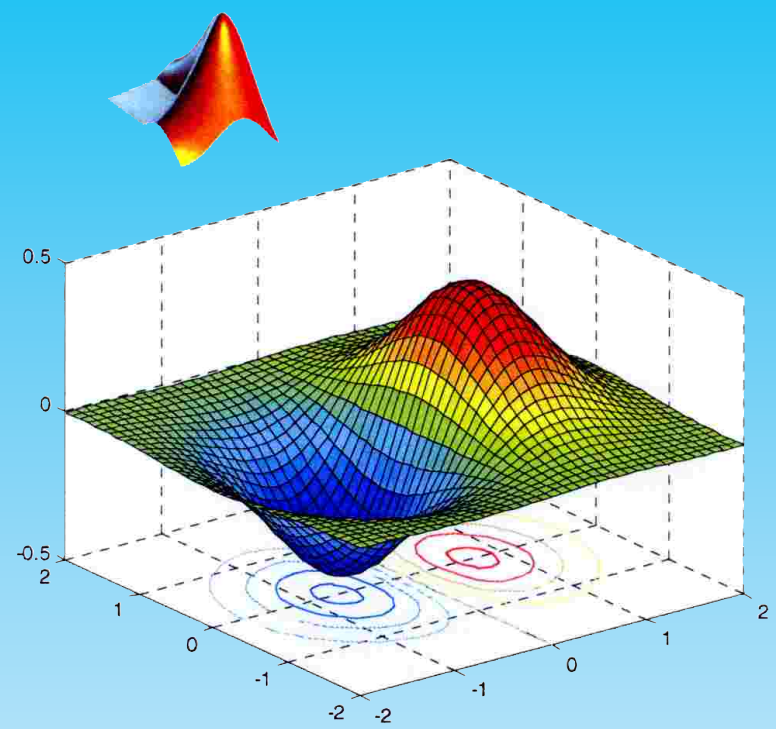
Εισαγωγή στη
MATLAB

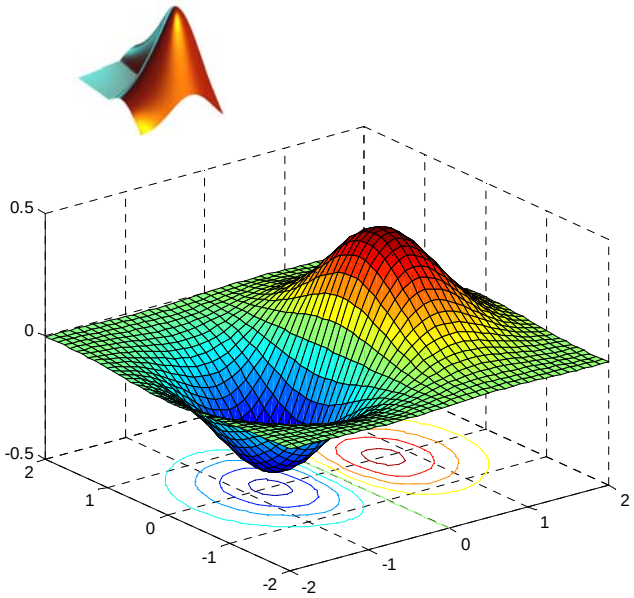


Γ. ΓΕΩΡΓΙΟΥ - Χ. ΞΕΝΟΦΩΝΤΟΣ

ΛΕΥΚΩΣΙΑ
2007

Εισαγωγή στη

MATLAB



Γ. ΓΕΩΡΓΙΟΥ - Χ. ΞΕΝΟΦΩΝΤΟΣ

Λευκωσία
2007

ISBN 978-9963-644-57-5
Kantzilaris
Nicosia, Cyprus
2007

Αφιερώνεται στους φοιτητές

και τις οικογένειές μας

Πρόλογος

Οι σημειώσεις αυτές γράφτηκαν κατά τη διάρκεια του εαρινού εξαμήνου του 2007 όταν διδάξαμε το μάθημα *Μαθηματικά με Υπολογιστές* στο Τμήμα Μαθηματικών και Στατιστικής του Πανεπιστημίου Κύπρου. Στην παρούσα τους μορφή θεωρούνται πρόχειρες. Αν και δεν αποτελούν ολοκληρωμένο βιβλίο, είναι πιστεύουμε χρήσιμες αφού δείχνουν κατά κάποιο τρόπο τη δική μας διαδρομή εκμάθησης της MATLAB.

Τα πρώτα επτά κεφάλαια καλύπτουν τις βασικές εντολές της MATLAB. Τα κεφάλαια 7 έως 11 αφορούν σε εφαρμογές και απαιτούν γνώσεις αριθμητικής ανάλυσης και διαφορικών εξισώσεων.

Για λόγους πρακτικούς η εκτύπωση του βιβλίου δεν είναι έγχρωμη. Η έγχρωμη εκδοχή του είναι διαθέσιμη στις ιστοσελίδες μας

http://www.ucy.ac.cy/~georgios

και

http://www.ucy.ac.cy/~xenophontos

όπου είναι επίσης προσβάσιμα αρκετά m-files που χρησιμοποιούνται στα παραδείγματα και λυμένες ασκήσεις.

Σχόλια και εισηγήσεις για τη βελτίωση των σημειώσεων είναι ευπρόσδεκτα.

Λευκωσία
Ιούλιος 2007

Γιώργος Γεωργίου & Χρίστος Ξενοφώντος
Τμήμα Μαθηματικών και Στατιστικής
Πανεπιστήμιο Κύπρου
ΤΘ 20537, 1678 Λευκωσία
ΚΥΠΡΟΣ

Περιεχόμενα

1.	**ΕΙΣΑΓΩΓΗ**	**1**
1.1	Ξεκινώντας με τη MATLAB	2
1.1.1	Βασικές πράξεις	4
1.1.2	Μεταβλητές	7
1.2	Βαθμωτές συναρτήσεις βιβλιοθήκης	12
1.3	Εντολές διαχείρισης του χώρου εργασίας	15
1.3.1	Οι εντολές quit και exit	15
1.3.2	Οι εντολές clear και clc	15
1.3.3	Η εντολή help	16
1.3.4	Οι εντολές who και whos	16
1.3.5	Οι εντολές save, load και diary	19
1.4	Άλλες χρήσιμες εντολές	20
1.4.1	Οι εντολές demo και doc	20
1.4.2	Οι εντολές dir, ls και what	20
1.4.3	Εντολές ημερομηνίας και ώρας	21
1.5	Είσοδος και έξοδος δεδομένων	22
1.5.1	Η εντολή disp	22
1.5.2	Η εντολή format	23
1.5.3	Η εντολή input	26
1.6	Ειδικές σταθερές και μεταβλητές	28
1.7	Ασκήσεις	31
2.	**ΔΙΑΝΥΣΜΑΤΑ ΚΑΙ ΠΙΝΑΚΕΣ**	**35**
2.1	Γενικά	35
2.1.1	Αριστερή και δεξιά διαίρεση	40
2.2	Στοιχειώδεις πίνακες	42
2.3	Ορισμός διανυσμάτων και πινάκων με βήμα	46
2.4	Συνένωση πινάκων	53
2.4.1	Η εντολή cat	57
2.5	Πράξεις κατά τα στοιχεία διανύσματος ή πίνακα	59
2.6	Συναρτήσεις βιβλιοθήκης για διανύσματα	63
2.6.1	Οι εντολές dot και cross	65
2.7	Χρήσιμες συναρτήσεις βιβλιοθήκης για πίνακες	67
2.7.1	Πινακοσυναρτήσεις	70
2.8	Ασκήσεις	
3.	**M-FILES**	**81**
3.1	Αρχεία script	82
3.2	Αρχεία συναρτήσεων	84
3.3	Μαθηματικές συναρτήσεις	90
3.3.1	Ανώνυμες συναρτήσεις	93
3.3.2	Η εντολή feval	96
3.3	Ασκήσεις	99
4.	**ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΣΤΗ MATLAB**	**103**
4.1	Σχεσιακοί τελεστές	104
4.2	Λογικοί τελεστές	111
4.3	Βρόχοι for	120

4.4 Βρόχοι while 120
4.5 Η εντολή if 122
4.6 Η εντολή switch 127
4.6.1 Η συνάρτηση menu 130
4.7 Ασκήσεις 132

5. ΓΡΑΦΙΚΑ 137
5.1 Η εντολή plot 137
5.1.1 Χρήσιμες συναρτήσεις για γραφικά 139
5.2 Η εντολή ezplot 143
5.2.1 Η εντολή comet 148
5.3 Η εντολή plot 148
5.4 Χρώματα, σύμβολα και γραμμές 150
5.5 Πολλαπλά γραφήματα 153
5.6 Άλλες χρήσιμες εντολές για γραφικά 156
5.6.1 Λογαριθμικοί άξονες 156
5.6.2 Πολλαπλά γραφήματα στο ίδιο παράθυρο 158
5.6.3 Γραφήματα σε πολικές συντεταγμένες 160
5.6.4 Ραβδοδιαγράμματα και εμβαδογράμματα 162
5.6.5 Τομεογράμματα 165
5.6.6 Ιστογράμματα 166
5.7 Γραφήματα στις 3 διαστάσεις 167
5.7.1 Ισοϋψείς καμπύλες 169
5.8 Τρισδιάστατες καμπύλες 174
5.9 Ασκήσεις 176

6 ΠΟΛΥΩΝΥΜΑ 185
6.1 Γενικά περί πολυωνύμων 185
6.2 Χρήσιμες συναρτήσεις για πολυώνυμα 187
6.3 Προσαρμογή δεδομένων και η εντολή polyfit 191
6.4 Ασκήσεις 200

7 ΕΚΤΥΠΩΣΗ ΚΑΙ ΑΝΑΓΝΩΣΗ ΔΕΔΟΜΕΝΩΝ 203
7.1 Εισαγωγή δεδομένων από τον χρήστη 203
7.1.1 Η εντολή pause 203
7.1.2 Η εντολή ginput 205
7.2 Εκτύπωση δεδομένων στην οθόνη 207
7.2.1 Η εντολή sprintf 208
7.3 Ανάγνωση από και γράψιμο σε αρχεία 218
7.3.1 Οι εντολές fopen και fclose 218
7.3.2 Η εντολή fprintf 219
7.3.3 Η εντολή fscanf 221
7.3.4 Οι εντολές fgetl και fgets 224
7.3.5 Οι εντολές fread και fwrite 224
7.4 Ασκήσεις 225

8 ΕΠΙΛΥΣΗ ΜΗ ΓΡΑΜΜΙΚΩΝ ΕΞΙΣΩΣΕΩΝ 227
8.1 Η μέθοδος της διχοτόμησης 227
8.2 Η μέθοδος Newton 229
8.3 Η συνάρτηση fzero 231
8.4 Ασκήσεις 232

9 ΕΠΙΛΥΣΗ ΓΡΑΜΜΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ 233
9.1 Γενικά περί γραμμικών συστημάτων 233
9.2 Απαλοιφή Gauss 235
9.2.1 Παραγοντοποίηση LU 238
9.2.2 Η μέθοδος Gauss 240
9.3 Παραγοντοποίηση Cholesky 241
9.4 Νόρμες διανυσμάτων και πινάκων 245
9.4.1 Νόρμες διανυσμάτων 245
9.4.2 Νόρμες πινάκων 246
9.5 Δείκτης κατάστασης αντιστρέψιμου πίνακα 248
9.6 Επαναληπτικές μέθοδοι 251
9.6.1 Η μέθοδος Jacobi 251
9.6.2 Η μέθοδος Gauss-Seidel 253
9.7 Ασκήσεις 254

10 ΑΡΙΘΜΗΤΙΚΗ ΟΛΟΚΛΗΡΩΣΗ 257
10.1 Αθροίσματα Riemann 257
10.2 Η μέθοδος του τραπεζίου 263
10.2.1 Η εντολή trapz 265
10.3 Οι εντολές quad και quadl 268
10.4 Ασκήσεις 270

11 ΣΥΝΗΘΕΙΣ ΔΙΑΦΟΡΙΚΕΣ ΕΞΙΣΩΣΕΙΣ 273
11.1 Γενικά περί διαφορικών εξισώσεων 273
11.2 Η εντολή ode45 277
11.3 Συστήματα ΣΔΕ 279
11.4 Ασκήσεις 282

ΒΙΒΛΙΟΓΡΑΦΙΑ 285

ΕΥΡΕΤΗΡΙΟ 287

1 ΕΙΣΑΓΩΓΗ

Το **MATLAB** είναι ένα σύγχρονο ολοκληρωμένο μαθηματικό λογισμικό πακέτο που χρησιμοποιείται σε πανεπιστημιακά μαθήματα αλλά και ερευνητικές και άλλες εφαρμογές με επιστημονικούς υπολογισμούς (scientific computing). Το όνομά του προέρχεται από τα αρχικά γράμματα των λέξεων **MATtrix LABoratory** *(εργαστήριο πινάκων*). Το MATLAB είναι ένα **διαδραστικό** (interactive) πρόγραμμα για αριθμητικούς υπολογισμούς και οπτικοποίηση δεδομένων (data visualization) με δυνατότητες προγραμματισμού που το καθιστούν ένα ισχυρό και χρήσιμο εργαλείο στις μαθηματικές και φυσικές επιστήμες. Σε αντίθεση με τα λογισμικά Maple και Mathematica, το MATLAB στις αρχικές του εκδοχές δεν έκανε συμβολικούς υπολογισμούς. Στις νεότερες εκδοχές του, το πακέτο περιλαμβάνει εργαλειοθήκες που επιτρέπουν συμβολικούς υπολογισμούς.

Όπως υποδηλώνεται και από το όνομά του, το MATLAB είναι ειδικά σχεδιασμένο για υπολογισμούς με πίνακες, όπως η επίλυση γραμμικών συστημάτων, η εύρεση ιδιοτιμών και ιδιοδιανυσμάτων, η αντιστροφή τετραγωνικών πινάκων κλπ. Επιπλέον το πακέτο αυτό είναι εφοδιασμένο με πολλές επιλογές για γραφικά (δηλ. την κατασκευή γραφικών παραστάσεων) και προγράμματα γραμμένα στη δική του γλώσσα προγραμματισμού για την επίλυση άλλων προβλημάτων όπως η εύρεση των ριζών μη γραμμικής εξίσωσης, η επίλυση μη γραμμικών συστημάτων, η επίλυση προβλημάτων αρχικών τιμών με συνήθεις διαφορικές εξισώσεις κα.

Η γλώσσα προγραμματισμού του MATLAB δίνει την ευχέρεια στον χρήστη να το επεκτείνει με δικά του προγράμματα. Συχνά θα γράφουμε η MATLAB (εννοώντας τη γλώσσα προγραμματισμού) και όχι το (πακέτο) MATLAB.

Το MATLAB είναι σχεδιασμένο για την αριθμητική επίλυση προβλημάτων σε *αριθμητική πεπερασμένης ακρίβειας* (finite-precision arithmetic), δηλαδή δεν βρίσκει την ακριβή αλλά μια προσεγγιστική λύση ενός προβλήματος. Αυτή είναι και η βασική του διαφορά από τα συστήματα συμβολικών υπολογισμών όπως η Maple και το Mathematica.

Στόχος του πρώτου κεφαλαίου είναι η εξοικείωση του αναγνώστη με τα βασικά χαρακτηριστικά της MATLAB. Κάποια θέματα θα τα δούμε μόνο επιφανειακά αφού θα τα συζητήσουμε σε μεγαλύτερο βάθος σε επόμενα κεφάλαια.

Ας σημειωθεί ότι ο καλύτερος (και ουσιαστικά ο μόνος) τρόπος εκμάθησης της MATLAB είναι η συστηματική ενασχόληση με αυτή και η διερεύνησή της από τον ίδιο τον χρήστη. Το πακέτο είναι εφοδιασμένο με ένα εκτενές σύστημα βοήθειας όπου κάθε εντολή επεξηγείται αναλυτικά και με αντιπροσωπευτικά παραδείγματα. Η πιο σημαντική εντολή της MATLAB είναι η **help** (βοήθεια)! Επίσης, στην επίσημη ιστοσελίδα της MATLAB:

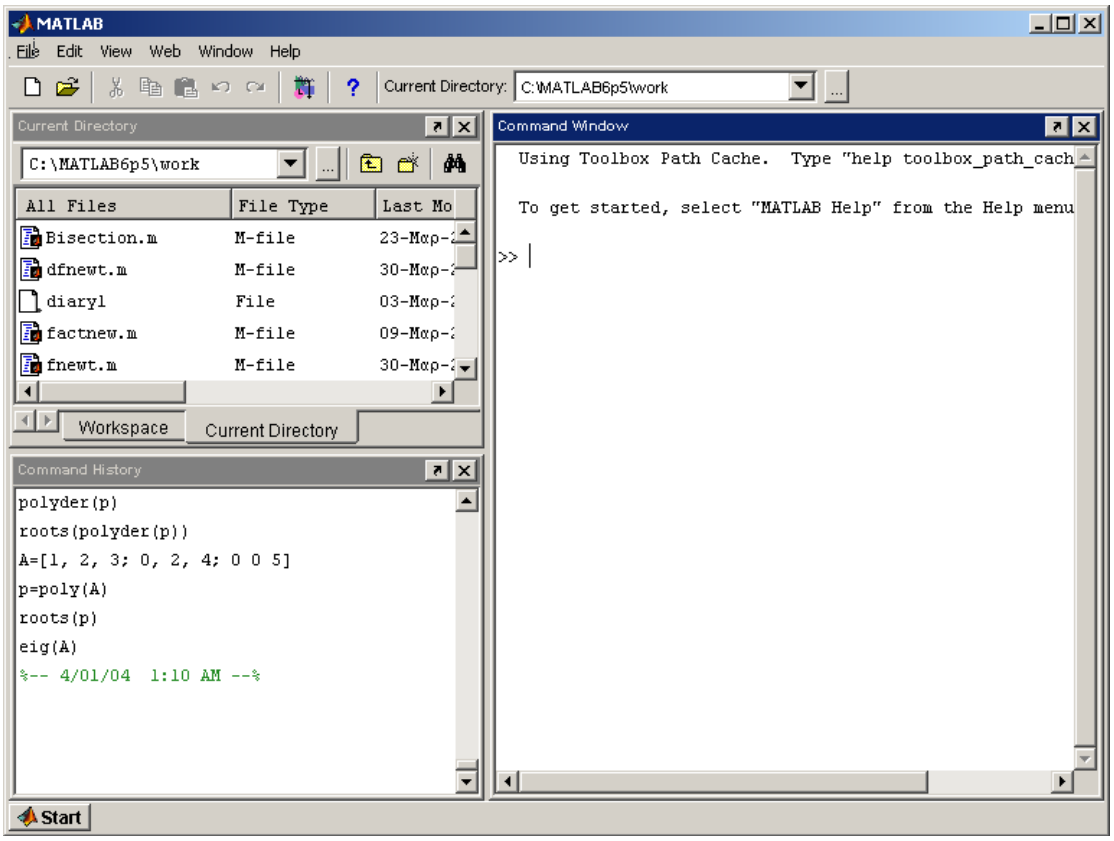
http://www.mathworks.com

μπορεί κάποιος να βρει μια πληθώρα πληροφοριών τόσο για αρχάριους όσο και προχωρημένους

1.1 Ξεκινώντας με τη MATLAB

Για να χρησιμοποιήσουμε το MATLAB πρέπει να το εγκαταστήσουμε πρώτα στον υπολογιστή μας. Το εικονίδιο του πακέτου για την εκδοχή 6.5 φαίνεται πιο κάτω:

Μπορούμε να ξεκινήσουμε το πρόγραμμα με διπλό κλικ πάνω στο εικονίδιο αυτό. Μετά από λίγο, αφού το πρόγραμμα φορτώσει, θα εμφανιστεί στην οθόνη μας το παράθυρο έναρξης της MATLAB (MATLAB opening window):



Γενικά εμφανίζονται τέσσερα παράθυρα:

1) Το μεγάλο **παράθυρο εντολών** (Command Window) στα δεξιά. Οι εντολές της MATLAB εισάγονται σ' αυτό το παράθυρο μετά την *προτροπή* (prompt) >>. Τα αποτελέσματα επίσης τυπώνονται στο παράθυρο αυτό (προεπιλογή).

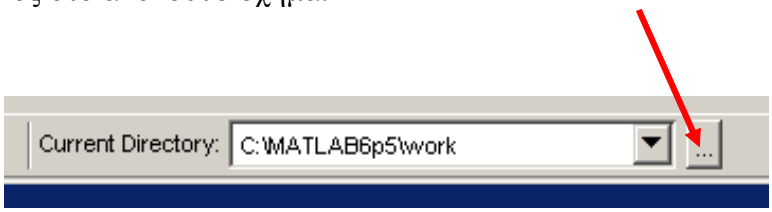
2) Ένα μικρό παράθυρο πάνω αριστερά που δείχνει τον **τρέχοντα φάκελο** (Current Directory) και τα αρχεία που εμφανίζονται σ' αυτόν. Αν το παράθυρο είναι κρυμμένο, επιλέξτε *Current Directory*.

3) Ένα παράθυρο που εναλλάσσεται με το παράθυρο τρέχοντα φακέλου ανάλογα με την επιλογή *Workspace* ή *Current Directory* είναι το παράθυρο του **χώρου εργασίας** (workspace). Αν το παράθυρο είναι κρυμμένο, επιλέξτε *Workspace*.

4) Ένα παράθυρο κάτω αριστερά που δείχνει το **ιστορικό εντολών** (Command History). Αν δεν εμφανιστεί το παράθυρο αυτό επιλέξτε *Command History* στην επιλογή *View*. Με τον ίδιο τρόπο μπορείτε να κλείσετε το παράθυρο αυτό.

Κατά τη διάρκεια μιας εργασίας στη MATLAB μπορεί να εμφανιστούν αυτόματα και άλλα παράθυρα όταν αυτό απαιτείται όπως **παράθυρα κειμένου** (document windows), **παράθυρα γραφικών** (graphics windows) και **παράθυρα σύνταξης αρχείων** (editing windows).

Σημειώνεται ότι μπορείτε να αλλάξετε τον τρέχοντα φάκελο επιλέγοντας τον προσωπικό σας φάκελο κάνοντας την κατάλληλη επιλογή στο τετραγωνίδιο που δείχνει το βέλος στο ακόλουθο σχήμα:



Προσοχή: Αν εργάζεστε στο Εργαστήριο του Τμήματος Μαθηματικών και Στατιστικής επιλέξτε ως τρέχοντα φάκελο τον

C:\Documents and Settings\youraccount ή τον **C:\temp**.

Η έξοδος από το πρόγραμμα μπορεί να γίνει με τους εξής τρόπους:

- με την εντολή **quit** ή την εντολή **exit** στο παράθυρο αντολών,
- με κλικ στο τετραγωνίδιο [x] που βρίσκεται πάνω δεξιά στο παράθυρο της MATLAB, και
- με την επιλογή *File* → *Exit MATLAB* στο παράθυρο εργασίας.

Στο κεφάλαιο αυτό θα δούμε πως μπορούμε να αποθηκεύσουμε την εργασία μας και τις μεταβλητές που υπολογίσαμε σε αρχείο (για να μπορούμε να τις χρησιμοποιήσουμε αργότερα).

Τα παραδείγματα που παραθέτουμε έχουν γίνει με την εκδοχή 6.5 της MATLAB. Με την εντολή **version** μπορούμε να μάθουμε την εκδοχή της MATLAB που χρησιμοποιούμε.

```
>> version
ans =
6.5.1.199709 (R13) Service Pack 1
```

Με την εντολή **ver** παίρνουμε περισσότερες πληροφορίες, όπως το **λειτουργικό σύστημα** (operating system), την εκδοχή του **μεταγλωττιστή** (compiler) και τις εκδοχές των **εργαλειοθηκών** (toolboxes) του πακέτου:

```
>> ver
-----------------------------------------------------------------------------
---------
MATLAB Version 6.5.1.199709 (R13) Service Pack 1
MATLAB License Number: 201749
Operating System: Microsoft Windows XP Version 5.1 (Build 2600: Service Pack
2)
Java VM Version: Java 1.3.1_01 with Sun Microsystems Inc. Java HotSpot(TM)
Client VM
-----------------------------------------------------------------------------
---------
MATLAB                                  Version 6.5.1      (R13SP1)
Financial Toolbox                       Version 2.3        (R13SP1)
MATLAB Compiler                         Version 3.0.1      (R13SP1)
Optimization Toolbox                    Version 2.3        (R13SP1)
Signal Processing Toolbox               Version 6.1        (R13SP1)
Statistics Toolbox                      Version 4.1        (R13SP1)
Symbolic Math Toolbox                   Version 3.0.1      (R13SP1)
```

1.1.1 Βασικές πράξεις

Το MATLAB μπορεί να χρησιμοποιηθεί σαν απλή αριθμομηχανή. Για τις βασικές πράξεις χρησιμοποιούνται τα σύμβολα που φαίνονται στον πιο κάτω πίνακα:

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

Παράδειγμα 1.1.1

```
>> 1+2

ans =
     3

>> 3.14567-3

ans =
    0.1457

>> 8/2.2

ans =
    3.6364

>> 3*4

ans =
    12

>> 3^4

ans =
    81
```

Επειδή τα αποτελέσματα γράφονται αραιά στη συνέχεια θα χρησιμοποιήσουμε την εντολή **format compact** για εξοικονόμηση χώρου. Την εντολή **format** θα τη συζητήσουμε αναλυτικά στην παράγραφο 1.5. Αν επαναλάβουμε λοιπόν το παράδειγμά μας σε format compact θα διαπιστώσουμε ότι δεν εμφανίζονται οι ενδιάμεσες κενές γραμμές στο παράθυρο εντολών:

```
>> format compact
>> 1+2
ans =
     3
>> 3.14567-3
ans =
    0.1457
>> 8/2.2
ans =
    3.6364
>> 3*4
ans =
    12
>> 3^4
ans =
    81
```

Ο επιστημονικός συμβολισμός

Εκτός από το **δεκαδικό συμβολισμό** (decimal notation) η MATLAB (όπως και μια αριθμομηχανή) χρησιμοποιεί το **λεγόμενο επιστημονικό συμβολισμό** (scientific notation). Ας πάρουμε σαν παράδειγμα τους (σε δεκαδική μορφή) αριθμούς

$$0.0001234 \text{ και } 4567.89$$

Στον επιστημονικό συμβολισμό γράφουμε τους πιο πάνω αριθμούς σαν

$$1.234\ 10^{-4} \text{ και } 4.56789\ 10^{3}.$$

Στη MATLAB γράφουμε τους δύο αριθμούς ως εξής:

1.234e-4 και **4.56789e3**

δηλ. χρησιμοποιούμε το γράμμα e για να δείξουμε ότι ακολουθεί ο εκθέτης του 10. Θα μπορούσαμε ακόμα να γράψουμε

$$0.1234\text{e-}3 \text{ και } 0.456789\text{e}4$$

Ο επιστημονικός συμβολισμός είναι ιδιαίτερα βολικός όταν χρησιμοποιούμε πολύ μεγάλους ή πολύ μικρούς αριθμούς. Για παράδειγμα, για τον αριθμό Avogadro είναι προτιμότερο να γράψουμε ότι αυτός είναι ίσος με

$$6.022\ 10^{23}$$

και όχι

602200000000000000000000!

Παράδειγμα 1.1.2

Θα υπολογίσουμε το άθροισμα των 12345.67 και 0.000012345 οι οποίοι σε επιστημονικό συμβολισμό γράφονται

$$1.234567\ 10^{4} \text{ και } 1.2345\ 10^{-5}$$

```
>> 1.234567e4+1.2345e-5
```

```
ans =
    1.2346e+004
```

Παρατηρούμε ότι δεν έχουμε επαρκή αριθμό δεκαδικών για να δούμε το πραγματικό αποτέλεσμα της πράξης. Για να δούμε περισσότερα δεκαδικά ψηφία χρησιμοποιούμε την εντολή **format long**:

```
>> format long
>> 1.234567e4+1.2345e-5
ans =
    1.234567001234500e+004
```

Είναι σημαντικό να μην υπάρχει κενό μεταξύ του δεκαδικού αριθμού και του e ή μεταξύ του e και του εκθέτη. Αν αφήσουμε κενό η MATLAB θα διαβάσει δύο αριθμούς (αντί ένα)!

Παράδειγμα 1.1.3

Θα δώσουμε τον αριθμό $1.2\ 10^{-5}$. Αφήνοντας κενό μεταξύ του δεκαδικού αριθμού και του e και μεταξύ του e και του εκθέτη παίρνουμε μήνυμα λάθους από τη MATLAB

```
>> 1.2 e-5
??? 1.2 e-5
        |
Error: Missing operator, comma, or semicolon.

>> 1.2e -5
??? 1.2e -5
     |
Error: "End of Input" expected, "incomplete floating-point
number" found.
```

Θα γράψουμε τώρα σωστά τον αριθμό:

```
>> 1.2e-5
ans =
  1.2000e-005
```

Προτεραιότητα πράξεων

Όπως και στις γλώσσες προγραμματισμού FORTRAN και C, η MATLAB ακολουθεί τους συνήθεις αλγεβρικούς κανόνες για την σειρά εκτέλεσης πράξεων:

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

Παράδειγμα 1.1.4

Ο σωστός τρόπος υπολογισμού της παράστασης $\dfrac{1.1^{2+3}-1}{3\cdot 2}$

είναι ο εξής:

```
>> (1.1^(2+3)-1)/(3*2)
ans =
    0.1018
```

Ας δούμε τώρα και μερικούς πιθανούς λάθος τρόπους:

```
>> (1.1^2+3-1)/(3*2)
ans =
    0.5350

>> (1.1^(2+3)-1)/3*2
ans =
    0.4070
```

1.1.2 Μεταβλητές

Η εκχώρηση τιμής σε μια μεταβλητή γίνεται με το σύμβολο =.

Παράδειγμα 1.1.5

```
>> x=1
x =
     1
>> y=2
y =
     2
>> w=z^y
w =
     9
```

Για τα ονόματα μεταβλητών χρησιμοποιούνται κυρίως γράμματα του αγγλικού αλφαβήτου. **Η MATLAB κάνει διάκριση μεταξύ κεφαλαίων και μικρών γραμμάτων**. Για παράδειγμα οι μεταβλητές *y* και *Y* είναι διαφορετικές μεταξύ τους. Για τα ονόματα μεταβλητών ισχύουν οι πιο κάτω κανόνες:

- *Το όνομα αρχίζει με γράμμα (του αγγλικού αλφαβήτου).*
- *Το όνομα περιέχει μόνο γράμματα, αριθμούς και υποπαύλες (underscore).*
- *Δεν χρησιμοποιούνται ονόματα που έχουν δεσμευτεί από τη MATLAB (π.χ. συναρτήσεις βιβλιοθήκης και εργαλειοθηκών).*
- *Προτιμούνται μικρά ονόματα για πρακτικούς λόγους αν και δεν υπάρχει περιορισμός στο μήκος των ονομάτων.*

Παράδειγμα 1.1.6

```
>> Y=20.2
Y =
   20.2000
>> y=2;
:>> Y+y
ans =
   22.2000
```

Παρατηρούμε ότι η MATLAB επιστρέφει μετά από κάθε εντολή το αποτέλεσμά της. Αν δεν θέλουμε να εμφανίζεται στο παράθυρο εργασίας το αποτέλεσμα μιας εντολής γράφουμε στο τέλος της εντολής το ερωτηματικό '**;**' (semicolon). Οποτεδήποτε θέλουμε να δούμε στο παράθυρο εργασίας την τιμή μιας ενεργής μεταβλητής, γράφουμε απλώς το όνομά της.

Παράδειγμα 1.1.7

```
>> x=1.234567

x =
    1.2346

>> y=2006;
>> z=x/y;

>> y
y =
        2006
>> z
z =
  6.1544e-004
```

Μπορούμε να γράψουμε περισσότερες από μια εντολές σε μια γραμμή τις οποίες χωρίζουμε είτε με κόμματα είτε με ερωτηματικά (αν δεν θέλουμε να τυπωθεί το αποτέλεσμα στο παράθυρο εντολών).

Παράδειγμα 1.1.7 (συνέχεια)

```
>> x=1; y=2, z=3; sum=x+y+z, w=x*y*z;
y =
     2
sum =
     6
```

Παρατηρούμε ότι στο παράθυρο εντολών τυπώθηκαν μόνο τα y και sum αφού μετά τις αντίστοιχες εντολές χρησιμοποιήσαμε κόμμα και όχι ερωτηματικό.

Θα έχετε ήδη προσέξει ότι όταν το αποτέλεσμα μιας εντολής δεν εκχωρείται σε μια μεταβλητή, τότε αυτό εκχωρείται στην προεπιλεγμένη μεταβλητή **ans**. Η μεταβλητή αυτή ανακυκλώνεται κάθε φορά που δίνουμε εντολή το αποτέλεσμα της οποίας δεν εκχωρείται σε κάποια άλλη μεταβλητή.

Παράδειγμα 1.1.7 (συνέχεια)

```
>> y-2000
ans =
     6

>> ans^2
ans =
    36

>> ans/10
ans =
    3.6000
```

Οι μεταβλητές στη MATLAB μπορεί να είναι όχι μόνο πραγματικές αλλά και μιγαδικές ή **αλφαριθμητικές**, δηλ. να έχουν ως τιμές **ακολουθίες χαρακτήρων** (strings)[1], ή ακόμα **λογικές** (logical), δηλ. να παίρνουν τις τιμές **true** (αληθής) και **false** (ψευδής).

[1] Ο αγγλικός όρος string μεταφράζεται στα ελληνικά ως **αλφαριθμητικό** (με κίνδυνο σύγχυσης με το alphanumeric) ή **συμβολοσειρά** ή **ακολουθία χαρακτήρων**.

Η MATLAB επεξεργάζεται με φυσικό τρόπο και μιγαδικούς αριθμούς. Αυτοί ορίζονται απλά ως εξής:

a+bi

όπου οι a και b είναι πραγματικοί αριθμοί και το *i* συμβολίζει τη φανταστική μονάδα: $i = \sqrt{-1}$. Ο συζυγής ενός μιγαδικού αριθμού **z** είναι ο **z'**.

Παράδειγμα 1.1.8

Έστω οι μιγαδικοί αριθμοί $z_1 = 1 + 4i$ και $z_2 = 2 - 3i$. Θα υπολογίσουμε τα εξής: $z_1 + z_2$, $z_1\, z_2$, z_2/z_1, $\overline{z}_1$, $\overline{z}_2$ και $\overline{(z_1 - z_2)}$.

```
>> z1=1+4i
z1 =
   1.0000 + 4.0000i
>> z2=2-3i
z2 =
   2.0000 - 3.0000i
>> z1+z2
ans =
   3.0000 + 1.0000i
>> z1*z2
ans =
  14.0000 + 5.0000i
>> z1'
ans =
   1.0000 - 4.0000i
>> z2'
ans =
   2.0000 + 3.0000i
>> (z1-z2)'
ans =
  -1.0000 - 7.0000i
```

Οι πίνακες και τα διανύσματα αποτελούν τις κύριες μεταβλητές της MATLAB όπως δηλώνεται και από το όνομά της. Μάλιστα στις αρχικές εκδοχές της MATLAB (μέχρι και την εκδοχή 3) όλες οι μεταβλητές ήταν πίνακες, αφού οι αριθμοί αντιμετωπίζονται σαν 1×1 πίνακες και τα διανύσματα σαν 1×*n* πίνακες. Οι πίνακες (και τα διανύσματα) θα μελετηθούν διεξοδικά στο Κεφάλαιο 2. Σε επόμενα κεφάλαια θα συζητήσουμε και τους νέους τύπους δεδομένων της MATLAB (που δεν είναι πίνακες).

