχείων ενός διανύσματος x .
- **[i,j] = *find*(X)** : επιστρέφει τους δείκτες γραμμών και στηλών των μη μηδενικών στοιχείων ενός πίνακα X.
- **[i,i,v] = *find*(X)** : επιστρέφει επίσης το διάνυσμα-στήλη v με τα μη μηδενικά στοιχεία του X.

Παραδείγματα Επανερχόμενοι στα τελευταία παραδείγματα, έχουμε:

```
» find(A>2)                  {μονοδιάστατες διευθύνσεις του A}
ans =
    3
```

```
    8

» [i,j]=find(A>2)            {τα διανύσματα i και j  περιέχουν τους δείκτες γραμμών και στηλών }
i =
    3
    2
j =
    1
    3

» [i,j]=find(A>B)
i =
    1
    3
    2
j =
    2
    2
    3

» [i,j,v]=find(A);           {το v περιέχει τα μη μηδενικά στοιχεία του A, ενώ τα i,j τις συντεταγμένες
τους}
» [i,j,v]=find(A>2);         {όλα  τα στοιχεία του v είναι 1}
```

Ειδικοί πίνακες

Για την κατασκευή ειδικών πινάκων υπάρχουν ειδικές συναρτήσεις. Συγκεκριμένα μπορούμε μα κατασκευάζουμε τους εξής τετραγωνικούς και μη πίνακες:

- *Μηδενικούς* πίνακες: ***zeros(k)***, ή ***zeros(n,m)***
- *Μοναδιαίους* πίνακες: ***eye(k)*** ή ***zeros(n,m)***
- Πίνακες με *όλα* τα *στοιχεία* τους 1: ***ones(k),*** ή ***ones(n,m)***
- Πίνακες *τυχαίων στοιχείων:* ***randn(k),*** ή ***randn(n,m)***. Η randn παράγει τυχαίους αριθμούς μεταξύ 0 και 1.

όπου k το μέγεθος τετραγωνικού πίνακα και n και m ο αριθμός γραμμών και στηλών ενός πίνακα nxm.

Παραδείγματα:

```
» zeros(3)
ans =
    0    0    0
    0    0    0
    0    0    0
» I=eye(3)
I =
    1    0    0
    0    1    0
    0    0    1

 » eye(3,4)
   ans =
       1    0    0    0
       0    1    0    0
       0    0    1    0
» ones(2,4)
ans =
    1    1    1    1
```

```
    1     1     1     1

» ones(size(A))
ans =

    1     1     1
    1     1     1
    1     1     1

» ones(size([A B]))
ans =
    1     1     1     1     1     1
    1     1     1     1     1     1
    1     1     1     1     1     1

» randn(2,3)
ans =
   1.1650    0.0751   -0.6965
   0.6268    0.3516    1.6961
```

Χρήσιμες συναρτήσεις

Μερικές πολύ χρήσιμες συναρτήσεις που απλουστεύουν το χειρισμό πινάκων είναι:

- ***size(A)***: επιστρέφει το μέγεθος του πίνακα (αριθμός γραμμών και αριθμός στηλών).
- ***length(v)*** ή ***length(A):*** επιστρέφει τον αριθμό των στοιχείων ενός διανύσματος v, ή την μεγαλύτερη από τις διαστάσεις ενός πίνακα A mxn.
- ***flipud(A)***: επιστρέφει το συμμετρικό ενός πίνακα ως προς τις γραμμές.
- ***fliplr(A)***: επιστρέφει το συμμετρικό ενός πίνακα ως προς τις στήλες.
- ***rot90(A)***: *περιστρέφει ένα πίνακα κατά 90 μοίρες.*
- ***reshape(A, m ,n)***: μετασχηματίζει ένα πίνακα A ixj σε ένα πίνακα διαστάσεων mxn=ixj.
- ***diag(v)*** και ***diag(v)***: κατασκευάζει διαγώνιο πίνακα με κύρια διαγώνιο ίση με το διάνυσμα v. Αν αντί του v δοθεί σαν όρισμα πίνακας A διαστάσεων mxn, εξάγει την κύρια διαγώνιό του. Με κατάλληλη παραμετροποίηση μπορεί να κατασκευάσει ή να εξάγει και δευτερεύουσες διαγώνιους (βλ. Παράδειγμα και on-line βοήθεια).
- ***trace(A)***: Επιστρέφει το ίχνος (άθροισμα διαγώνιων στοιχείων) ενός πίνακα δύο διαστάσεων.
- ***triu(A)*** : Άνω τριγωνικός πίνακας που αντιστοιχεί στον πίνακα A.
- ***sum(v), sum(A)*** : Για ένα διάνυσμα v υπολογίζει το άθροισμα των στοιχείων του, ενώ για έναν πίνακα δύο διατάσεων A, επιστρέφει ένα διάνυσμα-γραμμή με τα αθροίσματα των στηλών του.

Παραδείγματα

```
» A=[0 2 1;1 -2 4;4 -1 2];
» flipud(A)
  ans =
    4    -1     2
    1    -2     4
    0     2     1

  » fliplr(A)
  ans =
      1     2     0
      4    -2     1
      2    -1     4
```

```
» rot90(A)
  ans =
     1    4    2
     2   -2   -1
     0    1    4

» [A B]
ans =
    0    2    1    1    1    1
    1   -2    4    2    3   -1
    4   -1    2    4   -2    6

» U=reshape([A B],2,9)                {αναμόρφωση του A∪B σε πίνακα 2x9}

U =

    0    4   -2    1    2    2    1   -2   -1
    1    2   -1    4    1    4    3    1    6

» v=[3 15 -10];
» diag(v)
  ans =
     3    0    0
     0   15    0
     0    0  -10

  » diag(A)
  ans =
     1
    -2
     4

» diag(-2:2) + diag(ones(2*2,1),1) + diag(ones(2*2,1),-1)        {κατασκευή τρισδιαγώνιου πίνακα 5x5}
ans =
   -2    1    0    0    0
    1   -1    1    0    0
    0    1    0    1    0
    0    0    1    1    1
    0    0    0    1    2
```

B.1.5 Συμβολικές Εκφράσεις

Αναπαράσταση και Πράξεις

Το "Symbolic Toolbox" είναι μια συλλογή εργαλείων που επιτρέπει το χειρισμό λειτουργιών και πράξεων πάνω σε *συμβολικές εκφράσεις*, χωρίς να απαιτείται ο προκαθορισμός των μεταβλητών. Μια αλγεβρική έκφραση (ή συνάρτηση) μπορεί να ορισθεί με τη συνάρτηση ***sym*** και να πάρει όνομα μέ μια εντολή εκχώρηση:

sym*('έκφραση'),*
*f=****sym****('έκφραση'),* ή: *f = 'έκφραση'*

Η *έκφραση* αναπαρίσταται σαν σειρά χαρακτήρων και όχι σαν αριθμητική τιμή και συμβολίζει μια μαθηματική έκφραση ή πίνακα που περιλαμβάνει σύμβολα (ή μεταβλητές).

Οι εντολές του "Symbolic Toolbox" διαχειρίζονται τις εκφράσεις αυτές και εξάγουν αποτελέσματα σε συμβολική μορφή. Όλες οι γνωστές αλγεβρικές πράξεις μεταξύ συμβολικών εκφράσεων υποστηρίζονται

στο "Symbolic Toolbox" και μάλιστα κατά ανάλογο τρόπο όπως και στους πίνακες. Στις πράξεις αυτές μπορούν συμμετάσχουν και αριθμητικές εκφράσεις ή πίνακες, αλλά το αποτέλεσμα πάντα δίνεται σε συμβολική μορφή. Αν e1 και e2 είναι συμβολικές ή αριθμητικές εκφράσεις ή και σταθερές, τότε

symadd(e1, e2) : Άθροισμα e1+e2
symsub(e1,e2) : Διαφορά e1-e2
symmul(e1,e2) : Γινόμενο e1*e2
symdiv(e1,e2) : Πηλίκο e1/e
sympow(e1,e2) : Δύναμη $e1^{e2}$

Παραδείγματα

```
» f1='(x^2-x+1)/(x+1)'          {ορισμός δύο συμβολικών συναρτήσεων}
f1 =
(x^2-x+1)/(x+1)
» f2='x-1/(x+1)';

»  f=symadd(f1,f2)
f =
 (x^2-x+1)/(x+1)+x-1/(x+1)

» symmul(f1,'sin(x)')
ans =
 (x^2-x+1)/(x+1)*sin(x)

» symdiv('x^3-3*x+2','y*x-y')
ans =
 (x^3-3*x+2)/(y*x-y)

»  f=symadd(4,f2)
f =
4+x-1/(x+1)

»  f=sympow(f1,f2)
f =
 ((x^2-x+1)/(x+1))^(x-1/(x+1))
```

Συμβολικές Μεταβλητές

Μια συμβολική έκφραση μπορεί φυσικά να έχει πολλές μεταβλητές, αλλά στο Matlab *μόνον μία* νοείται σαν *ανεξάρτητη μεταβλητή*. Ως προς αυτήν μπορούν στη συνέχεια να εφαρμοσθούν σημαντικές λειτουργίες, όπως παραγώγιση και ολοκλήρωση. Υπάρχουν συγκεκριμένοι κανόνες για τον τρόπο επιλογής της ανεξάρτητης μεταβλητής. Αν η έκφραση περιέχει ακριβώς ένα μικρό γράμμα διάφορο του i και j που δεν αποτελεί μέρος κάποιας λέξης, τότε αυτό ορίζεται σαν η ανεξάρτητη μεταβλητή. Αν δεν υπάρχει τέτοιο, επιλέγεται το x. Αν υπάρχουν περισσότερα, τότε επιλέγεται το πλησιέστερο αλφαβητικά προς το x. Η εντολή symvar εμφανίζει ή καθορίζει την ανεξάρτητη μεταβλητή μιάς συμβολικής έκφρασης:

- ***symvar(expr)*** εμφανίζει ποια είναι η θεωρούμενη από το Matlab ανεξάρτητη μεταβλητή.
- ***symvar(expr,'v')*** την θέτει ίση με v. Αν δεν υπάρχει τέτοια μεταβλητή επιλέγεται η πλησιέστερη προς την v.