Οι πίνακες στη MATLAB εισάγονται με βάση τους εξής κανόνες:

1. **Τα στοιχεία του πίνακα γράφονται ανάμεσα σε αγκύλες […..]. Μόνο στην περίπτωση 1×1 πινάκων, δηλ. μόνο στην περίπτωση αριθμών (!), οι αγκύλες είναι προαιρετικές.**
2. **Τα στοιχεία μιας γραμμής του πίνακα χωρίζονται είτε με κόμματα είτε με κενό.**
3. **Η αλλαγής γραμμής στον πίνακα δηλώνεται είτε με ερωτηματικό (;) είτε με αλλαγή γραμμής στο παράθυρο εντολών της MATLAB.**

Τα πιο πάνω θα φανούν πολύ πιο εύκολα αν δούμε τα παραδείγματα που ακολουθούν.

Παράδειγμα 1.1.9

Θα προσθέσουμε τα διανύσματα $u = (4, 0, -1, 2)$ και $v = (1, 2, -3, 1)$.

```
>> u=[4 0 -1 2];
>> v=[1, 2, -3, 1];
>> u+v
ans =
     5     2    -4     3
```

Προσέξτε ότι στο u χωρίσαμε τα στοιχεία με κενά ενώ στο v τα χωρίσαμε με κόμματα. Οι δύο τρόποι είναι ισοδύναμοι.

Παράδειγμα 1.1.10

Θα υπολογίσουμε την παράσταση $3A + 2B$ όπου A και B οι πίνακες

$$A = \begin{bmatrix} -5 & 4 & 0 \\ 1 & -3 & 6 \end{bmatrix} \quad \text{και} \quad B = \begin{bmatrix} 1 & 0 & -2 \\ 0 & 3 & 1 \end{bmatrix}$$

```
>> A=[-5 4 0
1 -3 6]
A =
    -5     4     0
     1    -3     6

>> B=[1 0 -2; 0 3 1]
B =
     1     0    -2
     0     3     1

>> 3*A+2*B
ans =
   -13    12    -4
     3    -3    20
```

Προσέξτε ότι στον A αλλάζαμε γραμμή σε κάθε νέα γραμμή του πίνακα ενώ στον B χωρίσαμε τις γραμμές με ερωτηματικό. Οι δύο τρόποι είναι ισοδύναμοι.

Παράδειγμα 1.1.11

Θα βρούμε τα γινόμενα uv και vu όπου

$$u = \begin{bmatrix} 1 & -1 & 4 \end{bmatrix} \quad \text{και} \quad v = \begin{bmatrix} -3 \\ 0 \\ 2 \end{bmatrix}$$

```
>> u=[1 -1 4];
>> v=[-3;0;2];
>> u*v
ans =
     5
>> v*u
ans =
    -3     3   -12
     0     0     0
     2    -2     8
```

Η MATLAB χρησιμοποιεί επίσης αλφαριθμητικές μεταβλητές οι οποίες έχουν σαν τιμές ακολουθίες χαρακτήρων οι οποίες εισάγονται ανάμεσα σε τόνους όπως φαίνεται και στο παράδειγμα που ακολουθεί.

Παράδειγμα 1.1.12

```
>> stringvar='Keimeno se 1 grammi'
stringvar =
Keimeno se 1 grammi

>> stringvar2='Lastname firstname'
stringvar2 =
Lastname firstname
```

Τέλος η MATLAB χρησιμοποιεί λογικές μεταβλητές. Στην τιμή true (αληθής) η MATLAB αντιστοιχεί τη μονάδα 1 ενώ στην τιμή false (ψευδής) αντιστοιχεί το 0. Έτσι οι παραστάσεις **true, logical(1)** και **logical(true)** μας δίνουν την τιμή 1, ενώ οι παραστάσεις **false, logical(0)** και **logical(false)** μας δίνουν την τιμή 0.

Παράδειγμα 1.1.13

```
>> x=true
x =
     1

>> y=false
y =
     0

>> logical(true)
ans =
     1

>> logical(1)
ans =
     1

>> logical(false)
ans =
     0

>> logical(0)
ans =
     0
```

1.2 Βαθμωτές συναρτήσεις βιβλιοθήκης

Η MATLAB είναι εφοδιασμένη με αρκετές **συναρτήσεις βιβλιοθήκης** οι οποίες περιλαμβάνουν τους λογαρίθμους, τις εκθετικές συναρτήσεις, τις τριγωνομετρικές, τις υπερβολικές συναρτήσεις, συναρτήσεις στατιστικής ανάλυσης και άλλες. Στον πίνακα φαίνονται μόνο μερικές από αυτές.

Συνάρτηση	Ερμηνεία
sin	ημίτονο
cos	συνημίτονο
tan	εφαπτομένη
asin	τόξο ημιτόνου
acos	τόξο συνημιτόνου
atan	τόξο εφαπτομένης
exp	εκθετική συνάρτηση
log	φυσικός λογάριθμος
log10	λογάριθμος με βάση το 10
abs	απόλυτη τιμή
sqrt	τετραγωνική ρίζα
mod	προσημασμένο υπόλοιπο διαίρεσης (modulus)
rem	υπόλοιπο διαίρεσης
round	στρογγύλευση στον πλησιέστερο ακέραιο
ceil	στρογγύλευση στον αμέσως μεγαλύτερο ακέραιο
floor	στρογγύλευση προς το μείον άπειρο
fix	στρογγύλευση προς το μηδέν

Αρκετά από τα ονόματα των συναρτήσεων βιβλιοθήκης στην MATLAB είναι τα ίδια μα αυτά που χρησιμοποιούνται στη γλώσσα C αλλά και τη FORTRAN, όπως sin, exp, sqrt, log κα. Με την εντολή

help elfun

η MATLAB μας δίνει τον κατάλογο των στοιχειωδών μαθηματικών συναρτήσεων (elementary math functions). Με την εντολή

help specfun

παίρνουμε κατάλογο των ειδικών μαθηματικών συναρτήσεων (specialized math functions) της MATLAB.

Παράδειγμα 1.2.1

Μπορούμε να βρούμε πληροφορίες για τις πιο πάνω συναρτήσεις με την εντολή **help**. Για παράδειγμα:

```
>> help rem

    REM    Remainder after division.
    REM(x,y) is x - n.*y where n = fix(x./y) if y ~= 0.  If y is not an
    integer and the quotient x./y is within roundoff error of an integer,
    then n is that integer.  By convention, REM(x,0) is NaN.  The input
    x and y must be real arrays of the same size, or real scalars.

    REM(x,y) has the same sign as x while MOD(x,y) has the same sign as y.
```

```
REM(x,y) and MOD(x,y) are equal if x and y have the same sign, but
differ by y if x and y have different signs.

See also MOD.
```

Σε αδρές γραμμές, η συνάρτηση **rem** μας δίνει το υπόλοιπο της διαίρεσης *y*/*x*.

```
>> rem(10,3)
ans =
     1
>> rem(11,4)
ans =
     3
>> rem(10,-3)
ans =
     1
>> rem(-11,4)
ans =
    -3
```

Όπως φαίνεται και στη βοήθεια πιο πάνω, οι εντολές **rem** και **mod** δίνουν τα ίδια αποτελέσματα όταν οι *x* και *y* είναι ομόσημοι. Τα αποτελέσματα διαφέρουν στην αντίθετη περίπτωση.

```
>> mod(10,3)
ans =
     1
>> mod(11,4)
ans =
     3
>> mod(10,-3)
ans =
    -2
>> mod(-11,4)
ans =
     1
```

Παράδειγμα 1.2.2

Η MATLAB έχει προεπιλέξει το όνομα **pi** για τον αριθμό π. Στο παράδειγμα αυτό θα χρησιμοποιήσουμε πρώτα την εντολή **format long** για να βλέπουμε 15 και όχι 4 σημαντικά ψηφία μετά την υποδιαστολή.

Θα βρούμε τώρα τα sin(π/3), cos(π/6) και tan(π/4):

```
>> format long

>> pi
ans =
   3.14159265358979

>> sin(pi/3)
ans =
   0.86602540378444

>> cos(pi/6)
ans =
   0.86602540378444

>> tan(pi/4)
ans =
   1.00000000000000
```

Παράδειγμα 1.2.3

Οι εντολές round, floor, ceil και fix στρογγυλεύουν ένα αριθμό όπως φαίνεται στον πίνακα που δώσαμε πιο πάνω. Ας δούμε τα αποτελέσματα που παίρνουμε για τους αριθμούς $x = 10.51$ και $y = -0.51$:

```
>> x=10.51
x =
   10.5100

>> round(x)
ans =
    11

>> floor(x)
ans =
    10

>> ceil(x)
ans =
    11

>> fix(x)
ans =
    10

>> y=-10.51
y =
  -10.5100

>> round(y)
ans =
   -11

>> floor(y)
ans =
   -11

>> ceil(y)
ans =
   -10

>> fix(y)
ans =
   -10
```

Παράδειγμα 1.2.4

```
>> cos(.5)^2+sin(.5)^2
ans =
     1

>> exp(1)
ans =
    2.7183

>> log(ans)
ans =
     1

>> cos(pi/2)
ans =
  6.1232e-017
```

Σημείωση: σχολιάστε το τελευταίο αποτέλεσμα.

1.3 Εντολές διαχείρισης του χώρου εργασίας

Στην παράγραφο αυτή θα συζητήσουμε χρήσιμες εντολές για τη διαχείριση του παραθύρου εργασίας και των ενεργών μεταβλητών που έχουμε δημιουργήσει. Αυτές φαίνονται στον πίνακα που ακολουθεί.

Εντολή	Ερμηνεία
exit, quit	**έξοδος από το πρόγραμμα**
clear	**διαγραφή ενεργών μεταβλητών**
clc	**καθαρισμός παραθύρου εργασίας**
diary	**αποθήκευση εργασίας σε αρχείο**
help	**βοήθεια**
who, whos	**κατάλογος ενεργών μεταβλητών εργασίας**
load	**φόρτωση από αρχείο των μεταβλητών εργασίας**
save	**αποθήκευση σε αρχείο των μεταβλητών εργασίας**

Αξίζει τον κόπο να θυμόμαστε τις εξής βασικές οδηγίες για το παράθυρο εντολών:

- *Στην MATLAB υπάρχει διάκριση μεταξύ μικρών και κεφαλαίων γραμμάτων (οι μεταβλητές A και a είναι διαφορετικές μεταξύ τους).*
- *Όταν γράψουμε το όνομα μιας μεταβλητής, η MATLAB τυπώνει στην οθόνη την τιμή της.*
- *Αν γράψουμε το σύμβολο ';' στο τέλος μιας εντολής, το αποτέλεσμά της δεν τυπώνεται στην οθόνη.*
- *Μπορούμε να γράψουμε μια ακολουθία εντολών της MATLAB σε μια γραμμή χωρίζοντάς τις με κόμματα ή ερωτηματικά.*
- *Πατώντας τα πλήκτρα με τα πάνω και κάτω βέλη ([↑] και [↓]) μπορούμε να διατρέξουμε όλες τις προηγούμενες εντολές. Επίσης μια προηγούμενη εντολή μπορεί να επαναληφθεί αν γράψουμε τα πρώτα γράμματα και μετά πατήσουμε το πλήκτρο με το πάνω βέλος* [↑].

1.3.1 Οι εντολές quit και exit

Με τις εντολές **quit** και **exit** τερματίζεται η τρέχουσα εργασία. Πολλές φορές είναι καλό να αποθηκεύσουμε την εργασία που κάναμε ή/και τις τιμές των μεταβλητών που δημιουργήσαμε στην εργασία. Το πώς γίνεται η αποθήκευση θα το συζητήσουμε στη συνέχεια.

1.3.2 Οι εντολές clear και clc

Η εντολή **clear var** διαγράφει τη μεταβλητή **var** του χώρου εργασίας. Η εντολή **clear** διαγράφει όλες τις ενεργές μεταβλητές. Το ίδιο κάνει και η εντολή **clear variables**.

Αν θέλουμε να διαγράψουμε μόνο τις μεταβλητές var1, var2 και var3 μπορούμε να γράψουμε

clear var1 var2 var3

(οι var1, var2 και var3 χωρίζονται με κενά και όχι με κόμματα). Αν θέλουμε να διαγράψουμε όλες τις μεταβλητές που αρχίζουν από Z γράφουμε

clear Z*

Αν τέλος το όνομα της προς διαγραφή μεταβλητής είναι αποθηκευμένο σε αλφαριθμητικό, για παράδειγμα τη name, μπορούμε να χρησιμοποιήσουμε τη συναρτησιακή μορφή της εντολής **clear(name)**.

Η εντολή **clc** καθαρίζει απλώς το παράθυρο εργασίας (δεν διαγράφονται οι μεταβλητές).

1.3.3 Η εντολή help

Έχουμε ήδη χρησιμοποιήσει την εντολή αυτή (βοήθεια). Η MATLAB είναι εφοδιασμένη με ένα εκτεταμένο επιγραμμικό (on line) σύστημα βοήθειας. Η εντολή **help topic** μας δίνει βοήθεια για το θέμα **topic**.

Παράδειγμα 1.3.1
Θα δούμε τη βοήθεια που παίρνουμε για την εντολή clc που είδαμε πιο πάνω.

```
>> help clc

 CLC    Clear command window.
 CLC clears the command window and homes the cursor.
```

Κάποιος θα μπορούσε να αρχίσει με την εντολή **help help** η οποία εξηγεί πως λειτουργεί το σύστημα βοήθειας και αναφέρει επίσης κάποιες σχετικές εντολές. Γράφοντας απλά **help** παίρνουμε ένα κατάλογο θεμάτων για τα οποία υπάρχει διαθέσιμη βοήθεια. Σ' αυτόν τον κατάλογο μπορούμε να βρούμε για παράδειγμα το θέμα «elfun - elementary math functions» (στοιχειώδεις μαθηματικές συναρτήσεις). Αν γράψουμε τώρα **help elfun** παίρνουμε τον κατάλογο των διαθέσιμων μαθηματικών συναρτήσεων. Για παράδειγμα, βλέπουμε ότι εκτός από το φυσικό λογάριθμο **log** και το λογάριθμο με βάση το 10, **log10,** υπάρχει και ο λογάριθμος με βάση το 2**, log2.** :

```
log         - Natural logarithm.
log10       - Common (base 10) logarithm.
log2        - Base 2 logarithm and dissect floating point number.
```

1.3.4 Οι εντολές who και whos

Η εντολή **who** μας δίνει απλώς κατάλογο των ενεργών μεταβλητών (χωρίς άλλες πληροφορίες). Η εντολή **whos** μαζί με τον κατάλογο μας δίνει πληροφορίες για όλες τις ενεργές μεταβλητές. Υπάρχει επίσης η δυνατότητα να χρησιμοποιήσουμε τις δύο εντολές για μεμονωμένες μεταβλητές, όπως φαίνεται πιο κάτω:

who var1 var2 var3: *κατάλογος των var1, var2 και var3*
who ab*: *κατάλογος των μεταβλητών με όνομα που αρχίζει από ab*
who *z: *κατάλογος των μεταβλητών με όνομα που λήγει σε z.*
who –file filename: *κατάλογος των μεταβλητών που είναι αποθηκευμένες στο αρχείο filename.mat*

Παράδειγμα 1.3.2

Θα εισαγάγουμε τις ακόλουθες μεταβλητές: xreal = π και zcomp = 2 – 3*i*, το πραγματικό διάνυσμα wvec = (3 0 2 –1), το μιγαδικό πίνακα

$$A=\begin{bmatrix} 2+i & i \\ 3 & 1-i \end{bmatrix}$$

το αλφαριθμητικό varstr = 'xreal' (είναι το όνομα της πρώτης μεταβλητής μας) και τη λογική μεταβλητή varlog = true.

```
>> xreal=pi;
>> zcomp=2 -3i;
>> wvec=[3 0 2 -1];
>> A=[2+i i; 3 1-i];
>> varstr='xreal';
>> varlog=true;
```

Ας δούμε τώρα τι παίρνουμε με τις εντολές who και whos:

```
>> who

Your variables are:

A       varlog  varstr  wvec    xreal   zcomp

>> whos

  Name         Size                Bytes  Class

  A            2x2                    64  double array (complex)
  varlog       1x1                     1  logical array
  varstr       1x5                    10  char array
  wvec         1x4                    32  double array
  xreal        1x1                     8  double array
  zcomp        1x1                    16  double array (complex)

Grand total is 16 elements using 131 bytes
```

Παρατηρούμε ότι όλες οι μεταβλητές αντιμετωπίζονται σαν πίνακες (arrays). Στην πρώτη στήλη έχουμε το όνομα κάθε μεταβλητής, στη δεύτερη τη διάσταση κάθε πίνακα (size), στην τρίτη το πλήθος των bytes και στην τέταρτη την κλάση (class) της μεταβλητής. Η προεπιλογή για τους αριθμούς είναι η διπλή ακρίβεια (double precision). Η MATLAB μας προσδιορίζει αν ένας πίνακας είναι μιγαδικός (complex) και κατά πόσο αυτός είναι λογικός (logical) ή αλφαριθμητικός (*πίνακας χαρακτήρων*, char array). Η διάσταση του κάθε πίνακα καθορίζει και τον τύπο της μεταβλητής. Έτσι ο αριθμός xreal είναι 1×1 πίνακας και το διάνυσμα wvec είναι 1×4 πίνακας.

Παράδειγμα 1.3.2 (συνέχεια)

Θα διαγράψουμε πρώτα τις μεταβλητές zcomp και wvec και θα δούμε τι παίρνουμε με την εντολή whos.

```
>> clear zcomp wvec
>> whos

  Name         Size                Bytes  Class

  A            2x2                    64  double array (complex)
  varlog       1x1                     1  logical array
  varstr       1x5                    10  char array
  xreal        1x1                     8  double array

Grand total is 11 elements using 83 bytes
```

Ας διαγράψουμε τώρα την μεταβλητή xreal όχι με την εντολή clear xreal αλλά με την ισοδύναμή της clear(varstr) αφού varst r = 'xreal':

```
>> clear(varstr)
>> whos

  Name         Size                    Bytes  Class

  A            2x2                        64  double array (complex)
  varlog       1x1                         1  logical array
  varstr       1x5                        10  char array

Grand total is 10 elements using 75 bytes
```

Ας διαγράψουμε τέλος τις varstr και varlog με την εντολή clear var*:

```
>> clear var*
>> whos

  Name      Size                    Bytes  Class

  A         2x2                        64  double array (complex)

Grand total is 4 elements using 64 bytes
```

Παράδειγμα 1.3.3

Ας θεωρήσουμε την περίπτωση που έχουμε 9 μεταβλητές όπως φαίνεται πιο κάτω:

```
>> who

Your variables are:

a11  a12  a13  a21  a22  a23  a31  a32  a33
```

Για να δούμε τις μεταβλητές a11, a22 και a33 γράφουμε:

```
>> who a11 a22 a33

Your variables are:

a11  a22  a33
```

Για να δούμε τις μεταβλητές με όνομα που αρχίζει από a2 γράφουμε:

```
>> who a2*

Your variables are:

a21  a22  a23
```

Για να δούμε τις μεταβλητές με όνομα που τελειώνει σε 3 γράφουμε:

```
>> who *3

Your variables are:

a13  a23  a33
```

Για να δούμε τις μεταβλητές με όνομα που περιέχει το 2 μετά το πρώτο γράμμα γράφουμε:

```
>> whos *2*
```

```
Name      Size                Bytes  Class
  a12       1x1                   8  double array
  a21       1x1                   8  double array
  a22       1x1                   8  double array
  a23       1x1                   8  double array
  a32       1x1                   8  double array

Grand total is 5 elements using 40 bytes
```

1.3.5 Οι εντολές save, load και diary

Η εντολή **save filename** αποθηκεύει όλες τις ενεργές μεταβλητές στο δυαδικό αρχείο **filename.mat**. Το αρχείο αυτό μπορούμε να το φορτώσουμε με την εντολή **load filename** (χωρίς το επίθεμα .mat) και να συνεχίσουμε την εργασία μας από το σημείο που διακόψαμε. Άλλες επιλογές που έχουμε με τις δύο εντολές φαίνονται πιο κάτω:

save: *αποθήκευση όλων των μεταβλητών στο αρχείο matlab.mat*
load: *φόρτωση όλων των μεταβλητών από το αρχείο matlab.mat*
save filename x y z: *αποθήκευση στο αρχείο filename.mat μόνο των μεταβλητών x, y και z*
load filename x y z: *φόρτωση από το αρχείο filename.mat μόνο των μεταβλητών x, y και z*
save filename A*: *αποθήκευση στο αρχείο filename.mat μόνο των μεταβλητών με όνομα που αρχίζει από A**
load filename A*: *φόρτωση από το αρχείο filename.mat μόνο των μεταβλητών με όνομα που αρχίζει από A**
save filename -ascii: *αποθήκευση όλων των μεταβλητών στο αρχείο filename σε μορφή ASCII με 8 σημαντικά ψηφία*
save filename –ascii -double: *αποθήκευση όλων των μεταβλητών στο αρχείο filename σε μορφή ASCII με 16 σημαντικά ψηφία*
save filename x y z–ascii : *αποθήκευση μόνο των μεταβλητών x, y και z στο αρχείο filename σε μορφή ASCII με 8 σημαντικά ψηφία*

Αν θέλουμε να δημιουργήσουμε ένα ASCII αρχείο που να περιέχει όλες τις πληροφορίες της εργασίας μας δίνουμε την εντολή **diary filename.txt** στην αρχή και το τέλος της εργασίας μας. Το αρχείο **filename.txt** περιέχει όλες τις εντολές μας και τα αποτελέσματα που εμφανίζονται στο παράθυρο εργασίας (εκτός από τα γραφικά) και μπορούμε να το τυπώσουμε. Αν παραλείψουμε το όνομα του αρχείου τότε η MATLAB δημιουργεί αρχείο με το όνομα diary. Άλλες επιλογές φαίνονται πιο κάτω:

diary off: *προσωρινή διακοπή του ημερολογίου*
diary on: *επανέναρξη του ημερολογίου*
diary(stringfname): *έναρξη του ημερολογίου και αποθήκευση του σε αρχείου το όνομα του οποίου έχει εκχωρηθεί στο αλφαριθμητικό stringfname*

Το ημερολόγιο **filename.txt** θα το βρείτε στο **φάκελο εργασίας** (Working directory).

Προσοχή: Αν εργάζεστε στο Εργαστήριο του Τμήματος Μαθηματικών και Στατιστικής και το σύστημα δεν δέχεται την εντολή **diary** επιλέξτε ως τρέχοντα φάκελο τον **C:\Documents and Settings\youraccount** ή τον **C:\temp**.

1.4 Άλλες χρήσιμες εντολές

Θα συζητήσουμε εντολές που έχουν να κάνουν με τα κείμενα βοήθειας που διαθέτει η MATLAB, τη διαχείριση του φακέλου εργασίας και την ημερομηνία και την ώρα.

Εντολή	Ερμηνεία
demo	**επιδείξεις (demonstrations)**
doc	**πρόσβαση στα html κείμενα του φυλλομετρητή βοήθειας (help browser)**
dir	**κατάλογος των αρχείων του φακέλου εργασίας (για windows)**
ls	**κατάλογος των αρχείων του φακέλου εργασίας (για unix)**
what	**κατάλογος των αρχείων MATLAB στον τρέχοντα φάκελλο ομαδοποιημένων σύμφωνα με τον τύπο τους**
tic, toc	**έναρξη και διακοπή χρονομέτρου**
date	**ημερομηνία**
clock	**ώρα**

1.4.1 Οι εντολές demo και doc

Με την εντολή **demo** μπορούμε να φυλλομετρήσουμε τα αρχεία επίδειξης της MATLAB. Η εντολή **doc** μας οδηγεί στα html κείμενα του φυλλομετρητή βοήθειας (help browser). Σημειώνεται ότι εναλλακτικά μπορούμε να κάνουμε κλικ στην επιλογή help του παραθύρου της MATLAB και να προχωρήσουμε ανάλογα.

Δεν θα αναφέρουμε περισσότερα για τις εντολές αυτές. Ο χρήστης της MATLAB θα πρέπει να τις δοκιμάσει στην πράξη για να διαπιστώσει ότι το πακέτο παρέχει ένα τεράστιο όγκο βοηθητικού υλικού.

1.4.2 Οι εντολές dir, ls και what

Η εντολή **dir** μας δίνει κατάλογο των αρχείων που περιέχονται στον φάκελο εργασίας. Το ίδιο κάνει και η εντολή **ls** που αντιστοιχεί στο σύστημα unix. Η εντολή **what** μας δίνει κατάλογο μόνο των αρχείων της MATLAB. Ισχύουν τα γνωστά για την επιλογή μόνο κάποιων αρχείων. Για παράδειγμα, με την εντολή **dir A*** παίρνουμε *κατάλογο των αρχείων με όνομα που αρχίζει από A*.

Παράδειγμα 1.4.1
Στο παράδειγμα που ακολουθεί βλέπουμε ότι οι dir και ls μας δίνουν τον κατάλογο όλων των αρχείων που περιέχονται στο φάκελο εργασίας. Η εντολή what μας δίνει ξεχωριστά τα αρχεία της MATLAB με επίθεμα m (m-files) και mat.

```
>> dir

.               class1.txt      mandelbrot.m    whoami.m
..              diar1           mandelbrot.txt
cdiary.mat      inputexample.m  whoami.asv

>> ls

.               class1.txt      mandelbrot.m    whoami.m
```

```
..              diar1           mandelbrot.txt
cdiary.mat      inputexample.m  whoami.asv

>> what

M-files   in   the   current   directory   C:\Documents   and
Settings\T42\My Documents\gg\MATLAB\work

inputexample    mandelbrot      whoami

MAT-files   in   the   current   directory   C:\Documents   and
Settings\T42\My Documents\gg\MATLAB\work

cdiary
```

1.4.3 Εντολές ημερομηνίας και ώρας

Με τις εντολές **tic** (έναρξη) και **toc** (λήξη) μπορούμε να χρονομετρήσουμε το χρόνο της εργασίας που κάνουμε στην MATLAB. Έτσι με τις εντολές

tic, Αεντολή, toc

μπορούμε να χρονομετρήσουμε το χρόνο που απαιτεί η *Αεντολή*.

Παράδειγμα 1.4.2
Θα βρούμε το χρόνο που απαιτεί η αντιστροφή ενός δοσμένου πίνακα Α. Για το σκοπό αυτό θα χρησιμοποιήσουμε τη συνάρτηση **inv(A**).

```
>> tic, inv(A); toc
elapsed_time =
    0.0100
>> tic
```

Η εντολή **date** μας δίνει την ημερομηνία σαν αλφαριθμητικό στη μορφή *dd-mmm-yyyy*. Η ημερομηνία όταν γράφονταν αυτές οι γραμμές ήταν η 6η Ιανουαρίου 2007:

```
>> date
ans =
06-Jan-2007
```

Τέλος η εντολή **clock** μας δίνει τόσο την ημερομηνία όσο και την ώρα σε ένα διάνυσμα της μορφής

CLOCK = [year month day hour minute seconds]

Παράδειγμα 1.4.3
Όταν γράφονταν αυτές οι γραμμές πήραμε

```
>> clock
ans =
  1.0e+003 *
    2.0070    0.0010    0.0060    0.0110    0.0390    0.0225
```

Επειδή τα στοιχεία του διανύσματος πολλαπλασιάζονται με 1.0e+003 το διάνυσμα ans δεν διαβάζεται εύκολα. Με τη συνάρτηση **fix** μπορούμε να στρογγυλεύσουμε τα στοιχεία του διανύσματος ans σε ακεραίους:

```
>> fix(ans)
ans =
     2007           1           6          11          39          22
```

1.5 Είσοδος και έξοδος δεδομένων

Στην παράγραφο αυτή θα συζητήσουμε τρείς εντολές που έχουν να κάνουν με την απεικόνιση των μεταβλητών και των τιμών τους στην οθόνη, δηλαδή στο παράθυρο εντολών (Command window) της MATLAB. Αυτές φαίνονται στον πιο κάτω πίνακα:

Εντολή	Ερμηνεία
disp	**απεικόνιση μεταβλητών στην οθόνη**
format	**μορφή εκτύπωσης μεταβλητών στην οθόνη**
input	**υποβολέας για είσοδο δεδομένων**

Υπάρχουν δύο πολύ σημαντικές εντολές εξόδου, η **fprintf** (εκτύπωση σε αρχείο) και η **sprintf** (εκτύπωση σε αλφαριθμητικό) στις οποίες θα αφιερώσουμε επόμενο κεφάλαιο.

1.5.1 Η εντολή disp

Γνωρίζουμε ήδη ότι αν θέλουμε να δούμε την τιμή μιας μεταβλητής **var** στην οθόνη αρκεί να γράψουμε το όνομα της. Εμφανίζεται τότε στην οθόνη σε μια γραμμή

var =

και μετά ακολουθεί η τιμή της var που μπορεί να είναι αριθμός ή πίνακας ή αλφαριθμητικό.

Η εντολή **disp(var)** απεικονίζει την τιμή της μεταβλητής var (χωρίς το όνομα) στην οθόνη ενώ η εντολή **disp var** απεικονίζει το **όνομα** της μεταβλητής (χωρίς την τιμή).

Επίσης η εντολή **disp('…….')** τυπώνει στην οθόνη το αλφαριθμητικό που περιέχεται μεταξύ των τόνων.

Παράδειγμα 1.5.1

Θεωρούμε τον πίνακα

$$A=\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$$

```
>> A=[1 2; 3 4];
>> A
A =
     1     2
     3     4
>> disp(A)
     1     2
     3     4
>> disp A
A
```

Παράδειγμα 1.5.2

Θεωρούμε εδώ το αλφαριθμητικό **x = 'my name':**

```
>> x='my name';
```

```
>> x
x =
my name
>> disp(x)
my name
>> disp x
x
```

1.5.2 Η εντολή format

Έχουμε ήδη συναντήσει την εντολή **format** την οποία χρησιμοποιήσαμε στα προηγούμενα παραδείγματα για εξοικονόμηση χώρου. Η προεπιλογή της MATLAB στην απεικόνιση των αποτελεσμάτων στο παράθυρο εντολών είναι να αφήνει κενές ενδιάμεσες γραμμές, έχουμε δηλαδή αραιή απεικόνιση. Με την εντολή **format compact** παίρνουμε πυκνή απεικόνιση αφού οι ενδιάμεσες κενές γραμμές παραλείπονται. Με την εντολή **format loose** επανερχόμαστε στην αραιή απεικόνιση.

Παράδειγμα 1.5.3

Θα υπολογίσουμε το άθροισμα των διανυσμάτων $u = (-1 , 2, 0)$ και $v = (1, -4, -5)$ πρώτα σε αραιή (προεπιλογή ή format loose) και μετά σε πυκνή απεικόνιση (format compact):

```
>> u=[-1, 2, 0];
>> v=[1, -4, 5];

>> u+v

ans =
     0    -2     5

>> format compact

>> u+v

ans =
     0    -2     5

>> format loose

>> u+v

ans =
     0    -2     5
```

Η εντολή **format** καθορίζει επίσης τον τρόπο με τον οποίο απεικονίζονται οι πραγματικοί αριθμοί καθώς και το πλήθος των σημαντικών ψηφίων που θα εμφανιστούν στο παράθυρο εντολών.

Όλες οι πράξεις στη MATLAB γίνονται **με διπλή ακρίβεια** (double precision). Για την εκτύπωση ενός αριθμού στην οθόνη η προεπιλογή της MATLAB είναι αυτή της **σταθερής υποδιαστολής με 4 σημαντικά ψηφία** μετά την υποδιαστολή. Αν ένας μη ακέραιος αριθμός είναι πολύ μεγάλος τότε η MATLAB χρησιμοποιεί τον εκθετικό συμβολισμό, δηλ. ο αριθμός εμφανίζεται σε **μορφή κινητής υποδιαστολής, πάλι με 4 σημαντικά ψηφία** μετά την υποδιαστολή. Αυτό φαίνεται καθαρά στο παράδειγμα που ακολουθεί.

Παράδειγμα 1.5.4

```
>> x=0.123456789
x =
    0.1235
>> y=12345678.9
y =
  1.2346e+007
```

Η προεπιλογή της MATLAB είναι να εμφανίζει 4 σημαντικά ψηφία μετά την υποδιαστολή. Με την εντολή **format long** η MATLAB μας δίνει 15 σημαντικά ψηφία. Με την εντολή **format short** ή απλώς **format** μπορούμε να επανέλθουμε στην προεπιλογή των 4 σημαντικών ψηφίων μετά την υποδιαστολή.

Παράδειγμα 1.5.5

```
>> x=1234567.89
x =
  1.2346e+006

>> y=0.000000123456789
y =
  1.2346e-007

>> z=x+y
z =
  1.2346e+006

>> format long

>> z
z =
    1.234567890000123e+006

>> x=1
x =
     1

>> epsilon=1.e-20
epsilon =
    1.000000000000000e-020

>> x+epsilon
ans =
     1
```

Σχολιάστε το τελευταίο αποτέλεσμα.

Παράδειγμα 1.5.6

```
>> 1+1e-15-1
ans =
  1.1102e-015
```

Σχολιάστε το τελευταίο αποτέλεσμα.

Με την εντολή **help format** μπορούμε να δούμε όλες τις επιλογές που μας παρέχει η MATLAB. Για ευκολία, αυτές φαίνονται και στον πίνακα που ακολουθεί.

Οι επιλογές της εντολής format.

Εντολή	Ερμηνεία
format	**Προεπιλογή της MATLAB. Ισοδύναμη με το format short**
format short	**Σταθερή υποδιαστολή με 4 σημαντικά ψηφία μετά την υποδιαστολή**
format long	**Σταθερή υποδιαστολή με 15 σημαντικά ψηφία μετά την υποδιαστολή**
format short e	**Κινητή υποδιαστολή με 4 σημαντικά ψηφία μετά την υποδιαστολή (επιστημονικός συμβολισμός)**
format long e	**Κινητή υποδιαστολή με 15 σημαντικά ψηφία μετά την υποδιαστολή (επιστημονικός συμβολισμός)**
format short g	**Η προτιμότερη από τις επιλογές της σταθερής και κινητής υποδιαστολής με 4 σημαντικά ψηφία μετά την υποδιαστολή**
format long g	**Η προτιμότερη από τις επιλογές της σταθερής και κινητής υποδιαστολής με 15 σημαντικά ψηφία μετά την υποδιαστολή**
format hex	**Δεκαεξαδική μορφή**
format +	**Εκτύπωση του προσήμου (+ ή –)**
format bank	**Τραπεζικός συμβολισμός (μόνο 2 δεκαδικά ψηφία για τα σεντ)**
format rat	**Προσέγγιση αριθμών με κλάσματα μικρών ακέραιων**
format compact	**Πυκνή απεινόνιση χωρίς ενδιάμεσες κενές γραμμές**
format loose	**Αραιή απεικόνιση με ενδιάμεσες κενές γραμμές**

Παράδειγμα 1.5.7

Στον πιο κάτω πίνακα φαίνονται οι επιλογές της εντολής format και πως τυπώνεται σε κάθε περίπτωση ο αριθμός π^{π}.

Ο αριθμός pi^pi σε διάφορα format.

format short	36.4622
format long	36.46215960720790
format short e	3.6462e+001
format long e	3.646215960720790e+001
format short g	36.462
format long g	36.4621596072079
format hex	40423b280bc73ebd
format bank	36.46
format rat	4339/119

1.5.3 Η εντολή input

Η εντολή input έχει τη γενική μορφή

R=input('prompt')

όπου **prompt** ένα αλφαριθμητικό. Με την εντολή αυτή εμφανίζεται στην οθόνη η προτροπή prompt και το σύστημα αναμένει από τον χρήστη να εισαγάγει την τιμή της μεταβλητής R που μπορεί να είναι αριθμός ή αλφαριθμητικό ή διάνυσμα ή πίνακας ή ακόμα το αποτέλεσμα μιας ολόκληρης παράστασης σε γλώσσα MATLAB. Αν η R είναι διάνυσμα ή πίνακας τότε τα στοιχεία του εισάγονται κατά τα γνωστά μέσα σε αγκύλες. Αν η R είναι αλφαριθμητικό τότε η τιμή της πρέπει να δοθεί μέσα σε τόνους ' …….. '. Ένας τρόπος να αποφύγουμε τους τόνους είναι να δηλώσουμε μέσω της εντολής ότι η μεταβλητή είναι αλφαριθμητικό. Αυτό γίνεται ως εξής:

R=input('prompt', 's')

Παράδειγμα 1.5.8

```
>> R=input('Enter variable: ')
Enter variable: -10.234
R =
  -10.2340

>> R=input('Enter variable: ')
Enter variable: 3/7
R =
    0.4286

>> R=input('Enter complex variable: ')
Enter complex variable: -3 +2i
R =
  -3.0000 + 2.0000i

>> u=input('Enter an 1x4 vector: ')
Enter an 1x4 vector: [1 -2 0 5]
u =
     1    -2     0     5

>> B=input('Enter a 2x3 array: ')
Enter a 2x3 array: [ -1 2 0
4 -5 1]
B =
    -1     2     0
     4    -5     1

>> S=input('Enter a string: ')
Enter a string: 'My name'
S =
My name
```

Παράδειγμα 1.5.9

```
>> Lastname=input('Enter lastname: ','s')
Enter lastname: Cyprianou
Lastname =
Cyprianou

>> Firstname=input('Enter firstname: ','s')
Enter firstname: Andreas
Firstname =
Andreas
```

```
>> disp(Firstname)
Andreas
```

Έχουμε ήδη δεί ότι η προτροπή prompt μπορεί να αποτελείται από διάφορες λέξεις. Η input μας δίνει την επιλογή να αλλάξουμε και γραμμή χρησιμοποιώντας το σύμβολο **\n**. Αν θέλουμε να γράψουμε μια πρόταση σε 3 γραμμές τότε η εντολή μας θα πρέπει να είναι της μορφής:

R=input('line 1 \n line 2 \n line3')

Παράδειγμα 1.5.10

```
>> X=input('Enter \n Full name: ', 's')
Enter
 Full name: Matlabiou Matlabios
X =
Matlabiou Matlabios
```

1.6 Ειδικές σταθερές και μεταβλητές

Στη παράγραφο αυτή συζητούμε σε συντομία σταθερές ή μεταβλητές που έχουν προεπιλεχθεί από τη MATLAB. Αυτές φαίνονται στον πιο κάτω πίνακα. Τις περισσότερες τις έχουμε ήδη συναντήσει στα προηγούμενα παραδείγματα.

Μεταβλητή ή σταθερά	Ερμηνεία
ans	**η πιο πρόσφατη τιμή**
eps	**σχετική ακρίβεια κινητής υποδιαστολής**
i, j	**φανταστική μονάδα ($i=\sqrt{-1}$)**
Inf, inf	∞
NaN, nan	**μη αριθμός (not a number)**
pi	**π**

Η μεταβλητή **ans** (από τη λέξη ANSwer) περιέχει την πιο πρόσφατη απάντηση της MATLAB (μπορεί δηλ. να είναι αριθμός ή πίνακας ή αλφαριθμητικό κοκ), όταν αυτή δεν έχει εκχωρηθεί σε κάποια άλλη μεταβλητή.

Παράδειγμα 1.6.1

```
>> sin(1.5)+cos(2.5)
ans =
    0.1964
>> [ 1 2 3 ]+ [ -2 0 -3]
ans =
    -1     2     0
```

Η σταθερά **eps** είναι η σχετική ακρίβεια κινητής υποδιαστολής της MATLAB, δηλ. η απόσταση του 1 από τον αμέσως επόμενο αριθμό που διακρίνει η MATLAB. Η eps είναι η προεπιλεγμένη **ανοχή** (tolerance) για διάφορα προγράμματα της MATLAB. Μπορούμε να πούμε ότι ένας αριθμός μικρότερος κατ'απόλυτη τιμή από την eps είναι για τη MATLAB ίσος με μηδέν.

Παράδειγμα 1.6.2

```
>> format long g
>> eps
ans =
     2.22044604925031e-016
>> 1+eps
ans =
                         1
```

Τα σύμβολα **i** και **j** συμβολίζουν και τα δύο τη φανταστική μονάδα και χρησιμοποιούνται για την εισαγωγή μιγαδικών αριθμών. Για παράδειγμα, ο αριθμός **2-5i** μπορεί να γραφεί με τους εξής τρόπους:

2-5i ή 2-5*i ή 2-5*sqrt(-1) ή 2-5j ή 2-5*j

Τα σύμβολα i και j μπορούν να αξιοποιηθούν από τον χρήστη για άλλους σκοπούς (π.χ. σε βρόχους ή για απαρίθμηση). Σ' αυτή την περίπτωση η παράσταση **2-5i** είναι προτιμότερη από την **2-5*i** όπως φαίνεται και στο πιο κάτω παράδειγμα.

Παράδειγμα 1.6.3

```
>> i
ans =
        0 + 1.0000i
>> j
ans =
        0 + 1.0000i
>> z=5-2i
z =
   5.0000 - 2.0000i
>> w=4+3*j
w =
   4.0000 + 3.0000i
>>
>> w+i
ans =
   4.0000 + 4.0000i
>> i=1.13
i =
    1.1300
>> w+i
ans =
   5.1300 + 3.0000i
>> 1+2i
ans =
   1.0000 + 2.0000i
>> 1+2*i
ans =
    3.2600
```

Το σύμβολο **Inf** (από τη λέξη infinity) έχει δεσμευτεί για να δηλώνει το $+\infty$ σύμφωνα με την αριθμητική παράσταση της IEEE. Αυτό μπορεί να προκύψει με διάφορους τρόπους όπως 1/0, exp(10000) κα. Το σύμβολο **inf** είναι ισοδύναμο με το **Inf**.

Το σύμβολο **NaN** παριστάνει ένα **μη αριθμό** (not a number) σύμφωνα με τη σύμβαση της IEEE. Αυτός μπορεί να προκύψει από πράξεις όπως οι 0/0 και inf – inf. Το σύμβολο **nan** είναι ισοδύναμο με το **NaN**.

Παράδειγμα 1.6.4

```
>> 1/0
Warning: Divide by zero.
(Type "warning off MATLAB:divideByZero" to suppress this
warning.)
ans =
   Inf

>> 1/-0
Warning: Divide by zero.
(Type "warning off MATLAB:divideByZero" to suppress this
warning.)
ans =
  -Inf

>> log(0)
Warning: Log of zero.
ans =
  -Inf
```

```
>> 0/0
Warning: Divide by zero.
(Type "warning off MATLAB:divideByZero" to suppress this
warning.)
ans =
   NaN

>> Inf-Inf
ans =
   NaN
```

Τέλος το **pi** συμβολίζει τον αριθμό π = 3.1415926535897…….

Παράδειγμα 1.6.5

```
>> format long

>> pi
ans =
   3.14159265358979

>> cos(pi)
ans =
    -1

>> tan(pi/6)
ans =
   0.57735026918963

>> cos(pi/2)
ans =
  6.1232e-017
```

Παρατηρούμε ότι η MATLAB δεν δίνει ακριβώς μηδέν για το cos(π/2). Η τιμή που παίρνουμε είναι πάντως μικρότερη από τη σταθερά eps.

1.7 Ασκήσεις

1.1 Επαναλάβετε στη MATLAB όλα τα παραδείγματα της παραγράφου 1.1 και αποθηκεύστε την εργασία σας σε αρχείο με το όνομά σας.
Υπόδειξη: χρησιμοποιήστε την εντολή **diary onomasas.txt** στην αρχή και στο τέλος της εργασίας σας.

1.2 Να υπολογιστούν τα εξής στη MATLAB:

(α) 1000/11, (β) $12345 \cdot 6789$, (γ) $(1.13\ 10^{-3} + 17.24\ 10^{-7})/24$

1.3 Αν x ο αριθμός της ταυτότητάς σας, $y = x/10^5$ και $z = 125$ να υπολογιστούν τα εξής: $(x + y)/100$, y^6, $(x + y)/z$, xyz.

1.4 Έστω οι μιγαδικοί αριθμοί $z = 1 - 3i$, $u = 2 + 5i$ και $w = 4 + i$. Να υπολογιστούν τα εξής: $z + w$, zwu, $z/(u + w)$, ο συζυγής του z και ο συζυγής του $(z - w)$.

1.5 Ορίστε στη MATLAB τα ακόλουθα αλφαριθμητικά (strings):

fname:	το όνομά σας
lname:	το επίθετό σας
tmima:	το Τμήμα σας
apt:	αριθμός ταυτότητας

1.6 (α) Ποια τιμή έχει η x αν x = false;
(β) Ποια τιμή έχει η true*false;
(γ) Ποια τιμή μας δίνει η acos(-true);

1.7 Αν $x = 1.2345$, $y = 0.0002006$ και z ο αριθμός της ταυτότητάς σας υπολογίστε τα εξής: $w = x^2 - 2x + 5$, sum $= x + y + z$, adiff $= |x - y|$, $\sin(x^3 - z/100000)$, $\log(z - 100x)$, $s = \exp(x^3)$ και

$$t = \frac{\sqrt{z^2 - 100} - x / y}{\ln(x^2 + y^2)} e^x$$

Αποθηκεύστε την εργασία σας σε αρχείο με το όνομα **epitheto_onoma.txt.**

1.8 Επαναλάβετε στη MATLAB όλα τα παραδείγματα της παραγράφου 1.2 και αποθηκεύστε την εργασία σας σε αρχείο με το όνομά σας.

1.9 Βρείτε το υπόλοιπο της διαίρεσης $x/13$, όπου x ο αριθμός της ταυτότητάς σας.

1.10 Με την εντολή **help specfun** παίρνουμε τη βοήθεια για τις **ειδικές μαθηματικές συναρτήσεις** (specialized math functions) της MATLAB. Βρείτε τις τρεις από αυτές που συμβολίζονται με ελληνικά γράμματα, δώστε σύντομη περιγραφή τους και υπολογίστε με τη MATLAB τις τιμές τους για $x = \pi$.
Υπόδειξη: Όταν ολοκληρώσετε την εργασία αυτή δοκιμάστε τις εντολές: **load handel**, **sound(y,Fs)**.

1.11 Να υπολογίσετε τις $J_0(0)$, $J_0(e)$, $J_1(e)$ και $J_1(\pi)$ όπου J_0 και J_1 οι γνωστές μας **συναρτήσεις Bessel**.

1.12 Επαναλάβετε στη MATLAB όλα τα παραδείγματα της παραγράφου 1.3 και αποθηκεύστε την εργασία σας σε αρχείο με το όνομά σας.

1.13 Να συμπληρωθούν οι εντολές MATLAB που λείπουν:

```
>> A=[ 1+2i ; -3+4i];
```

```
>> Εντολή 1
A
>> Εντολή 2
   1.0000 + 2.0000i
  -3.0000 + 4.0000i

>> Εντολή 3
A =
   1.0000 + 2.0000i
  -3.0000 + 4.0000i
```

1.14 Να συμπληρωθούν οι εντολές MATLAB που λείπουν:

```
>> who

Your variables are:

X11  X12  X13  X21  X22  X23  Xa   Ya

>> Εντολή 1

Your variables are:

X11  X12  X13  X21  X22  X23  Xa

>> Εντολή 2

Your variables are:

X11  X12  X13

>> Εντολή 3

Your variables are:

Xa  Ya
```

1.15 Δίνεται ότι με την εντολή whos σε κάποιο παράθυρο εργασίας η MATLAB τυπώνει τα εξής:

```
>> whos

  Name       Size                    Bytes  Class

  A          3x3                        72  double array
  B             3x3                         144  double array
(complex)
  name       1x8                        16  char array
  newn       1x8                        16  char array
  u             5x1                          80  double array
(complex)
  uu       100x100                  80000  double array
  v          1x2                        2  logical array
  vec        1x6                       48  double array
  x1         1x1                        8  double array
  x2         1x1                        8  double array
  x3         1x1                        8  double array

Grand total is 10050 elements using 80402 bytes
```

(α) Πόσες είναι οι ενεργές μεταβλητές στο χώρο εργασίας;

(β) Ποιες μεταβλητές είναι πίνακες;
(γ) Ποιες μεταβλητές είναι διανύσματα;
(δ) Ποιες μεταβλητές είναι μιγαδικές;
(ε) Ποιες μεταβλητές είναι αλφαριθμητικά;
(στ) Ποιες μεταβλητές είναι λογικές;

1.16 Να συμπληρωθούν οι εντολές MATLAB που λείπουν:

```
>> who

Your variables are:

name  u     uu    v     vec   x1    x2    xint  yint  zcat

>> Εντολή 1
>> who

Your variables are:

name  u     uu    v     vec   yint  zcat

>> Εντολή 2
>> who

Your variables are:

name  u     uu    v     vec
```

1.17 Να συμπληρωθεί η εντολή MATLAB που λείπει:

```
>> who

Your variables are:

A   B   ss  x   y   z

>> save svfile A ss x y z
>> clear
>> who
>> Εντολή 1
>> who

Your variables are:

x  y  z
```

1.18 Επαναλάβετε στη MATLAB όλα τα παραδείγματα της παραγράφου 1.4 και αποθηκεύστε την εργασία σας σε αρχείο με το όνομά σας.

1.19 Επαναλάβετε στη MATLAB όλα τα παραδείγματα της παραγράφου 1.5 και αποθηκεύστε την εργασία σας σε αρχείο με το όνομά σας.

1.20 Με δεδομένο ότι η τιμή του x δεν αλλάζει, συμπληρώστε τις εντολές MATLAB που λείπουν:

```
>> x=log(1.11e+06)
x =
   13.9199
```

```
>> Εντολή 1
>> x
x =
  13.91987057328852

>> Εντολή 2
>> x
x =
    1.391987057328852e+001

>> Εντολή 3
>> x
x =
         13.92

>> Εντολή 4
>> x
x =
  4343/312
```

1.21 Βρείτε με ποια κλάσματα προσεγγίζει η MATLAB τους αριθμούς π και e σε format rat και υπολογίστε τα σφάλματα προσέγγισης σε format long e.

1.22 Υπολογίστε τον αριθμό 10^{e}, τυπώστε τον σε διάφορα format και συμπληρώστε τον πίνακα:

Ο αριθμός 10^exp(1) σε διάφορα format.

format short **format long** **format short e** **format long e** **format short g** **format long g** **format hex** **format bank** **format rat**	

1.23 Να συμπληρωθούν οι εντολές MATLAB που λείπουν:

```
>> Εντολή 1
Enter 2x2 array:[1 2; 3 4]
A =
      1            2
      3            4

>> Εντολή 2
Enter last name:'Matlabius'
B =
Matlabius

>> Εντολή 3
Enter first name:Matlabios
C =
Matlabios
```

1.24 Επαναλάβετε στη MATLAB όλα τα παραδείγματα της παραγράφου 1.6 και αποθηκεύστε την εργασία σας σε αρχείο με το όνομά σας.

2 ΔΙΑΝΥΣΜΑΤΑ ΚΑΙ ΠΙΝΑΚΕΣ

2.1 Γενικά

Στις αρχικές εκδοχές της MATLAB (μέχρι και την εκδοχή 3) κάθε μεταβλητή ήταν ένας διδιάστατος, δηλ. $m\times n$, πίνακας με (μιγαδικούς) αριθμούς διπλής ακρίβειας. Τα διανύσματα και οι αριθμοί ήταν απλώς ειδικές περιπτώσεις ($1\times n$ ή $n\times 1$ και 1×1 πίνακες αντίστοιχα). Από την εκδοχή 4, η MATLAB δέχεται επιπλέον n-διάστατους πίνακες καθώς και άλλους τύπους δεδομένων. Οι νέοι αυτοί τύποι δεδομένων περιλαμβάνουν τις **δομές** (structures), τις **τάξεις** (classes) και τους **πίνακες κελλίων** (cell arrays), οι οποίοι είναι πίνακες με στοιχεία όχι αναγκαστικά του ίδιου τύπου. Για παράδειγμα σ' ένα μονοδιάστατο πίνακα, το πρώτο στοιχείο μπορεί να είναι ένας αριθμός, το δεύτερο ένα αλφαριθμητικό (string) , το τρίτο ένα διάνυσμα κοκ.

Στο Κεφάλαιο αυτό θα συζητήσουμε τους διδιάστατους πίνακες. Θα ασχοληθούμε με άλλους τύπους δεδομένων σε επόμενο κεφάλαιο.

Ένα διάνυσμα $\mathbf{u = (u_1, u_2, \ldots, u_n)}$ εισάγεται στη MATLAB ως εξής:

>> u=[u_1, u_2 , …, u_n] ή **>> u=[u_1 u_2 … u_n]**

Έτσι οι συνιστώσες βρίσκονται ανάμεσα σε αγκύλες (όχι παρενθέσεις) και διαχωρίζονται από κόμματα ή απλώς με διαστήματα.

Οι πίνακες ορίζονται με παρόμοιο τρόπο: δίνουμε τα στοιχεία κάθε γραμμής και για να υποδείξουμε την αλλαγή γραμμής χρησιμοποιούμε το σύμβολο ';' ή απλά αλλάζουμε γραμμή.

Παράδειγμα 2.1.1
Θα ορίσουμε τους πίνακες

$$a=[-3,4,0],\qquad b=\begin{bmatrix}1\\2\\-4\\3\end{bmatrix}\qquad \text{και}\qquad A=\begin{bmatrix}1&2&3&4\\1&0&1&0\\-1&2&-2&1\\4&1&0&0\end{bmatrix}$$

στο παράθυρο εργασίας με όλους τους πιθανούς τρόπους. Στην αρχή βλέπουμε το μήνυμα λάθους που παίρνουμε όταν χρησιμοποιήσουμε παρενθέσεις αντί αγκύλες:

```
>> a=(-3, 4, 0)
??? a=(-3, 4, 0)
          |
Error: ")" expected, "," found.

>> a=[ -3, 4, 0]
a =
```

```
    -3     4     0

>> a=[-3 4 0 ]
a =
    -3     4     0

>> b=[ 1
2
-4
3]

b =
     1
     2
    -4
     3

>> b=[1; 2; -4; 3]
b =
     1
     2
    -4
     3

>> A=[ 1 2 3 4
1 0 1 0
-1 2 -2 1
4 1 0 0]

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

Είναι φανερό ότι τα διανύσματα είναι ειδικές περιπτώσεις πινάκων. Ένα διάνυσμα στήλης είναι ένας $m\times1$ πίνακας ενώ ένα διάνυσμα γραμμής είναι ένας $1\times n$ πίνακας. Επίσης, ένας αριθμός αντιστοιχεί σε ένα 1×1 πίνακα. Η εντολή *whos* μας δίνει (μεταξύ άλλων) και τις διαστάσεις των *a*, *b* και *A* που μόλις ορίσαμε:

```
>> whos
  Name      Size                    Bytes  Class

  A         4x4                       128  double array
  a         1x3                        24  double array
  b         4x1                        32  double array

Grand total is 23 elements using 184 bytes
```

Οι πράξεις μεταξύ πινάκων γίνονται με τα σύμβολα που φαίνονται στον πίνακα:

Σύμβολο	Πράξη
+	**Πρόσθεση**
-	**Αφαίρεση**
*	**Πολλαπλασιασμός**
\	**Αριστερή διαίρεση**
/	**Δεξιά διαίρεση**
^	**Ύψωση σε δύναμη**

Εννοείται βέβαια ότι οι χρησιμοποιούμενοι πίνακες πρέπει να είναι **συμβιβαστοί** ως προς την πράξη που κάνουμε. Έτσι, η ύψωση σε δύναμη είναι δυνατή μόνο για τετραγωνικούς πίνακες. Σημειώνουμε επίσης τα εξής:

- ο **ανάστροφος** A^{T} ενός **πραγματικού** πίνακα Α, συμβολίζεται με **A'.**
- Οι εντολές

  A*A*A και **A^3**

  όπου Α τετραγωνικός πίνακας είναι ισοδύναμες.
- Μπορούμε να πολλαπλασιάσουμε **όλα τα στοιχεία** ενός πίνακα A και ενός διανύσματος u με ένα αριθμό x (βαθμωτός πολλαπλασιασμός)

  x*A και **x*u**

  Με τον ίδιο τρόπο μπορούμε να διαιρέσουμε όλα τα στοιχεία με κάποιο μη μηδενικό αριθμό x:

  A/x και **u/x**
- Μπορούμε να προσθέσουμε ή να αφαιρέσουμε ένα αριθμό x **από όλα τα στοιχεία** ενός πίνακα A και ενός διανύσματος u:

  A – x και **u + x**

Παράδειγμα 2.1.2
Ορίζουμε εκ νέου το 1×3 διάνυσμα a, το 4×1 διάνυσμα στήλης b και τον 4×4 πίνακα A.

```
>> a=[-3 4 0];
>> b=[1 ; 2 ; -4; 3];
>> A=[ 1 2 3 4; 1 0 1 0; -1 2 -2 1; 4 1 0 0];
```

Θα βρούμε πρώτα τους αναστρόφους a^{T}, b^{T} και A^{T}. Στη συνέχεια θα υπολογίσουμε το συμμετρικό και το αντισυμμετρικό τμήμα του A:

$$(A + A^{\mathrm{T}})/2 \quad \text{και} \quad (A - A^{\mathrm{T}})/2.$$

```
>> a'
ans =
    -3
     4
     0

>> b'
ans =
     1     2    -4     3

>> A'
ans =
     1     1    -1     4
     2     0     2     1
     3     1    -2     0
     4     0     1     0

>> (A+A')/2
ans =
    1.0000    1.5000    1.0000    4.0000
    1.5000         0    1.5000    0.5000
    1.0000    1.5000   -2.0000    0.5000
    4.0000    0.5000    0.5000         0
```

```
>> (A-A')/2
ans =
         0    0.5000    2.0000         0
   -0.5000         0   -0.5000   -0.5000
   -2.0000    0.5000         0    0.5000
         0    0.5000   -0.5000         0
```

Ας εκτελέσουμε τώρα τους πιο κάτω πολλαπλασιασμούς:

(–3) a, $4A$, $b\,b^{\mathrm{T}}$, $b^{\mathrm{T}}b$, Ab, $b^{\mathrm{T}}A$, AA, A^3

```
>> (-3)*a
ans =
     9   -12     0

>> 4*A
ans =
     4     8    12    16
     4     0     4     0
    -4     8    -8     4
    16     4     0     0

>> b*b'
ans =
     1     2    -4     3
     2     4    -8     6
    -4    -8    16   -12
     3     6   -12     9

>> b'*b
ans =
    30
```

Παρατηρούμε ότι το γινόμενο b'*b αντιστοιχεί στο εσωτερικό γινόμενο και μας δίνει ένα αριθμό, ενώ το b*b' αντιστοιχεί στο λεγόμενο "εξωτερικό γινόμενο" και μας δίνει ένα $n\times n$ πίνακα.

```
>> A*b
ans =
     5
    -3
    14
     6

>> b'*A
ans =
    19    -3    13     0

>> A*A
ans =
    16    12    -1     7
     0     4     1     5
     7    -5     3    -6
     5     8    13    16

>> A^3
ans =
    57    37    62    63
    23     7     2     1
   -25    14    10    31
    64    52    -3    33
```

Ας αφαιρέσουμε τέλος τον 1 από όλα τα στοιχεία του *a* και ας προσθέσουμε τον 5 σε όλα τα στοιχεία του *A*:

```
>> a
a =
    -3     4     0

>> a-1
ans =
    -4     3    -1

>> A
A =
     1     2     3     4
     1     0     1     0
    -1     2    -2     1
     4     1     0     0

>> A+5
ans =

     6     7     8     9
     6     5     6     5
     4     7     3     6
     9     6     5     5
```

Είδαμε πιο πάνω ότι η απόστροφος ' δηλώνει τον **ανάστροφο** (transpose) ενός **πραγματικού πίνακα** (ή διανύσματος). Στην περίπτωση **μιγαδικού πίνακα** η απόστροφος δηλώνει τον **αναστροφοσυζυγή**. Έτσι αν ο *A* είναι ένας μιγαδικός πίνακας :

- Ο **A'** είναι ο **αναστροφοσυζυγής** του A
- Ο **A.'** είναι ο **ανάστροφος** του A (δεν παίρνουμε τα συζυγή στοιχεία)
- Ο **A.''** είναι ο **συζυγής** του A (χωρίς αναστροφή)

Παράδειγμα 2.1.3

Θα βρούμε τον αναστροφοσυζυγή, τον (απλό) ανάστροφο και το συζυγή του μιγαδικού πίνακα

$$A=\begin{bmatrix} 1-2i & 2+3i & 3 \\ 4i & -2+i & 4-3i \end{bmatrix}$$

```
>> A=[
1-2i  2+3i 3
4i    -2+i  4-3i]

A =
   1.0000 - 2.0000i   2.0000 + 3.0000i   3.0000
        0 + 4.0000i  -2.0000 + 1.0000i   4.0000 - 3.0000i

>> A'

ans =
   1.0000 + 2.0000i        0 - 4.0000i
   2.0000 - 3.0000i  -2.0000 - 1.0000i
   3.0000             4.0000 + 3.0000i
```

```
>> A.'

ans =
   1.0000 - 2.0000i        0 + 4.0000i
   2.0000 + 3.0000i  -2.0000 + 1.0000i
   3.0000             4.0000 - 3.0000i

>> A.''

ans =
   1.0000 + 2.0000i   2.0000 - 3.0000i   3.0000
        0 - 4.0000i  -2.0000 - 1.0000i   4.0000 + 3.0000i
```

2.1.1 Αριστερή και δεξιά διαίρεση

Η MATLAB είναι ένα αρκετά προχωρημένο λογισμικό και επιτρέπει την επίλυση γραμμικών συστημάτων με πολλούς τρόπους. Αν ο A είναι ένας αντιστρέψιμος πίνακας έχουμε τις πιο κάτω "διαιρέσεις" πινάκων:

A\b μας δίνει τη λύση του συστήματος $Ax = b$ **(αριστερή διαίρεση)**

A/b μας δίνει τη λύση του συστήματος $xA = b$ **(δεξιά διαίρεση)**

όπου οι πίνακες A, x και b **έχουν συμβιβαστές διαστάσεις**. Η διαφορά της αριστερής από τη δεξιά διαίρεση φαίνεται στον πίνακα:

Συμβολισμός MATLAB	**Μαθηματικό ισοδύναμο**
Αριστερή διαίρεση: **a\b**	$\frac{b}{a}$
Δεξιά διαίρεση: **a/b**	$\frac{a}{b}$

Παράδειγμα 2.1.4

Θεωρούμε τους πίνακες A και b του προηγούμενου παραδείγματος. Θα λύσουμε τα συστήματα $Ax = b$ και $xA = b^{\mathrm{T}}$.

```
>> A
A =
     1     1     1
     1     0     2
     0     2     1

>> b
b =
     6
     7
     7

>> A\b
```

```
ans =
     1
     2
     3

>> b'/A
ans =
    5.6667    0.3333    0.6667
```

Παράδειγμα 2.1.5

Θα βρούμε τον πίνακα C για τον οποίο ισχύει $AC = B$, όπου

$$A=\begin{bmatrix} 1 & 1 & 1 \\ 2 & 0 & 1 \\ -3 & 1 & 4 \end{bmatrix} \quad \text{και} \quad B=\begin{bmatrix} 6 & 5 & 4 \\ 3 & -1 & 2 \\ 4 & 0 & 1 \end{bmatrix},$$

```
>> format rat
>> A=[ 1 1 1; 2 0 1; -3 1 4];

>> B=[6 5 4; 3 -1 2; 4 0 1];

>> C=A\B

C =
    11/10         1/5          9/10
    41/10        31/5         29/10
     4/5         -7/5          1/5
```

Επαληθεύουμε το αποτέλεσμα:

```
>> A*C

ans =
     6             5             4
     3            -1             2
     4             0             1
```

2.2 Στοιχειώδεις πίνακες

Αρκετοί στοιχειώδεις πίνακες μπορούν να παραχθούν με τις συναρτήσεις της MATLAB. Οι σημαντικότερες από αυτές φαίνονται στον πίνακα που ακολουθεί:

Συνάρτηση	Ερμηνεία
eye	**πίνακας με 1 στη κύρια διαγώνιο και 0 αλλού**
zeros	**μηδενικός πίνακας**
ones	**πίνακας με 1 σε όλες τις θέσεις**
rand	**ομοιόμορφα ψευδο-τυχαίος πίνακας**
randn	**κανονικά ψευδο-τυχαίος πίνακας**
pascal	**(τετραγωνικός) πίνακας Pascal**
magic	**(τετραγωνικός) μαγικός πίνακας**
hilb	**(τετραγωνικός) πίνακας Hilbert**
invhilb	**αντίστροφος πίνακας Hilbert**

Στις συναρτήσεις **eye**, **zeros, ones rand** και **randn** πρέπει να ορίσουμε τις επιθυμητές διαστάσεις του πίνακα. Για παράδειγμα,

eye(m,n) ή **eye([m n])**.

Αν ο πίνακάς μας είναι τετραγωνικός μπορούμε να γράψουμε **eye(n,n)** ή πιο απλά **eye(n)**. Είναι φανερό ότι ο τελευταίος πίνακας είναι ο ταυτοτικός $n \times n$ πίνακας. Στις συναρτήσεις που παράγουν τετραγωνικούς, δηλ. n×n, πίνακες (**pascal**, **magic**, **hilb** και **invhilb**) γράφουμε απλά τη διάσταση n. Για παράδειγμα

pascal(n) και **magic(4)**.

Παράδειγμα 2.2.1

Ας δούμε τι μας δίνουν οι eye(2,3), zeros(4,5) και ones(4,2).

```
>> eye(2,3)
ans =
     1     0     0
     0     1     0

>> eye(3,3)
ans =
     1     0     0
     0     1     0
     0     0     1

>> zeros(4,5)
ans =
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0

>> ones(4,2)
ans =
     1     1
     1     1
     1     1
     1     1
```

Η **rand(m,n)** μας δίνει ένα $m \times n$ πίνακα με "τυχαία" στοιχεία επιλεγμένα από μια ομοιόμορφη κατανομή στο διάστημα [0, 1]. Με αυτή την κατανομή, το ποσοστό των αριθμών που βρίσκονται στο διάστημα $[a, b]$, όπου $0 < a < b < 1$, είναι $b - a$. Η **randn(m,n)** μας δίνει ένα $m \times n$ πίνακα με "τυχαία" στοιχεία επιλεγμένα από τη συνήθη κανονική κατανομή με μέση τιμή το 0, διασπορά 1 και τυπική απόκλιση 1. Καλώντας τις δύο συναρτήσεις με ένα όρισμα n, π.χ. rand(n), παίρνουμε ένα $n \times n$ τυχαίο πίνακα. Καλώντας τις χωρίς όρισμα παίρνουμε ένα τυχαίο αριθμό, π.χ.

```
>> rand
ans =
    0.5013
>> randn
ans =
    0.6457
```

Παράδειγμα 2.2.2

Θα κατασκευάσουμε ένα *ψευδοτυχαίο* 4×4 πίνακα:

```
>> rand(4)
ans =
    0.9501    0.8913    0.8214    0.9218
    0.2311    0.7621    0.4447    0.7382
    0.6068    0.4565    0.6154    0.1763
    0.4860    0.0185    0.7919    0.4057
```

Τα στοιχεία του πίνακα είναι ψευδοτυχαίοι αριθμοί μεταξύ 0 και 1. Θα κατασκευάσουμε τώρα ακόμα ένα ψευδοτυχαίο 4×4 πίνακα.

```
>> rand(4)
ans =
    0.9355    0.0579    0.1389    0.2722
    0.9169    0.3529    0.2028    0.1988
    0.4103    0.8132    0.1987    0.0153
    0.8936    0.0099    0.6038    0.7468
```

Αν ξεκινήσουμε εκ νέου τη MATLAB και πάμε να κατασκευάσουμε ένα τυχαίο 4×4 πίνακα θα πάρουμε ακριβώς τον πρώτο πίνακα, οπότε ο πίνακας μας δεν είναι εντελώς τυχαίος. Γι' αυτό το λόγο χρησιμοποιήσαμε τον όρο ψευδοτυχαίος.

Παράδειγμα 2.2.3

Οι συναρτήσεις **pascal** και **magic** επίσης ορίζουν τετραγωνικούς πίνακες. Η πρώτη μας δίνει τον κλασικό πίνακα του Pascal ενώ η magic επιστρέφει μαγικά τετράγωνα.

```
>> pascal(4)
ans =
     1     1     1     1
     1     2     3     4
     1     3     6    10
     1     4    10    20

>> magic(3)
ans =
     8     1     6
     3     5     7
     4     9     2
```

Μπορείτε να μάθετε περισσότερα για τις δύο αυτές συναρτήσεις με την εντολή help.

Παράδειγμα 2.2.4

Η συνάρτηση hilb ορίζει τον πίνακα Hilbert που έχει ως γενικό στοιχείο το

$$h_{ij} = \frac{1}{i+j-1}$$

Ας κατασκευάσουμε τον 3×3 πίνακα Hilbert:

```
>> H=hilb(3)
H =
    1.0000    0.5000    0.3333
    0.5000    0.3333    0.2500
    0.3333    0.2500    0.2000
```

Αν χρησιμοποιήσουμε την εντολή **format rat** παίρνουμε:

```
>> format rat

>> H
H =
      1          1/2          1/3
     1/2         1/3          1/4
     1/3         1/4          1/5
```

Ο αντίστροφος πίνακας Hilbert δίνεται από τη συνάρτηση **invhilb**. Ένα ενδιαφέρον χαρακτηριστικό του αντίστροφου πίνακα Hilbert είναι ότι όλα τα στοιχεία του είναι ακέραιοι.