Παραδείγματα

```
» symvar('2*c*x+y^2')
```

```
ans=
     x
» symvar('s/3*t+u')                          {το u  είναι το πλησιέστερο προς το x}
ans=
    u
» symvar('st/lamda+2*pi')            {δεν υπάρχει μεταβλητή με ένα γράμμα}
ans=
   x
» symvar('2*c*x+y^2','y')                    {σαν ελεύθερη μεταβλητή ορίζεται η y}
ans=
   y
```

Χρήσιμες Συναρτήσεις

Ένας μεγάλος αριθμός εντολών υποστηρίζουν το χειρισμό των συμβολικών εκφράσεων, εφαρμόζοντας αλγεβρικές πράξεις για την ανάλυση, απλοποίηση και βελτιστοποίηση της μορφής τους. Οι κυριότερες είναι οι εξής:

- ***isstr(f)***: Επιστρέφει 1 αν το f αναπαριστά συμβολική έκφραση (σειρά χαρακτήρων) και 0 διαφορετικά.
- ***numeric(s)***: Μετατροπή συμβολικής παράστασης s (που δεν περιέχει σύμβολα) σε αριθμητική. Η s μπορεί να είναι και συμβολικός πίνακας.
- ***simple(expr)***: Απλοποίηση συμβολικής έκφρασης με μετατροπή της σε μια άλλες απλούστερη κάνοντας παραγοντοποίηση, ανάπτυξη, απλοποίηση, χρήση τριγωνομετρικών και άλλων μετασχηματισμών κ.λ.π. Επιστρέφεται η συντομότερη στο μήκος έκφραση.
- ***pretty(f)***: Μετατροπή συμβολικής έκφρασης f σε ευανάγνωστη μορφή.
- ***simplify(f)***: Συμβολική απλοποίηση μιας συμβολικής έκφρασης ή συμβολικού πίνακα.
- ***expand(f)***: Ανάπτυξη μιας συμβολικής έκφρασης ή των στοιχείων συμβολικού πίνακα, με εκτέλεση πράξεων και χρήση ταυτοτήτων και μετασχηματισμών.
- ***factor(f)***: Παραγοντοποίηση της συμβολικής έκφρασης f, ή ενός πίνακα ακεραίων (σε πρώτους παράγοντες), ή των στοιχείων ενός συμβολικού πίνακα.
- ***subc(f, exp, x)***: Αντικατάσταση σε μια συμβολική έκφραση ή πίνακα f μιας συμβολικής μεταβλητής x με την συμβολική έκφραση expr. Η **subc(f,expr)** αντικαθιστά την εξ' ορισμού συμβολική μεταβλητή της f με την έκφραση expr. Αν η f περιέχει μια μόνον μεταβλητή και στη θέση expr δοθεί μια αριθμητική τιμή v, τότε υπολογίζεται η τιμή f(v).
- ***[n, d] = numden(f)***: Εξαγωγή αριθμητή n και παρονομαστή d ενός ρητού πολυωνυμικού κλάσματος.
- ***finnverse(f)*** : Επιστρέφει την αντίστροφο συνάρτηση της συνάρτησης μιας μεταβλητής f.

Παραδείγματα

```
» pretty(symadd('(x^2-x+1)/(x+1)', 'x-1/(x+1)'))    {ευανάγνωστη μορφή έκφρασης}
                    2
                   x  - x + 1        1
                   ---------- + x - -----
                     x + 1         x + 1
»g1='x^3-3*x+2','y*x-y';                    {ορισμός δύο συμβολικών συναρτήσεων}
»g2='y*x-y';
»g=symdiv(g1,g2)

» simplify(g)                               {απλοποίηση της g}
ans =
(x^2+x-2)/y

» simple(g)                                 {απλούστερη μορφή της g μετά από πράξεις}
simplify:
```

```
(x^2+x-2)/y
combine(trig):
1/(y*x-y)*x^3-3/(y*x-y)*x+2/(y*x-y)
factor:
(x+2)*(x-1)/y
expand:
1/(y*x-y)*x^3-3/(y*x-y)*x+2/(y*x-y)
ans =
(x^2+x-2)/y

» factor(g)                          {παραγοντοποίηση της g}
ans =
(x+2)*(x-1)/y

» factor(10:14);                     {επιστρέφει (2)*(5), (11), (2)^2*(3), (2)*(7)}

» subs('sin(x)',pi/3)                {συμβολική αντικατάσταση}
ans =
sin(1/3*pi)

» f = 'F(a*r^2)'                     {ορισμός συμβολικής συνάρτησης F}
f =
F(a*r^2)

» subs(f,  'sqrt(x^2+y^2)' ,'r')     {αντικατάσταση της μεταβλητής r με μια έκφραση}
ans =
F(a*(x^2+y^2))

» subs('x^2+y^2', 2, 'y')            {αντικατάσταση του y με 2}
ans =
x^2+4

» subs('x^2+y^2', 2)                 {αντικατάσταση της εξ ορισμού ελεύθερης μεταβλητής  x με 2}
ans =
4+y^2

» expand('cos(x+y)');                {επιστρέφει cos(x)*cos(y)-sin(x)*sin(y)}
» expand('(x+y)^4');                 {επιστρέφει ανάπτυγμα του (x+y)^4)}

» fininverse('1/tan(x)');              {επιστρέφει arctan(1/x)}
```

Ολοκλήρωση και Παραγώγιση

Τα εργαλεία συμβολικού υπολογισμού του Matlab υποστηρίζουν την παραγώγιση και ολοκλήρωση. συναρτήσεων. Χρησιμοποιούνται για το σκοπό αυτό οι συναρτήσεις ***diff*** και ***int*** αντίστοιχα.

- ***diff(f,'v',n):*** **Παράγωγος συνάρτησης**. F είναι η συμβολική αναπαράσταση της συνάρτησης f(x), v η μεταβλητή ως προς την οποία γίνεται η παραγώγιση και n η τάξη της παραγώγου. Οι επιλογές 'v' και n μπορούν να παραλειφθούν, οπότε εννοείται η πρώτη παράγωγος ως προς την προκαθορισμένη ελεύθερη μεταβλητή (με την συνάρτηση symvar).
- **int(f,'v',a,b): Αόριστο ολοκλήρωμα συνάρτησης**. f είναι η συμβολική αναπαράσταση της συνάρτησης f(x), v η μεταβλητή ως προς την οποία γίνεται η ολοκλήρωση. Η επιλογή 'v' μπορεί να παραλειφθεί.
- **int(f,'v',a,b): Ορισμένο ολοκλήρωμα συνάρτησης**. Τα a,b είναι τα όρια της ολοκλήρωσης

Παραδείγματα

```
» f='x/(x^2+1)'                      {δημιουργία μιας συμβολικής συνάρτησης}
f =
x/(x^2+1)
```

```
» diff(f,'x')                          {παραγώγιση ση ως προς x}
ans =
1/(x^2+1)-2*x^2/(x^2+1)^2

» int(f, 'x')                          {ολοκλήρωση ως προς x}
ans =
1/2*log(x^2+1)

» int(f, 'x',-1, 2)                    {ολοκλήρωση ως προς x από -1 έως 2}
ans =1/2*log(5)-1/2*log(2)
```

Συμβολική Επίλυση αλγεβρικών εξισώσεων

Στο Matlab είναι δυνατή η επίλυση συμβολικών εξισώσεων ως προς μια μεταβλητή και συστημάτων συμβολικών εξισώσεων. Για το σκοπό αυτό παρέχεται η συνάρτηση ***solve(x)*** σε διαφορετικές μορφές.

- **Μια εξίσωση με ένα άγνωστο:**
  - solve(S) : όπου S συμβολική εξίσωση, ή συμβολική έκφραση, επιλύει τη δοθείσα εξίσωση ή την S=0, ως προς την ελεύθερη μεταβλητή που ορίσθηκε από την *symvar*.
  - solve(S,'v') : επιλύει τη δοθείσα εξίσωση ως προς τη μεταβλητή v.

- **Συστήματα εξισώσεων:**
  - solve(S1,S2,..,Sn) επιλύει n συμβολικές εξισώσεις ως προς n αγνώστους που ορίσθηκαν από την *symvar*.
  - solve(S1,S2,..,Sn,'v1,v2,..,vn') επιλύει n συμβολικές εξισώσεις ως προς τους n αγνώστους v1,v2,..,vn.
  - Οι κλήσεις **[X1,X2,..,XN]= solve(S1,S2,..,Sn)** και **[X1,X2,..,XN] = solve(S1,S2,..,Sn,'v1,v2,..,vn')** επιστρέφουν n συμβολικά διανύσματα-λύσεις που αντιστοιχούν στις δοθείσες μεταβλητές.

Αν δεν μπορεί να βρεθεί συμβολική αναλυτική λύση, τότε υπολογίζονται οι λύσεις αριθμητικά.

Σε πολλές περιπτώσεις, στην έκφραση των λύσεων περιέχεται το string *RootOf(expr)*. Η έκφραση αυτή απλοποιεί την μορφή των λύσεων και αναφέρεται σε όλες τις ρίζες της έκφρασης expr. Για τον υπολογισμό όλων των λύσεων x που περιέχουν την υπο-έκφραση 'RootOf' χρησιμοποιείται η συνάρτηση **allvalues(x)**, η οποία βρίσκει όλες τις ρίζες της expr, και υπολογίζει την x για κάθε ρίζα.

Παραδείγματα

```
» x = solve('x^3  +x =10')
x =
[     2]
[-1+2*i]
[-1-2*i]

» t=solve('tan(2*x)=cos(x)')
t =                                    {εύρεση λύσεων σε συμβολική μορφή}
[              1/2*pi]
[asin(-1/2-1/2*3^(1/2))]
[asin(-1/2+1/2*3^(1/2))]
» numeric(t)                           {εύρεση αριθμητικών λύσεων}
ans =
   1.5708
  -1.5708 + 0.8314i
   0.3747

» f='x*sin(x)-1'                       {αριθμητική επίλυση εξίσωσης}
f =
x*sin(x)-1
» solve(f,')
```

```
Warning: No analytic solution found.
       Returning numeric solution.
ans =
2.772604708265991

» [u,v] = solve('a*u^2 + v^2', 'u - v = 1', 'u,v')     {επίλυση μη γραμμικού συστήματος}
u =
RootOf((a+1)*_Z^2+2*a*_Z+a)+1                          {βοηθητική συμβολική έκφραση ρίζας της συνάρτησης}
v =                                                    {(a+1)x^2+2az+a }
RootOf((a+1)*_Z^2+2*a*_Z+a)
» allvalues(u)                                         {υπολογίζονται όλες οι λύσεις u για κάθε ρίζα «_Z»}
ans =
[1/2/(a+1)*(-2*a+2*(-a)^(1/2))+1]
[1/2/(a+1)*(-2*a-2*(-a)^(1/2))+1]
» allvalues(v)                                         {υπολογίζονται όλες οι λύσεις v για κάθε ρίζα «_Z»}
ans =
[1/2/(a+1)*(-2*a+2*(-a)^(1/2))]
[1/2/(a+1)*(-2*a-2*(-a)^(1/2))]
```

B.1.6 Συμβολικοί Πίνακες

Αναπαράσταση και προσπέλαση συμβολικών πινάκων

Συμβολικοί πίνακες είναι πίνακες με στοιχεία συμβολικές εκφράσεις. Το "Symbolic Toolbox" χειρίζεται τους πίνακες αυτούς. Ένας συμβολικός πίνακας είναι ένας πίνακας κειμένου στον οποίο κάθε γραμμή αρχίζει με '[' τελειώνει με ']' και περιέχει σειρές χαρακτήρων χωρισμένες με ',' οι οποίες και αναπαριστούν τα διακεκριμένα στοιχεία. Για την αναπαράσταση, κατασκευή και προσπέλαση των συμβολικών πινάκων χρησιμοποιείται η εντολή ***sym***. Υπάρχουν τρεις τρόποι κατασκευής συμβολικών πινάκων:

- ***sym*(X)** : Μετατρέπει ένα αριθμητικό πίνακα X στην συμβολική του μορφή.