```
>> Hinv=invhilb(3)
Hinv =
      9         -36          30
    -36         192        -180
     30        -180         180
```

Θα επαληθεύσουμε τώρα ότι πράγματι ο Hinv είναι ο αντίστροφος του H:

```
>> H*Hinv
ans =
      1           0           0
      0           1           0
      0           0           1
```

Με τις πιο πάνω συναρτήσεις μπορούμε με μια απλή εντολή να κατασκευάσουμε τεράστιους πίνακες. Αν δεν χρησιμοποιήσουμε το σύμβολο ';' στο τέλος της εντολής η MATLAB θα τυπώνει το αποτέλεσμα στο παράθυρο εργασίας αλλά ο χρήστης θα μπορεί να δει μόνο το τελευταίο κομμάτι του πίνακα. Μια επιλογή του χρήστη είναι να διατρέξει προς τα πάνω το παράθυρο εργασίας χρησιμοποιώντας είτε με το πλήκτρο [↑] είτε με το πλήκτρο [PgUp]. Αν όμως ο πίνακας είναι αρκετά μεγάλος δεν θα καταφέρει με τον τρόπο αυτό να δει τις πρώτες γραμμές.

Με την εντολή **more on** η MATLAB τυπώνει μόνο μια οθόνη κάθε φορά και ο χρήστης μπορεί να ελέγχει το αποτέλεσμα με την άνεσή του πατώντας το <CR> όποτε θέλει να προχωρήσει. Με την εντολή **more off** η MATLAB επανέρχεται στην κανονική της λειτουργία. ☺

Υπάρχουν πολλές άλλες συναρτήσεις στοιχειωδών και άλλων ειδικών πινάκων. Με την εντολή

help elmat

η MATLAB μας δίνει τον κατάλογο κατάλογό τους. Στον πίνακα που ακολουθεί παραθέτουμε ένα ακόμα μικρό κατάλογο προτρέποντας τον αναγνώστη να ανακαλύψει περισσότερα με την εντολή help:

Συνάρτηση	Ερμηνεία
compan	συνοδός πίνακας
gallery	συλλογή πινάκων του Ingham
hadamard	πίνακας Hadamard
hankel	πίνακας Hankel
rosser	πίνακας Rosser
toeplitz	πίνακας Toeplitz
vander	πίνακας Vandermonde
wilkinson	πίνακας Wilkinson

Ο N.J. Ingham [2] δημιούργησε μια μεγάλη συλλογή από συναρτήσεις πινάκων η οποία είναι ενσωματωμένη στο πακέτο της MATLAB. Η συνάρτηση **gallery** μας δίνει πρόσβαση σε όλες αυτές τις συναρτήσεις. Με την εντολή help gallery βλέπουμε τόσο τον κατάλογο αυτών των συναρτήσεων και άλλες χρήσιμες πληροφορίες. Έτσι αν matname είναι το όνομα μιας συνάρτησης του καταλόγου, μπορούμε να μάθουμε περισσότερα με την εντολή

help private\matname

2.3 Ορισμός διανυσμάτων και πινάκων με βήμα

Η MATLAB μας δίνει τη δυνατότητα να ορίσουμε τα στοιχεία ενός διανύσματος με κάποιο βήμα:

$$u = [\, u_1 : b : u_{last} \,]$$

πρώτο στοιχείο — βήμα — τελευταίο επιτρεπόμενο στοιχείο

Αν το βήμα b είναι ίσο με τη μονάδα, τότε αυτό μπορεί να παραλειφθεί:

$$u = [\, u_1 : u_{last} \,]$$

Παράδειγμα 2.3.1
Θα κατασκευάσουμε πρώτα το $u = (-1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$ με δύο τρόπους:

```
>> u=[-1:1:10]
u =
  -1     0     1     2     3     4     5     6     7     8
9    10

>> u=[-1:10]
u =
  -1     0     1     2     3     4     5     6     7     8
9    10
```

Θα κατασκευάσουμε τώρα το $u = (-1, 1, 3, 5, 7, 9)$.

```
>> u=[-1:2:9]
u =
    -1     1     3     5     7     9
```

Το ίδιο αποτέλεσμα το παίρνουμε και ως εξής:

```
>> u=[-1:2:10]
u =
    -1     1     3     5     7     9
```

Θα κατασκευάσουμε τέλος το $u = (-1, 2, 5, 8)$.

```
>> u=[-1:3:10]
u =
    -1     2     5     8
```

Όπως φαίνεται στο παράδειγμα που ακολουθεί η ίδια ιδέα μπορεί να χρησιμοποιηθεί για την κατασκευή πινάκων.

Παράδειγμα 2.3.2
Θα κατασκευάσουμε τους πίνακες

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 10 & 8 & 6 & 4 & 2 \end{bmatrix} \quad \text{και} \quad B = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 4 & 6 & 8 & 10 \\ 3 & 6 & 9 & 12 & 15 \end{bmatrix}.$$

```
>> A=[1:5;10:-2:2]

A =
     1     2     3     4     5
    10     8     6     4     2

>> B=[1:5;2:2:10;3:3:15]
B =
     1     2     3     4     5
     2     4     6     8    10
     3     6     9    12    15
```

Αν το *u* είναι ένας $n\times 1$ ή $1\times n$ πίνακας

$$u=\begin{bmatrix}u_1 & u_2 & \cdots & u_n\end{bmatrix} \qquad \text{ή} \qquad u=\begin{bmatrix}u_1\\ u_2\\ \vdots\\ u_n\end{bmatrix}$$

τότε το

$$u(i) \qquad \text{όπου} \quad 1 \le i \le n$$

μας δίνει το στοιχείο u_i.

Επίσης αν το *u* είναι ένας$1\times n$ πίνακας το

u(i : k : j)

μας δίνει το υποδιάνυσμα

$$u=\begin{bmatrix}u_i & u_{i+k} & u_{i+2k} & \cdots & u_j\end{bmatrix}$$

Αν το βήμα k είναι ίσο με τη μονάδα, μπορούμε να το παραλείψουμε. Έτσι το

u(i : j)

μας δίνει το υποδιάνυσμα

$$u=\begin{bmatrix}u_i & u_{i+1} & u_{i+2} & \cdots & u_j\end{bmatrix}$$

Παράδειγμα 2.3.3

Θα πάρουμε ομοιόμορφα κατανεμημένα σημεία στο [0,1] με βήμα 0.1:

```
>> u=[0:0.1:1]

u =
  Columns 1 through 7
         0    0.1000    0.2000    0.3000    0.4000    0.5000
0.6000
  Columns 8 through 11
    0.7000    0.8000    0.9000    1.0000

>> u(3)

ans =
    0.2000

>> u(1:3)

ans =
```

```
         0    0.1000    0.2000
>> u(2:6)
ans =
    0.1000    0.2000    0.3000    0.4000    0.5000
>> u(1:2:11)
ans =
         0    0.2000    0.4000    0.6000    0.8000    1.0000
>> b=u(1:5)
b =
         0    0.1000    0.2000    0.3000    0.4000
```

Σημειώνουμε εδώ ότι **το βήμα μπορεί να είναι αρνητικό**:

```
>> u=[10:-3:-10]
u =
    10     7     4     1    -2    -5    -8
```

Τα πιο πάνω γενικεύονται και για $m \times n$ πίνακες. Έτσι αν

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}$$

τότε

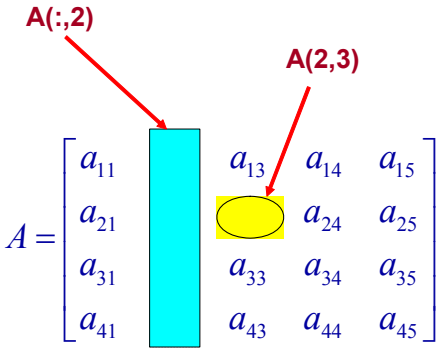
- το **A(i,j)** μας δίνει το a_{ij}
- το **A(:,j)** μας δίνει την ***j*-στήλη του *A***
- το **A(i, :)** μας δίνει την ***i*-γραμμή του *A***
- το **A(m:n,p:s)** μας δίνει **τον υποπίνακα του A που ορίζεται από τις γραμμές *m* έως *n* και τις στήλες *p* έως *s*.**

Η λέξη-κλειδί **end** (τέλος) δηλώνει την τελευταία γραμμή ($1^η$ διάσταση) ή την τελευταία στήλη ($2^η$ διάσταση). Έτσι

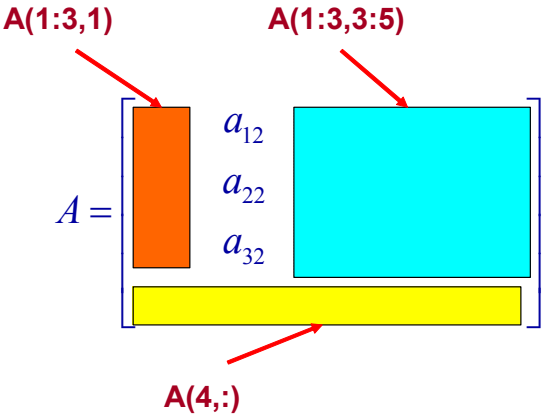
- το **A(end,:)** μας δίνει την **τελευταία γραμμή του A**
- το **A(:, end)** μας δίνει την **τελευταία στήλη του A**
- το **A(end, 1:2:5)** μας δίνει το διάνυσμα που περιέχει το 1^0, το 3^0 και το 5^0 στοιχείο της τελευταίας γραμμής του A (δεδομένου ότι ο A έχει τουλάχιστον 5 στήλες)

Παράδειγμα 2.3.4

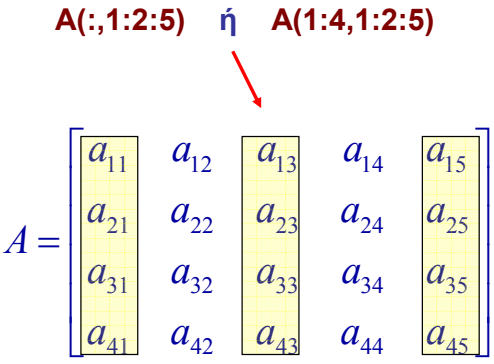
Έστω A ένας 4×5 πίνακας. Το A(2,3) μας δίνει το στοιχείο a_{23} ενώ το A(:,2) μας δίνει τη $2^η$ στήλη του A.



Το A(4,:) μας δίνει την 4η γραμμή και τα A(1:3,1) και A(1:3,3:5) τους υποπίνακες που φαίνονται στο σχήμα:



Το A(:,1:2:5) είναι ισοδύναμο με το A(1:5,1:2:5) και μας δίνει τον υποπίνακα που αποτελείται από την 1η την 3η και την 5η στήλη του A:



Παράδειγμα 2.3.5

Θεωρούμε τον πίνακα

$$A=\begin{bmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

Θα πάρουμε διαδοχικά τα στοιχεία a_{13} και $a_{32,}$ τη δεύτερη στήλη καθώς και την τρίτη γραμμή του A:

```
>> A=[ 1 2 3; 1 1 1; 1 0 1];
>> A(1,3)
ans =
     3
>> A(3,2)
ans =
     0
>> A(:,2)
ans =
     2
     1
     0
>> A(3,:)
ans =
     1     0     1
```

Θα κατασκευάσουμε τώρα δύο υποπίνακες του *A*:

```
>> A(1:2,2:3)
ans =
     2     3
     1     1
>> A(:,2:3)
ans =
     2     3
     1     1
     0     1
```

Παράδειγμα 2.3.6

Θεωρούμε ένα πίνακα *A* με άγνωστες διαστάσεις. Ο υποπίνακας των τριών τελευταίων στηλών είναι ο A(:, end-2:end). Ομοίως ο 3×3 κάτω δεξιά υποπίνακας του *A* είναι ο A(end-2:end, end-2:end):

```
>> A=magic(6)
A =
    35     1     6    26    19    24
     3    32     7    21    23    25
    31     9     2    22    27    20
     8    28    33    17    10    15
    30     5    34    12    14    16
     4    36    29    13    18    11
>> A(end-2:end,end-2:end)
ans =
    17    10    15
    12    14    16
    13    18    11
```

Η MATLAB μας δίνει τη δυνατότητα να επιλέξουμε ονομαστικά τους δείκτες γραμμών και στηλών. Έτσι

- το **A([1 3 5], [2 4])** είναι ο υποπίνακας που προκύπτει από την τομή των γραμμών 1, 3 και 5 και τον στηλών 2 και 4 του *A*

- το **A(end, [2 4 7])** είναι ο υποπίνακας που περιέχει το $2^{ο}$, $4^{ο}$ και $7^{ο}$ στοιχείο της τελευταίας γραμμής του *A*
- το **A([1 3 4], 1:2:5)** μας δίνει τον υποπίνακα που προκύπτει από την τομή των γραμμών 1, 3 και 4 και των στηλών 1, 3 και 5

Παράδειγμα 2.3.7

Ο υποπίνακας που προκύπτει από την τομή των γραμμών 2, 3 και 6 και των στηλών 1, 4 και 7 ενός 7×7 πίνακα είναι ο *A*([2 3 6], [1 4 7]) ή εναλλακτικά ο *A*([2 3 6], 1:3:7).

Αναφέρουμε τέλος ότι

- το **A(:)** είναι το διάνυσμα που περιέχει όλα τα στοιχεία του *A* κατά στήλη, από την πρώτη μέχρι την τελευταία.

Παράδειγμα 2.3.8

```
>> A=[ 1 3 2; 4 -2 1; 0 1 0]
A =
     1     3     2
     4    -2     1
     0     1     0
>> u=A(:)
u =
     1
     4
     0
     3
    -2
     1
     2
     1
     0
```

Επίσης όταν το **A(:)** μπει αριστερά σε μια εντολή εκχώρησης τιμής,

A(:) = v,

Τότε ο *A* γεμίζει με τα στοιχεία του *v* διατηρώντας τη μορφή του. Θα πρέπει φυσικά ο *A* και το *v* να έχουν το ίδιο πλήθος στοιχείων (δηλ. τα γινόμενα των διαστάσεων των *A* και *v* να είναι ίσα).

Παράδειγμα 2.3.9

Ένας εύκολος τρόπος να κατασκευάσουμε τον

$$A=\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

είναι ο εξής:

```
>> u=1:16
```

```
u =
  Columns 1 through 13
     1     2     3     4     5     6     7     8     9    10
11    12    13
  Columns 14 through 16
    14    15    16

>> A=zeros(4);

>> A(:)=u

A =
     1     5     9    13
     2     6    10    14
     3     7    11    15
     4     8    12    16

>> A=A'

A =
     1     2     3     4
     5     6     7     8
     9    10    11    12
    13    14    15    16
```

2.4 Συνένωση πινάκων

Στη γραμμική άλγεβρα, άρα και στη MATLAB, συχνά μας βολεύει να χρησιμοποιήσουμε **σύνθετους πίνακες** (block matrices). Οι σύνθετοι αυτοί πίνακες προκύπτουν με τη **συνένωση** (*concatenation*) μικρότερων υποπινάκων. Για την κατασκευή ενός σύνθετου πίνακα χειριζόμαστε τους υποπίνακες όπως και τα απλά στοιχεία ενός πίνακα. Έτσι, αν για παράδειγμα οι A, B, C και D είναι $m \times n$ πίνακες τότε

- η **[A B]** μας δίνει τον σύνθετο πίνακα
$$[A\ B]$$
- η **[A; B; C]** μας δίνει τον
$$\begin{bmatrix} A \\ B \\ C \end{bmatrix}$$
- η **[A B; C D]** μας δίνει τον
$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

Παράδειγμα 2.4.1
Έστω οι πίνακες
$$A=\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix},\quad B=\begin{bmatrix} -4 & 1 \\ 2 & 0 \end{bmatrix},\quad I=\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{και} \quad O=\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$
Θα κατασκευάσουμε τους σύνθετους πίνακες
$$E=[A\ B],\quad F=\begin{bmatrix} A \\ B \end{bmatrix} \quad \text{και} \quad G=\begin{bmatrix} A & B \\ I & O \end{bmatrix}.$$

```
>> A=[1 2; 3 4];
>> B=[-4 1; 2 0];
>> I=eye(2);
>> O=zeros(2);

>> E=[A B]
E =
     1     2    -4     1
     3     4     2     0

>> F=[A;B]
F =
     1     2
     3     4
    -4     1
     2     0

>> G=[ A B; I O]
G =
     1     2    -4     1
     3     4     2     0
     1     0     0     0
     0     1     0     0
```

Παράδειγμα 2.4.2

Έστω το γραμμικό σύστημα $Ax = b$. Είναι γνωστό ότι ο σύνθετος πίνακας $[A\ b]$ καλείται **επαυξημένος πίνακας** (augmented matrix) του συστήματος. Αν γνωρίζουμε τα A και b, τότε είναι πολύ εύκολο να κατασκευάσουμε τον επαυξημένο πίνακα στη MATLAB γράφοντας **[A b]** .

Παράδειγμα 2.4.3

Η συνάρτηση βιβλιοθήκης **det** υπολογίζει την ορίζουσα ενός τετραγωνικού πίνακα. Για παράδειγμα για τον πίνακα

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 2 \\ 0 & 2 & 1 \end{pmatrix}$$

έχουμε:

```
>> A=[1 1 1; 1 0 2; 0 2 1];
>> det(A)
ans =
    -3
```

Έστω τώρα το σύστημα ***Ax = b*** όπου $b = [6, 7, 7]^T$. Θα λύσουμε το σύστημα με τη **μέθοδο Cramer**.

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

>> D1=[b,A(:,2:3)]
D1 =
     6     1     1
     7     0     2
     7     2     1

>> D2=[A(:,1),b,A(:,3)]
D2 =
     1     6     1
     1     7     2
     0     7     1

>> D3=[A(:,1:2),b]
D3 =
     1     1     6
     1     0     7
     0     2     7

>> x1=det(D1)/det(A)
x1 =
     1

>> x2=det(D2)/det(A)
x2 =
     2

>> x3=det(D3)/det(A)
x3 =
     3
```

Γνωρίζουμε ήδη ότι η MATLAB επιτρέπει την επίλυση γραμμικών συστημάτων με πολλούς τρόπους. Η μέθοδος Cramer χρησιμοποιήθηκε πιο πάνω για διδακτικούς και μόνο λόγους.

Παράδειγμα 2.4.4

Θα κατασκευάσουμε τον πίνακα

$$A=\begin{bmatrix} 2 & 0 & 3 & 0 & 0 \\ 0 & 2 & 0 & 3 & 0 \\ 0 & 0 & 4 & 4 & 4 \\ 0 & 0 & 4 & 4 & 4 \end{bmatrix}$$

ως εξής:

```
>> A=[2*eye(2) 3*eye(2,3); zeros(2) 4*ones(2,3)]
A =
     2     0     3     0     0
     0     2     0     3     0
     0     0     4     4     4
     0     0     4     4     4
```

Ας θεωρήσουμε τώρα τον σύνθετο (κατά μπλοκ) διαγώνιο πίνακα

$$A=\begin{bmatrix} 1 & 2 & 0 & 0 & 0 & 0 \\ 3 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 3 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 3 & 4 \end{bmatrix}$$

Ένας τρόπος για να κατασκευάσουμε τον A στη MATLAB είναι να ορίσουμε πρώτα τον υποπίνακα

$$B=\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

και μετά να κατασκευάσουμε τον σύνθετο 3×3 σύνθετο πίνακα A με υποπίνακες τον B και το μηδενικό 2×2 πίνακα zeros(2):

```
>> B=[1 2; 3 4];
>> A=[ B zeros(2) zeros(2); zeros(2) B zeros(2); zeros(2) zeros(2) B]
A =
     1     2     0     0     0     0
     3     4     0     0     0     0
     0     0     1     2     0     0
     0     0     3     4     0     0
     0     0     0     0     1     2
     0     0     0     0     3     4
```

Θα μπορούσαμε εναλλακτικά να χρησιμοποιήσουμε τη συνάρτηση **blkdiag** η οποία κατασκευάζει κατά μπλοκ διαγώνιους πίνακες. Γράφουμε απλώς:

```
>> blkdiag(B, B, B)

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

Μια άλλη σχετική συνάρτηση είναι η

repmat(A,m,n)

η οποία μας δίνει ένα σύνθετο $m \times n$ πίνακα με υποπίνακα τον A σε κάθε θέση. Η

repmat(A,n)

μας δίνει τον αντίστοιχο $n \times n$ πίνακα.

Παράδειγμα 2.4.5

Έστω ο σύνθετος πίνακας

$$A = \begin{bmatrix} I & I & I \\ I & I & I \end{bmatrix}$$

όπου I ο 2×2 ταυτοτικός πίνακας. Μπορούμε να κατασκευάσουμε τον A ως εξής:

```
>> format compact
>> I=eye(2); A=repmat(I,2,3)
A =
     1     0     1     0     1     0
     0     1     0     1     0     1
     1     0     1     0     1     0
     0     1     0     1     0     1
```

Διαφορετικά θα μπορούσαμε να γράψουμε A = [I I I; I I I].

Πολλές φορές για να κατασκευάσουμε ένα πίνακα είναι βολικό να τροποποιήσουμε ένα πίνακα που ορίζεται με συναρτήσεις βιβλιοθήκης της MATLAB. Αν για παράδειγμα θέλουμε να κατασκευάσουμε τον πίνακα

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

μπορούμε να μηδενίσουμε πρώτα όλα τα στοιχεία του A μέσω της συνάρτησης zeros και μετά να εκχωρήσουμε την επιθυμητή τιμή στο στοιχείο a_{31}:

```
>> A=zeros(4);
>> A(3,1)=4;
>> A
A =
```

```
     0     0     0     0
     0     0     0     0
     4     0     0     0
     0     0     0     0
```

Αξίζει να σημειωθεί εδώ ότι με την εντολή A = zeros(4) καθορίσαμε επίσης και τις επιθυμητές διαστάσεις του *A*.

Η MATLAB μας επιτρέπει μάλιστα να εκχωρήσουμε **μία τιμή** σε τμήματα ή υποπίνακες του *A* που ορίζονται σύμφωνα με τους κανόνες που αναφέραμε στην παράγραφο αυτή. Έτσι

- η **A(1:3,1:3) = 0** δίνει την τιμή 1 στον πάνω αριστερά 3×3 υποπίνακα του *A*
- η **A(3,:) = 0** μηδενίζει την $3^{η}$ γραμμή του *A*
- η **A(:,3:4) = pi** δίνει την τιμή π στα στοιχεία της $3^{ης}$ και της $4^{ης}$ στήλης του *A* κοκ.

Παράδειγμα 2.4.6
Θα κατασκευάσουμε τον πίνακα

$$A=\begin{bmatrix} 2 & 2 & 2 & 2 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix}$$

```
>> A=2*ones(5);
>> A(2:4,2:4)=1
A =
     2     2     2     2     2
     2     1     1     1     2
     2     1     1     1     2
     2     1     1     1     2
     2     2     2     2     2
```

2.4.1 Η συνάρτηση cat

Η συνάρτηση **cat** συνενώνει δύο ή περισσότερους πίνακες **κατά μήκος μιας διάστασης.** Για να εξηγήσουμε τη συνάρτηση θα πάρουμε σαν παράδειγμα τρεις τετραγωνικούς *n*×*n* πίνακες *A*, *B* και *C*, σημειώνοντας ότι η συνάρτηση μπορεί να χρησιμοποιηθεί και για πολυδιάστατους πίνακες:

- η **cat(1,A,B)** συνενώνει τους *A* και *B* στη διεύθυνση των *y* σχηματίζοντας τον σύνθετο πίνακα

  $$\begin{bmatrix} A \\ B \end{bmatrix}.$$

  Άρα είναι ισοδύναμη με την **[A;B]**. Ομοίως η **cat(1,A,B,C)** είναι ισοδύναμη με την **[A; B; C].**
- η **cat(2,A,B)** συνενώνει τους *A* και *B* στη διεύθυνση των *x* σχηματίζοντας τον σύνθετο πίνακα

  $$[A\ B].$$

Άρα είναι ισοδύναμη με την **[A B]**. Ομοίως η **cat(2,A,B,C)** είναι ισοδύναμη με την **[A, B, C]**.

- η **cat(3,A,B)** συνενώνει τους A και B στη διεύθυνση των z σχηματίζοντας ένα ***n*×*n*×2** πίνακα. Ομοίως η **cat(3, A, B, C)** σχηματίζει ένα ***n*×*n*×3** πίνακα.

Παράδειγμα 2.4.7
Έστω οι πίνακες

$$A=\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{bmatrix} \quad \text{και} \quad B=\begin{bmatrix}2 & 1 & -3\\ 0 & 4 & -1\\ 5 & 1 & 0\end{bmatrix}$$

Θα δούμε τα αποτελέσματα των cat(1,A,B), cat(2,A,B) και cat(3,A,B).

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

>> B=[2 1 -3; 0 4 -1; 5 1 0];

>> C=cat(1,A,B)

C =
     1     2     3
     4     5     6
     7     8     9
     2     1    -3
     0     4    -1
     5     1     0

>> D=cat(2,A,B)

D =
     1     2     3     2     1    -3
     4     5     6     0     4    -1
     7     8     9     5     1     0

>> E=cat(3,A,B)

E(:,:,1) =
     1     2     3
     4     5     6
     7     8     9

E(:,:,2) =
     2     1    -3
     0     4    -1
     5     1     0
```

Με την εντολή who βλέπουμε τις διαστάσεις των πινάκων που δημιουργήσαμε και ειδικά ότι ο E είναι 3×3×2 πίνακας.

```
>> whos
  Name      Size                    Bytes  Class

  A         3x3                        72  double array
  B         3x3                        72  double array
  C         6x3                       144  double array
  D         3x6                       144  double array
  E         3x3x2                     144  double array

Grand total is 72 elements using 576 bytes
```

2.5 Πράξεις κατά τα στοιχεία διανύσματος ή πίνακα

Η MATLAB μας δίνει τη δυνατότητα να κάνουμε τις ίδιες πράξεις **σε όλα τα στοιχεία** ενός διανύσματος ή ενός πίνακα. Οι προκύπτουσες τιμές αποθηκεύονται αντίστοιχα σε διάνυσμα ή πίνακα. Για παράδειγμα αν θέλουμε να υπολογίσουμε τα ημίτονα όλων των στοιχείων του διανύσματος u_i του u αρκεί να γράψουμε

sin(u)

Με τον ίδιο τρόπο μπορούμε να χρησιμοποιήσουμε όλες τις συναρτήσεις βιβλιοθήκης της MATLAB (exp, abs κα).

Παράδειγμα 2.5.1

```
>> u=[0:0.2:1]
u =
         0    0.2000    0.4000    0.6000    0.8000    1.0000

>> sin(u)
ans =
         0    0.1987    0.3894    0.5646    0.7174    0.8415

>> exp(u)
ans =
    1.0000    1.2214    1.4918    1.8221    2.2255    2.7183
```

Παράδειγμα 2.5.2

Θα κατασκευάσουμε τον 5×5 πίνακα A με γενικό στοιχείο το

$$e^{1/(i+j-1)}$$

```
>> A=exp(hilb(5))
A =
    2.7183    1.6487    1.3956    1.2840    1.2214
    1.6487    1.3956    1.2840    1.2214    1.1814
    1.3956    1.2840    1.2214    1.1814    1.1536
    1.2840    1.2214    1.1814    1.1536    1.1331
    1.2214    1.1814    1.1536    1.1331    1.1175
```

Αξιοποιήσαμε εδώ το ότι $1/(i+j-1)$ είναι το γενικό στοιχείο του πίνακα Hilbert.

Η MATLAB έχει προβλέψει τη δυνατότητα μια πράξη, όπως ο πολλαπλασιασμός (*), η διαίρεση (/) και η ύψωση σε δύναμη (^) να γίνεται κατά τα στοιχεία ενός πίνακα ένα προς ένα αρκεί πριν από το σύμβολο της πράξης να εμφανίζεται μια τελεία. Έτσι αν το u είναι ένα διάνυσμα και ο A ένας $n×n$ πίνακας, τότε

- **u.^2** είναι το διάνυσμα με γενικό στοιχείο το u(i)^2. Η παράσταση είναι ισοδύναμη με την **u.*u**.
- **A.^3** είναι ο πίνακας με γενικό στοιχείο το A(i,j)^3
- **u.*4** είναι το διάνυσμα με γενικό στοιχείο το u(i)*4 που μπορούμε να πάρουμε πιο απλά με **u*4.**
- **u./5** είναι το διάνυσμα με γενικό στοιχείο το u(i)/5 που μπορούμε να πάρουμε πιο απλά με **u/5** κοκ.

Παρατήρηση: αν το k είναι ένας αριθμός και A ένας $m×n$ πίνακας τότε στις εντολές

k+A, A+k, A-k, -k+A, A*k, k*A, A/k

οι πράξεις γίνονται ξεχωριστά σε κάθε στοιχείο του A. **Έτσι η μόνη εξαίρεση στον κανόνα αυτό είναι η ύψωση σε δύναμη**:

A^k.

Παράδειγμα 2.5.3

```
>> u=[1:6]

u =
     1     2     3     4     5     6
```

Υψώνουμε τώρα κάθε στοιχείο του *u* στο τετράγωνο με την u.^2 αλλά και την ισοδύναμή της u.*u:

```
>> a=u.^2

a =
     1     4     9    16    25    36
>> b=u.*u

b =
     1     4     9    16    25    36
```

Παίρνοντας τις τετραγωνικές ρίζες των στοιχείων του *b* βρίσκουμε ξανά το *u*.

```
>> c=sqrt(b)

c =
     1     2     3     4     5     6
```

Αν έχουμε δύο διανύσματα *u* και *v* ίσου μήκους τότε μπορούμε να κάνουμε πράξεις μεταξύ των ομοθέσιων στοιχείων τους εφόσον βάλουμε τελεία πριν από το σύμβολο της πράξης. Έτσι

- **w = u.*v** είναι το διάνυσμα με γενικό στοιχείο το w(i) = u(i)*v(i). Η παράσταση είναι ισοδύναμη με την **v.*u**.
- **w = u./v** είναι το διάνυσμα με γενικό στοιχείο το w(i) = u(i)/v(i).
- **w = u.\v** είναι το διάνυσμα με γενικό στοιχείο το v(i)/u(i). Είναι φανερό ότι η εντολή είναι ισοδύναμη με την **w = v./u** .
- **w = 2.^v** είναι το διάνυσμα με γενικό στοιχείο το w(i) = 2^v(i).
- **w = u.^v** είναι το διάνυσμα με γενικό στοιχείο το w(i) = u(i)^v(i) κοκ.

Εντελώς ανάλογα εκτελούνται πράξεις κατά τα ομοθέσια στοιχεία δύο $m \times n$ πινάκων *A* και *B*:

- **C = A.*B** είναι ο πίνακας με γενικό στοιχείο το C(i,j) = A(i,j)*B(i,j). Η παράσταση είναι ισοδύναμη με την **B.*A**.
- **C = A./B** είναι ο πίνακας με γενικό στοιχείο το C(i,j) = A(i,j)/B(i,j).
- **C = A.\B** είναι ο πίνακας με γενικό στοιχείο το C(i,j) = B(i,j)/A(i,j). Είναι φανερό ότι η εντολή είναι ισοδύναμη με την **C = B./A** .
- **C = 2.^B** είναι ο πίνακας με γενικό στοιχείο το C(i,j) = 2^B(i,j).
- **C = A.^B** είναι ο πίνακας με γενικό στοιχείο το C(i,j) = A(i,j)^B(i,j).

Παράδειγμα 2.5.5

Θα δοκιμάσουμε τις 5 πράξεις που φαίνονται πιο πάνω με $u = (1, 2, 3, 4)$ και $v = (8, 6, 4, 2)$.

```
>> u=1:4
u =
     1     2     3     4

>> v=8:-2:2
v =
     8     6     4     2

>> u.*v
ans =
     8    12    12     8

>> u./v
ans =
    0.1250    0.3333    0.7500    2.0000

>> u.\v
ans =
    8.0000    3.0000    1.3333    0.5000

>> 2.^v
ans =
   256    64    16     4

>> u.^v
ans =
     1    64    81    16
```

Παράδειγμα 2.5.6

Θα επαναλάβουμε τις πράξεις με τους πίνακες

$$A=\begin{bmatrix}1 & 1 & 1\\ 2 & 2 & 2\\ 3 & 3 & 3\end{bmatrix} \text{ και } B=\begin{bmatrix}3 & 1 & 1\\ 1 & 3 & 1\\ 1 & 1 & 3\end{bmatrix}$$

```
>> A=[1 1 1; 2 2 2; 3 3 3];
>> B=2*eye(3)+1;

>> A.*B
ans =
     3     1     1
     2     6     2
     3     3     9

>> A./B
ans =
    0.3333    1.0000    1.0000
    2.0000    0.6667    2.0000
    3.0000    3.0000    1.0000

>> A.\B
ans =
    3.0000    1.0000    1.0000
    0.5000    1.5000    0.5000
    0.3333    0.3333    1.0000
```

```
>> 2.^B
ans =
     8     2     2
     2     8     2
     2     2     8

>> A.^B
ans =
     1     1     1
     2     8     2
     3     3    27
```

Είναι φανερή η διαφορά μεταξύ του **A^2** και του **A.^2**. Ενώ το **A^2 (=A*A)** είναι το τετράγωνο του πίνακα, το **A.^2** είναι ο πίνακας με στοιχεία τα τετράγωνα των στοιχείων του A.

Παράδειγμα 2.5.7
Θα δούμε τη διαφορά μεταξύ A^2 και του A.^2 όταν

$$A=\begin{bmatrix}1 & -2+i\\ 3 & i\end{bmatrix}$$

```
>> A=[1 -2+i; 3 i];

>> A^2
ans =
  -5.0000 + 3.0000i  -3.0000 - 1.0000i
   3.0000 + 3.0000i  -7.0000 + 3.0000i

>> A.^2
ans =
   1.0000            3.0000 - 4.0000i
   9.0000           -1.0000
```

2.6 Συναρτήσεις βιβλιοθήκης για διανύσματα

Στον πίνακα που ακολουθεί φαίνονται κάποιες από τις συναρτήσεις βιβλιοθήκης για διανύσματα.

Συνάρτηση	Ερμηνεία
max	**μέγιστο στοιχείο διανύσματος**
min	**ελάχιστο στοιχείο διανύσματος**
length	**μήκος διανύσματος**
sort	**ταξινόμηση σε αύξουσα σειρά**
sum	**άθροισμα στοιχείων**
prod	**γινόμενο στοιχείων**
norm	**νόρμα διανύσματος**
median	**διάμεσος**
mean	**μέση τιμή**
std	**τυπική απόκλιση**

Μπορείτε να μάθετε περισσότερα για τις πιο πάνω συναρτήσεις και τη χρήση τους με την εντολή **help.**

Παράδειγμα 2.6.1
Έστω το διάνυσμα $u = (3, 4, 6, 3.4, 2.5, –7, –2, 5, 2.7, 6, 4)$. Θα βρούμε το ελάχιστο, το μέγιστο, το ελάχιστο κατ'απόλυτη τιμή και το μέγιστο κατ' απόλυτη τιμή στοιχείο του.

```
>> u=[ 3 4 6 3.4 2.5 -7 -2 5 2.7 6 4]

u =
  Columns 1 through 7
    3.0000    4.0000    6.0000    3.4000    2.5000   -7.0000   -
2.0000
  Columns 8 through 11
    5.0000    2.7000    6.0000    4.0000

>> min(u)

ans =
    -7

>> max(u)

ans =
     6

>> min(abs(u))

ans =
     2

>> max(abs(u))

ans =
     7
```

Ας βρούμε τώρα το μήκος και ας ταξινομήσουμε τα στοιχεία του *u*:

```
>> length(u)

ans =
```

```
    11

>> v=sort(u)

v =
  Columns 1 through 7
   -7.0000    -2.0000     2.5000     2.7000     3.0000     3.4000
4.0000
  Columns 8 through 11
    4.0000    5.0000    6.0000    6.0000
```

Ας βρούμε τώρα το άθροισμα και το γινόμενο των στοιχείων του *u*:

```
>> sum(u)

ans =
   27.6000

>> prod(u)

ans =
  2.7760e+006
```

Παράδειγμα 2.6.2

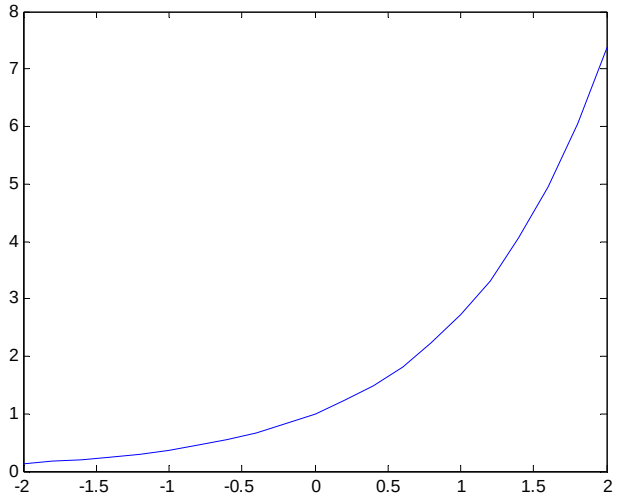
Θα διαμερίσουμε το διάστημα [–2, 2] σε ίσα υποδιαστήματα με βήμα 0.2 και θα αποθηκεύσουμε τις συντεταγμένες των σημείων στο διάνυσμα *x*. Στη συνέχεια θα υπολογίσουμε τη συνάρτηση e^x σε όλα τα σημεία του *x* και θα αποθηκεύσουμε τις τιμές στο διάνυσμα *y*.

```
>> x=[-2:0.2:2]
x =
  Columns 1 through 7
   -2.0000   -1.8000   -1.6000   -1.4000   -1.2000   -1.0000   -
0.8000
  Columns 8 through 14
   -0.6000   -0.4000   -0.2000         0    0.2000    0.4000
0.6000
  Columns 15 through 21
    0.8000    1.0000    1.2000    1.4000    1.6000    1.8000
2.0000

>> y=exp(x)
y =
  Columns 1 through 7
    0.1353    0.1653    0.2019    0.2466    0.3012    0.3679
0.4493
  Columns 8 through 14
    0.5488    0.6703    0.8187    1.0000    1.2214    1.4918
1.8221
  Columns 15 through 21
    2.2255    2.7183    3.3201    4.0552    4.9530    6.0496
7.3891
```

Αν δοκιμάσουμε τώρα την εντολή

plot(x,y)

θα πάρουμε σε ξεχωριστό παράθυρο την γραφική παράσταση:



Η εντολή **plot** θα μας απασχολήσει σε επόμενο κεφάλαιο.

Παράδειγμα 2.6.3
Θα κατασκευάσουμε τυχαία διανύσματα μήκους 10, 1000 και 10000 και θα βρούμε τις μέσες τιμές των στοιχείων τους:

```
>> mean(rand(1,10))
ans =
    0.5326
>> mean(rand(1,1000))
ans =
    0.4907
>> mean(rand(1,100000))
ans =
    0.5000
```

Πως ερμηνεύετε τα πιο πάνω αποτελέσματα;

2.6.1 Οι εντολές dot και cross

Το **εσωτερικό γινόμενο *a*·*b* δύο ίσου μήκους διανυσμάτων *a* και *b*** υπολογίζεται με την εντολή

dot(a,b)

Η μόνη προϋπόθεση για να γίνει η πράξη είναι τα *a* και *b* να έχουν το ίδιο μήκος. Έτσι μπορεί να είναι και τα δύο διανύσματα γραμμές ή και τα δύο διανύσματα στήλες ή το ένα διάνυσμα γραμμή και το άλλο διάνυσμα στήλη.

Το **εξωτερικό γινόμενο *u*×*v* δύο διανυσμάτων *u* και *v* του R^3** υπολογίζεται με την εντολή

cross(u,v)

Το εξωτερικό γινόμενο μας δίνει ένα διάνυσμα γραμμή εκτός αν και τα δύο διανύσματα είναι διανύσματα στήλες.

Παράδειγμα 2.6.4
Θα υπολογίσουμε το εσωτερικό και το εξωτερικό γινόμενο των διανυσμάτων
$a = (-2, 1, 3)^T$ και $b = (0, 2, 1)^T$:

```
>> a=[-2;1;3];
>> b=[0;2;1];
>> dot(a,b)
ans =
     5
>> cross(a,b)
ans =
    -5
     2
    -4
```

Παρατηρούμε ότι το εξωτερικό γινόμενο είναι και αυτό διάνυσμα στήλη όπως και τα *a* και *b*. Ένας άλλος τρόπος να υπολογίσουμε το εσωτερικό γινόμενο των *a* και *b* είναι ο εξής:

```
>> a'*b
ans =
     5
```

2.7 Χρήσιμες συναρτήσεις βιβλιοθήκης για πίνακες

Πιο κάτω φαίνονται κάποιες από τις συναρτήσεις βιβλιοθήκης για πίνακες:

Συνάρτηση	Ερμηνεία
max	**διάνυσμα μέγιστων στοιχείων κάθε στήλης**
min	**διάνυσμα ελάχιστων στοιχείων κάθε στήλης**
diag	**διαγώνιος πίνακας ή η διαγώνιος πίνακα!**
triu	**άνω τριγωνικό μέρος πίνακα**
tril	**κάτω τριγωνικό μέρος πίνακα**
size	**μέγεθος πίνακα**
length	**η μεγαλύτερη διάσταση πίνακα**
norm	**νόρμα πίνακα**
det	**ορίζουσα τετραγωνικού πίνακα**
trace	**ίχνος πίνακα**
rank	**βαθμός πίνακα**
inv	**αντίστροφος αντιστρέψιμου πίνακα**
rref	**ανηγμένη κλιμακωτή μορφή πίνακα**
eig	**ιδιοτιμές (και ιδιοδιανύσματα) πίνακα**
poly	**χαρακτηριστικό πολυώνυμο τετραγωνικού πίνακα**
cond	**δείκτης κατάστασης τετραγωνικού πίνακα**

Ένας πλήρης κατάλογος των συναρτήσεων βιβλιοθήκης για πίνακες δίνεται με την εντολή

help matfun

Μπορείτε να μάθετε περισσότερα για τις πιο πάνω συναρτήσεις και τη χρήση τους με την εντολή **help.**

Έχουμε ήδη συναντήσει την συνάρτηση **length** όταν μιλούσαμε για διανύσματα. Η συνάρτηση αυτή μας δίνει τη μεγαλύτερη από τις διαστάσεις ενός πίνακα. Αν x είναι $n\times 1$ ή $1\times n$ διάνυσμα, η length(x) έχει την τιμή n. Η συνάρτηση **size** μας δίνει (σε διάνυσμα γραμμή) το μέγεθος ενός πίνακα. Ας πάρουμε ένα 4×3 πίνακα:

```
>> A=ones(4,3);
>> length(A)
ans =
     4
>> size(A)
ans =
     4     3
```

Αν θέλουμε να βρούμε το μέγιστο στοιχείο ενός πίνακα Α αρκεί να γράψουμε

max(max(A)).

Ομοίως αν θέλουμε να βρούμε το ελάχιστό του στοιχείο γράφουμε

min(min(A)).

Παράδειγμα 2.7.1
Έστω ο πίνακας

$$C = \begin{bmatrix} 1 & -2 & 3 & 0 & 1 \\ -2 & 9 & 4 & 2 & 1 \\ -7 & -8 & 1 & 0 & 1 \\ 1 & -1 & 2 & -2 & 3 \end{bmatrix}$$

Θα βρούμε το διάνυσμα με τα μέγιστα στοιχεία κάθε στήλης καθώς και το μέγιστο στοιχείο του πίνακα.

```
>> C=[
1 -2 3 0 1
-2 9 4 2 1
-7 -8 1 0 1
1 -1 2 -2 3]

C =
     1    -2     3     0     1
    -2     9     4     2     1
    -7    -8     1     0     1
     1    -1     2    -2     3

>> max(C)
ans =
     1     9     4     2     3

>> max(max(C))
ans =
     9
```

Θα βρούμε το διάνυσμα με τα ελάχιστα στοιχεία κάθε στήλης καθώς και το ελάχιστο στοιχείο του πίνακα.

```
>> min(C)
ans =
    -7    -8     1    -2     1

>> min(min(C))
ans =
    -8
```

Η συνάρτηση **size** μας δίνει σ'ένα διάνυσμα μήκους v τις διαστάσεις ενός v-διάστατου πίνακα.

Παράδειγμα 2.7.2
Θα βρούμε το μέγεθος (**size**), την ορίζουσα (**det**), το ίχνος (**trace**), τον βαθμό (**rank**), και τον αντίστροφο (**inv**) του τετραγωνικού πίνακα

$$A = \begin{bmatrix} 1 & -3 & 3 \\ 3 & -5 & 3 \\ 6 & -6 & 4 \end{bmatrix}$$

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

```
     3    -5     3
     6    -6     4

>> size(A)
ans =
     3     3

>> det(A)
ans =
    16

>> trace(A)
ans =
     0

>> rank(A)
ans =
     3

>> inv(A)
ans =
   -0.1250   -0.3750    0.3750
    0.3750   -0.8750    0.3750
    0.7500   -0.7500    0.2500
```

Αν το v είναι διάνυσμα τότε η **diag(v)** μας δίνει τετραγωνικό διαγώνιο πίνακα με τα στοιχεία του v στη διαγώνιο. Από την άλλη αν ο A είναι τετραγωνικός πίνακας η **diag(A)** μας δίνει το διάνυσμα των διαγώνιων στοιχείων του A. Αν τώρα ο A δεν είναι τετραγωνικός πίνακας το **diag(A)** είναι το διάνυσμα $(a_{11}, a_{22}, \ldots.)$.

Παράδειγμα 2.7.3
Θα κατασκευάσουμε το διαγώνιο πίνακα

$$\begin{bmatrix} -5 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -3 & 0 \\ 0 & 0 & 0 & 0 & 4 \end{bmatrix}$$

```
>> diag([-5 3 1 -3 4])
ans =
    -5     0     0     0     0
     0     3     0     0     0
     0     0     1     0     0
     0     0     0    -3     0
     0     0     0     0     4
```

Η συνάρτηση **rref(A)** μας δίνει την ανηγμένη κλιμακωτή μορφή ενός πίνακα A.

Παράδειγμα 2.7.4
Θα βρούμε την ανηγμένη κλιμακωτή μορφή του πίνακα

$$A = \begin{bmatrix} 1 & 2 & 0 & -1 \\ 3 & 4 & 1 & 2 \\ -2 & 3 & 2 & 5 \end{bmatrix}$$

```
>> A=[1 2 0 -1; 3 4 1 2; -2 3 2 5];

>> R=rref(A)
R =
    1.0000         0         0    0.2727
         0    1.0000         0   -0.6364
         0         0    1.0000    3.7273
```

Ας δούμε τον πίνακα σε κλασματική μορφή (format rat):

```
>> format rat
>> R
R =
      1           0           0          3/11
      0           1           0         -7/11
      0           0           1         41/11
```

Με την **eig(A)** παίρνουμε σε ένα διάνυσμα στήλη τις ιδιοτιμές ενός τετραγωνικού πίνακα A. Με την **[V, D] = eig(A)** παίρνουμε τον διαγώνιο πίνακα D που έχει στη διαγώνιο τις ιδιοτιμές του A και τον πίνακα V που έχει σε στήλες τα αντίστοιχα ιδιοδιανύσματα, έτσι ώστε

$$AV = VD$$

Παράδειγμα 2.7.5
Θα βρούμε τις ιδιοτιμές του πίνακα

$$A=\begin{bmatrix} 2 & 0 & -2 \\ 0 & 4 & 0 \\ -2 & 0 & 5 \end{bmatrix}$$

```
>> A=[2 0 -2; 0 4 0; -2 0 5];

>> eig(A)
ans =
      1
      4
      6
```

Ας βρούμε τώρα τόσο τις ιδιοτιμές όσο και τα αντίστοιχα ιδιοδιανύσματα του A:

```
>> [P,D]=eig(A)
P =
   -0.8944         0   -0.4472
         0    1.0000         0
   -0.4472         0    0.8944
D =
     1     0     0
     0     4     0
     0     0     6
```

2.7.1 Πινακοσυναρτήσεις

Έχουμε ήδη πει ότι όταν μια συνάρτηση βιβλιοθήκης ,όπως οι sin, exp και sqrt, έχει σαν όρισμα ένα πίνακα, τότε αυτή μας δίνει τον πίνακα των τιμών της συνάρτησης σε

κάθε στοιχείο του. Συχνά συμβαίνει οι ίδιες αυτές συναρτήσεις να έχουν διαφορετικό νόημα στη Γραμμική Άλγεβρα.

Για παράδειγμα, μπορούμε να ορίσουμε την τετραγωνική ρίζα τετραγωνικού πίνακα A. Αν

$$X^2 = A \quad \text{τότε} \quad \sqrt{A} = X\,.$$

Κάθε μη ιδιάζων πίνακας έχει τουλάχιστον δύο τετραγωνικές ρίζες. Επίσης μπορούμε να ορίσουμε τον εκθετικό πίνακα ως εξής:

$$e^A = I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \cdots$$

όπου I ο ταυτοτικός πίνακας. Ο λογάριθμος ενός πίνακα είναι αντίστροφος της εκθετικής συνάρτησης

Οι πινακοσυναρτήσεις ή συναρτήσεις πινάκων της MATLAB χαρακτηρίζονται από το τελικό *m* που εμφανίζεται στο όνομά τους. Έτσι η **sqrtm** μας δίνει την τετραγωνική ρίζα πίνακα, η **expm** τον εκθετικό πίνακα, και η **logm** τον λογάριθμο πίνακα.

Παράδειγμα 2.7.6

Ας δούμε τι μας δίνουν οι sqrt(A) και sqrtm(A) όταν

$$A = \begin{bmatrix} 17 & 8 \\ 8 & 17 \end{bmatrix}$$

```
>> A=[17 8; 8 17];
>> sqrt(A)
ans =
    4.1231    2.8284
    2.8284    4.1231
>> X=sqrtm(A)
X =
    4.0000    1.0000
    1.0000    4.0000
>> X^2
ans =
   17.0000    8.0000
    8.0000   17.0000
```

Παράδειγμα 2.7.7

Θα βρούμε τον e^X, όπου

$$X = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}$$

```
>> X=[1 2;0 1];
>> A=expm(X)
A =
    2.7183    5.4366
         0    2.7183
```

Αν χρησιμοποιήσουμε την αντίστροφη συνάρτηση logm για τον $A = e^X$ θα πρέπει φυσικά να πάρουμε τον X:

```
>> logm(A)
ans =
    1.0000    2.0000
    0.0000    1.0000
```

Αν αντί των expm και logm χρησιμοποιήσουμε τις exp και log, παίρνουμε τα πιο κάτω αποτελέσματα:

```
>> exp(X)
ans =
    2.7183    7.3891
    1.0000    2.7183

>> log(A)

Warning: Log of zero.

ans =
    1.0000    1.6931
      -Inf    1.0000
```

Μια άλλη χρήσιμη συνάρτηση είναι η **funm** με την οποία μπορούμε να υπολογίσουμε άλλες πινακοσυναρτήσεις. Η

funm(A,@fname)

μας δίνει την εκδοχή της συνάρτησης fname για πίνακες, με την προϋπόθεση ότι η (μαθηματική) συνάρτηση fname(x) έχει σειρά Taylor η οποία συγκλίνει για όλα τα x. Για παράδειγμα, οι **funm(A,@exp)** και **expm(A)** είναι ισοδύναμες, όπως είναι και οι **funm(A,@log)** και **logm(A)**. Η funm όμως είναι γενική και δουλεύει και για άλλες πινακοσυναρτήσεις που δεν έχουν την αντίστοιχή τους m-συνάρτηση στη MATLAB, όπως οι cos, sin, cosh και sinh. Θα πρέπει πάντως να γνωρίζουμε ότι η **funm** δεν δίνει πάντα αξιόπιστα αποτελέσματα.

Παράδειγμα 2.7.8

Θα υπολογίσουμε την πινακοσυνάρτηση sin(A), όπου

$$A=\begin{bmatrix} 1 & 2 & 3 \\ 0 & \frac{\pi}{2} & \pi \\ 0 & 0 & 1 \end{bmatrix}$$

```
>> A=[ 1 2 3; 0 pi/2 pi; 0 0 1];

>> funm(A,@sin)

ans =
    0.8415    0.5555         0
         0    1.0000    0.8725
         0         0    0.8415
```

Ας συγκρίνουμε τώρα τα αποτελέσματα με την εντολή sin(A) της MATLAB:

```
>> sin(A)

ans =
    0.8415    0.9093    0.1411
         0    1.0000    0.0000
         0         0    0.8415
```

2.8 Ασκήσεις

2.1 Επαναλάβετε στη MATLAB όλα τα παραδείγματα της παραγράφου 2.1 και αποθηκεύστε την εργασία σας σε αρχείο με το όνομά σας.

2.2 Αν ο A είναι ένας $m \times n$ πίνακας και ο x ένας αριθμός τι μας δίνουν οι πιο κάτω πράξεις στην MATLAB:

(α) x*A (β) A*x (γ) A/x (δ) A+x (ε) A-x

2.3 Επαναλάβετε στη MATLAB όλα τα παραδείγματα της παραγράφου 2.2 και αποθηκεύστε την εργασία σας σε αρχείο με το όνομά σας.

2.4 Για το διάνυσμα *u* η συνάρτηση βιβλιοθήκης **length** μας έδωσε το εξής:

```
>> length(u)
ans =
    15
```

Να βρεθούν τα διανύσματα που προκύπτουν από τις πιο κάτω παραστάσεις:

(α) u(2:10) (β) u(1:3:15) (γ) u(2:2:15)

2.5 Για τον πίνακα A η συνάρτηση βιβλιοθήκης **size** μας έδωσε το εξής:

```
>> size(A)
ans =
    11    10
```

Να βρεθούν τα διανύσματα που προκύπτουν από τις πιο κάτω παραστάσεις:

(α) A(:,2:10)
(β) A(1:2:10,1:3:10)
(γ) A(2:7,:)
(δ) A(1:4:12,:)

2.6 Έστω το διάστημα [0, 2] το οποίο διαμερίζουμε ομοιόμορφα με 101 σημεία. Κατασκευάστε το διάνυσμα x που περιέχει τις συντεταγμένες των σημείων αυτών.

2.7 Με την εντολή help μπορείτε να μάθετε περισσότερα στοιχεία για τις πιο κάτω συναρτήσεις:

Συνάρτηση	Ερμηνεία
hadamard	**πίνακας Hadamard**
hankel	**πίνακας Hankel**
rosser	**πίνακας Rosser**
toeplitz	**πίνακας Toeplitz**
vander	**πίνακας Vandermonde**
wilkinson	**πίνακας Wilkinson**

(α) Δώστε σύντομη περιγραφή κάθε συνάρτησης.
(β) Δώστε δύο παραδείγματα για τη καθεμιά με ένα 4×4 και ένα 5×5 πίνακα.

2.8 Κατασκευάστε το διάνυσμα x που περιέχει τις συντεταγμένες των σημείων που ξεκινούν από το 1 και φτάνουν στο 3 με βήμα 0.1.

2.9 Έστω το διάστημα [0, 5] το οποίο διαμερίζουμε ομοιόμορφα με βήμα 0.1. Κατασκευάστε το διάνυσμα που περιέχει τις τιμές της συνάρτησης $\cos(x)$ στα σημεία της διαμέρισης.

2.10 Έστω το διάστημα [0, 4] το οποίο διαμερίζουμε ομοιόμορφα με βήμα 0.1. Κατασκευάστε το διάνυσμα που περιέχει τις τιμές της συνάρτησης $x^2 - 3x$ στα σημεία της διαμέρισης.

2.11 Έστω το διάστημα [0, 2] το οποίο διαμερίζουμε ομοιόμορφα με βήμα 0.05. Κατασκευάστε το διάνυσμα που περιέχει τις τιμές της συνάρτησης $x^2 = 1$ στα σημεία της διαμέρισης.

2.12 Ποιους πίνακες παίρνουμε με τις εκφράσεις

(α) [1:3,2:4]
(β) [1:3;2:4]
(γ) [1:6;5:-1:0]
(δ) [1:6;2:7;3:8;4:9;5:10]
(ε) [1:3;2:3,1;3:-1:1]

2.13 Κατασκευάστε με σύντομο τρόπο τους πιο κάτω πίνακες:

(α) $\begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 3 & 5 & 7 \\ 1 & 4 & 7 & 10 \\ 1 & 5 & 9 & 13 \end{bmatrix}$

(β) $(19,16,13,10,7,4,1,-2,-5,-8)$

(γ) $\begin{bmatrix} 1 & 3 & 5 & 7 & 9 & 11 \\ 10 & 8 & 6 & 4 & 2 & 0 \\ 4 & 8 & 12 & 16 & 20 & 24 \end{bmatrix}$

(δ) $\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \\ 9 & 10 \\ 11 & 12 \\ 13 & 14 \\ 15 & 16 \end{bmatrix}$

2.14 Κατασκευάστε με σύντομο τρόπο τους πιο κάτω πίνακες:

(α) $\begin{bmatrix} 1 & \sqrt{2} & \sqrt{3} & \sqrt{4} \\ 1 & \sqrt{3} & \sqrt{5} & \sqrt{7} \\ 1 & \sqrt{4} & \sqrt{7} & \sqrt{10} \\ 1 & \sqrt{5} & \sqrt{9} & \sqrt{13} \end{bmatrix}$

(β) $(1,4,9,16,25,36,49,64,81,100,121,144)$

(γ) $\begin{bmatrix} 1 & e & e^2 & e^3 \\ 1 & e^2 & e^4 & e^6 \\ 1 & e^3 & e^6 & e^9 \end{bmatrix}$

2.15 Με μια μόνο εντολή της MATLAB κατασκευάστε

(α) τον υποπίνακα που αποτελείται από τη $2^η$ και την $3^η$ στήλη του 4×4 πίνακα *A*.

(β) τον υποπίνακα που αποτελείται από τη $1^η$ και την $3^η$ γραμμή του 4×5 πίνακα *A*.

(γ) τον υποπίνακα που αποτελείται από τις στήλες 2 μέχρι 4 του 4×5 πίνακα *A*.

(δ) τον υποπίνακα που αποτελείται από τη $2^η$ και την $4^η$ γραμμή του 4×5 πίνακα *A*.

(ε) τον υποπίνακα του *A* που αποτελείται από τις γραμμές 1, 3 και 5 του 5×6 πίνακα *A*.

2.16 Κατασκευάστε τους κάτωθι υποπίνακες του πίνακα *A*:

(α) το διάνυσμα που περιέχει το $1^ο$, το $4^ο$, το $7^ο$ και το $10^ο$ στοιχείο της τελευταίας γραμμής του *A* (δεδομένου ότι ο *A* έχει τουλάχιστον 10 στήλες)

(β) το διάνυσμα που περιέχει το $9^ο$, το $7^ο$, το $5^ο$ και το $3^ο$ στοιχείο της τελευταίας στήλης του *A* (δεδομένου ότι ο *A* έχει τουλάχιστον 9 γραμμές)

(γ) τον υποπίνακα που προκύπτει από την τομή των γραμμών 1, 4, 7 και 10 και τον στηλών 3, 5, 7 και 9 του *A*

(δ) τον υποπίνακα που περιέχει το $1^ο$, $2^ο$, $3^ο$ και $7^ο$ στοιχείο της τελευταίας γραμμής του *A*.

(ε) τον υποπίνακα που προκύπτει από την τομή των γραμμών 5, 6 και 9 και των στηλών 1, 4 και 5.

2.17 Τι παριστάνει ο πίνακας B = A(1:end,:);

2.18 Έστω *A* ένας *m*×*n* πίνακας MATLAB.

(α) Πως κατασκευάζεται το διάνυσμα που περιέχει όλα τα στοιχεία του *A* κατά στήλη, από την πρώτη μέχρι την τελευταία;

(β) Πως κατασκευάζεται το διάνυσμα που περιέχει όλα τα στοιχεία του *A* κατά γραμμή, από την πρώτη μέχρι την τελευταία;

2.19 Έστω ο *m*×*p* πίνακας *A* και ο *p*×*m* πίνακας *B*. Τι παριστάνουν τα πιο κάτω;

(α) A(i,:) (β) B(:,j) (γ) A(i,:)*B(:,j)

2.20 Κατασκευάστε με απλό τρόπο τους πιο πίνακες:

$$A=\begin{bmatrix}2&2&0&0&0\\2&2&0&0&0\\3&0&4&0&0\\0&4&0&4&0\end{bmatrix},\quad B=\begin{bmatrix}1&0&0&0&0\\0&2&2&2&2\\0&2&2&2&2\\0&2&2&2&2\end{bmatrix},\quad C=\begin{bmatrix}1&2\\3&4\\1&2\\3&4\\1&2\\3&4\\1&2\\3&4\end{bmatrix}$$

2.21 Κατασκευάστε με απλό τρόπο τους πιο πίνακες: πίνακες:

$$A=\begin{bmatrix}2&0&0&0&0\\0&1&1&1&1\\0&1&1&1&1\\0&0&0&2&0\end{bmatrix},\quad B=\begin{bmatrix}1&1&1&1&1\\1&2&2&2&2\\1&2&2&2&2\\1&2&2&2&2\end{bmatrix},\quad C=\begin{bmatrix}1&0\\0&1\\1&0\\0&1\\1&0\\0&1\\1&0\\0&1\end{bmatrix}$$

2.22 Κατασκευάστε με δύο τρόπους τον πιο πίνακα:

$$A=\begin{bmatrix}2&0&0&0&0&0\\0&2&0&0&0&0\\0&0&2&2&0&0\\0&0&2&2&0&0\\0&0&0&0&2&0\\0&0&0&0&0&2\end{bmatrix}$$

2.23 Αν οι A, B και C είναι $m\times n$ πίνακες βρείτε τις διαστάσεις των πιο κάτω πινάκων:

(α) cat(1,A,B)
(β) cat(1,A,B,C)
(γ) cat(2,A,B)
(δ) cat(2,A,B,C)
(ε) cat(3,A,B)
(στ) cat(3,A,B,A,B)

2.24 Επαναλάβετε στη MATLAB όλα τα παραδείγματα της παραγράφου 2.3 και αποθηκεύστε την εργασία σας σε αρχείο με το όνομά σας.

2.25 Επαναλάβετε στη MATLAB όλα τα παραδείγματα της παραγράφου 2.4 και αποθηκεύστε την εργασία σας σε αρχείο με το όνομά σας.

2.26 Υπολογίστε το εσωτερικό και το εξωτερικό γινόμενο των διανυσμάτων:

(α) $u = (3, -1, 2)$, $v = (1, 2, -1)$

(β) $u = (2, -4, 0), \quad v = (3, 2, -1)$
(γ) $u = (1, 0, 1), \quad v = (4, 3, -5)$

2.27 Επαναλάβετε στη MATLAB όλα τα παραδείγματα της παραγράφου 2.5 και αποθηκεύστε την εργασία σας σε αρχείο με το όνομά σας.

2.28 Να βρεθούν οι ανηγμένοι κλιμακωτοί πίνακες των

$$A=\begin{bmatrix} 1 & 1 & -3 & -1 \\ 2 & 1 & -2 & 1 \\ 1 & 1 & 1 & 3 \\ 1 & 2 & -3 & 1 \end{bmatrix}, \quad B=\begin{bmatrix} 1 & -1 & 1 & 1 & 1 \\ 1 & -1 & 2 & 3 & 1 \\ 2 & -2 & 1 & 0 & 2 \\ 1 & 1 & -1 & -3 & 3 \end{bmatrix}, \quad C=\begin{bmatrix} 1 & 1 & 0 & -3 & -1 \\ 1 & -1 & 2 & -1 & 0 \\ 4 & -2 & 6 & 3 & -4 \\ 2 & 4 & -2 & 4 & -7 \end{bmatrix}$$

και να τυπωθούν σε **format rat**.

2.29 Να βρεθούν οι ανηγμένοι κλιμακωτοί πίνακες των

$$A=\begin{bmatrix} 1 & -1 & 2 & 1 \\ 2 & 1 & -1 & 1 \\ 1 & -2 & 1 & 1 \end{bmatrix}, \quad B=\begin{bmatrix} i & 1-i & i & 0 \\ 1 & -2 & 0 & i \\ 1-i & -1+i & 1 & 1 \end{bmatrix}, \quad C=\begin{bmatrix} 0 & 1 & 1 & 1 & 2 & 2 \\ -1 & 4 & 3 & 3 & 4 & 7 \\ 2 & 1 & 3 & 2 & 8 & 3 \\ 3 & 1 & 4 & -1 & 4 & 0 \\ 5 & 2 & 7 & 0 & 10 & 2 \end{bmatrix}$$

και να τυπωθούν σε **format rat**.

2.30 Να λυθεί το γραμμικό σύστημα:

$$\begin{aligned} x_1 - 2x_2 + 3x_3 - 4x_4 &= -8 \\ 2x_1 - 3x_2 + 4x_3 - x_4 &= 2 \\ 3x_1 - 4x_2 + x_3 - 2x_4 &= -8 \\ 4x_1 - x_2 + 2x_3 - 3x_4 &= -6 \end{aligned}$$

2.31 Να λυθεί το γραμμικό σύστημα:

$$\begin{aligned} (1+i)\,x_1 + (2+i)\,x_2 &= 5 \\ (2-2i)\,x_1 + i\,x_2 &= 1+2i \end{aligned}$$

2.32 Να βρεθούν και να γραφούν σε format rat οι αντίστροφοι των πιο κάτω πινάκων (αν υπάρχουν)

$$A=\begin{bmatrix} 1 & -1 & 2 \\ 3 & 2 & 4 \\ 0 & 1 & -2 \end{bmatrix}, \quad B=\begin{bmatrix} 1 & 3 & 1 \\ 0 & 1 & -2 \\ 2 & 5 & 4 \end{bmatrix}, \quad C=\begin{bmatrix} 6 & -1 & 2 \\ 0 & -2 & 5 \\ 2 & 1 & -2 \end{bmatrix}$$

2.33 Να βρεθούν και να γραφούν σε format rat οι αντίστροφοι των πιο κάτω πινάκων (αν υπάρχουν).

$$A=\begin{bmatrix} -1 & 1 & 1 \\ 12 & 2 & -5 \\ -4 & 2 & 3 \end{bmatrix}, \quad B=\begin{bmatrix} 1 & 2 & -3 & 1 \\ 2 & 4 & 1 & 3 \\ -1 & 1 & 1 & 0 \\ -2 & 2 & -5 & -1 \end{bmatrix}, \quad C=\begin{bmatrix} 1 & i & 0 \\ i & -1 & 1+i \\ 1-i & 0 & 2 \end{bmatrix}$$

2.34 Να βρεθούν οι βαθμοί των πιο κάτω πινάκων

$$A=\begin{bmatrix}2 & -1 & 0 & 1\\ 1 & 2 & 1 & -1\\ 2 & 9 & 4 & -5\end{bmatrix},\quad B=\begin{bmatrix}1 & 1 & 2 & -3 & 4\\ 0 & 2 & 4 & -6 & 6\\ 1 & 3 & 4 & -5 & 8\\ 1 & 3 & 5 & -7 & 9\end{bmatrix},\quad C=\begin{bmatrix}1 & -1 & 2\\ 3 & 2 & 4\\ 0 & 1 & -2\end{bmatrix}$$

2.35 Να βρεθούν οι βαθμοί των πιο κάτω πινάκων

$$A=\begin{bmatrix}1 & 2 & 4 & -1\\ 0 & 1 & 1 & 2\\ -2 & 3 & -1 & 1\end{bmatrix},\quad B=\begin{bmatrix}1 & 0 & -2\\ 2 & 1 & 3\\ 4 & 1 & -1\\ -1 & 2 & 1\end{bmatrix},\quad C=\begin{bmatrix}0 & 1 & -2 & 3\\ 1 & -3 & 2 & 1\\ 1 & -1 & -2 & 7\\ 1 & 0 & -4 & 10\end{bmatrix}$$

2.36 Αν $AX = B$ όπου

$$A=\begin{bmatrix}1 & 0 & -1\\ 2 & 1 & -2\\ 3 & 4 & 1\end{bmatrix}\quad \text{και}\quad B=\begin{bmatrix}1 & 4\\ 2 & 3\\ 0 & 6\end{bmatrix},$$

να βρεθεί και να γραφεί ο X σε format rat.

2.37 Αν $AX = B$ όπου

$$A=\begin{bmatrix}1 & 2\\ 3 & 5\end{bmatrix}\quad \text{και}\quad B=\begin{bmatrix}5 & -6\\ 4 & 4\end{bmatrix},$$

να βρεθεί και να γραφεί ο X σε format rat.

2.38 Αν $AX = B$ όπου

$$A=\begin{bmatrix}3 & 2 & 1\\ -1 & 1 & -3\\ 2 & 1 & -0\end{bmatrix}\quad \text{και}\quad B=\begin{bmatrix}6 & 0 & 12\\ 0 & 3 & 0\\ 12 & 0 & 6\end{bmatrix},$$

να βρεθεί και να γραφεί ο X σε format rat.

2.39 Αν R είναι ο ανηγμένος κλιμακωτός του

$$A=\begin{bmatrix}2 & -1 & 0 & 1\\ 1 & 2 & 1 & -1\\ 2 & 9 & 4 & -5\end{bmatrix}$$

να βρεθεί τετραγωνικός P πίνακας τέτοιος ώστε

$$R = PA.$$

Γράψτε τον P σε format rat.