- ***sym*(m,n,'expr')**: Δημιουργεί ένα mxn συμβολικό πίνακα του οποίου κάθε στοιχείο είναι η έκφραση expr που παίρνει τιμές για i = 1:m και j = 1:n. Η *expr* είναι μια συμβολική έκφραση που συνήθως περιέχει τους χαρακτήρες 'i' και 'j', και ενδεχομένως και άλλες ελεύθερες μεταβλητές.
  Η εντολή **SYM(m,n,'r','c','expr')** έχει το ίδιο αποτέλεσμα, με τη διαφορά ότι οι γραμμές και οι στήλες συμβολίζονται με 'r' και 'c' αντίστοιχα.

- ***sym*('[s11,s12,...,s1n; s21,s22,...; ...,smn]')**: Δημιουργεί ένα mxn συμβολικό πίνακα με χρήση των συμβολικών στοιχείων s11, s12, ..., smn.

Υπάρχουν επίσης δύο τρόποι προσπέλασης των στοιχείων συμβολικού πίνακα:

- ***sym*(S,i,j,'expr')**: Δεν αλλάζει τον S, αλλά θέτει ans=S με ans(i,j) = 'expr'. Έτσι το ισοδύναμο της S(i,j)=expr είναι **S= SYM(S,i,j,'expr')**.

- **r = *sym*(S,i,j)** που ισοδυναμεί με r = S(i,j).

Για να βρούμε το μέγεθος ενός συμβολικού πίνακα A χρησιμοποιούμε τη συνάρτηση ***symsize(A)***.

Παραδείγματα:

```
» A= [0,2,1; 1,-2,4; 4,-1,2];          {ένας αριθμητικός πίνακας}
» SA=sym(A)              {μετατροπή αριθμητικού πίνακα A στον συμβολικό πίνακα σταθερών SA}
SA =
[0,  2, 1]
[1, -2, 4]
[4, -1, 2]
```

```
» S=sym('[a,b,c;c,d,e;e,d,c]')
S =
[a,b,c]
[c,d,e]
[e,d,c]

» elmt=sym(A,1,2)          {προσπέλαση του στοιχείου (1,2) του A}
elmt =
    2

» S=Sym(S,1,3,'0')         {αλλαγή του στοιχείου S(1,2) σε 0}
S =
[a,b,0]
[c,d,e]
[e,d,c]

» M = sym(hilb(3))         {παράγει τον πίνακα Hilbert με M(i,j)=1/(i+j-1), ο οποίος έχει κακή
κατάσταση}
M =
[  1, 1/2, 1/3]
[1/2, 1/3, 1/4]
[1/3, 1/4, 1/5]

M = sym(3,3,'1/(i+j-t)')   {παραγωγή πίνακα M με M(i,j)= 1/(i+j-t)}
[1/(2-t), 1/(3-t), 1/(4-t)]
[1/(3-t), 1/(4-t), 1/(5-t)]
[1/(4-t), 1/(5-t), 1/(6-t)]

M = sym(M,1,3,'1/t')       {αλλάξει το στοιχείο M(1,3) σε '1/t' }

» symsize(M)
ans =
   3    3

» [n,m]=symsize(M);        {θέτει n=3, m=3}
```

Πράξεις με Συμβολικούς Πίνακες

Οι βασικές αλγεβρικές πράξεις υποστηρίζονται και στους συμβολικούς πίνακες. Οι πράξεις αυτές λειτουργούν είτε μεταξύ πινάκων είτε μεταξύ πίνακα και στοιχείου. Επίσης ορίζεται και η δύναμη πίνακα. Αν A, B είναι συμβολικοί πίνακες και c συμβολικό στοιχείο, οι πράξεις ορίζονται:

symadd(A,'c') : $A+c$, συμβολική πρόσθεση
symadd(A,B) : $A+B$
symsub(A,'c') : $A-c$, συμβολική αφαίρεση
symsub(A,B) : $A-B$
symmul(A,'c') : $A*$, , συμβολικός πολ/σμός
symmul(A,B)) : $A*B$
symdiv(A,B) : AB^{-1}, συμβολική διαίρεση
symdiv(A,'c') : A/c
symdiv('c',B) : $c*B^{-1}$
sympow(A,n) : A^n (n = ακέραιος), ύψωση σε δύναμη

Μερικές χρήσιμες συναρτήσεις πάνω σε συμβολικούς τετραγωνικούς πίνακες είναι:

- ***transpose(S)***: Επιστρέφει τον ανάστροφο πίνακα.
- ***determ(S)*** : Επιστρέφει την ορίζουσα ενός πίνακα σαν συμβολική έκφραση. Αν ο πίνακας αποτελείται από συμβολικές σταθερές, την υπολογίζει.
- ***inverse(S)***: Επιστρέφει τον αντίστροφο πίνακα.

Παραδείγματα

```
»S=sym('[a,b,c;c,d,e;e,d,c]');
» symadd(S,'2')                                  {ans=S+2}
ans =
[a+2, b+2, c+2]
[c+2, d+2, e+2]
[e+2, d+2, c+2]

» symsub(S,'a')                                  {ans=S-a}
ans =
[  0, b-a, c-a]
[c-a, d-a, e-a]
[e-a, d-a, c-a]

» symmul(S,'(a-b)/b')                            {ans=S*(a-b)/b}
ans =
[(a-b)/b*a,       a-b, (a-b)/b*c]
[(a-b)/b*c, (a-b)/b*d, (a-b)/b*e]
[(a-b)/b*e, (a-b)/b*d, (a-b)/b*c]

» symdiv(S,'c')                                  {ans=S/c}
ans =
[1/c*a, 1/c*b,     1]
[    1, 1/c*d, 1/c*e]
[1/c*e, 1/c*d,     1]

» SA=symmul(SA,S)                                {SA=SA*S}
SA =
[  2*c+e,   3*d,    2*e+c]
[a-2*c+4*e, b+2*d, -2*e+4*c]
[4*a-c+2*e, 4*b+d,   -e+2*c]

» M1=symsub(sym(3,3,'1/(i+j-t)'),eye([3]))   {M1=M-I, M=γενικευμένος πίνακας Hilbert}
M1 =
[1/(2-t)-1,   1/(3-t),   1/(4-t)]
[  1/(3-t), 1/(4-t)-1,   1/(5-t)]
[  1/(4-t),   1/(5-t), 1/(6-t)-1]

» M2 = symadd(sym(3,3,'1/(i+j-t)'),ones(3))  {M2=M+1}
M2 =
[1/(2-t)+1, 1/(3-t)+1, 1/(4-t)+1]
[1/(3-t)+1, 1/(4-t)+1, 1/(5-t)+1]
[1/(4-t)+1, 1/(5-t)+1, 1/(6-t)+1]

» M3 = symmul(sym(3,3,'1/(i+j-t)'),sym('[0;-t;1]'))      {M3=M*(0;-t;1)^t }
M3=
[(-5*t+t^2+3)/(-3+t)/(-4+t)]
[(-6*t+t^2+4)/(-5+t)/(-4+t)]
[(-7*t+t^2+5)/(-6+t)/(-5+t)]

» x=symmul(inverse(sym(3,3,'1/(i+j-t)')),transpose(sym('[0,-t,1]')))
x =                                              {λύση συστήματος Mx=(0,-t,1)^t, x=M^-1(0,-t,1)^t }
[   -21*t-223*t^2+721/4*t^3-117/2*t^4+35/4*t^5-1/2*t^6+180]
[       -36*t+776*t^2-1041/2*t^3+147*t^4-39/2*t^5+t^6-720]
[80*t-1185/2*t^2+1407/4*t^3-179/2*t^4+43/4*t^5-1/2*t^6+600]

» determ(sym(3,3,'1/(i+j-t)'))                   {ορίζουσα πίνακα Hilbert}
ans =
-4/(-2+t)/(-4+t)^3/(-6+t)/(-5+t)^2/(-3+t)^2

» G=sym('[cos(t),sin(t);-sin(t),cos(t)]')        {συμβολικός πίνακας με συναρτήσεις}
G =
[ cos(t),sin(t)]
[-sin(t),cos(t)]
```

```
» G1=symadd(G,'t')                                  {ans=G+t}
G1 =
[ cos(t)+t, sin(t)+t]
[-sin(t)+t, cos(t)+t]

» inverse(G)                                        {ans=G^-1}
ans =
[cos(t)/(cos(t)^2+sin(t)^2), -sin(t)/(cos(t)^2+sin(t)^2)]
[sin(t)/(cos(t)^2+sin(t)^2),  cos(t)/(cos(t)^2+sin(t)^2)]

» symdiv(sym('[1,-1]'),G)                           {ans=(1,-1) G^-1}
ans =
[-(sin(t)-cos(t))/(cos(t)^2+sin(t)^2), -(cos(t)+sin(t))/(cos(t)^2+sin(t)^2)]

» G2=sympow(G,2)                                    {G2=G*G}
G2 =
[cos(t)^2-sin(t)^2,   2*cos(t)*sin(t)]
[ -2*cos(t)*sin(t), cos(t)^2-sin(t)^2]
```

Υπάρχει επίσης η δυνατότητα της επίλυσης συστημάτων συμβολικά, με τη χρήση της συνάρτησης ***linsolve(A,b)*** , όπου Α ο πίνακας των συντελεστών των αγνώστων και b το σταθερό διάνυσμα.