2.40 Να υπολογιστούν οι ορίζουσες των πιο κάτω πινάκων

$$A=\begin{bmatrix}3 & 2 & 1\\ -1 & 1 & -3\\ 2 & 1 & -0\end{bmatrix},\quad B=\begin{bmatrix}1 & i & 0\\ i & -1 & 1+i\\ 1-i & 0 & 2\end{bmatrix},\quad C=\begin{bmatrix}0 & 1 & -2 & 3\\ 1 & -3 & 2 & 1\\ 1 & -1 & -2 & 7\\ 1 & 0 & -4 & 10\end{bmatrix}$$

2.41 Να υπολογιστούν οι ορίζουσες των πιο κάτω πινάκων

$$A=\begin{bmatrix}0 & 1 & 3 & 1\\ 1 & -2 & -2 & 2\\ 3 & 4 & 2 & -2\\ 4 & 3 & -1 & 1\end{bmatrix},\quad B=\begin{bmatrix}2 & 4 & -2 & -2\\ 1 & 3 & 1 & 2\\ 1 & 3 & 1 & 3\\ -1 & 2 & 1 & 2\end{bmatrix},\quad C=\begin{bmatrix}3 & 1 & 1 & 1\\ 1 & 3 & 1 & 1\\ 1 & 1 & 3 & 1\\ 1 & 1 & 1 & 3\end{bmatrix}$$

2.42 Να υπολογιστούν οι ιδιοτιμές και τα ιδιοδιανύσματα των πιο κάτω πινάκων

$$A=\begin{bmatrix}3 & -2 & -2\\ 2 & -2 & -4\\ -1 & 2 & 4\end{bmatrix},\quad B=\begin{bmatrix}1 & -3 & 3\\ 3 & -5 & 3\\ 6 & -6 & 4\end{bmatrix},\quad C=\begin{bmatrix}3 & 1 & 1\\ 2 & 4 & 2\\ 1 & 1 & 3\end{bmatrix}$$

2.43 Βρείτε την ορίζουσα και τον δείκτη κατάστασης του πίνακα gallery(3).

2.44 Βρείτε τις ιδιοτιμές και τα ιδιοδιανύσματα του πίνακα gallery(5).

2.45 Έστω το γραμμικό σύστημα

$$\boldsymbol{Ax = b}$$

όπου ***b*** το διάνυσμα στήλη που περιέχει **τα τελευταία 5 ψηφία της ταυτότητάς σας.** Κατασκευάστε τον 5×5 πίνακα **A** δημιουργώντας πρώτα ένα τυχαίο 5×5 πίνακα **B** (θα χρησιμοποιήσετε τη συνάρτηση **rand**) και αντικαθιστώντας την $1^{η}$ του γραμμή με το $\mathbf{b}^{\mathbf{T}}$ (χρησιμοποιήστε κατάλληλη εντολή της MATLAB).

(α) Βρείτε το ελάχιστο, το μέγιστο, το ελάχιστο κατ'απόλυτη τιμή και το μέγιστο κατ'απόλυτη τιμή στοιχείο του ***b***.

(β) Ταξινομήστε το ***b***.

(γ) Υπολογίστε τα $\boldsymbol{b^{T}b}$ και $\boldsymbol{bb^{T}}$.

(δ) Κατασκευάστε τον **επαυξημένο πίνακα** $\boldsymbol{C = [A \mid b]}$ και βρείτε την **ανηγμένη κλιμακωτή μορφή** του.

(ε) Υπολογίστε τα $\boldsymbol{A + B}$, $\boldsymbol{AB}$, $\boldsymbol{AC}$, $\boldsymbol{Ab}$ και $\boldsymbol{A - I}$

(στ) υπολογίστε τις **διαστάσεις**, την **ορίζουσα**, το **βαθμό**, το **ίχνος**, τον **αντίστροφο** και τις **ιδιοτιμές** του ***A***.

(ζ) επιλύστε τα συστήματα $\boldsymbol{Ax = 0}$, $\boldsymbol{Ax = b}$ και $\boldsymbol{Ax = c}$ όπου ***c*** διάνυσμα στήλη με όλα τα στοιχεία του μονάδες.

Αποθηκεύστε την εργασία σας σε αρχείο με το όνομα **epitheto_onoma.txt**

3 M-FILES

Για να εκμεταλλευτούμε πλήρως τις ικανότητες της MATLAB, πρέπει να μάθουμε πώς να δημιουργούμε μεγάλες και συχνά πολύπλοκες ακολουθίες εντολών. Ο καλύτερος τρόπος για να επιτύχουμε αυτό το στόχο είναι με τη χρήση αρχείων που καλούνται "m-files" αφού έχουν ως επίθεμα (extension) το .m, π.χ.

script1.m, gausse.m και function2.m

Τα m-files της MATLAB είναι τα αντίστοιχα των συναρτήσεων (functions) και των υπορουτινών (subroutines) που συναντούμε σε άλλες γλώσσες προγραμματισμού όπως η C και η FORTRAN. Τα m-files που δημιουργεί ο χρήστης συμπληρώνουν τις συναρτήσεις βιβλιοθήκης της MATLAB που είναι επίσης m-files.

Τα m-files διακρίνονται σε

- **Αρχεία script** *ή* **αρχεία εντολών** (script m-files or command files) τα οποία δεν έχουν ορίσματα εισόδου και εξόδου αλλά εκτελούν μια ακολουθία εντολών σε μεταβλητές του χώρου εργασίας, και σε
- **Αρχεία συναρτήσεων** (function m-files) τα οποία περιλαμβάνουν μια γραμμή ορισμού συνάρτησης, δέχονται ορίσματα εισόδου και επιστρέφουν μεταβλητές εξόδου, και των οποίων οι εσωτερικές μεταβλητές είναι τοπικές (εκτός αν δηλωθούν ως ολικές με την εντολή global).

Τόσο τα αρχεία script όσο και τα αρχεία συναρτήσεων

- Δημιουργούνται ξεχωριστά με κάποιο συντάκτη (editor) όπως ο notepad ή ο wordpad ή ακόμα με το συντάκτη της MATLAB:

  Για νέο αρχείο: File → New → M-file
  Για υπάρχον αρχείο: File → Open

  Μπορούμε, επίσης, να ανοίξουμε τον editor της MATLAB και με την εντολή

  ```
  >> edit
  ```

- Πρέπει να βρίσκονται στον φάκελο εργασίας (working directory) ή στον φάκελο (directory) της MATLAB ή στον δρόμο αναζήτησης (research path) της MATLAB.
- Μπορούν να καλούν άλλα m-files ή ακόμα τον ίδιο τον εαυτό τους (*αναδρομικά* m-files).

3.1 Αρχεία script

Τα αρχεία τύπου script περιέχουν μια ακολουθία εντολών της MATLAB η οποία εκτελείται αν γράψουμε το όνομα του αρχείου (χωρίς το επίθεμα .m), π.χ.

```
>> script1
```

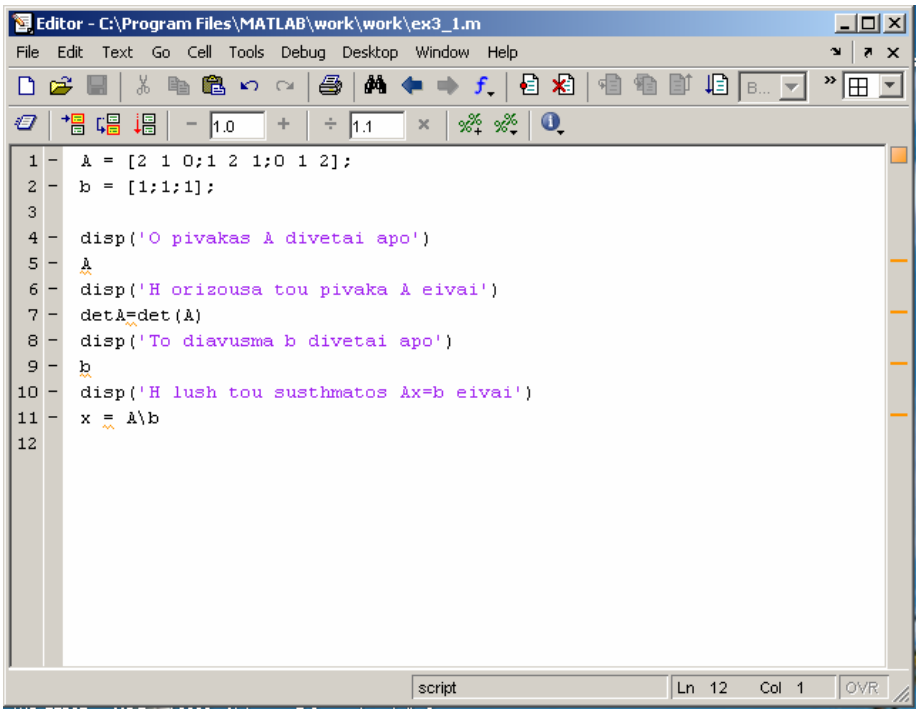
Τα script files είναι χρήσιμα για την εισαγωγή δεδομένων (π.χ. μεγάλων πινάκων) και για την επανάληψη μεγάλων ακολουθιών εντολών για διαφορετικά δεδομένα.

Παράδειγμα 3.1.1

Έστω ότι καταχωρήσαμε τις εξής εντολές σε ένα αρχείο (χρησιμοποιώντας τον editor της MATLAB) το οποίο ονομάσαμε ex3_1.m

```
A = [2 1 0;1 2 1;0 1 2];
b = [1;1;1];
disp('O pivakas A divetai apo')
A
disp('H orizousa tou pivaka A eivai')
detA=det(A)
disp('To diavusma b divetai apo')
b
disp('H lush tou susthmatos Ax=b eivai')
x = A\b
```

Το αρχείο φαίνεται πιο κάτω:



Τότε, αν γράψουμε το όνομα του αρχείου (χωρίς το επίθεμα .m) θα εκτελεστούν όλες οι εντολές και θα πάρουμε τα εξής:

```
>> ex3_1
```

```
O pivakas A divetai apo
A =
     2     1     0
     1     2     1
     0     1     2

H orizousa tou pivaka A eivai

detA =
     4

To diavusma b divetai apo

b =
     1
     1
     1

H lush tou susthmatos Ax=b eivai

x =
    0.5000
    0.0000
    0.5000
```

Παράδειγμα 3.1.2 Μετατροπή θερμοκρασίας από Κελσίου σε Φαρέναϊτ.

Το εξής script το έχουμε καταχωρήσει στο αρχείο C2F.m. Ζητά από τον χρήστη να δώσει την θερμοκρασία σε βαθμούς Κελσίου και την μετατρέπει σε βαθμούς Φάρεναϊτ.

```
%Metatroph 0ermokrasias apo Kelsiou se Farenheit
format compact
C = input('Dwste th 0ermokrasia se ba0mous Kelsiou: ');
disp('H 0ermokrasia se ba0mous Farenheit eivai ')
F = 9*C/5 + 32
%Telos tou C2F.m
```

Ας το τρέξουμε για διάφορες τιμές:

```
>> C2F
Dwste th 0ermokrasia se ba0mous Kelsiou: 40
H 0ermokrasia se ba0mous Farenheit eivai
F =
   104

>> C2F
Dwste th 0ermokrasia se ba0mous Kelsiou: 0
H 0ermokrasia se ba0mous Farenheit eivai
F =
    32

>> C2F
Dwste th 0ermokrasia se ba0mous Kelsiou: 18
H 0ermokrasia se ba0mous Farenheit eivai
F =
   64.4000
```

3.2 Αρχεία συναρτήσεων

Τα αρχεία συναρτήσεων (function m-files) περιέχουν μια ολοκληρωμένη ακολουθία εντολών της MATLAB με μεταβλητές εισόδου,

input1, input2, ….

με την οποία υπολογίζονται νέες μεταβλητές εξόδου

output1, output2, …..

Η δομή τους είναι η εξής:

1. **Επικεφαλίδα (header)**

   Μια γραμμή της μορφής:

   function [output1, output2, ….] = filename (input1, input2, ….)

   Το filename είναι το όνομα του function το οποίο αποθηκεύεται (υποχρεωτικά) στο m-file με όνομα filename.m. Προσέξτε ότι οι μεταβλητές εισόδου είναι σε παρενθέσεις ενώ οι μεταβλητές εξόδου βρίσκονται σε αγκύλες. Οι τελευταίες δεν είναι απαραίτητες αν έχουμε μόνο μια μεταβλητή εξόδου. Για παράδειγμα αντί

   function [out1] = parad(x, y)

   μπορούμε να γράψουμε

   function out1 = parad(x, y)

2. **Σχόλια (comments)**

   Αυτά ξεκινούν αναγκαστικά με το σύμβολο **%** και είναι προαιρετικά. Τα σχόλια μετά την επικεφαλίδα αποτελούν και το κείμενο βοήθειας για το function. Με διαφορετικά λόγια, αυτά εμφανίζονται αν γράψουμε:

   >> help filename

3. **Εντολές (statements)**

   Ακολουθία εντολών της MATLAB με την οποία υπολογίζονται οι μεταβλητές εξόδου

   Προκαταρκτικές εντολές
   output1 = …….
   output2 = …….
   κλπ

4. **Υποσυναρτήσεις (subfunctions)**

   Αυτές είναι εσωτερικές συναρτήσεις functions που περιέχονται στο αρχείο filename.m. Αναγνωρίζονται μόνο από τα functions που περιέχονται στο αρχείο αυτό. Ορίζονται με τον ίδιο ακριβώς τρόπο, π.χ.

   function [out1, out2,] = subfname (x, y, z)

Οι συναρτήσεις βιβλιοθήκης της MATLAB, όπως οι sin, max, exp και size, είναι function m-files. **Το όνομα μιας νέας function δεν πρέπει να συμπίπτει με όνομα συνάρτησης βιβλιοθήκης.**

Γενικά για την ονομασία μιας m-συνάρτησης ακολουθούμε τους κανόνες ονοματολογίας που ισχύουν και για τις μεταβλητές:

- *Το όνομα αρχίζει με γράμμα (του αγγλικού αλφαβήτου).*
- *Το όνομα περιέχει μόνο γράμματα, αριθμούς και υποπαύλες (underscore).*
- *Δεν χρησιμοποιούνται ονόματα που έχουν δεσμευτεί από τη MATLAB (π.χ. συναρτήσεις βιβλιοθήκης και εργαλειοθηκών).*
- *Προτιμούνται μικρά ονόματα για πρακτικούς λόγους αν και δεν υπάρχει περιορισμός στο μήκος των ονομάτων.*

Έστω για παράδειγμα ένα function με επικεφαλίδα

```
function [sum, prod] = onoma1( a, b, c)
```

όπου οι μεταβλητές εισόδου *a*, *b* και *c* είναι αριθμοί. Μπορούμε να καλέσουμε την onoma1 με διαφορετικούς τρόπους όπως:

```
>> onoma1( 3, 2, 1)
```

ή

```
>> [x,y] = onoma1( 3, 2, 1)
```

Αν για κάποιο λόγο γράψουμε

```
>> z = onoma1(3,2,1)
```

τότε στο *z* θα εκχωρηθεί η τιμή της πρώτης μεταβλητής εξόδου, δηλ. της sum.

Ακολουθεί η βοήθεια που παίρνουμε για το function. Το δεύτερο παράδειγμα παρουσιάζει ενδιαφέρον αφού περιλαμβάνει μια **υποσυνάρτηση** (subfunction).

```
>> help function

    FUNCTION Add new function.
    New functions may be added to MATLAB's vocabulary if they
    are expressed in terms of other existing functions. The
    commands and functions that comprise the new function must
    be put in a file whose name defines the name of the new
    function, with a filename extension of '.m'. At the top of
    the file must be a line that contains the syntax definition
    for the new function. For example, the existence of a file
    on disk called STAT.M with:

            function [mean,stdev] = stat(x)
            %STAT Interesting statistics.
            n = length(x);
            mean = sum(x) / n;
            stdev = sqrt(sum((x - mean).^2)/n);

    defines a new function called STAT that calculates the
    mean and standard deviation of a vector. The variables
    within the body of the function are all local variables.
    See SCRIPT for procedures that work globally on the work-
```

```
space.

A subfunction that is visible to the other functions in the
same file is created by defining a new function with the FUNCTION
keyword after the body of the preceding function or subfunction.
For example, avg is a subfunction within the file STAT.M:

       function [mean,stdev] = stat(x)
       %STAT Interesting statistics.
       n = length(x);
       mean = avg(x,n);
       stdev = sqrt(sum((x-avg(x,n)).^2)/n);

       %-------------------------
       function mean = avg(x,n)
       %MEAN subfunction
       mean = sum(x)/n;

Subfunctions are not visible outside the file where they are
defined.
Normally functions return when the end of the function is reached.
A RETURN statement can be used to force an early return.

See also SCRIPT, RETURN, VARARGIN, VARARGOUT, NARGIN,
NARGOUT, INPUTNAME, MFILENAME.
```

Ας δούμε ορισμένα παραδείγματα.

Παράδειγμα 3.2.1 sumprod.m

```
function [sum, prod] = sumprod (x1, x2, x3)

% function [sum, prod] = sumprod (x1, x2, x3)
%
% Paradeigma function m-file
% Ypologizei to a0roisma kai to ginomeno triwn ari0mwn
%
% Onoma function    : sumprod
% Onoma m-file      : sumprod.m
% Metablhtes eisodou: x1, x2, x3
% Metablhtes exodou : sum (a0roisma twn x1, x2, x3)
%                     prod (ginomeno twn x1, x2, x3)
%
    sum  = x1+x2+x3;
    prod = x1*x2*x3;

% Telos tou sumprod.m
```

Παρατηρούμε ότι ο editor της MATLAB χρησιμοποιεί διαφορετικά χρώματα για τις διάφορες δομές του αρχείου – μπλε για τη λέξη `function`, πράσινο για τα σχόλια και **μαύρο** για τις εντολές.

Ας δούμε τώρα τι γίνεται όταν ζητήσουμε βοήθεια για την εντολή (ή αρχείο) που δημιουργήσαμε:

```
>> help sumprod

  Paradeigma function m-file
  Ypologizei to a0roisma kai to ginomeno triwn ari0mwn

  Onoma function    : sumprod
  Onoma m-file      : sumprod.m
```

```
      Metablhtes eisodou: x1, x2, x3
      Metablhtes exodou : sum (a0roisma twn x1, x2, x3)
                          prod (ginomeno twn x1, x2, x3)
```

Τώρα, ας καλέσουμε το αρχείο με δεδομένα εισόδου, π.χ. 1, – 3 και 4:

```
>> sumprod(1, -3, 4)
ans =
     2
```

Μια και δεν καλέσαμε το αρχείο ζητώντας και τις δύο μεταβλητές εξόδου, πήραμε μόνο την πρώτη, δηλ. $x_1 + x_2 + x_3 = 1 - 3 + 4 = 2$. Για να πάρουμε και τις δύο μεταβλητές εξόδου, πρέπει να γράψουμε

```
>> [s,p] = sumprod(1, -3, 4)
s =
     2
p =
   -12
```

Σημειώνουμε ότι αν και μπορούμε να πάρουμε μόνο τη πρώτη μεταβλητή εξόδου (όπως είδαμε πιο πάνω) δεν μπορούμε να πάρουμε *μόνο* τη δεύτερη – το ίδιο ισχύει ακόμα και αν έχουμε περισσότερες από δύο μεταβλητές εξόδου.

Αν καλέσουμε το αρχείο μας με κάποια άλλα δεδομένα εισόδου, π.χ. – 12, 0 και 5, τότε θα πάρουμε

```
>> [s,p] = sumprod(-12, 0, 5)
s =
    -7
p =
     0
```

Τέλος, σχολιάστε το αποτέλεσμα του

```
>> sumprod(s,p,s+p)
ans =
   -14
```

Παράδειγμα 3.2.2 miscop.m

```
function [detA, rankA, Atrans, Ainv, A2] = miscop(A)

% function [detA, rankA, Atrans, Ainv, A2] = miscop(A)
%
% Briskei (a) thn orizousa
%         (b) ton ba0mo
%         (c) ton anastrofo
%         (d) ton antistrofo kai
%         (e) to tetragwno enos tetragwnikou pinaka.
%
% Shmeiwsh: den ginetai elegxos an o pinakac einai tetragwnikos
%           kai antistreyimos
%
    detA   = det(A)
    rankA  = rank(A)
    Atrans = A'
    Ainv   = inv(A)
    A2     = A*A

% Telos miscop.m
```

Ας δοκιμάσουμε να τρέξουμε το αρχείο μας με ένα τυχαίο 5×5 πίνακα *A* τον οποίο θα κατασκευάσουμε μέσω της εντολής `rand`, και θα δώσουμε σαν μεταβλητή εισόδου:

```
>> [detA, rankA, Atrans, Ainv, A2] = miscop(rand(5,5))

detA =
    0.1292

rankA =
     5

Atrans =
    0.2028    0.1987    0.6038    0.2722    0.1988
    0.0153    0.7468    0.4451    0.9318    0.4660
    0.4186    0.8462    0.5252    0.2026    0.6721
    0.8381    0.0196    0.6813    0.3795    0.8318
    0.5028    0.7095    0.4289    0.3046    0.1897

Ainv =
   -1.1396   -0.3483    2.6740   -0.5993   -0.7602
   -0.4075    0.0050   -0.4817    1.1704    0.2710
   -0.9387    0.7468    0.2439   -1.3784    1.3572
    0.8386   -0.6665   -0.4477    0.4721    0.5243
    1.8445    0.6295   -0.5205    0.5669   -1.7056

A2 =
    0.6250    1.2161    0.8255    1.1917    0.6430
    0.8460    1.2863    1.6404    1.3553    1.1332
    0.7987    1.4100    1.3315    1.4878    1.1335
    0.5266    1.2858    1.2905    0.7819    1.0582
    0.8029    1.5137    1.1266    1.1071    1.0082

detA =
    0.1292

rankA =
     5

Atrans =
    0.2028    0.1987    0.6038    0.2722    0.1988
    0.0153    0.7468    0.4451    0.9318    0.4660
    0.4186    0.8462    0.5252    0.2026    0.6721
    0.8381    0.0196    0.6813    0.3795    0.8318
    0.5028    0.7095    0.4289    0.3046    0.1897

Ainv =
   -1.1396   -0.3483    2.6740   -0.5993   -0.7602
   -0.4075    0.0050   -0.4817    1.1704    0.2710
   -0.9387    0.7468    0.2439   -1.3784    1.3572
    0.8386   -0.6665   -0.4477    0.4721    0.5243
    1.8445    0.6295   -0.5205    0.5669   -1.7056

A2 =
    0.6250    1.2161    0.8255    1.1917    0.6430
    0.8460    1.2863    1.6404    1.3553    1.1332
    0.7987    1.4100    1.3315    1.4878    1.1335
    0.5266    1.2858    1.2905    0.7819    1.0582
    0.8029    1.5137    1.1266    1.1071    1.0082
```

Παράδειγμα 3.2.3 stat1.m

```
function [mean,stdev] = stat1(x)

% function [mean,stdev] = stat1(x)
%
% Ypologizei tov meso oro kai thn tupikh
% apoklish evos diavusmatos x.
n = length(x);
mean = sum(x) / n;
stdev = sqrt(sum((x - mean).^2)/n);

% Telos tou stat.m
```

Ας χρησιμοποιήσουμε το πιο πάνω m-file για ένα τυχόν διάνυσμα x με 100 συνιστώσες. (Για να μην δούμε τις συνιστώσες του x βάζουμε ';' στο τέλος την εντολής που ορίζει το x).

```
>> x=rand(1,100);
>> [mean,stdev] = stat1(x)
mean =
    0.5286
stdev =
    0.2803
```

Παράδειγμα 3.2.4 stat2.m

Ας δούμε τώρα το ίδιο m-file όπως πιο πάνω, αλλά τώρα ας χρησιμοποιήσουμε μια *υποσυνάρτηση* για τον υπολογισμό του μέσου όρου (η οποία φαίνεται στο τέλος του m-file και καλείται avg) .

```
function [mean,stdev] = stat2(x)

% function [mean,stdev] = stat2(x)
%
% Ypologizei tov meso oro kai thn tupikh
% apoklish evos diavusmatos x.
%%

n = length(x);
mean = avg(x,n);
stdev = sqrt(sum((x-avg(x,n)).^2)/n);

%-------------------------
function mean = avg(x,n)
% uposuvarthsh pou upologizei tov meso oro
mean = sum(x)/n;
```

Χρησιμοποιώντας πάλι ένα (άλλο) τυχόν διάνυσμα με 100 συνιστώσες σαν δεδομένο εισόδου, έχουμε

```
>> y =rand(1,100);
>> [mean,stdev] = stat2(y)
mean =
    0.4652
stdev =
    0.2776
```

3.3 Μαθηματικές συναρτήσεις

Τα m-files είναι ιδιαίτερα χρήσιμα στον ορισμό (μαθηματικών) συναρτήσεων, όπως για παράδειγμα η

$$f_1(x) = \frac{4x^2}{x^4 + 2}.$$

Γράφουμε ένα m-file που ορίζει αυτή τη συνάρτηση, όπως φαίνεται πιο κάτω, και το αποθηκεύουμε σε ένα αρχείο με το όνομα f1.m το οποίο πρέπει να βρίσκεται στον φάκελο εργασίας (working directory) ή στον φάκελο (directory) της MATLAB ή στον δρόμο αναζήτησης (research path) της MATLAB.

```
function [F] = f1(x)
F = 4*x.^2./(x.^4+2);
% Telos tou f1.m
```

Για να πάρουμε την τιμή της f_1 για $x = 1$, γράφουμε απλά

```
>> f1(1)
ans =
    1.3333
```

Πράγματι,

```
>> 4*(1)^2/((1)^4+2)
ans =
    1.3333
```

Με τον ίδιο τρόπο παίρνουμε και άλλες τιμές:

```
>> f1(0)
ans =
     0
>> f1(-3)
ans =
    0.4337
>> f1(5)
ans =
    0.1595
```

Παρατήρηση

Η f1 δουλεύει και στην περίπτωση που το x είναι διάνυσμα αφού στο πιο πάνω m-file ορίσαμε τις **πράξεις κατά τα στοιχεία** του (δηλ. βάλαμε "." πριν από τα **"^"** και **"/"**). Για παράδειγμα, για το διάνυσμα

$$x = (-3, -2.5, -2, -1.5, -1, -0.5, 0, 0.5, 1, 1.5, 2, 2.5, 3)^T$$

έχουμε

```
>> x=[-3:0.5:3]';
```

```
>> f(x)
ans =
    0.4337
    0.6088
    0.8889
    1.2743
    1.3333
```

```
0.4848
     0
0.4848
1.3333
1.2743
0.8889
0.6088
0.4337
```

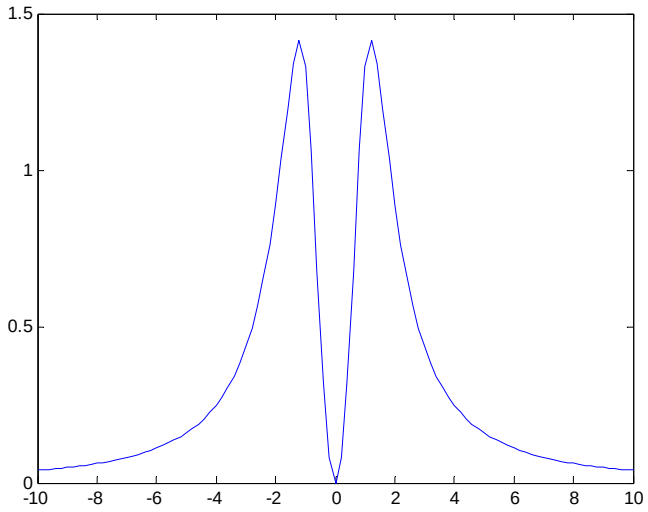
Αυτή η δυνατότητα θα μας φανεί πολύ χρήσιμη για την κατασκευή γραφημάτων στο Κεφ. 5. Προς το παρόν, σημειώνουμε ότι αν, για παράδειγμα, θέλουμε να κατασκευάσουμε τη γραφική παράσταση της πιο πάνω συνάρτησης για $x \in [-10,10]$, τότε διαμερίζουμε ομοιόμορφα το διάστημα αυτό με 101 σημεία (100 ίσα υποδιαστήματα) με την εντολή

```
>> x=linspace(-10,10,101);
```

και μετά χρησιμοποιούμε την εντολή

```
>> plot(x,f(x))
```

Παίρνουμε έτσι την γραφική παράστασητης f_1:



Η βοήθεια που δίνει η MATLAB για την εντολή linspace, έχει ως εξής:

```
>> help linspace

    LINSPACE Linearly spaced vector.
    LINSPACE(X1, X2) generates a row vector of 100 linearly
    equally spaced points between X1 and X2.

    LINSPACE(X1, X2, N) generates N points between X1 and X2.
    For N < 2, LINSPACE returns X2.

    Class support for inputs X1,X2:
       float: double, single

    See also logspace.
```

Παράδειγμα 3.3.1.
Η συνάρτηση $f_2(x) = \cos(4\pi x) - \sin(3\pi x)$ ορίζεται στο m-file f2.m που φαίνεται πιο κάτω:

```
function [G] = f2(x)
G = cos(4*pi*x) - sin(3*pi*x);
% Telos tou f2.m
```

Σημειώνουμε ότι για αυτή τη συνάρτηση δεν χρειάζεται να χρησιμοποιήσουμε **πράξεις κατά στοιχεία** μια και οι τριγωνομετρικές συναρτήσεις βιβλιοθήκης δουλεύουν και για διανύσματα.

Έχουμε

```
>> f2(-3.2)
ans =
   -1.7601
>> f2(0)
ans =
     1
>> f2(1.85)
ans =
    0.6787
>> f2(1)
ans =
    1.0000
>> f2([0:0.2:1])
ans =
    1.0000   -1.7601    0.8968    0.8968   -1.7601    1.0000
```

Σχολιάστε το τελευταίο αποτέλεσμα.

Παράδειγμα 3.3.2
Με τον πιο πάνω τρόπο μπορούμε να ορίσουμε συναρτήσεις πολλών μεταβλητών, αν χρησιμοποιήσουμε περισσότερες από μια μεταβλητές εισόδου. Έστω η συνάρτηση $f_3(x,y) = e^{-xy} + (x^2 + y^2)\cos(y)$ η οποία ορίζεται στο m-file f3.m που φαίνεται πιο κάτω:

```
function [F] = f3(x,y)
F = exp(-x.*y) + (x.^2 + y.^2).*cos(y);
%Telos tou f3.m
```

Έτσι, παίρνουμε τιμές της $f_3(x,y)$ ως εξής:

```
>> f3(0,0)
ans =
     1
```

```
>> f3(-4,1)

ans =
   63.7833
```

Ο πιο πάνω τρόπος ορισμού μαθηματικών συναρτήσεων, αν και δεν είναι ο μοναδικός, είναι ο πιο λειτουργικός ειδικά όταν γράφουμε προγράμματα τα οποία καλούν αυτές τις συναρτήσεις (όπως και θα δούμε στα επόμενα κεφάλαια).

3.3.1 Ανώνυμες συναρτήσεις

Η MATLAB περιλαμβάνει ένα απλό τύπο συνάρτησης η οποία καλείται **ανώνυμη συνάρτηση** (anonymous function) η οποία ορίζεται είτε στο παράθυρο εντολών είτε σ'ένα αρχείο script και είναι διαθέσιμη όσο διαρκεί η εργασία μας στο παράθυρο εντολών. Ας πάρουμε σαν παράδειγμα τη συνάρτηση

$$f_1 = \sin(2\pi x)$$

Η ανώνυμη συνάρτηση ορίζεται με την εντολή

f1 = @(x) sin(2*pi*x)

όπου

- *Το σύμβολο @ δηλώνει στη MATLAB ότι η f1 είναι συνάρτηση.*
- *Αμέσως μετά το σύμβολο @ ακολουθεί η μεταβλητή εισόδου της συνάρτησης.*
- *Τέλος ορίζεται η συνάρτηση.*

Το όνομα της συνάρτησης εμφανίζεται στο παράθυρο των μεταβλητών με τον χαρακτηρισμό **function_handle**. Αν καθαρίσουμε το χώρο εργασίας (με την εντολή clear) η ανώνυμη συνάρτηση θα διαγραφεί. Όπως και κάθε άλλη ενεργή μεταβλητή μια ανώνυμη συνάρτηση μπορεί να αποθηκευτεί σ'ένα αρχείο .mat με την εντολή save και να φορτωθεί εκ νέου με την εντολή load.

Οι ανώνυμες συναρτήσεις χρησιμοποιούνται όπως κάθε άλλη συνάρτηση. Για παράδειγμα, αφού ορίσουμε την f1 όπως είδαμε πιο πάνω, τότε μπορούμε να πούμε

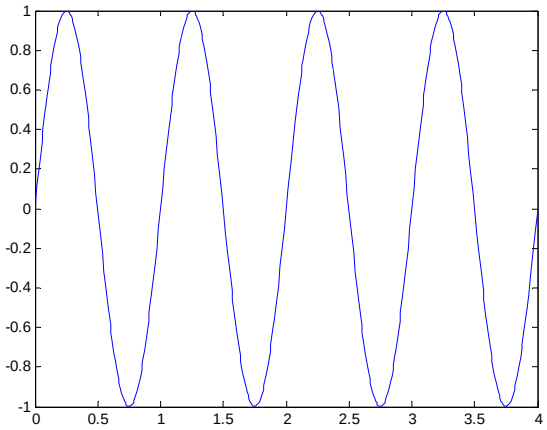
```
>> f1(0)
ans =
     0
>> f1(0.25)
ans =
     1
```

Η εντολή fplot

Η fplot είναι παράδειγμα συνάρτησης βιβλιοθήκης που έχει σαν όρισμα μια άλλη συνάρτηση. Η fplot έχει δύο ορίσματα εισόδου: μια συνάρτηση και το διάστημα στο οποίο θα κάνει το γράφημά της. Αν τη χρησιμοποιήσουμε με την ανώνυμη συνάρτηση που μόλις ορίσαμε

fplot(f1, [0 4])

παίρνουμε το ακόλουθο γράφημα:



Παράδειγμα 3.3.3

Θα ορίσουμε μια ανώνυμη συνάρτηση για την

$$g_1(x,y)=\sqrt{x^2+y^2}-2xy$$

Παρατηρούμε ότι τώρα έχουμε δύο μεταβλητές εισόδου, τις x και y. Άρα γράφουμε

g1 = @(x,y) sqrt(x.^2+y.^2) – 2*x.*y

Για να μπορεί να χρησιμοποιείται και με διανύσματα σαν μεταβλητές εισόδου ορίζουμε τις πράξεις κατά τα στοιχεία των (διανυσμάτων) x και y.

Ένας άλλος τρόπος ορισμού (μαθηματικών) συναρτήσεων είναι μέσω της εντολής **inline**, της οποίας η βοήθεια που δίνει η MATLAB φαίνεται πιο κάτω.

```
>> help inline

    INLINE Construct INLINE object.

    INLINE(EXPR) constructs an inline function object from the
    MATLAB expression contained in the string EXPR.  The input
    arguments are automatically determined by searching EXPR
    for variable names (see SYMVAR). If no variable exists,
    'x' is used.

    INLINE(EXPR, ARG1, ARG2, ...) constructs an inline
    function whose input arguments are specified by the
    strings ARG1, ARG2, ...  Multicharacter symbol names may
    be used.

    INLINE(EXPR, N), where N is a scalar, constructs an
    inline function whose input arguments are 'x', 'P1',
    'P2', ..., 'PN'.

     Examples:
      g = inline('t^2')
      g = inline('sin(2*pi*f + theta)')
      g = inline('sin(2*pi*f + theta)', 'f', 'theta')
      g = inline('x^P1', 1)
```

Παράδειγμα 3.3.4

Θα ορίσουμε τη συνάρτηση f1 που είδαμε πιο πάνω:

```
>> f1=inline('sin(2*pi*x)','x')
```

```
f1 =
     Inline function:
     f1(x) = sin(2*pi*x)

>> f1(0)
ans =
     0

>> f1(0.25)
ans =
     1
```

Παρατηρούμε ότι η εντολή inline μας δίνει ότι και η ανώνυμη συνάρτηση αλλά έχει κάπως πιο σύνθετη σύνταξη.