Παραδείγματα

```
 % Παράδειγμα 1: επίλυση συστήματος συμβολικών σταθερών
» A = sym('[3 1 -1;-2 -6 3;4 -2 8]');              { Άσκηση ΙΙΙ.5.1 }
» b=sym('[0;2;1]');
» linsolve(A,b)
ans =
 [ 5/42]
[-7/18]
[-2/63]

%Παράδειγμα 2: επίλυση του συστήματος x  -  y -kz  = 1, kx +  y +kz =  1-k,  kx +3y +3z = -1
» A=sym('[1 -1 -k;k 1 k;k 3 3]');
» b=sym('[1;1-k;-1]')
b =
[  1]
[1-k]
[ -1]
» x=linsolve(A,b)
x =
[        -(k-2)/(k+1)]
[1/3*(k^2-2*k+3)/(k+1)]
[     -1/3*(k+4)/(k+1)]
```

Β.1.7 Πολυώνυμα

Τα πολυώνυμα αναπαρίστανται σαν διανύσματα-γραμμές που περιέχουν τους συντελεστές κατά φθίνουσα διάταξη. Οι μηδενικοί όροι πρέπει προφανώς να λαμβάνονται υπ' όψη, γράφοντας 0 στις αντίστοιχες θέσεις. Π.χ. το $p(x)=2x^3 + 5x - 6$ δίνεται σαν p=[2 0 5 -6]. Οι πράξεις των πολυωνύμων υλοποιούνται ως εξής:

- **Πολλαπλασιασμός** : συνάρτηση **conv(p,q)**

- **Άθροισμα και διαφορά**: υποστηρίζεται από τις αντίστοιχες πράξεις μεταξύ πινάκων-γραμμών. Όταν τα πολυώνυμα δεν είναι του ίδιου βαθμού, θα πρέπει το πολυώνυμο του μικρότερου βαθμού να συμπληρώνεται στην αρχή με τον κατάλληλο αριθμό μηδενικών.
- **Διαίρεση**: συνάρτηση **[q,r]=deconv(a,b)**, όπου q είναι το πηλίκο και r το υπόλοιπο της διαίρεσης.

Βασικές λειτουργίες πάνω στα πολυώνυμα υποστηρίζονται από συναρτήσεις:

- Εύρεση ριζών: **r=roots(p)**, όπου r είναι ένα διάνυσμα-στήλη το οποίο και επιστρέφει τις ρίζες του πολυωνύμου p.
- Κατασκευή πολυωνύμου με δοθείσες ρίζες: **pp=poly(r)**, όπου r είναι ένα διάνυσμα-στήλη που περιέχει τις ρίζες του ζητούμενου πολυωνύμου pp.
- Παράγωγος πολυωνύμου: **h=polyder(p)**.
- Υπολογισμός τιμής p(x) πολυωνύμου: **polyval(p,x)**.

Παραδείγματα

```
» p=[2 0 5 -6];
» q=[0 2 -1 3];
» p+q
ans =
    2     2     4    -3

» conv(p,q)
ans =
    0     4    -2    16   -17    21   -18

» [quotient,rem]=deconv(p,q)
quotient =
   1.0000    0.5000
rem=
        0         0    2.5000   -7.5000

» r=roots(p)
r =
  -0.4521 + 1.7644i
  -0.4521 - 1.7644i
   0.9042

» h=polyder(p)
h =
   3.0000    0.0000    2.5000
» p=poly(r)
p =
   1.0000    0.0000    2.5000   -3.0000

» x=linspace(1,5);                    {δημιουργεί 100 σημεία μεταξύ 1 και 5}
» v=polyval(p,x);          {υπολογίζει το p στις τιμές του  x και αποθηκεύει το αποτέλεσμα στο v}
```

Β.1.8 Στοιχειώδεις εντολές της ενσωματωμένης γλώσσας προγραμματισμού

Όπως αναφέρθηκε, το Matlab παρέχει δυνατότητες προγραμματισμού, με τη βοήθεια μίας "Pascal-like" γλώσσας. Τα δημιουργούμενα προγράμματα αποθηκεύονται με την κατάληξη *.m (M-files) και δίνουν τη δυνατότητα στο χρήστη να φτιάξει τις δικές του συναρτήσεις, συχνά βασιζόμενος στις ήδη υπάρχουσες. Οι δομές ελέγχου που υποστηρίζονται στην γλώσσα αυτή είναι εν συντομία οι εξής:

- ***Εντολή if***. Ακολουθεί τη σύνταξη:

```
if  <έκφραση>
  <εντολή> ;
      :
  <εντολή> ;
end
```

Αν η <έκφραση> αληθεύει τότε εκτελούνται οι εντολές μέχρι το end. Η εντολή if μπορεί να πάρει και τη μορφή:

```
if  <έκφραση>
   <εντολή> ;
      :
   <εντολή> ;
else
  <εντολή> ;
     :
  <εντολή> ;
 end
```

Οι εντολές μεταξύ ***else*** και ***end*** εκτελούνται όταν η <έκφραση> είναι ψευδής. Η <έκφραση> είναι της μορφής:

<μεταβλητή ή έκφραση ή αριθμός> <λογικός τελεστής> <μεταβλητή ή έκφραση ή αριθμός>

Για την κατασκευή σύνθετων δομών if χρησιμοποιείται η εντολή ***elseif***, η οποία δεν απαιτεί στο τέλος της end.

```
if  <έκφραση1>
     <εντολές>
elseif <έκφραση2>
     <εντολές>
elseif <έκφραση3>
   <εντολές>
   . . . . . .
else
   <εντολές>
end
```

Παράδειγμα:

```
if  ( a[i , j] >= x )
  counter = counter + 1;
else
  if  ( a[i , j] > y )
    k = k + 1;
  else
    b[k] = a[i , j];
  end
 end
```

- ***Εντολή for***. Ακολουθεί τη σύνταξη:

```
for  <μεταβλητή> = <πεδίο τιμών>
  <εντολή> ;
     :
  <εντολή> ;
 end
```

όπου το <πεδίο τιμών> έχει τη μορφή: <αρχική τιμή> **:** <βήμα> **:** <τελική τιμή>. Οι εντολές του σώματος εκτελούνται για κάθε τιμή της μεταβλητής, η οποία κάθε φορά αυξάνει κατά <βήμα>,μέχρι αυτή να πάρει σαν τιμή και την <τελική τιμή>.

Παράδειγμα:

```
for  i = 1 : n
  for  j = 1 : k
     a[i , j] = b[i , j] ;
     c[i , j] = a[i , j] .* b[i , j];
   end
end
```

- ***Εντολή while***. Ακολουθεί τη σύνταξη:

```
while  <έκφραση>
 <εντολή>
     :
 <εντολή> ;
end
```

όπου η <έκφραση> έχει την ίδια μορφή με την <έκφραση> της εντολής **if**. Όσο η <έκφραση> είναι αληθής εκτελούνται οι εντολές του σώματος της **while.**

Παράδειγμα:

```
while  ( done == 0 ) & ( i ~= j)
     b[i , k] = a[i ,j];
     i = i + 1;
     k = k + 1;
     if  ( k > size)
       done =1;
     end
 end
```

Μια αποδοτική προγραμματιστική τακτική στο Matlab είναι η χρήση εντολών που χρησιμοποιούν διανύσματα αντί των συμβατικών ανακυκλώσεων while και for. Η προσέγγιση αυτή είναι ταχύτερη, αν και απαιτεί μεγαλύτερη μνήμη (βλ. παραδείγματα προγραμμάτων στο β' μέρος).

- ***Εντολή break***. Τερματίζει την εκτέλεση ενός βρόγχου. Στην περίπτωση που ο βρόγχος που περικλείει την εντολή **break** βρίσκεται στο εσωτερικό ενός ή περισσότερων βρόγχων, τότε τερματίζεται η εκτέλεση του πιο εσωτερικού.

Β.1.8 Σχεδίαση γραφικών παραστάσεων

Υποστηρίζεται ένα πλούσιο σύνολο συναρτήσεων για σχεδίαση γραφημάτων συναρτήσεων σε 2 και 3 διαστάσεις. Δίνονται παρακάτω οι αντίστοιχες εντολές.

- **Εντολή PLOT**

plot (x_1 , y_1 , $options_1$, x_2 , y_2 , $options_2$ ....,x_n , y_n , $options_n$**)** [**,grid**]

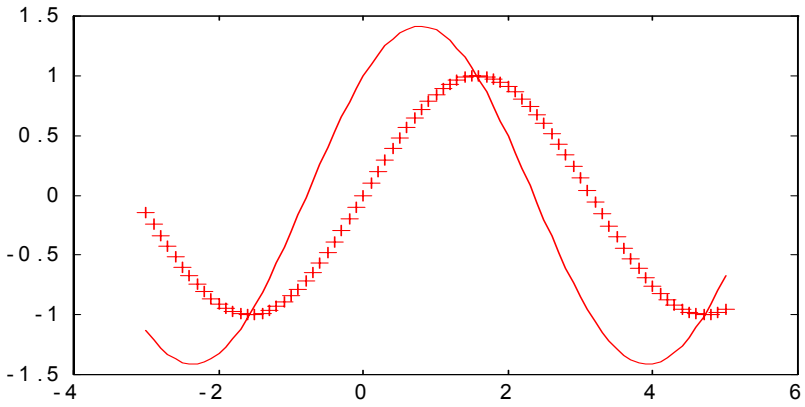
Η **plot** σχεδιάζει τη γραφική παράσταση μιας ή περισσότερων συναρτήσεων με τη βοήθεια πινάκων στους οποίους αποθηκεύουμε διακριτές τιμές. Στην μορφή αυτή της **plot** σχεδιάζουμε το y συναρτήσει του x_i i=1..n, πάνω στους ίδιους άξονες. Το y_i πρέπει να είναι διάνυσμα ή πίνακας της ίδιας διάστασης με το x_i (οι y_i , x_i μπορεί να είναι απλές μεταβλητές, οπότε η γραφική παράσταση θα είναι ένα σημείο).

Με τις επιλογές *options* καθορίζουμε το χρώμα και το σύμβολο σχεδίασης της γραφικής παράστασης. Η μορφή των επιλογών είναι: **'**cs**'** , όπου c το πρώτο γράμμα του χρώματος, π.χ. r για το red, g για το green, b για το blue, k για το black κ.τ.λ., και s για το σύμβολο σχεδίασης, π.χ. 'o', '+', 'x', '*', '-' κ.λ.π. Προαιρετικά μπορούμε να προσθέσουμε την εντολή **grid**.

Παράδειγμα:

```
»x = -3:0.1:5;          { Ορισμός του x σε μορφή διαστήματος. Το 0.1 είναι το βήμα }
```

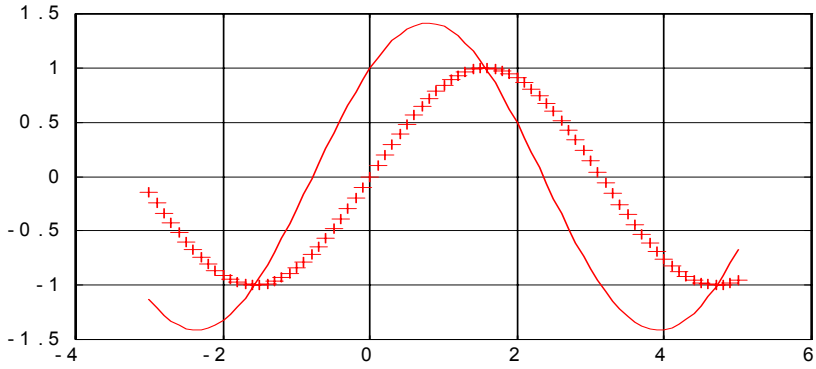
```
»y = sin(x);
»u = cos(x)+y;
»plot(x, y, ‘r+’, x , u ,'r-‘)
```



Προσθέτοντας την εντολή **grid** :

»plot(x, y, 'r+', x, u,' r-'),grid

παίρνουμε την παρακάτω γραφική παράσταση :



- **Εντολή PLOT3**

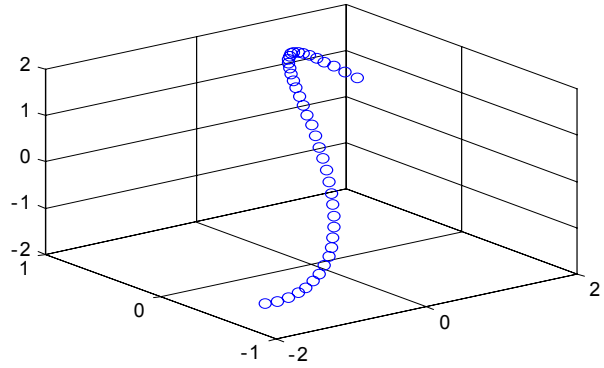
Για γραφικές παραστάσεις σε τρεις διαστάσεις, χρησιμοποιείται η εντολή **plot3** η οποία έχει τη μορφή:

plot3(x_1 , y_1 , z_1 , x_2 , y_2 , z_2 , ..., x_n , y_n , z_n **)** [**,grid**].

Στην plot3 μπορούμε να προσθέσουμε τις ίδιες επιλογές με αυτές της plot.

Παράδειγμα:

```
»a = -2 : 0.1 : 2 ;
»x = a ;
»y = sin(x) ;
»z = y + cos(x) ;
»plot3 ( x , y , z , 'bo' ) , grid
```



- **Εντολή FPLOT**

fplot (fname, limits [,marker] [,tol]))

Σχεδιάζει το γράφημα μιας ή περισσοτέρων συναρτήσεων που υποδεικνύονται από τη μεταβλητή τύπου string fname μεταξύ προσδιοριζόμενων ορίων του άξονα των x. Τα όρια σχεδίασης *limits* δίνονται σαν

[xmin xmax], ή:
[xmin xmax ymin ymax], για υποδήλωση ορίων και στον άξονα των y

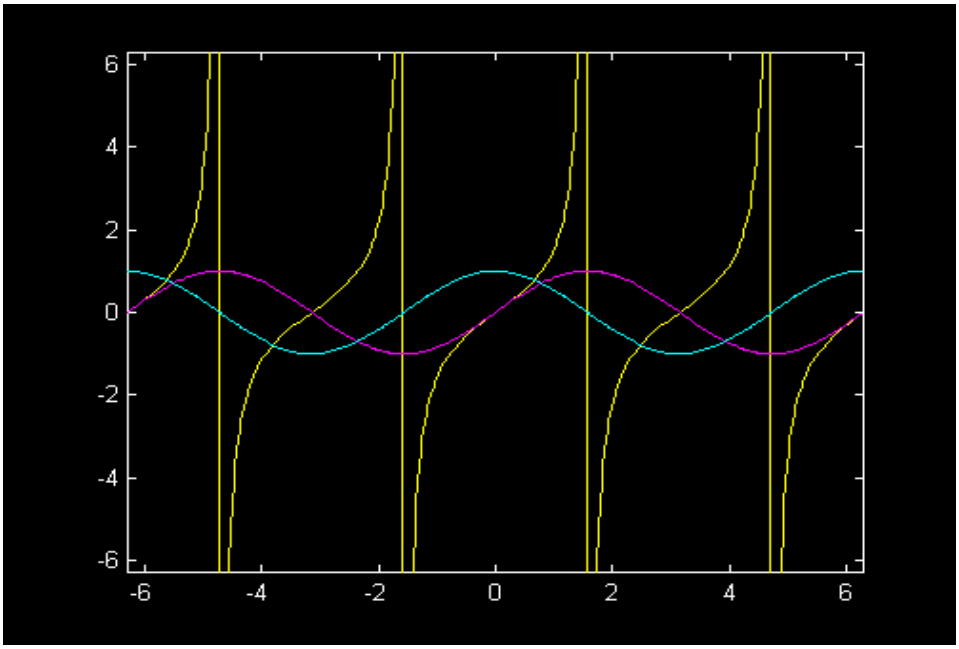
Η συνάρτηση fname μπορεί να είναι

- Μία ή περισσότερες εκτιμήσιμες συναρτήσεις μιας μεταβλητής x, όπως 'sin(x)', ή '[sin(x),cos(x)]'.
- Μια συνάρτηση της μορφής f= [f1(x) f2(x) ... fn(x)]. Η f θα επιστέφει τιμές υπό τη μορφή στηλών (όπως ακριβώς απαιτεί το Matlab). Αν π.χ. δοθεί σαν x το διάνυσμα [x1,x2,...,xm], η f θα επιστρέψει τον πίνακα f(x)=(fi(xj)), i=1,...,m, j=1,...,n.

Το προαιρετικό όρισμα **marker** προσδιορίζει τον τύπο του συμβόλου σχεδίασης και είναι ένα από τα strings '-+', '-x', '-o', '-*'. (default = '-'). Το όρισμα **tol** είναι η ανοχή του σχετικού σφάλματος (default = 2e-3). Το μέγιστο πλήθος βημάτων είναι (1/tol)+1.
Η κλήση [X,Y] = fplot(fname, limits,...) επιστρέφει στις στήλες X και Y τις συντεταγμένες x και y που πήραν μέρος στη σχεδίαση του fname.

Παράδειγμα:

» fplot('[tan(x),sin(x),cos(x)]',[-2*pi 2*pi -2*pi 2*pi])



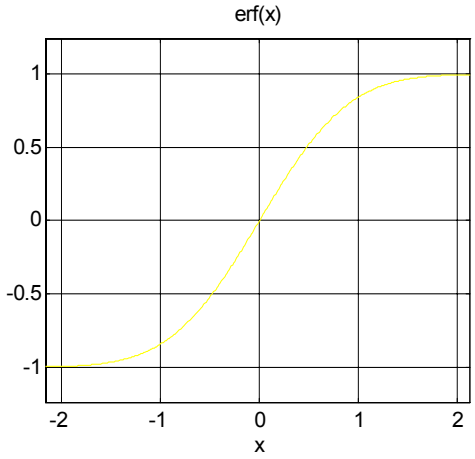
- **Εντολή EZPLOT**

ezplot(f,[xmin xmax])

όπου f η μια συμβολική συνάρτηση μιας μεταβλητής και [xmin xmax] το διάστημα σχεδίασης. Αν παραλείψουμε το διάστημα αυτό, το διάστημα που επιλέγεται είναι το [-2π,2π].

Παράδειγμα Η εντολή ezplot('erf(x)') σχεδιάζει το γράφημα της συνάρτησης σφάλματος

$$erf(x)=\frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}\,dt\,.$$



B.1.9 M-Files

Όταν για την επίλυση ενός προβλήματος έχουμε προς εκτέλεση μια μεγάλη ακολουθία εντολών (scripts), το Matlab δίνει τη δυνατότητα να τις αποθηκεύσουμε σε ένα αρχείο και να το εκτελούμε όποτε αυτό απαιτείται. Τα αρχεία αυτά ονομάζονται '**M-files**' και έχουν την προσθήκη '**.m**'. Τα M-files πρέπει να αποθηκεύονται σε έναν υποκατάλογο που ανήκει στο μονοπάτι εύρεσης του Matlab.

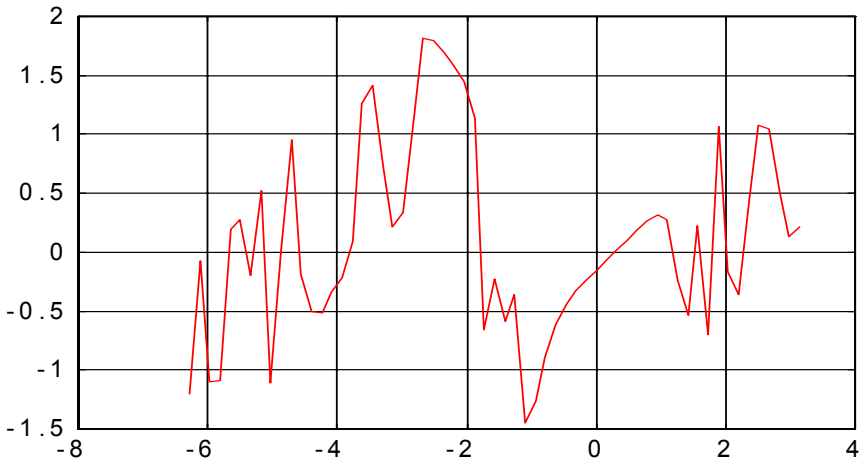
Δίνοντας το όνομα του αρχείου, αυτόματα το Matlab δέχεται ως είσοδο ακολουθιακά τις εντολές που είναι αποθηκευμένες σ' αυτό, τις οποίες και διερμηνεύει. Οι εντολές αυτές μπορεί να είναι οποιεσδήποτε μπορούν να δοθούν στη γραμμή εντολών του Matlab, οι εντολές ελέγχου ροής της γλώσσας script που είδαμε και κλήσεις άλλων M-files. Τα M-files προσφέρονται ιδιαίτερα για να περιλαμβάνουν συναρτήσεις στη γλώσσα script.