Παράδειγμα 3.3.5

Θα ορίσουμε τη συνάρτηση g1 του Παραδείγματος 3.3.3 έχουμε

```
>> g1=inline('sqrt(x.^2+y.^2)-2*x.*y','x','y')
g1 =
     Inline function:
     g1(x,y) = sqrt(x.^2+y.^2)-2*x.*y

>> g1(0,0)
ans =
     0

>> g1(1,0)
ans =
     1

>> g1(sqrt(2),sqrt(2))
ans =
   -2.0000
```

Παράδειγμα 3.3.6

Για την συνάρτηση του Παραδείγματος 3.3.1 μπορούμε να γράψουμε

```
>> f2 = inline('cos(4*pi*x) - sin(3*pi*x)','x')
f2 =
     Inline function:
     f2(x) = cos(4*pi*x) - sin(3*pi*x)
>> f2(0)
ans =
     1
>> f2(-4.1)
ans =
    1.1180
```

Ομοίως για την συνάρτηση του παραδείγματος 3.3.2 έχουμε:

```
>> f3 = inline('exp(-x.*y) + (x.^2 + y.^2).*cos(y)','x','y')
f3 =
     Inline function:
     f3(x,y) = exp(-x.*y) + (x.^2 + y.^2).*cos(y)
>> f3(0,0)
ans =
     1
>> f3(-4,1)
```

```
ans =
   63.7833
```

3.3.2 Η εντολή feval

Ένας άλλος τρόπος να καλέσουμε μια συνάρτηση που έχει οριστεί σε ένα m-file, είναι μέσω της εντολής **feval**. Αν για παράδειγμα έχουμε μια m-συνάρτηση fname με μεταβλητές εισόδου τις x1 και x2, ένας τρόπος να υπολογίσουμε την τιμή της είναι:

z = fname(x1, x2)

Εναλλακτικά, μπορούμε να χρησιμοποιήσουμε την εντολή feval

z = feval(@fname, x1, x2)

Η feval υπολογίζει λοιπόν την m-συνάρτηση που ακολουθεί μετά το σύμβολο @ για τις τιμές των μεταβλητών x1 και x2. Η feval χρησιμοποιείται συνήθως μέσα σε συναρτήσεις οι οποίες έχουν το όνομα μιας άλλης συνάρτησης σαν αλφαριθμητική μεταβλητή εισόδου. Η feval δουλεύει και με τις γνωστές συναρτήσεις βιβλιοθήκης της MATLAB.

Παράδειγμα 3.3.7

Για να υπολογίσουμε το sin(pi/2) μπορούμε να γράψουμε

```
>> sin(pi/2)
```

ή ισοδύναμα

```
>> feval(@sin, pi/2)
```

Επίσης, αντί της

```
>> rem(1234,11)
```

μπορούμε να γράψουμε

```
>> feval(@rem, 1234, 11)
```

Παράδειγμα 3.3.8

Έστω το m-file με όνομα myfunction.m:

```
function [f] = myfunction(x,y,z)
f = x.^3 + exp(y) + 1./(z.^2 + 2);
%Telos tou myfunction.m
```

Για να υπολογίσουμε την πιο πάνω συνάρτηση για $x = 1$, $y = 2$, $z = 3$, μπορούμε να γράψουμε

```
>> feval(@myfunction,1,2,3)
ans =
    8.4800
```

Ισοδύναμα, θα μπορούσαμε να γράψουμε

```
>> myfunction(1,2,3)
ans =
    8.4800
```

Παράδειγμα 3.3.9

Το πρόγραμμα function με όνομα **exfun** έχει σαν μεταβλητές εισόδου ένα αλφαριθμητικό strfun που είναι το όνομα μιας συνάρτησης $f(x)$ και ένα διάνυσμα x. Η συνάρτηση υπολογίζει την παράσταση:

$$f^2(x)-3f(x)+2$$

```
function z=exfun(strfun,x)

%EXFUN Briskei thn
%      f(x)^2 - 3 f(x) + 1
%      opou f(x) h sunartisi eisodou strfun.
%      Paradeigma klisis ths EXFUN: exfun(@sin,pi/2)
%

y=feval(strfun, x);
z=y.^2 -3*y + 2;

%End of exfun1
```

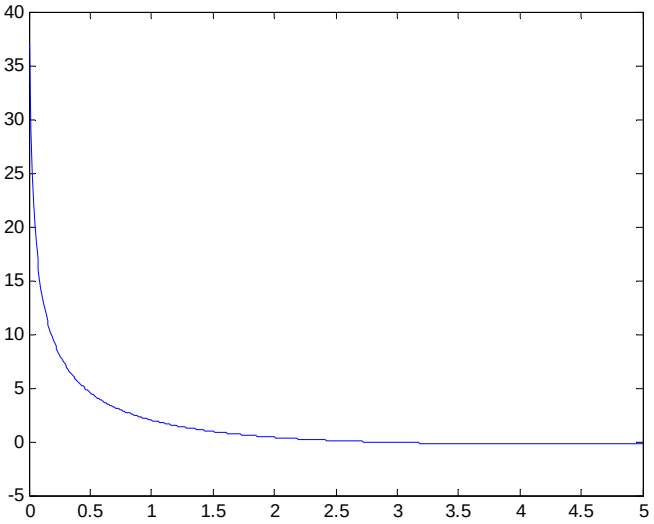
Εδώ χρησιμοποιήσαμε τη feval προκειμένου να υπολογίσουμε πρώτα την $f(x)$. Κατά τον υπολογισμό της z υψώσαμε τα στοιχεία του y στο τετράγωνο χρησιμοποιώντας τελεία πριν από το σύμβολο ^. Για να καλέσουμε την exfun πριν από το όνομα της συνάρτησης χρησιμοποιούμε το σύμβολο @.

```
>> exfun(@sin,pi/2)
ans =
     0

>> w=exfun(@log,1)
w =
     2
```

Και μια περίπτωση όπου το x είναι διάνυσμα:

```
>> x=[0.01:0.01:5];
>> y=exfun(@log,x);
>> plot(x,y)
```

Παίρνουμε έτσι την ακόλουθη γραφική παράσταση:



Σημείωση: Αν στο πιο πάνω παράδειγμα δώσουμε σαν δεδομένο εισόδου μια συνάρτηση που δεν είναι συνάρτηση βιβλιοθήκης (π.χ. ανώνυμη ή inline συνάρτηση), τότε το σύμβολο @ πριν από το όνομα της συνάρτησης μπορεί να παραλειφθεί. Για παράδειγμα,

```
>> f = @(x) 2*x - 3
f =
    @(x) 2*x - 3
>> exfun(f,1)
ans =
     6
```

Στην περίπτωση όμως που η συνάρτηση ορίζεται με m-file, η χρήση του συμβόλου @ είναι υποχρεωτική.

3.3 Ασκήσεις

3.1 Γράψτε ένα script file που να ονομάζεται `F2C.m` το οποίο να ζητά από τον χρήστη τη θερμοκρασία σε βαθμούς Φάρεναϊτ και να την μετατρέπει σε βαθμούς Κελσίου. Βεβαιωθείτε ότι το script file σας δουλεύει σωστά χρησιμοποιώντας το για τουλάχιστον 4 μετατροπές θερμοκρασίας.

3.2 Δημιουργήστε ένα νέο function m-file με το όνομα abssumprod.m το οποίο να υπολογίζει το άθροισμα και το γινόμενο των απόλυτων τιμών 4 αριθμών. Στα σχόλια να γράψετε το ονοματεπώνυμό σας. Βεβαιωθείτε ότι το m-file σας δουλεύει σωστά χρησιμοποιώντας το για τουλάχιστον 2 παραδείγματα.

3.3 Δημιουργήστε ένα νέο function m-file με το όνομα newmiscop.m το οποίο να υπολογίζει το τετράγωνο ενός τετραγωνικού πίνακα (σύμβολο A2) καθώς επίσης την ορίζουσα και τον βαθμό του A2. Στα σχόλια να γράψετε το ονοματεπώνυμό σας. Βεβαιωθείτε ότι το m-file σας δουλεύει σωστά χρησιμοποιώντας το για τουλάχιστον 3 παραδείγματα.

3.4 Να γράψετε ένα function m-file με όνομα euclnrm.m το οποίο να υπολογίζει την τετραγωνική ρίζα του αθροίσματος των τετραγώνων των στοιχείων ενός πίνακα. Στα σχόλια να γράψετε το ονοματεπώνυμό σας. Βεβαιωθείτε ότι το m-file σας δουλεύει σωστά χρησιμοποιώντας το για τουλάχιστον 2 παραδείγματα.

3.5 Γράψτε ένα script file που να μετατρέπει τις μοίρες σε ακτίνια και ένα άλλο που να μετατρέπει τα ακτίνια σε μοίρες. Βεβαιωθείτε ότι το script file σας δουλεύει σωστά χρησιμοποιώντας το για τουλάχιστον 4 μετατροπές.

3.6 Γράψτε ένα function file, που να καλείται `nthroot.m`, το οποίο να παίρνει σαν δεδομένα εισόδου τα x και n και να δίνει σαν δεδομένο εξόδου τη νιοστή ρίζα του x. Βεβαιωθείτε ότι το m-file σας δουλεύει σωστά χρησιμοποιώντας το για τουλάχιστον 3 παραδείγματα.

3.7 (α) Γράψτε ένα function file που βρίσκει το μέγιστο κατ'απόλυτη τιμή στοιχείο πίνακα.

(β) Γράψτε ένα άλλο function file που βρίσκει το ελάχιστο κατ'απόλυτη τιμή στοιχείο πίνακα.

(γ) Γράψτε ένα function file που βρίσκει το μέγιστο και το ελάχιστο κατ'απόλυτη τιμή στοιχείο πίνακα.

Βεβαιωθείτε ότι τα m-file σας δουλεύει σωστά χρησιμοποιώντας τα για τουλάχιστον 2 παραδείγματα (για το καθένα).

3.8 Γράψτε ένα function file που βρίσκει το ίχνος και την ορίζουσα του γινομένου δύο τετραγωνικών πινάκων. Βεβαιωθείτε ότι τα m-file σας δουλεύει σωστά χρησιμοποιώντας τα για τουλάχιστον 2 παραδείγματα.

3.9 Γράψτε ένα function file που να υπολογίζει το πολυώνυμο $p(x)=3x^3+5x^2-2x+1$. Χρησιμοποιώντας το m-file σας, υπολογίστε τα $p(-3), p(-1), p(0), p(1), p(4), p(12)$.

3.10 Γράψτε ένα function m-file με όνομα `ex3_2.m`, το οποίο να παίρνει σαν δεδομένο εισόδου το x και να δίνει σαν δεδομένα εξόδου τα y, z, και w τα οποία ορίζονται από τις $y = x/10^5$, $z = (x + y)/100$ και $w = xyz$.

3.11 Έστω ότι έχετε ένα κυλινδρικό δοχείο ύψους h με διάμετρο βάσης b στο οποίο θέλετε να βάλετε μπάλες του πιγκ πογκ με διάμετρο d. Έστω $V_C = h\pi\left(b/2\right)^2$ ο όγκος του δοχείου και $V_B = d^3$ ο όγκος του κύβου που περιφράσσει μια μπάλα. Τότε, ο ελάχιστος αριθμός μπάλων που χωρούν στο δοχείο δίνεται από $N = V_C / V_B$. Γράψτε ένα script file με όνομα `balls1.m` το οποίο να λύνει αυτό το πρόβλημα (δηλ. να δίνει τη τιμή του N) για $d = 1.54cm, b = 8cm, h = 14cm$.

3.12 Επαναλάβετε την πιο πάνω άσκηση, αλλά αυτή τη φορά γράψτε ένα function m-file με όνομα `balls2.m` το οποίο να λύνει το ίδιο πρόβλημα για οποιαδήποτε d, b και h. (Δηλ. τα d, b και h θα είναι δεδομένα εισόδου και το N θα είναι το δεδομένο εξόδου.) Βεβαιωθείτε ότι το m-file σας δουλεύει σωστά χρησιμοποιώντας το για τουλάχιστον 2 παραδείγματα της επιλογής σας.

3.13 Αν καταθέσουμε £P σε ένα λογαριασμό με ετήσιο επιτόκιο r το οποίο ανατοκίζεται n φορές το χρόνο, τότε μετά από t χρόνια το υπόλοιπο στον λογαριασμό θα είναι $B = P(1 + r/n)^{nt}$. Έστω ότι καταθέτουμε £5000 σε ένα λογαριασμό με ετήσιο ανατοκισμό για 17 χρόνια, και έστω ότι καταθέτουμε £5000 σε έναν άλλα λογαριασμό με μηνιαίο ανατοκισμό. Αν και για τους δύο λογαριασμούς το επιτόκιο είναι 8.5%, να γράψετε ένα script file με όνομα `ex3_5a.m` το οποίο να βρίσκει πόσα χρόνια (και πόσους μήνες) θα πρέπει να περάσουν για να έχουμε στον δεύτερο λογαριασμό το ίδιο υπόλοιπο με τον πρώτο.

3.14 Να επαναλάβετε την προηγούμενη άσκηση, αλλά αυτή τη φορά γράψτε ένα function m-file με όνομα `ex3_5b.m` το οποίο να λύνει το ίδιο πρόβλημα για οποιαδήποτε P, r και t. (Δηλ. τα P, r και t θα είναι δεδομένα εισόδου και το απαιτούμενο χρονικό διάστημα θα είναι το δεδομένο εξόδου.) Βεβαιωθείτε ότι το m-file σας δουλεύει σωστά χρησιμοποιώντας το για τουλάχιστον 2 παραδείγματα της επιλογής σας.

3.15 Η απόσταση d ενός σημείου (x_0, y_0) από μια ευθεία με εξίσωση $Ax + By + C = 0$ δίνεται από την $d = \left|Ax_0 + By_0 + C\right| / \sqrt{A^2 + B^2}$. Γράψτε ένα function m-file με όνομα `ex3_7.m` το οποίο να παίρνει σαν δεδομένα εισόδου τα x_0, y_0, A, B, και C, και να δίνει σαν δεδομένου εξόδου το d. Χρησιμοποιείστε το για να βρείτε την απόσταση του σημείου (2, –3) από την ευθεία $3x + 5y - 6 = 0$.

3.16 Ορίστε τις πιο κάτω συναρτήσεις χρησιμοποιώντας ένα m-file για την κάθε μια, και βρείτε τις ζητούμενες τιμές, χρησιμοποιώντας την εντολή feval.

(α) $f_1(x) = \cos^2(x) - \sin^2(x)$, $f_1(0), f_1(\pi/2), f_1(\pi)$

(β) $f_2(x) = 3x^3 - 2x^2 - 1$, $f_2(-1), f_2(1/2), f_2(3)$

(γ) $f_3(x, y) = y^2x^4 + xy - 5$, $f_3(0, -1), f_3(1, 2), f_3(3, -6)$

(δ) $f_4(x,y,z)=\dfrac{xyz}{x^2+y^2+z^2}$, $f_4(1,1,1)$, $f_4(-1,2,-7)$, $f_4(\pi,e,\sqrt{2})$

3.17 (α) Ορίστε τη συνάρτηση $g(x,y)=e^{y^2-x^2}$ με την εντολή inline, και βρείτε τα $g(0,1)$, $g(1,0)$, $g(\sqrt{\ln 3},\sqrt{\ln 2})$.
(β) Επαναλάβετε την άσκηση ορίζοντας μια ανώνυμη συνάρτηση.

3.18 Γράψτε μια m-συνάρτηση με όνομα compfun.m που υπολογίζει τη σύνθεση μιας συνάρτησης *f* με τον εαυτό της, δηλ. *f*(*f*(*x*)). Το όνομα της *f* πρέπει να είναι μεταβλητή εισόδου για την compfun. Βεβαιωθείτε ότι το m-file σας δουλεύει σωστά χρησιμοποιώντας το για τουλάχιστον 2 παραδείγματα της επιλογής σας.

3.19 Γράψτε μια m-συνάρτηση με όνομα prodfun που υπολογίζει το γινόμενο δύο συναρτήσεων *f*(*x*) και *g*(*x*). Τα ονόματα της *f* και της *g* πρέπει να είναι μεταβλητές εισόδου για την prodfun. Βεβαιωθείτε ότι το m-file σας δουλεύει σωστά χρησιμοποιώντας το για τουλάχιστον 2 παραδείγματα της επιλογής σας.

4 ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΣΤΗ MATLAB

Για να μπορέσουμε να γράψουμε δικά μας προχωρημένα προγράμματα (δηλ. m-files που θα παίζουν το ρόλο "νέων εντολών" στη MATLAB) πρέπει να μελετήσουμε καλά τις τέσσερις **δομές ελέγχου ροής** (flow control structures) που διαθέτει η MATLAB:

- **βρόχοι for** (for loops)
- **βρόχοι while** (while loops)
- **εντολή if** (if statement)
- **εντολή switch** (switch statement)

Πριν πάμε στις δομές ελέγχου ροής θα πρέπει να συζητήσουμε πρώτα τους λεγόμενους **σχεσιακούς** (relational operators) και **λογικούς τελεστές** (logical operators).

Ένας από τους τύπους δεδομένων της MATLAB είναι ο **λογικός** (logical data type) με πιθανές τιμές τη λογική μονάδα 1 (αληθής, true) και το λογικό 0 (ψευδής, false). Οι λογικές μεταβλητές παράγονται από σχεσιακούς και λογικούς τελεστές ή συναρτήσεις και από τις συναρτήσεις **true** και **false**. Για παράδειγμα, ας πάρουμε τις πραγματικές μεταβλητές x και y και τις λογικές μεταβλητές a και b:

```
>> x=1
x =
     1
>> y=0
y =
     0
>> a=true
a =
     1
>> b=false
b =
     0
>> whos
  Name      Size                    Bytes  Class

  a         1x1                         1  logical array
  b         1x1                         1  logical array
  x         1x1                         8  double array
  y         1x1                         8  double array

Grand total is 4 elements using 18 bytes
```

Παρατηρούμε ότι ενώ οι πραγματικές μεταβλητές καταλαμβάνουν 8 bytes, οι λογικές μεταβλητές καταλαμβάνουν 1 byte.

4.1 Σχεσιακοί τελεστές

Οι **σχεσιακοί τελεστές** (relational operators) της MATLAB είναι οι πιο κάτω:

<	less than	μικρότερος
>	greater than	μεγαλύτερος
< =	less than or equal	μικρότερος ή ίσος
> =	greater than or equal	μεγαλύτερος ή ίσος
= =	equal	ίσος
~ =	not equal	άνισος

Παρατηρούμε ότι σε αντίθεση με την εκχώρηση τιμής όπου χρησιμοποιείται το '=', για την ισότητα στις λογικές προτάσεις χρησιμοποιείται το '= =' .

Η σύγκριση μεταξύ δύο αριθμών μας δίνει τη λογική μονάδα αν αυτή είναι αληθής και το λογικό μηδέν αν είναι ψευδής:

```
>> 1==2
ans =
     0

>> 1+1==2
ans =
     1

>> 3<5
ans =
     1

>> 3>6
ans =
     0

>> 10>=10
ans =
     1

>> 10<=5
ans =
     0

>> 10~=5
ans =
     1
```

Θα πρέπει να σημειώσουμε ότι στην περίπτωση που συγκρίνουμε μιγαδικούς αριθμούς τότε στην περίπτωση ισότητας ('= =') και ανισότητας ('~=') συγκρίνονται τόσο τα πραγματικά όσο και τα φανταστικά μέλη. Σε όλες τις άλλες περιπτώσεις η σύγκριση γίνεται ως προς τα πραγματικά μέλη. Έτσι για παράδειγμα η σχέση

1+5i < 2 + i

είναι αληθής όπως επίσης και η

1+5i >= 1-i

Παράδειγμα 4.1.1

```
>> x=1+4i; y=3-2i; z=1+i; w=1+4i;

>> x ~=y
ans =
     1

>> x == z
ans =
     0

>> x == w
ans =
     1

>> x > z
ans =
     0

>> y > z
ans =
     1

>> x >= z
ans =
     1
```

Μπορούμε επίσης να συγκρίνουμε δύο πίνακες ίσων διαστάσεων ή τα στοιχεία ενός πίνακα με ένα αριθμό. Παράγεται τότε ένας πίνακας με λογικά στοιχεία:

- *στη σύγκριση πίνακα με πίνακα συγκρίνονται τα ομοθέσια στοιχεία, ενώ*
- *στη σύγκριση πίνακα με αριθμό συγκρίνεται κάθε στοιχείο του πίνακα με τον αριθμό αυτό.*

Παράδειγμα 4.1.2

```
>> A=[1 2; -1 1]
A =
     1     2
    -1     1

>> B=[3 2; -1 4]
B =
     3     2
    -1     4

>> A==B
ans =
     0     1
     1     0

>> A>B
ans =
     0     0
     0     0

>> A>=B
ans =
     0     1
     1     0

>> A==1
```

```
ans =
     1     0
     0     1

>> A>=1
ans =
     1     1
     0     1
```

Οι πίνακες που παράγονται με σύγκριση είναι λογικοί πίνακες, δηλ. τα στοιχεία τους είναι λογικές μεταβλητές.

Παράδειγμα 4.1.3

```
>> A=[ 1 2 3; 2 3 1; 3 1 2]
A =
     1     2     3
     2     3     1
     3     1     2

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

Ας βρούμε τώρα το λογικό πίνακα που μας δείχνει ποια στοιχεία των A και B είναι ίσα:

```
>> C=A==B
C =
     0     1     0
     1     0     0
     0     0     1
```

Ο D είναι ο λογικός πίνακας που μας δείχνει ποια στοιχεία του A είναι μεγαλύτερα των αντίστοιχων στοιχείων του B:

```
>> D=A>B
D =
     0     0     1
     0     1     0
     1     0     0
```

Ο E είναι ο λογικός πίνακας που μας δείχνει ποια στοιχεία του A είναι μεγαλύτερα ή ίσα του 2:

```
>> E=A>=2
E =
     0     1     1
     1     1     0
     1     0     1
     1     0     0
```

Με την εντολή whos βλέπουμε ότι κατασκευάσαμε δύο πίνακες διπλής ακρίβειας (A και B) και τρεις λογικούς πίνακες, τους C, D και E:

```
>> whos
  Name      Size             Bytes  Class      Attributes

  A         3x3                 72  double
  B         3x3                 72  double
  C         3x3                  9  logical
```

```
D          3x3                  9  logical
E          3x3                  9  logical
```

Στις νέες εκδοχές της MATLAB έχουμε τη δυνατότητα να χρησιμοποιήσουμε ισοδύναμες **σχεσιακές συναρτήσεις** αντί των λογικών τελεστών. Για παράδειγμα, οι παραστάσεις **A==B** και **eq(A,B)** είναι ισοδύναμες. Οι συναρτήσεις που αντιστοιχούν στους λογικούς τελεστές φαίνονται στον πιο κάτω πίνακα:

lt	<	less than	μικρότερος
gt	>	greater than	μεγαλύτερος
le	<=	less than or equal	μικρότερος ή ίσος
ge	>=	greater than or equal	μεγαλύτερος ή ίσος
eq	==	equal	ίσος
ne	~=	not equal	άνισος

Παράδειγμα 4.1.4

Οι πιο κάτω παραστάσεις είναι ισοδύναμες:

x=y	eq(x,y)
1>= 5	ge(1,5)
A~=B	ne(A,B)
(a+b) < c/d	lt(a=b, c/d)

Οι is-συναρτήσεις

Για τον έλεγχο της ισότητας δύο πινάκων ίσων διαστάσεων, μπορούμε να χρησιμοποιήσουμε τη ***λογική συνάρτηση***

isequal(A,B)

Για παράδειγμα,

```
>> A=ones(2,3)
A =
     1     1     1
     1     1     1

>> B=eye(2,3)
B =
     1     0     0
     0     1     0

>> isequal(A,B)
ans =
     0
```

Η isequal επιστρέφει την τιμή 1 αν οι δύο πίνακες είναι ίσοι και το 0 διαφορετικά. Είναι φανερή η διαφορά της **isequal(A,B)** από την **A= =B** ή την **eq(A,B)**: στην πρώτη περίπτωση παίρνουμε μια λογική μεταβλητή ενώ στη δεύτερη ένα λογικό πίνακα.

Η MATLAB είναι εφοδιασμένη με πλήθος **λογικών συναρτήσεων** (logical functions) το όνομα των οποίων αρχίζει με **'is'** όπως η isequal. Κάποιες από αυτές φαίνονται στον πίνακα που ακολουθεί. Η MATLAB μας δίνει τον πλήρη κατάλογο αυτών των συναρτήσεων με την εντολή **doc is**.

ischar	Έλεγξε αν είναι αλφαριθμητικό (string)
isempty	Έλεγξε αν πίνακας είναι άδειος
isequal	Έλεγξε αν οι πίνακες είναι ίσοι
isfinite	Βρες τα πεπερασμένα στοιχεία πίνακα
isinf	Βρες τα άπειρα στοιχεία πίνακα
isinteger	Έλεγξε αν ο πίνακας είναι ακέραιος
iskeyword	Βρες αν το όνομα είναι λέξη-κλειδί της MATLAB
isletter	Βρες τα στοιχεία που είναι αλφαβητικά γράμματα
islogical	Έλεγξε αν ο πίνακας είναι λογικός
isnan	Βρες τα στοιχεία πίνακα που είναι NaN
isreal	Έλεγξε αν ο πίνακας είναι πραγματικός
isscalar	Έλεγξε αν ο πίνακας είναι βαθμωτός
issorted	Έλεγξε αν το διάνυσμα είναι ταξινομημένο
isvarname	Έλεγξε αν το όνομα μεταβλητής είναι αποδεκτό
isvector	Έλεγξε αν είναι διάνυσμα

Παράδειγμα 4.1.5

Ας δούμε τι μας δίνουν οι συναρτήσεις isfinite, isinf και isnan για τον πίνακα

$$A=\begin{bmatrix} 1 & 2 & \text{inf} \\ NaN & -\text{inf} & 0.5 \\ NaN & \text{inf} & 1 \end{bmatrix}$$

```
>> A=[ 1 2 Inf; NaN -inf 0.5; NaN inf 1]
A =
    1.0000    2.0000       Inf
       NaN      -Inf    0.5000
       NaN       Inf    1.0000

>> isfinite(A)
ans =
     1     1     0
     0     0     1
     0     0     1

>> isinf(A)
ans =
     0     0     1
     0     1     0
     0     1     0

>> isnan(A)
ans =
     0     0     0
     1     0     0
     1     0     0
```

Παράδειγμα 4.1.6

Έστω τα αλφαριθμητικά s1, s2 και s3:

```
>> s1='2wrong-name1';
>> s2='okname';
```

```
>> s3='end';
```

Το s1 είναι φυσικά αλφαριθμητικό, δεν είναι λέξη κλειδί αλλά δεν είναι αποδεκτό όνομα μεταβλητής (γιατί;)

```
>> ischar(s1)
ans =
     1

>> iskeyword(s1)
ans =
     0

>> isvarname(s1)
ans =
     0
```

Το s2 είναι και αυτό αλφαριθμητικό, δεν είναι λέξη κλειδί και είναι αποδεκτό όνομα μεταβλητής:

```
>> ischar(s2)
ans =
     1
>> iskeyword(s2)
ans =
     0
>> isvarname(s2)
ans =
     1
```

Το s3 είναι επίσης αλφαριθμητικό, είναι λέξη κλειδί και έτσι δεν είναι αποδεκτό όνομα μεταβλητής:

```
>> ischar(s3)
ans =
     1

>> iskeyword(s3)
ans =
     1

>> isvarname(s3)
ans =
     0
```

Η λογική συνάρτηση find

Η πιο σημαντική λογική συνάρτηση είναι η **find.** Η συνάρτηση αυτή ερευνά ένα διάνυσμα ή ένα πίνακα και βρίσκει ποια στοιχεία του ικανοποιούν ένα δοσμένο κριτήριο. Για παράδειγμα για το διάνυσμα

```
>> u=[1 -3 4 0 -2 5 3 2 -4 6]

u =
     1    -3     4     0    -2     5     3     2    -4     6
```

παίρνουμε

```
>> y=find(u>=1)

y =
     1     3     6     7     8    10
```

Παρατηρούμε ότι η find επιστρέφει τους δείκτες των στοιχείων που ικανοποιούν το κριτήριο.

Παράδειγμα 4.1.7

Θα πάρουμε ένα τυχαίο πίνακα:

```
>> A=rand(4,3)
A =
    0.8462    0.8381    0.8318
    0.5252    0.0196    0.5028
    0.2026    0.6813    0.7095
    0.6721    0.3795    0.4289
```

Θα βρούμε τα στοιχεία του A που είναι μεγαλύτερα ή ίσα με το0.5:

```
>> z=find(A>0.5)
z =
     1
     2
     4
     5
     7
     9
    10
    11
```

Παρατηρούμε ότι οι θέσεις που παίρνουμε προκύπτουν αν διατρέχουμε προς τα κάτω κάθε στήλη και πάμε από αριστερά προς τα δεξιά. Αν θέλουμε να βρούμε τις θέσεις των στοιχείων στον πίνακα A μπορούμε να γράψουμε

```
>> [row,column]=find(A>0.5)
row =
     1
     2
     4
     1
     3
     1
     2
     3
column =
     1
     1
     1
     2
     2
     3
     3
     3
```

Παράδειγμα 4.1.8

Έστω grades το διάνυσμα των βαθμών 12 φοιτητών σ'ένα μάθημα:

```
>> grades=[8.5 8 2 6 3 2.5 8 1.5 7.5 4.5 9.5 5 ]
grades =
  Columns 1 through 6
    8.5000    8.0000    2.0000    6.0000    3.0000    2.5000
  Columns 7 through 12
    8.0000    1.5000    7.5000    4.5000    9.5000    5.0000
```

Για να επιτύχει κάποιος στο μάθημα πρέπει να πετύχει βαθμό τουλάχιστον 5. Το διάνυσμα pass μας δείχνει ποιοι φοιτητές έχουν επιτύχει.

```
>> pass=find(grades>=5)
pass =
     1     2     4     7     9    11    12
```

Μπορούμε να δούμε και τους βαθμούς των επιτυχόντων φοιτητών:

```
>> grades(pass)
ans =
  Columns 1 through 6
    8.5000    8.0000    6.0000    8.0000    7.5000    9.5000
  Column 7
    5.0000
```

4.2 Λογικοί τελεστές

Οι **λογικοί τελεστές** (logical operators) φαίνονται στον πιο κάτω πίνακα μαζί με τις αντίστοιχες λογικές συναρτήσεις:

Συμβ.	Συνάρτηση	Περιγραφή
&	**and**	logical and λογικό και (κατά στοιχεία)
|	**or**	logical or λογικό ή (κατά στοιχεία)
~	**not**	logical not λογικό όχι
xor	**xor**	logical exclusive or λογικό αποκλειστικό ή
all	**all**	true if all vector elements are nonzero αληθής αν όλα τα στοιχεία διανύσματος είναι μη μηδενικά
any	**any**	true if any element is nonzero αληθής αν ένα στοιχείο διανύσματος δεν είναι μηδέν
&&	**relop**	logical and (for scalars) with short circuiting λογικό και μικρού κυκλώματος
||	**relop**	logical or (for scalars) with short circuiting λογικό ή μικρού κυκλώματος

Στην παράγραφο αυτή θα ασχοληθούμε κυρίως με τους τρεις πρώτους λογικούς τελεστές: **&**, **|** και ~. Αν οι p και q είναι λογικές προτάσεις, ισχύουν τα εξής:

- Η **p & q** μας δίνει την τιμή 1 μόνο αν οι p και q είναι (και οι δύο) αληθείς.
- Η **p | q** μας δίνει την τιμή 1 αν τουλάχιστο μια από τις p και q είναι αληθής.

- Η **~ p** μας δίνει την τιμή 1 μόνο αν η p είναι ψευδής και την τιμή 0 αν η p είναι αληθής.

Παράδειγμα 4.2.1

Παίρνουμε μια αληθή και μια ψευδή πρόταση.

```
>> p=1==1
p =
     1
>> q=2<1
q =
     0
```

Ας δούμε τι παίρνουμε με τους τρεις λογικούς τελεστές.

```
>> p & q
ans =
     0

>> p | q
ans =
     1

>> ~p
ans =
     0

>> ~q
ans =
     1
```

Αντί των τελεστών &, | και ~ μπορούμε να χρησιμοποιήσουμε και τις συναρτησιακές τους μορφές:

- Η **and(p,q)** είναι ισοδύναμη με την **p & q**.
- Η **or(p,q)** είναι ισοδύναμη με την **p | q**.
- Η **not(p)** είναι ισοδύναμη με την **~ p**.

Παράδειγμα 4.2.2

Παίρνουμε τώρα δύο αληθείς προτάσεις:

```
>> p=1+1==2
p =
     1
>> q=1+1>=1
q =
     1
>> and(p,q)
ans =
     1
>> or(p,q)
ans =
     1
>> not(p)
ans =
     0
```

Μπορούμε επίσης να συνδυάσουμε τις τελεστικές με τις συναρτησιακές μορφές:

```
>> and(p,~q)
ans =
     0
>> or(~p,q)
ans =
     1
>> not(~p)
ans =
     1
```

Όπως και οι σχεσιακοί τελεστές, οι λογικοί τελεστές &, | και ~ παράγουν πίνακες με λογικά 0 και 1 όταν ένα από τα ορίσματα είναι πίνακας.

Παράδειγμα 4.2.3

```
>> A=[-2 0; 3 1]; B=[2 -3; 1 2];
>> A
A =
    -2     0
     3     1
>> B
B =
     2    -3
     1     2
>> A>1 & B <2
ans =
     0     0
     1     0
>> A>1 | B <2
ans =
     0     1
     1     0
>> ~(A>1)
ans =
     1     1
     0     1
```

Προσοχή: ποια η διαφορά μεταξύ της **~(A>1)** και της **~A>1**;

Η συνάρτηση **all** όταν το όρισμά της είναι διάνυσμα επιστρέφει το 1 αν όλα τα στοιχεία του διανύσματος είναι μη μηδενικά και το 0 όταν αυτό δεν ισχύει. Η συνάρτηση **any** επιστρέφει το 1 αν τουλάχιστον ένα στοιχείο του διανύσματος είναι μη μηδενικό και το 0 στην αντίθετη περίπτωση.

Παράδειγμα 4.2.4

```
>> u=[1 2 3 4 5];
>> v=[-1 3 0 1 4];
>> w=[0 0 1 0 0];
>> all(u)
```

```
ans =
     1
>> all(v)
ans =
     0
>> all(w)
ans =
     0
>> any(u)
ans =
     1
>> any(v)
ans =
     1
>> any(w)
ans =
     1
```

Στην περίπτωση που το όρισμά της είναι πίνακας, η συνάρτηση all επιστρέφει ένα διάνυσμα με τα αποτελέσματα της all εφαρμοσμένα σε κάθε στήλη του πίνακα. Είναι τότε εύκολο να δούμε ότι η

all(all(A==B))

είναι ένας άλλος τρόπος να ελέγξουμε την ισότητα των πινάκων A και B. Μια άλλη ισοδύναμη επιλογή είναι η **all(A(:)==B(:))**. Ομοίως η συνάρτηση **any(any(A==B))** παίρνει την τιμή 1 αν οι A και B έχουν έστω και σε μια θέση ίσα στοιχεία και την τιμή 0 διαφορετικά. Η **any(A(:)==B(:))** είναι ισοδύναμή της.

4.3 Βρόχοι for

Οι βρόχοι for έχουν την εξής δομή:

```
for  index = initial value (: step) : final value
        statements
end
```

Οι λέξεις "for" και "end" χρησιμοποιούνται στην αρχή και στο τέλος του βρόχου, ο μετρητής *index* παίρνει τις τιμές από *initial value* μέχρι *final value* με βήμα *step*, και οι εντολές (statements) εκτελούνται για όλες τις τιμές του μετρητή *index*. Αν παραλείψουμε το βήμα, τότε η MATLAB χρησιμοποιεί το 1 σαν βήμα. Σημειώνουμε επίσης ότι το βήμα μπορεί να είναι αρνητικό.

Για παράδειγμα, αν θέλουμε να υπολογίσουμε την παράσταση

$$j+2, \quad j=1, 2, 3, 4$$

μπορούμε να γράψουμε:

```
>> for j=1:4
j+2
```

```
end
ans =
     3
ans =
     4
ans =
     5
ans =
     6
```

Παράδειγμα 4.3.1

Θα χρησιμοποιήσουμε ένα βρόχο για να υψώσουμε τα στοιχεία του διανύσματος $x = [1, 2, 3, 4, 5]$ στο τετράγωνο.