Επεξηγηματικά σχόλια σε ένα M-file μπορούν να εισαχθούν με το σύμβολο «**%**». Τα σχόλια αρχίζουν αμέσως μετά το '%' και τελειώνουν στο τέλος της γραμμής. Οι πρώτες γραμμές σχολίων που υπάρχουν στο αρχείο μέχρι την πρώτη γραμμή που δεν αρχίζει με '%', αποτελούν το κείμενο βοήθειας που εμφανίζεται όταν δίνουμε την εντολή **help** <όνομα M-file>.

Παράδειγμα
Έστω ένα M-file με όνομα example.m και περιεχόμενα τα εξής:

```
% Το αρχείο αυτό σχεδιάζει τη γραφική παράσταση της συνάρτησης
% y=sin(cos(x.^2)+tan(x))-cos(x) στο διάστημα [-2π,π]
  x=-2*pi:pi/20:pi;              % Καθορισμός των τιμών του x με βήμα π/20
  y=sin(cos(x.^2)+tan(x))-cos(x);
  plot(x,y,'r'),grid
```

Όταν δώσουμε στην γραμμή εντολών την εντολή *example*, θα πάρουμε την εξής γραφική παράσταση:



Αν δώσουμε την εντολή *help example,* θα εμφανισθεί στην οθόνη:

```
Το αρχείο αυτό σχεδιάζει τη γραφική παράσταση της συνάρτησης
y=sin(cos(x.^2)+tan(x))-cos(x) στο διάστημα [-2π,π]
```

Αν αφαιρέσουμε τα '**;**' από το τέλος των γραμμών θα εμφανιστούν και οι τιμές των x και y.

B.1.10 Συναρτήσεις ορισμένες από το χρήστη

Ορισμός συναρτήσεων

Συναρτήσεις ορισμένες από το χρήστη μπορούν να εισαχθούν στο λεξιλόγιο του Matlab αν εκφραστούν με τη βοήθεια άλλων συναρτήσεων ή εντολών. Οι εντολές και συναρτήσεις που θα αποτελούν την νέα συνάρτηση πρέπει να τοποθετηθούν σε ένα M-file. Στην αρχή του αρχείου πρέπει να υπάρχει μία γραμμή που περιέχει την σύνταξη της συνάρτησης. Πρέπει να τονισθεί ότι το όνομα της συνάρτησης πρέπει να είναι ίδιο με το όνομα του αρχείου που την περιέχει.

Παράδειγμα 1 Το αρχείο με όνομα ' **root1.m** ' που περιέχει τη συνάρτηση root1:

```
function x0=root1(a, b)
%Υπολογίζει την ρίζα της πρωτοβάθμιας εξίσωσης ax+b=0.
%Κλήση: x0=root1(a,b).
if a~=0  x0=-b/a;
  elseif b~=0 x0=-sign(b)*Inf;
      else x0=NaN;     %undefined solution
end
end
```

ορίζει μία νέα συνάρτηση root1, παίρνει παραμέτρους τους συντελεστές μιας πρωτοβάθμιας εξίσωσης και επιστρέφει την ρίζα της.

Όλες οι μεταβλητές που δημιουργούνται μέσα στο αρχείο που περιέχει τη συνάρτηση είναι απομονωμένες (τοπικές μεταβλητές) από το workspace της Matlab. Η μόνη σύνδεση μεταξύ των τοπικών μεταβλητών και του workspace είναι οι μεταβλητές εισόδου και εξόδου. Αν μία συνάρτηση αλλάζει την τιμή οποιασδήποτε από τις μεταβλητές εισόδου, οι αλλαγές δεν επηρεάζουν τις μεταβλητές του workspace.

Αν μία συνάρτηση έχει περισσότερες από μία μεταβλητές εξόδου, τότε οι μεταβλητές εξόδου περικλείονται σε αγκύλες, όπως φαίνεται στο παρακάτω παράδειγμα:

Παράδειγμα 2

```
function [r1,r2] = root2(a,b,c)
%Επίλυση της δευτεροβάθμιας εξίσωσης ax^2+bx+c=0
%Κλήση: [r1,r2] = root2(a,b,c)
format long e
if a==0  r1=root1(b,c);
 else
      d=b^2-4*a*c;
      if d==0
           r1=-b/(2*a); r2=r1; disp('duble solution');
      else
            r1=(-b+sqrt(d))/(2*a);
           r2=(-b-sqrt(d))/(2*a);
      end
end
end
```

Δύο χρήσιμες προκαθορισμένες μεταβλητές είναι οι **nargin**, **nargout** που κρατάνε των αριθμό των μεταβλητών εισόδου και εξόδου αντίστοιχα. Για το παραπάνω παράδειγμα είναι nargin=3, nargout=2. Με κατάλληλο χειρισμό των μεταβλητών αυτών μπορούμε να δώσουμε σε μια συνάρτηση ποικίλες μορφές κλήσεων, οι οποίες να αντιστοιχούν σε μεταβλητό αριθμό παραμέτρων εισόδου/εξόδου.

Παράδειγμα 3 Για να συνενώσουμε τις λειτουργίες των συναρτήσεων root1 και root2 των παραδειγμάτων 1 και 2 αντίστοιχα, σε μια ενιαία συνάρτηση root, γράφουμε την παρακάτω συνάρτηση.

```
function [r1,r2,Det] = root(a,b,c)
%Επίλυση πρωτοβάθμιας ή δευτεροβάθμιας εξίσωσης
%Κλήσεις:  r1 = root(a,b), (ax+b=0)
%              [r1,r2] = root(a,b,c), (ax^2+bx+c=0)
%              [r1,r2,det] = root(a,b,c), det=διακρίνουσα

if nargin==2 r1=root1(a,b);            %πρωτοβάθμια εξίσωση, αριθμός παραμέτρων εισόδου=2
else
if a==0  r1=root1(b,c);                %πρωτοβάθμια εξίσωση
 else                                  %δευτεροβάθμια εξίσωση, αριθμός παραμέτρων εισόδου=3
      d=b^2-4*a*c;
      if d==0
           r1=-b/(2*a); r2=r1; disp('duble solution');
      else
        r1=(-b+sqrt(d))/(2*a);
       r2=(-b-sqrt(d))/(2*a);
      end;
      if nargout==3 det=d; end  %αν δοθούν τρεις παράμετροι εξόδου, τότε εξάγεται και η διακρίνουσα
end
end
end
```

Χρήσιμες εντολές για συναρτήσεις και M - Files

Για την ενεργοποίηση ακολουθιών εντολών και συναρτήσεων χρησιμοποιούνται οι βασικές συναρτήσεις eval και feval.

- **eval(s)** : όπου s ένα string. Εκτελεί το string s που περιέχει μια έκφραση. Προκαλεί την εκτέλεση του string σαν να είναι κανονική εντολή ή δήλωση. Το s μπορεί να είναι το όνομα ενός M-file.
- **[X,Y,Z,...] = eval(s)**. Επιστρέφει αποτελέσματα στα ορίσματα X,Y,Z από την έκφραση που περιέχεται στο string s.
- **feval(F, x_1, x_2,..., x_n)**, όπου F ένα string που περιέχει μία συνάρτηση και $x_1,x_2,...,x_n$ τα ορίσματά της. Υπολογίζει την τιμή της συνάρτησης F για τα $x_1,x_2,...,x_n$. Η συνάρτηση F μπορεί να καθορίζεται από ένα M-file.

Παραδείγματα

Σε ένα M-file δίνουμε τις εντολές:

```
% Έκδοση demo για μία από 6 συναρτήσεις του matlab\numerics
% ορισμός ενός πίνακα 6x1 από strings που πρέπει να έχουν τον ίδιο αριθμό χαρακτήρων
% (=στήλες πίνακα)
d=['rrefmovie'; 'zerodemo '; 'spline2d '; 'fitdemo '; 'fplotdemo'; 'eigmovie '];
n=input(' Διάλεξε έναν αριθμό για έκδοση demo 1-6: ');      %  αναμονή εισόδου από το χρήστη
eval(d(n, : ))                  %  εκτελείται το d(n , : ) σαν να είναι μία κανονική εντολή του αρχείου.
```

Αν F='cos', τότε η κλήση feval(F,pi) ισοδυναμεί με την cos(pi) ;

Επίσης, αν S='root' (του παραδείγματος 3), η κλήση [x1,x2,d] =feval(S,1,1,1) ισοδυναμεί με [x1,x2, d]=root(1,1,1).

Τέλος η εντολή **disp** εμφανίζει στην οθόνη τα στοιχεία του πίνακα (χωρίς να εμφανίζει το όνομά του) ή το κείμενο που περιέχεται ανάμεσα στα εισαγωγικά.

- **disp(x)**, όπου x είναι ένας πίνακας με στοιχεία οποιουδήποτε τύπου, ή
- **disp('<text>')**, όπου <*text*> είναι μια ή περισσότερες γραμμές κειμένου.

Παράδειγμα

```
» t=[5.0 6.3 ;5.1 7.7];
» disp(t)
   5.0000    6.3000
   5.1000    7.7000
» disp('Matlab is a powerful language')
Matlab is a powerful language
```

B.2 Εφαρμογές στις Αριθμητικές Μεθόδους και Γραμμική Άλγεβρα

B.2.1 Επίλυση μη Γραμμικών Εξισώσεων

Μέθοδος Διχοτόμησης

Η σύνταξη της υποστηριζόμενης εντολής για τη μέθοδο αυτή είναι:

part(f,a,b,t,i)

όπου f το όνομα ενός M-file στο οποίο έχουμε σώσει την προς μελέτη συνάρτηση. Τα a,b είναι τα άκρα του διαστήματος εγκλεισμού της ρίζας, t ο αριθμός των σημαντικών ψηφίων και i ο μέγιστος αριθμός επαναλήψεων.

Παράδειγμα Αν έχουμε σώσει τη συνάρτηση στο αρχείο fun.m (το οποίο πρέπει να βρίσκεται μέσα σε ένα κατάλογο που έχει δηλωθεί στο path του Matlab), ψάχνουμε τη ρίζα στο διάστημα [0,2] και επιθυμούμε ακρίβεια 8 σημαντικών ψηφίων και μέγιστο αριθμό επαναλήψεων 10.000, δίνουμε:

part('fun',0,2,8,10000)

Μέθοδος Newton-Raphson

Η σύνταξη της υποστηριζόμενης συνάρτησης είναι:

newtrap(f,x,t,i)

όπου f το όνομα ενός m-file που περιέχει την προς μελέτη συνάρτηση, x μια προσέγγιση της ρίζας, t ο αριθμός των σημαντικών ψηφίων και i ο μέγιστος αριθμός επαναλήψεων.

Παράδειγμα Αν έχουμε σώσει μια συνάρτηση στο αρχείο fun1.m, ψάχνουμε την ρίζα στη περιοχή του 0.5 και επιθυμούμε ακρίβεια 8 σημαντικών ψηφίων και μέγιστο αριθμό επαναλήψεων 10.000, δίνουμε:

newtrap('fun1',0.5,8,10000)

Μέθοδος τέμνουσας

Η σύνταξη της εντολής έχει ως εξής:

[p1,y1,err]=secant('f',p0,p1,delta,epsilon,max1) ή
[p1,y1,err,P]=secant('f',p0,p1,delta,epsilon,max1)

όπου f το όνομα ενός αρχείου κειμένου που περιέχει την προς μελέτη συνάρτηση, p0 και p1 τα αρχικά σημεία, delta η ανοχή σύγκλισης για το p1, epsilon η ανοχή σύγκλισης για το y1 και max1 ο μέγιστος αριθμός επαναλήψεων. p1 είναι η ρίζα, y1 η τιμή της συνάρτησης για x=p1, err το σφάλμα κατά τον υπολογισμό της ρίζας και P το διάνυσμα όπου αποθηκεύεται κάθε επανάληψη.

Μέθοδος Horner

Όπως είδαμε, για να υπολογίσουμε μια τιμή ενός πολυωνύμου p δίνουμε την polyval(P,x), ενώ για να παραγωγίσουμε δίνουμε την polyder(x). Και οι δύο συναρτήσεις χρησιμοποιούν το σχήμα Horner. Ας σημειωθεί τέλος ότι η συνάρτηση **horner(p)** μετασχηματίζει το συμβολικό πολυώνυμο p στη μορφή Horner.

Παράδειγμα

```
» p = 'x^3-6*x^2+11*x-6';
» horner(p)
ans =
-6+(11+(-6+x)*x)*x
```

Β.2.2 Γραμμικά Συστήματα

Μέθοδος Gauss

Με την εντολή ***rref*** ένα σύστημα Ax=b μετατρέπεται σε ένα ισοδύναμο σύστημα Ux=c, όπου ο U είναι πάνω τριγωνικός, σύμφωνα με τη μέθοδο απαλοιφής Gauss-Jordan με μερική οδήγηση.

Παράδειγμα: Για το σύστημα:

$$x_1 + x_2 + x_3 = 3$$
$$2x_1 + 3x_2 - x_3 = 15$$
$$4x_1 - 2x_2 + 6x_3 = -10$$

παίρνουμε :

```
 » A=[1 1 1 3;2 3 -1 15;4 -2 6 -10]
A =
     1     1     1     3
     2     3    -1    15
     4    -2     6   -10
» rref(A)
ans =
     1     0     0     2
     0     1     0     3
     0     0     1    -2
```

Η εντολή ***rrefmovie*** ανιχνεύει τα βήματα του αλγορίθμου:

```
Original matrix
A =
     1          1          1          3
     2          3         -1         15
     4         -2          6        -10
Press any key to continue. . .
swap rows 1 and 3
A =
     4         -2          6        -10
     2          3         -1         15
     1          1          1          3
Press any key to continue. . .
A =
     1       -1/2        3/2       -5/2
     0          4         -4         20
     0        3/2       -1/2       11/2
Press any key to continue. . .
A =
     1          0          1          0
     0          1         -1          5
     0          0          1         -2
Press any key to continue. . .
A =
     1          0          0          2
     0          1          0          3
     0          0          1         -2
```

Η μέθοδος Gauss παρέχεται με τη χρήση του τελεστή \. Η επίλυση ενός συστήματος Ax=b ανάγεται στην πράξη x=A\b, όπου A ο πίνακας των συντελεστών των αγνώστων και b το σταθερό διάνυσμα.

Ας αναφερθεί ότι στην περίπτωση που οι εξισώσεις είναι περισσότερες από τους αγνώστους τότε για την επίλυση του συστήματος χρησιμοποιείται η μέθοδος ελαχίστων τετραγώνων για την ελαχιστοποίηση του σφάλματος.

Όταν ο αριθμός των εξισώσεων είναι μικρότερος από τον αριθμό των αγνώστων τότε από τις άπειρες λύσεις υπολογίζονται μόνο δύο. Συγκεκριμένα υπολογίζουμε x=pinv(A)*b, όπου pinv(A) ο ψευδοαντίστροφος του πίνακα A, και βρίσκουμε τη λύση με τη μικρότερη νόρμα.

Παράδειγμα (Άσκηση III.5.1)

Για την επίλυση του συστήματος: .

$$\{3x + y - z = 0,\ -2x - 6y + 3z = 2,\ 4x - 2y + 8z = 1\}$$

με τη μέθοδο Gauss δίνουμε:

```
» A = [ 3 1 -1 ; -2 -6 3 ; 4 -2 8 ];
» b = [ 0 ; 2 ; 1 ];
» x=A\b
x =
   0.1190
  -0.3889
  -0.031
```

Παραγοντοποίηση LU

Η μέθοδος παραγοντοποίησης υλοποιείται στο περιβάλλον του Matlab με την συνάρτηση ***lu*** που συντάσσεται ως εξής:

- **[L,U]=lu(A)** Ο πίνακας L επιστρέφει έναν κάτω τριγωνικό πίνακα και η U έναν πάνω άνω τριγωνικό, ώστε LU=PA, όπου P είναι ένας μεταθετικός πίνακας.
- **[L,U,P]=lu(A)** Ο πίνακας P επιστρέφει τον μεταθετικό πίνακα P.

Παράδειγμα

```
» A=[1 1 1 ;2 3 -1;4 -2 6];
» [L,U]=lu(A)
L =
   0.2500    0.3750    1.0000
   0.5000    1.0000         0
   1.0000         0         0
U =
    4   -2    6
    0    4   -4
    0    0    1

» A=[0 2 1;1 -2 4;4 -1 2];
» [L,U,P]=lu(A)
L =
   1.0000         0         0
        0    1.0000         0
   0.2500   -0.8750    1.0000
U =
   4.0000   -1.0000    2.0000
        0    2.0000    1.0000
        0         0    4.3750
P =
    0    0    1
    1    0    0
    0    1    0
```

Μέθοδος Cholesky

Αν ο Α είναι θετικά ορισμένος πίνακας, η εντολή

R=chol(A)

επιστρέφει έναν άνω τριγωνικό πίνακα U, για τον οποίο ισχύει U^t *U=A
Παρακάτω δίνουμε μία συνάρτηση προγραμματισμένη στη γλώσσα script που υλοποιεί την μέθοδο Choleski. Το πρόγραμμα διασπά ένα θετικά ορισμένο Α στη μορφή $A=LL^T$ και είναι βασισμένο στον αλγόριθμο Cholesky του Κεφ. 3 (χωρίς οδήγηση και χωρίς αντιμεταθέσεις γραμμών). Έλεγχος για το αν ο Α είναι θετικά ορισμένος δεν πραγματοποιείται.

```
function L=choleski(A);
%Cholesky factorization
n=length(A);
L=zeros(n);
L(1,1)=sqrt(A(1,1));
for i=2:n
        L(i,1)=A(i,1)/L(1,1);
end
for j=2:(n)
        temp=0;
        for k=1:(j-1)
                temp=temp+(L(j,k))^2;
        end
        L(j,j)=sqrt(A(j,j)-temp);
        if j<n
                for i=(j+1):n
                        temp1=0;
                        for k=1:(j-1)
                                temp1=temp1+L(i,k)*A(j,k);
                        end
                L(i,j)=(A(i,j)-temp1)/L(j,j);
                end
        end
end
```

Το πρόγραμμα αυτό εκτελείται με την εντολή *choleski(A)*, και δίνει τον κάτω τριγωνικό πίνακα L, για τον οποίο ισχύει $A=LL^T$.

<u>*Παράδειγμα*</u> Εφαρμόζουμε τη παραπάνω συνάρτηση *choleski* για δύο πίνακες.

```
» A=[4 12;12 45];
» L=chole(A)
L =
    2    0
    6    3

» L*L'
ans =
    4   12
   12   45

» B=[1 2 0 0;2 6 -2 0;0 -2 5 -2;0 0 -2 3];
» L=chole(B)
```

```
L =
   1.0000         0         0         0
   2.0000    1.4142         0         0
        0   -1.4142    1.7321         0
        0         0   -1.1547    1.2910
» L*L'
ans =
   1.0000    2.0000         0         0
   2.0000    6.0000   -2.0000         0
        0   -2.0000    5.0000   -2.0000
        0         0   -2.0000    3.0000
```

Ιδιοτιμές και ιδιοδιανύσματα

Οι ιδιοτιμές και τα ιδιοδιανύσματα ενός πίνακα υπολογίζονται με τη συνάρτηση ***eig***.

- **poly(A)** Επιστρέφει το *χαρακτηριστικό πολυώνυμο* ενός πίνακα A.
- **eig(A) :** Επιστρέφει τις ιδιοτιμές του πίνακα A.
- **[V,D] = eig(A) :** Παράγεται ένας διαγώνιος πίνακας D με τις ιδιοτιμές και ένας πίνακας V, του οποίου οι στήλες αντιστοιχούν στα ιδιοδιανύσματα, ώστε να ισχύει A*V=V*D . Μια πρόσθετη παράμετρος, το 'nobalance' ([V,D]= eig(A, 'nobalance')), υπάρχει για την καλύτερη αναπαράσταση των τιμών σε ορισμένες περιπτώσεις.

Παραδείγματα

```
» A=[1 1 0; 0 2 0; 0 1 3];
» poly(A)
ans =
    1   -6    11    -6

» eig(A)
ans =
    1
    3
    2

» [V,D]=eig(A)
V =
   1.0000         0    0.5774
        0         0    0.5774
        0    1.0000   -0.5774
D =
    1    0    0
    0    3    0
    0    0    2
```

Οι σχετικές συναρτήσεις που υποστηρίζονται στο *Symbolic Toolbox* είναι:

- ***eigensys(A)*** : Υπολογίζει τις ιδιοτιμές του συμβολικού ή αριθμητικού πίνακα A.
- ***[V, E]=eigensys(A)*** : Υπολογίζει στο διάνυσμα E τις ιδιοτιμές και στον πίνακα V τα ιδιοδιανύσματα του πίνακα A.
- ***charpoly(A)*** : Επιστρέφει το *χαρακτηριστικό πολυώνυμο* του A.
- Η *κανονική μορφή Jordan* ενός πίνακα A είναι ο διαγώνιος πίνακας των ιδιοτιμών J και προκύπτει από τον μετασχηματισμό $V^{-1}AV=D$, όπου V ένας αντιστρεπτός πίνακας του οποίου οι στήλες είναι ιδιοδιανύσματα του A. Η συνάρτηση ***jordan(A)*** βρίσκει τον πίνακα V και έχει δύο μορφές: jordan(A) και [V,J]=jordan(F).

Παράδειγμα

```
» A=sym('a b c;b c a;c a b')
A =
[a,b,c]
[b,c,a]
[c,a,b]

» eigensys(A)
ans =
[  a+b+c                                        ]
[-(a^2-b*a-c*a-c*b+b^2+c^2)^(1/2)]
[ (a^2-b*a-c*a-c*b+b^2+c^2)^(1/2)]

» [V,E]=eigensys(A)
V =
[1, -(a+E(2)-c)/(a-b), -(a+E(3)-c)/(a-b)]
[1,              1       ,                  1     ]
[1,  (b-c+E(2))/(a-b),  (b-c+E(3))/(a-b)]

E =
[ a+b+c                                        ]
[ (a^2-b*a-c*a-c*b+b^2+c^2)^(1/2)]
[-(a^2-b*a-c*a-c*b+b^2+c^2)^(1/2)]

» F=sym(A);

» jordan(F)
ans =
[1, 0, 0]
[0, 2, 0]
[0, 0, 3]

» [V,J]=jordan(F)
V =
[-1,  2, 0]
[ 0,  2, 0]
[ 0, -2, 2]
J =
[1, 0, 0]
[0, 2, 0]
[0, 0, 3]
```

Νόρμα και δείκτης κατάστασης

Η νόρμα ενός πίνακα υπολογίζεται με την εντολή ***norm*** που συντάσσεται ως εξής:

- **norm(A)** Υπολογίζει αυτόματα τη νόρμα $||A||_2$.
- **norm(A, p)** Υπολογίζεται η νόρμα του πίνακα A της οποίας ο τύπος καθορίζεται από την παράμετρο p:
  - *norm(A, 1):* Υπολογίζει τη νόρμα $||A||_1$.
  - *norm(A, 2):* Υπολογίζει τη νόρμα $||A||_2$.
  - *norm(A, inf):* Υπολογίζει τη νόρμα $||A||_\infty$.
  - *norm(A, 'fro'):* Υπολογίζει την *F-νόρμα* του πίνακα A (=sqrt(sum(diag(A'*A)))).

Ο *δείκτης κατάστασης* πίνακα υπολογίζεται με την εντολή ***cond*** και ***condest***. Η πρώτη εντολή υπολογίζει το δείκτη κατάστασης με βάση τη νόρμα $||A||_2$, ενώ η δεύτερη με βάση τη νόρμα $||A||_1$.

Παραδείγματα

```
» A=[1 1 0;0 2 0;0 1 3]
A =
    1    1    0
    0    2    0
    0    1    3
» norm(A)
ans =
   3.2988
» norm(A,1)
ans =
    4
» norm(A,2)
ans =
   3.2988
» norm(A,inf)
ans =
    4
» norm(A,'fro')
ans =
    4

» A=[0.8064 0.7904;0.7904 0.8144]
A =
   0.8064    0.7904
   0.7904    0.8144
» cond(A)
ans =
  80.0810
» condest(A)
ans =
  80.4807
```

Προσαρμογή καμπυλών

Η πολυωνυμική προσαρμογή ελαχίστων τετραγώνων υλοποιείται από τη συνάρτηση *polyfit*:

p=polyfit(x,y,n)

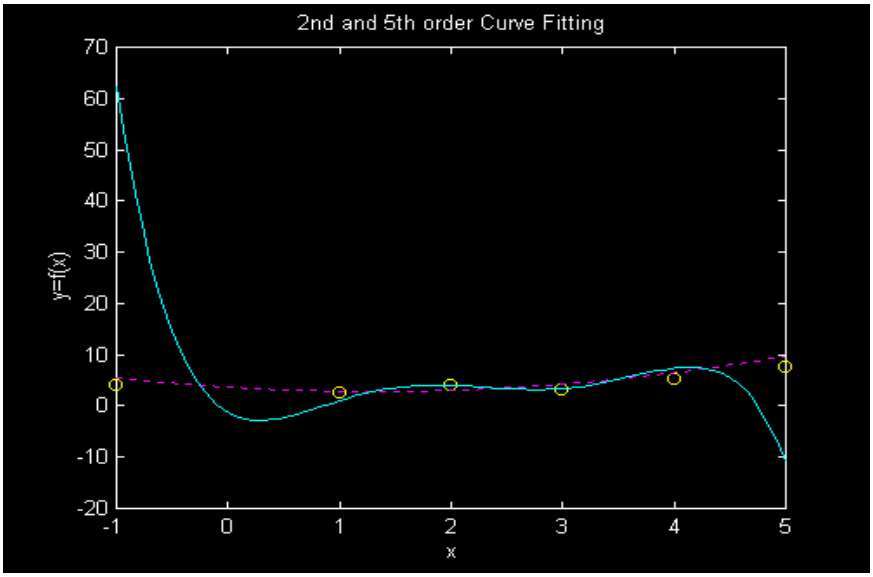
η οποία βρίσκει τους συντελεστές του πολυωνύμου ελαχίστων τετραγώνων p(x) βαθμού n που προσαρμόζεται στα σημεία (x(i), y(i)). Το αποτέλεσμα είναι ένα διάνυσμα-γραμμή με τους συντελεστές του πολυωνύμου σε φθίνουσα διάταξη.
Για να δώσουμε ένα γράφημα της παρεμβολής, πρέπει να παράγουμε σημεία στον άξονα των x και με βάση αυτά να παράγουμε τις τιμές του p(x) με τη συνάρτηση polyval. Η διαδικασία αυτή φαίνεται στο παρακάτω παράδειγμα:

Παράδειγμα Για τα δεδομένα (-1,4), (1,2.5), (2,4), (3,3.2), (4,5.1), (5,7.4) (άσκηση VI.5) θα υπολογίσουμε τα πολυώνυμα ελαχίστων τετραγώνων βαθμών 2 και 5. Στη συνέχεια , σχεδιάζουμε τα γραφήματά τους σε μια ενιαία γραφική παράσταση.

```
» x=[-1,1,2,3,4,5] ;
» y=[4,2.5,4,3.2,5.1,7.4];
» p2=polyfit(x,y,2)                    %κατασκευή πολυώνυμο ελαχ. τετραγώνων 2ου βαθμού
p2 =
   0.2845   -0.6081    3.1300
» p5=polyfit(x,y,5)                    %κατασκευή πολυώνυμο ελαχ. τετραγώνων 5ου βαθμού
p5 =
```

```
  -0.0943    1.1104   -4.1410    4.4229    3.4853   -2.2833

» xi=linspace(-1,5,100);                  %δημιουργία δεδομένων στον άξονα των x
» z2=polyval(p2,xi);                      %δημιουργία δεδομένων στον άξονα των y για το p2
» z5=polyval(p5,xi);                      %δημιουργία δεδομένων στον άξονα των y για το p5
» plot(x,y,'o',xi,z2,':',xi,z5)           %σχεδίαση γραφημάτων για τα p2 και p5
                                          %το γράφημα του p2 είναι με διακεκομμένη γραμμή
» xlabel('x'),ylabel('y=f(x)'),           %ενδείξεις στους άξονες
» title('2nd and 5th order Curve Fitting')%έκδοση τίτλου γραφήματος
```



Παρεμβολή

Η παρεμβολή μονοδιάστατων δεδομένων υλοποιείται από τη εντολή *interp1*. Τα σημεία (x_i, y_i) που θα παρεμβληθούν πρέπει να αποθηκευθούν σε δύο διανύσματα x (για τα x_i)και y (για τα y_i).

- ***yi=interp1(x,y,xi)***: υπολογίζει στο διάνυσμα yi τις τιμές που αντιστοιχούν στα δεδομένα του διανύσματος xi και οι οποίες καθορίζονται έπειτα από εφαρμογή γραμμικής παρεμβολής στα σημεία (x_i,y_i) των διανυσμάτων x και y.
- ***yi=interp(x,y,xi, 'method')***: προσδιορίζει την μέθοδο παρεμβολής που θα εφαρμοσθεί, όπου method μπορεί να είναι:

  -'linear' για γραμμική παρεμβολή
  -'spline' για παρεμβολή με splines
  -'cubic' για κυβική παρεμβολή. Εδώ απαιτείται οι τιμές του x να είναι ομοιόμορφα κατανεμημένες.

Σε όλες τις περιπτώσεις, οι τιμές του διανύσματος x θα πρέπει να δίνονται διατεταγμένες.

Παράδειγμα
Για τα δεδομένα του προηγούμενου παραδείγματος έχουμε:

```
»  x=[-1,1,2,3,4,5] ;
» y=[4,2.5,4,3.2,5.1,7.4];
» xi=-1:0.1:5;                  %δημιουργία διαστήματος ομοιόμορφα κατανεμημένων τιμών
» y0=interp1(x,y,0,'spline')    %υπολογισμός τιμής της παρεμβολής splines στο σημείο 0
ans =
 y0= 0.9453
```

```
» y0=interp1(x,y,0)              %υπολογισμός τιμής της γραμμικής παρεμβολής στο σημείο 0
y0 =
   3.2500
                                 %Εφαρμογή τριών μεθόδων παρεμβολής
» Yspl=interp1(x,y,xi,'spline');
» Yl=interp1(x,y,xi,'linear');
» Yc=interp1(x,y,xi,'cubic');
» plot(x,y,'o',xi,Yl,xi,Yc,':',xi,Yspl)  %σχεδίαση κοινού γραφήματος
» xlabel('x'),ylabel('Y=f(x)'),
» title('Spline interpolation versus Linear and Cubic interpolation')
```