```
>> for i=1:5
       x(i) = i^2;
   end
>> x
x =
     1     4     9    16    25
```

Η εντολή

```
x(i) = i^2;
```

μας επιτρέπει να αποθηκεύσουμε τις τιμές *i* του μετρητή, στο τετράγωνο, στις θέσεις *i* του διανύσματος *x*. Σημειώνουμε, επίσης, ότι βάλαμε ";" στο τέλος αυτής της εντολής για να μην τυπωθούν τα (ενδιάμεσα) αποτελέσματα στην οθόνη.

Αξίζει να πούμε ότι στην πράξη, ο πιο πάνω τρόπος δεν είναι ο πιο κατάλληλος για να υψώσουμε τα στοιχεία ενός διανύσματος σε μια δύναμη. Ο ενδεδειγμένος τρόπος στη MATLAB είναι ο εξής:

```
>> x=(1:5).^2
x =
     1     4     9    16    25
```

Παράδειγμα 4.3.2

Θα υπολογίσουμε τις τετραγωνικές ρίζες των 1, 3, 5, 7, 9, 11 και θα τις αποθηκεύσουμε στο διάνυσμα *y*.

```
>> format long
>> for i=1:2:11
       y(i)=sqrt(i);
   end
>> y'
ans =
   1.000000000000000
                   0
   1.732050807568877
                   0
   2.236067977499790
                   0
   2.645751311064591
                   0
   3.000000000000000
                   0
   3.316624790355400
```

Παρατηρούμε εδώ ότι τα στοιχεία του *y* που αντιστοιχούν στις άρτιες τιμές του μετρητή (δηλ.
i = 2, 4, …, 10 οι οποίες παραλήφθηκαν) έχουν πάρει την (εξ ορισμού) τιμή 0. Μπορούμε να αποφύγουμε αυτό το φαινόμενο ως εξής:

```
>> format long
>> clear('y')
>> n=1:2:11;
>> for i=1:length(n)
     y(i) = sqrt(n(i));
   end
>> y'
ans =
   1.000000000000000
   1.732050807568877
   2.236067977499790
   2.645751311064591
   3.000000000000000
   3.316624790355400
```

Παρατηρήσεις

Αν θέλουμε να γράψουμε ένα βρόχο σε μια γραμμή του παραθύρου εργασίας, χρησιμοποιούμε οπωσδήποτε κόμμα ή ερωτηματικό μετά την εντολή for και μετά γράφουμε κατά τα γνωστά τις επόμενες εντολές χωρίζοντάς τις επίσης με κόμματα ή ερωτηματικά. Στην τελευταία εντολή (end) δεν χρειάζονται ερωτηματικά. Έτσι οι εντολές

```
>> for i=1:10
x(i)=1/i;
end
```

γράφονται σε μια γραμμή ως εξής:

```
>> for i=1:10, x(i)=1/i; end
```

ή

```
>> for i=1:10; x(i)=1/i; end
```

Οι βρόχοι χρησιμοποιούνται συχνά για τον υπολογισμό αθροισμάτων της μορφής

$$\sum_{i=1}^{n} a_i$$

ή γινομένων της μορφής

$$\prod_{i=1}^{n} a_i$$

Στην περίπτωση αθροίσματος μηδενίζουμε πρώτα το άθροισμα (π.χ. sum1) και προσθέτουμε σ'αυτό τους διαδοχικούς όρους μ'ένα βρόχο for:

```
sum1=0;
for i=1:n
      sum1=sum1+a(i);
end
```

Στην περίπτωση γινομένου θέτουμε το γινόμενο (π.χ. product) ίσο με 1 και στη συνέχεια το πολλαπλασιάζουμε με τους διαδοχικούς όρους με ένα βρόχο for:

```
product=1;
```

```
for i=1:n
      product=product*a(i);
end
```

Θα πρέπει πάντως να σημειώσουμε ότι η MATLAB μας παρέχει τη δυνατότητα να υπολογίσουμε με απλό τρόπο το άθροισμα και το γινόμενο των στοιχείων ενός διανύσματος.

Παράδειγμα 4.3.3

Μπορούμε να υπολογίσουμε το άθροισμα

$$\sum_{i=1}^{10}(k+1)^k$$

ως εξής:

```
>> sum1=0;
>> for i=1:10
     sum1=sum1+(i+1)^i;
   end
>> sum1
sum1 =
  2.6983e+010
```

Μπορούμε εναλλακτικά να χρησιμοποιήσουμε τη συνάρτηση sum για διανύσματα αποφεύγοντας το βρόχο for:

```
>> sum1=sum(((1:10)+1).^(1:10))
sum1 =
  2.6983e+010
```

Παράδειγμα 4.3.4

Έστω x_i, $i = 1, \ldots, 11$ οι κόμβοι που προκύπτουν αν διαμερίσουμε το διάστημα [0, 1] σε 10 ισομήκη διαστήματα:

$$x_i = 0.1(i-1), \qquad i = 1, \cdots, 11$$

Μπορούμε να υπολογίσουμε το γινόμενο

$$\prod_{i=1,\ i\neq 5}^{11}(x_5 - x_i)$$

ως εξής:

```
>> x=(0:10)/10;
>> product=1;
>> for i=1:4, product=product*(x(5)-x(i)); end
>> for i=6:11, product=product*(x(5)-x(i)); end
>> product
product =
  1.7280e-006
```

Παρατηρούμε ότι χρησιμοποιήσαμε δύο βρόχους για να αποφύγουμε το $i = 5$. Όπως θα δούμε στη συνέχεια μπορούμε να εργαστούμε με ένα βρόχο και να αποφύγουμε το $i = 5$ με μια εντολή if.

Συχνά στις εφαρμογές απαιτείται μέσα σ'ένα βρόχο να δημιουργήσουμε ένα άλλο (εσωτερικό) βρόχο και σ'αυτόν ένα άλλο βρόχο κοκ. Έχουμε έτσι τους λεγόμενους πολλαπλούς ή **εγκιβωτισμένους βρόχους** (nested loops). Για παράδειγμα αν πρέπει να επαναλάβουμε μια διαδικασία για κάθε στοιχείο ενός m×n πίνακα, μπορούμε να σαρώσουμε τα στοιχεία του πίνακα ως εξής:

```
for i=1:m
      for j=1:n
            διαδικασία για i και j
      end
end
```

Παράδειγμα 4.3.5

Θα χρησιμοποιήσουμε ένα **διπλό βρόχο** για να υψώσουμε τα στοιχεία ενός πίνακα στο τετράγωνο.

```
>> A=[ 1 5 -3 ; 2 4 0 ; -1 6 9]
A =
     1     5    -3
     2     4     0
    -1     6     9

>> for i=1:3
       for j=1:3
           A2(i,j) = A(i,j)^2;
       end
   end

>> A2
A2 =
     1    25     9
     4    16     0
     1    36    81
```

Εναλλακτικός (και πιο κατάλληλος) τρόπος:

```
>> A=[ 1 5 -3 ; 2 4 0 ; -1 6 9];
>> A2=A.^2
A2 =
     1    25     9
     4    16     0
     1    36    81
```

Παράδειγμα 4.3.6 gabssum.m

```
function [xsum] = gabssum(A)
%
% function [xsum] = gabssum(A)
%
% Upologizei to apoluto a0roisma twn stoixeiwn enos pinaka A
%
[m,n]=size(A);
xsum=0;
for i=1:m
    for j=1:n
        xsum = xsum + abs(A(i,j));
    end
end

% Telos toy gabssum.m
```

Εναλλακτικός τρόπος:

```
function [xsum] = gabssum(A)
%
% function [xsum] = gabssum(A)
%
% Upologizei to apoluto a0roisma twn stoixeiwn enos pinaka A
%
[m,n]=size(A);
xsum=0;
B=abs(A);
for i=1:m
        xsum = xsum + sum( B(i, : ) );
end

% Telos toy gabssum.m
```

Παράδειγμα 4.3.7

Έστω ο A = ones(200). Θα υπολογίσουμε πόσος χρόνος χρειάζεται για να υπολογίσουμε τον
B = π A. Για τον σκοπό αυτό θα χρησιμοποιήσουμε τις συναρτήσεις **clock** και **etime**.

Αν δεν χρησιμοποιήσουμε βρόχο ο χρόνος είναι πρακτικά 0:

```
>> t0=clock; B=A*pi; time=etime(clock,t0)
time =
     0
```

Αν χρησιμοποιήσουμε βρόχο παρατηρούμε ότι ο απαιτούμενος χρόνος υπολογισμού είναι μεγαλύτερος.

```
>> t0=clock;
>> for k=1:200, for j=1:200, B(k,j)=A(k,j)*pi; end, end
>> time=etime(clock,t0)
time =
    0.1700
```

4.4 Βρόχοι while

Οι βρόχοι while είναι της μορφής:

```
while relation
        statements
end
```

Οι λέξεις "while" και "end" χρησιμοποιούνται στην αρχή και στο τέλος του βρόχου. Η ακολουθία εντολών «statements» εκτελούνται εφόσον η συνθήκη *relation* ικανοποιείται (δηλ. είναι αληθής) και σταματούν όταν αυτή παύει να ισχύει.

Για να γράψουμε τη συνθήκη relation χρησιμοποιούμε τους σχεσιακούς και λογικούς τελεστές που συζητήσαμε στα εδάφια 4.1 και 4.2.

Παράδειγμα 4.4.1

Το function m-file xlgmin.m, που φαίνεται πιο κάτω, βρίσκει τον ελάχιστο ακέραιο για τον οποίο ισχύει

$$\log n \geq x$$

όπου x δοσμένος αριθμός.

```
function [n] = xlgmin(x)
% function [n] = xlgmin(x)
%
% Briskei ton elaxisto akeraio n gia ton opoio isxuei
%              log n >= x
% O x prepei na einai gnhsia 0etikos
%

n = 1 ;
while log(n) < x
    n = n+1;
end

% Telos tou xlgmin.m
```

Τρέχουμε το πιο πάνω m-file για διάφορες τιμές του x:

```
>> xlgmin(1)
ans =
     3
>> xlgmin(3)
ans =
    21
>> xlgmin(8)
ans =
        2981
```

Παράδειγμα 4.4.2

Η συνάρτηση $f(x) = e^x$ προσεγγίζεται με σειρά Taylor ως εξής: $f(x) = e^x \approx \sum_{n=0}^{N} \frac{x^n}{n!}$.

Μπορεί να δειχτεί ότι $f(x) = e^x = \lim_{N\to\infty} \sum_{n=0}^{N} \frac{x^n}{n!}$, άρα για κάποιο πεπερασμένο N, η πιο πάνω σειρά δίνει μια «καλή» προσέγγιση για την συνάρτηση. Το function m-file

exp1.m, που ακολουθεί, υλοποιεί αυτή την προσέγγιση για την $f(x)$ για κάποιο δεδομένο x με ακρίβεια 0.0001.

```
function [S] = exp1(x)
% function [S] = exp1(x)
%
% Briskei mia proseggish gia thv suvarthsh exp(x) me
% akribeia 0.0001.
n = 1;
S = 1;
an = 1;
while abs(an) >= 0.0001
    an = x.^n/factorial(n);
    S = S + an;
    n = n + 1;
end
```

Εδώ χρησιμοποιήσαμε τη συνάρτηση βιβλιοθήκης **factorial(n)** για τον υπολογισμό του παραγοντικού n!.

Ας τρέξουμε το exp1.m για διάφορες τιμές του x, και κάθε φορά θα ελέγχουμε την απάντηση μας με την συνάρτηση βιβλιοθήκης exp της MATLAB:

```
>> exp1(2)
ans =
   7.389046015712681
>> exp(2)
ans =
   7.389056098930650

>> exp1(4)
ans =
  54.598136483106295
>> exp(4)
ans =
  54.598150033144236

>> exp1(-3)
ans =
   0.049796294665156
>> exp(-3)
ans =
   0.049787068367864
```

Οι εντολές break, return και continue

Οι βρόχοι for και while μπορούν να διακοπούν με την εντολή **break** η οποία μεταφέρει τον έλεγχο στην πρώτη εντολή μετά το τελικό end του βρόχου στον οποίο χρησιμοποιείται. Έτσι αν έχουμε εγκιβωτισμένους βρόχους ο έλεγχος περνά στον επόμενο (εξωτερικό βρόχο). Η εντολή break ορίζεται μόνο μέσα σε βρόχους for και while.

Μια παρόμοια εντολή που μπορεί να χρησιμοποιηθεί εναλλακτικά και εκτός βρόχων for και while είναι η εντολή **return**. Γενικά η εντολή αυτή μεταφέρει τον έλεγχο στην καλούσα συνάρτηση (π.χ. στο καλόν m-file) ή στο πληκτρολόγιο. Χρησιμοποιείται επίσης για τερματισμό της εντολής **keyboard** που μεταφέρει τον έλεγχο από ένα m-

file στο πληκτρολόγιο. Όπως θα δούμε στη συνέχεια με την εντολή return μπορούμε να διακόψουμε όταν χρειάζεται την εκτέλεση των εντολών ενός m-file. Αν για παράδειγμα ένας πίνακας δεν είναι αντιστρέψιμος θα ήταν καλό να μην προχωρήσουμε στην εύρεση του αντιστρόφου του.

Τέλος η εντολή **continue** μεταβιβάζει τον έλεγχο στην επόμενη επανάληψη ενός βρόχου for ή while χωρίς να εκτελεστούν οι εναπομένουσες εντολές του βρόχου.

Θα δούμε παραδείγματα χρήσης αυτών των εντολών στις επόμενες παραγράφους.

4.5 Η εντολή if

Η εντολή if μας επιτρέπει να ελέγξουμε αν μια (ή περισσότερες) συνθήκες ισχύουν και να εκτελέσουμε σε κάθε περίπτωση την επιθυμητή ακολουθία εντολών και πράξεων. Η εντολή έχει την γενική μορφή:

```
if relation_1
        statement(s)
elseif relation_2
        statement(s)
else
        statement(s)
end
```

Οι συνθήκες ελέγχονται με τη χρήση σχεσιακών και λογικών τελεστών. Σημειώνουμε επίσης ότι η εντολή elseif γράφεται σαν μια λέξη (δεν πρέπει να υπάρχει κενό μεταξύ του else και του if).

Η απλούστερη μορφή της εντολής if είναι η πιο κάτω:

```
if relation
        statement(s)
end
```

Οι εντολές εκτελούνται μόνο αν ικανοποιείται η συνθήκη relation. Διαφορετικά δεν εκτελείται καμιά εντολή στο σημείο αυτό του προγράμματος και η ροή του τελευταίου συνεχίζει κανονικά. Αν θέλουμε να γράψουμε στην ίδια γραμμή άλλες εντολές χρησιμοποιούμε κόμμα ή ερωτηματικό για να χωρίσουμε την εντολή if από την επόμενη. Για παράδειγμα αντί

```
if x>0
   x=sqrt(x)
end
```

γράφουμε

```
if x>0, x=sqrt(x); end
```

Παράδειγμα 4.5.1 workerpay.m

Το script m-file workerpay.m, που φαίνεται πιο κάτω, υπολογίζει τον μισθό ενός εργαζομένου βάσει των ωρών που έχει εργαστεί. Αφού ζητήσει από τον χρήστη να δώσει τις ώρες εργασίας και το ωρομίσθιο, το πρόγραμμα υπολογίζει το μισθό ως εξής: Μέχρι τις 40 ώρες εργασίας, ο μισθός είναι το γινόμενο των ωρών επί το ωρομίσθιο, ενώ πάνω από τις 40 ώρες (υπερωρίες), οι απολαβές είναι κατά 50% μεγαλύτερες.

```
% Script file workerpay.m

t = input('Poses wres exei ergastei to atomo? ');
h = input('Poios eivai o mis0os ava wra? ');

Pay = t*h;

if t > 40
    Pay = Pay + (t-40)*0.5*h;
end

disp('To eisodeima tou ergazomevou eivai ')
format bank
disp(Pay)

% Telos tou workerpay.m
```

Ας τρέξουμε το πρόγραμμα μερικές φορές:

```
>> workerpay

Poses wres exei ergastei to atomo? 46
Poios eivai o mis0os ava wra? 4.25
To eisodeima tou ergazomevou eivai
        208.25

>> workerpay

Poses wres exei ergastei to atomo? 35
Poios eivai o mis0os ava wra? 4
To eisodeima tou ergazomevou eivai
        140.00
```

Ας δούμε τώρα τι γίνεται στην περίπτωση η "απόφαση" που χρειάζεται να παρθεί συμπεριλαμβάνει εναλλακτική περίπτωση, δηλ. ή το ένα θα συμβεί ή το άλλο. Η δομή είναι

```
if relation 1
        statement(s)
else
        statement(s)
end
```

Παράδειγμα 4.5.2

Το m-file gee.m, που φαίνεται πιο κάτω, ορίζει την συνάρτηση

$$g(x)=\begin{cases} x^2 & , x \le 0.5 \\ 0.25 & , x > 0.5 \end{cases}$$

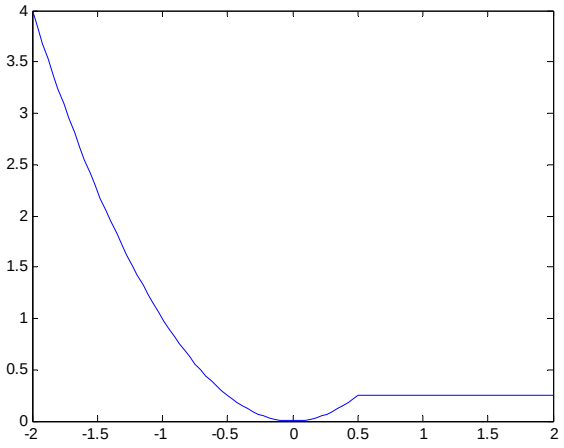
```
function [G] = gee(x)

for i=1:length(x)
    if x(i) <= 0.5
        G(i) = x(i)^2;
    else
        G(i) = 0.25;
    end
end

% Telos tou gee.m
```

Τρέχουμε το m-file και παίρνουμε τις εξής τιμές:

```
>> gee(0)
ans =
             0

>> gee(0.2)
ans =
          0.04

>> gee(3)
ans =
          0.25

>> gee([1:7]')
ans =
    0.2500    0.2500    0.2500    0.2500    0.2500    0.2500
0.2500

>> gee([-3:3])
ans =
    9.0000    4.0000    1.0000         0    0.2500    0.2500
0.2500
```

Μπορούμε, επίσης, να πάρουμε τη γραφική παράσταση της συνάρτησης g, π.χ. στο διάστημα $[-2, 2]$, ως εξής:

```
>> x=linspace(-2,2);
>> plot(x,gee(x))
```



Στο επόμενο παράδειγμα θα δούμε τη γενική δομή της εντολής if.

Παράδειγμα 4.5.3 Εύρεση του αντίστροφου πίνακα - invnew.m

```
function [Ainv] = invnew(A)
%
% Elegxei an (a) o pinakas A einai tetragwnikos (diaforetika stamata),
%            (b) o pinakas A einai antistreyimos (diaforetika stamata)
% kai an ikanopoiountai oi anwterw sun0hkes briskei ton antistrofo
% tou A.

% Ypologise tis diastaseis tou A
[m,n] = size(A);

% Elegkse an o A einai tetragwnikos
if m~=n
    disp('Matrix A is not square.');
    return %eksodos apo to programma
elseif rank(A) ~= n
    disp('Matrix is singular');
    return %eksodos apo to programma
else
    Ainv = inv(A);
end
% Telos tou invnew.m
```

Εδώ χρησιμοποιήσαμε την εντολή **return** που συζητήσαμε στην προηγούμενη παράγραφο.

Το πιο πάνω m-file έδωσε τα εξής για ένα ψευδοτυχαίο 4×4 πίνακα:

```
>> A=rand(4)
A =
    0.9501    0.8913    0.8214    0.9218
    0.2311    0.7621    0.4447    0.7382
    0.6068    0.4565    0.6154    0.1763
    0.4860    0.0185    0.7919    0.4057

>> invnew(A)
ans =
    2.2631   -2.3495   -0.4696   -0.6631
   -0.7620    1.2122    1.7041   -1.2146
   -2.0408    1.4228    1.5538    1.3730
    1.3075   -0.0183   -2.5483    0.6344

>> A=rand(3,2)
A =
    0.9355    0.8936
    0.9169    0.0579
    0.4103    0.3529

>> invnew(A)
Matrix A is not square.

>> A=zeros(3)
A =
     0     0     0
     0     0     0
     0     0     0

>> invnew(A)
Matrix is singular
```

Παράδειγμα 4.5.4 Υπολογισμός του *n*!

Στο παράδειγμα αυτό η factnew είναι αναδρομική (καλεί τον εαυτό της)!

```
function [N] = factnew(n)
% Elegxei av to n eivai 0etikos akaireow, kai av vai,
% upologizei to n! me anadromiko tropo:
% n! = n (n-1)!
% H function kalei ton eauto ths oses fores xreiazetai gia
% na upologisei to n!
%
if n < 0 | round(n) ~= n
    disp('To dedomevo eisodou prepei va eivai 0etikos akeraios.')
    return
elseif (n==1) | (n==0)
    % 1!=0!=1
    N=1;
else
    % n! = n (n-1)!
    N = n* factnew(n-1);
end
% Telos tou factnew.m
```

Χρησιμοποιήσαμε ξανά την εντολή return.

Θα τρέξουμε το πιο πάνω πρόγραμμα και θα ελέγξουμε τις απαντήσεις μας με τη συνάρτηση βιβλιοθήκης **factorial(n)**.

```
>> factnew(1/2)
To dedomevo eisodou prepei va eivai 0etikos akeraios.
>> factnew(-2)
To To dedomevo eisodou prepei va eivai 0etikos akeraios.
>> factnew(4)
ans =
    24
>> factorial(4)
ans =
    24
>> factnew(8)
ans =
       40320
>> factorial(8)
ans =
       40320
```

Παράδειγμα 4.5.5

Θα δούμε κάπως διαφορετικά το Παράδειγμα 4.3.4 όπου χρησιμοποιήσαμε δύο βρόχους για να υπολογίσουμε το γινόμενο

$$\prod_{i=1,\ i\neq 5}^{11} (x_5 - x_i)$$

όπου x_i, $i = 1, \ldots, 11$ οι κόμβοι που προκύπτουν αν διαμερίσουμε το διάστημα [0, 1] σε 10 ισομήκη διαστήματα:

$$x_i = 0.1(i-1), \qquad i = 1, \cdots, 11$$

Μπορούμε να αποφύγουμε την περίπτωση $i = 5$ με την εντολή if:

```
>> x=(0:10)/10;
>> product=1;
>> for i=1:11
       if i~=5
         product=product*(x(5)-x(i));
       end
    end
>> product
product =
  1.7280e-006
```

Η συνάρτηση beep

Η συνάρτηση **beep** κάνει ένα προειδοποιητικό beep και χρησιμοποιείται συνήθως μαζί με μηνύματα σφάλματος.

Παράδειγμα 4.5.6

```
>> x=-2;
>> if x >=0
     y=sqrt(x);
   else
     beep
     disp('x should be positive!')
   end

>> x should be positive!
```

4.6 Η εντολή switch

Η εντολή **switch-case** μας δίνει τη δυνατότητα να επιλέξουμε για εκτέλεση μια ομάδα εντολών από άλλες πιθανές ομάδες. Η γενική της δομή έχει ως εξής:

```
switch switch_expression
        case value_1
                statement(s)
        case value_2
                statement(s)
        case value_3
                statement(s)
        ⋮
        otherwise
                statement(s)
end
```

Σημειώνουμε τα εξής γι' αυτή την εντολή:

- Η πρώτη γραμμή περιέχει την λέξη κλειδί switch, ακολουθούμενη από το όνομα switch_expression, που θα δώσουμε εμείς, το οποίο μπορεί να είναι βαθμωτή ποσότητα, αλφαριθμητικό, ή ακόμα και μαθηματική παράσταση με προκαθορισμένες μεταβλητές που μπορεί να πάρει μια τιμή.
- Μετά από το switch, ακολουθούν οι διάφορες εντολές case. Η κάθε μια έχει ένα όνομα (π.χ. value _1, value_2 κλπ) το οποίο μπορεί να είναι βαθμωτή ποσότητα ή αλφαριθμητικό, και μετά ακολουθούν οι εντολές που θα εκτελεστούν αν βρεθούμε στη συγκεκριμένη περίπτωση.
- Μετά την τελευταία περίπτωση/εντολή case, ακολουθεί η προαιρετική περίπτωση/εντολή otherwise της οποίας οι εντολές θα εκτελεστούν αν καμιά από τις προηγούμενες περιπτώσεις δεν ισχύει.
- Σε αντίθεση με άλλες γλώσσες προγραμματισμού (όπως, π.χ. η C), στη MATLAB δεν χρειάζεται να διακόψουμε τη ροή της δομής μετά από κάθε case, μια και αυτό θα γίνει αυτόματα αφού μια από τις περιπτώσεις έχει επαληθευτεί.

Παράδειγμα 4.6.1 convertcm.m

Το πιο κάτω script m-file μετατρέπει μήκη από διάφορες μονάδες σε cm:

```
% Metatrepei mia posothta apo inches h milimeters, se ekatosta

x = input('Dwste th timh ths metablhths pou xreiazetai metatroph: ');
units = input('Dwste tis movades metrhshs ths (mesa se apostrofous):
');

switch units
    case {'in','inch'}
        y = 2.54*x;        % converts to centimeters
        disp(' ')
        disp('Se ekatosta (cm), h timh ths metablhths eivai')
        disp(y)
    case     {'m','meter'}
        y = x*100;        % converts to centimters
        disp(' ')
        disp('Se ekatosta (cm), h timh ths metablhths eivai')
        disp(y)
    case { 'millimeter','mm'}
        y = x/10;
        disp(' ')
        disp('Se ekatosta (cm), h timh ths metablhths eivai')
        disp(y)
    case {'cm','centimeter'}
        y = x;
        disp(' ')
        disp('Se ekatosta (cm), h timh ths metablhths eivai')
        disp(y)
    otherwise
        disp(' ')
        disp(['unknown units:' units])
        y = nan;
end
```

Τρέχουμε το πιο πάνω script για διάφορες τιμές:

```
>> convertcm
Dwste th timh ths metablhths pou xreiazetai metatroph: 3
Dwste tis movades metrhshs ths (mesa se apostrofous): 'in'
Se ekatosta (cm), h timh ths metablhths eivai
    7.6200

>> convertcm
Dwste th timh ths metablhths pou xreiazetai metatroph: 5
Dwste tis movades metrhshs ths (mesa se apostrofous): 'mm'
Se ekatosta (cm), h timh ths metablhths eivai
    0.5000

>> convertcm
Dwste th timh ths metablhths pou xreiazetai metatroph: 9
Dwste tis movades metrhshs ths (mesa se apostrofous): 'cm'
Se ekatosta (cm), h timh ths metablhths eivai
     9

>> convertcm
Dwste th timh ths metablhths pou xreiazetai metatroph: 7
Dwste tis movades metrhshs ths (mesa se apostrofous): 'feet'
unknown units:feet
```

Παράδειγμα 4.6.2

Το function m-file που ακολουθεί χρησιμοποιεί την εντολή switch μέσα σ'ένα βρόχο if. Αν ο πίνακας A είναι τετραγωνικός, υπολογίζει την ορίζουσα και αποφαίνεται αν ο πίνακας είναι ιδιάζων ή όχι. Το checkmat.m δεν έχει δεδομένα εξόδου!

```
function []=checkmat(A)
% CHECKMAT
% Elegxei an enas pinakas einai antistrepsimos
% Dedomeno eisodou: pinakas A
% Dedomeno eksodou: kanena
%
[m,n]=size(A);
if m ~=n
    disp('Matrix A is not square!')
else
    detA=det(A);
    switch detA
        case 0
            disp('Matrix A is singular!')
        otherwise
            disp('Matrix A is invertible!')
    end
end
```

Παράδειγμα 4.6.3

Το function m-file που ακολουθεί βρίσκει την τιμή του αεροπορικού εισιτηρίου ανάλογα με τον προορισμό.

```
function []=ticketp()
% TICKETP
% Returns the cost of an airfair depending on the
% destination
%
city=input('Enter destination city: ', 's')
switch city
```

```
        case 'Paphos'
            disp('50 CYP')
        case 'Athens'
            disp('150 CYP')
        case 'Heraklion'
            disp('110 CYP')
        case 'Thessaloniki'
            disp('200 CYP')
        otherwise
            sprintf('No flights for %s',city)
    end
```

Ακολουθούν κάποια αποτελέσματα που παίρνουμε με το ticketp.m.

```
>> ticketp
Enter destination city: Athens
city =
Athens
150 CYP

>> ticketp
Enter destination city: Paphos
city =
Paphos
50 CYP

>> ticketp
Enter destination city: Paris
city =
Paris
ans =
No flights for Paris
```

4.6.1 Η συνάρτηση menu

Η συνάρτηση menu παράγει ένα μενού επιλογών για τον χρήστη. Η δομή της είναι

choice=menu(header, item1, item2, ….)

όπου

- header είναι η επικεφαλίδα
- item1 είναι η πρώτη επιλογή, item2 η δεύτερη κοκ.

Η σύνταξη

choice=menu(header, itemlist)

όπου itemlist είναι ένας αλφαριθμητικός πίνακας κελίων είναι επίσης αποδεκτή.

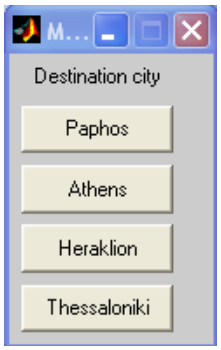
Η menu επιστρέφει τον αριθμό της επιλογής του χρήστη στη μεταβλητή choice.

Παράδειγμα 4.6.4

Με την εντολή

```
choice=menu('Destination city', 'Paphos', 'Athens', 'Heraklion',
'Thessaloniki')
```

εμφανίζεται ένα ξεχωριστό παράθυρο με τις 4 επιλογές.



Ο χρήστης επιλέγει με ένα απλό κλικ την προτίμησή του.

Το ίδιο αποτέλεσμα παίρνουμε αν ορίσουμε τις επιλογές σε ένα αλφαριθμητικό πίνακα κελίων:

```
>> list={'Paphos'} {'Athens'} {'Heraklion'} {'Thessaloniki'}}
list =
    'Paphos'    'Athens'    'Heraklion'    'Thessaloniki'

>> choice=menu('Destination city',list)
```

4.7 Ασκήσεις

4.1 Εξηγήστε τα πιο κάτω αποτελέσματα της MATLAB:

```
>> 3>2>1
ans =
     0
>> 3>2>0
ans =
     1
```

4.2 Εξηγήστε τα πιο κάτω αποτελέσματα της MATLAB:

```
>> 5>3+6>4
ans =
     0
>> (5>3)+(6>4)
ans =
     2
```

4.3 Βρείτε τις (λογικές) τιμές των πιο κάτω παραστάσεων:
(α) 3 = = 2
(β) 3 ~ = 4
(γ) 1/(1+cos1) <= 2
(δ) exp(3) >= 30
(ε) (1-3)/3^2 <= exp(1)/sin(pi/3)
Επαναλάβετε την άσκηση με τις αντίστοιχες σχεσιακές συναρτήσεις.

4.4 Βρείτε τις (λογικές) τιμές των πιο κάτω παραστάσεων:
(α) 13 >= 2
(β) 1+3i < –1 = 7i
(γ) 13 ~ = 13
(δ) (6+7) < 12
(ε) ones(3) ~ = eye(3)

4.5 Ορίστε στη MATLAB τους πίνακες

$$A=\begin{bmatrix}1 & 2 & -1\\ 2 & 0 & 1\\ 1 & 1 & 4\end{bmatrix} \text{ και } B=\begin{bmatrix}3 & 2 & -3\\ 2 & 0 & 1\\ 5 & 1 & -4\end{bmatrix}$$

και βρείτε τα αποτελέσματα που προκύπτουν με όλους τους σχεσιακούς τελεστές. Επαναλάβετε με τις αντίστοιχες σχεσιακές συναρτήσεις.

4.6 Ορίστε στη MATLAB τους πίνακες A=rand(4) και B=ones(4) και βρείτε τα αποτελέσματα που προκύπτουν με όλους τους σχεσιακούς τελεστές. Επαναλάβετε με τις αντίστοιχες σχεσιακές συναρτήσεις.

4.7 Γράψτε με διαφορετικό τρόπο τις πιο κάτω παραστάσεις στη MATLAB:
(α) 13 >= 2
(β) 1+3i < –1 +7i
(γ) 13 ~ = 13
(δ) (6+7) < 12
(ε) ones(3) ~ = eye(3)

4.8 Γράψτε με διαφορετικό τρόπο τις πιο κάτω παραστάσεις στη MATLAB:
(α) ne(a,b)
(β) ge(zeros(3,5), eye(3,5))
(γ) lt(rand(4), zeros(4))
(δ) eq(x, cos(x))
(ε) gt(err, 1e-6)

4.9 Έστω οι πίνακες

$$A=\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{bmatrix} \text{ και } B=\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & -6\\ 7 & 8 & 0\end{bmatrix}$$

Δοκιμάστε στη MATLAB τις πιο κάτω εντολές και σχολιάστε τα αποτελέσματά σας:
(α) A= =B
(β) all(A= =B)
(γ) any(A= =B)
(δ) all(all(A= =B))
(ε) all(A(:)= =B(:))
(στ) any(any(A= =B))
(ζ) any(A(:)= =B(:))

4.10 Γράψτε ένα διπλό βρόχο για να κατασκευάσετε τον 5×5 πίνακα Hilbert, και ελέγξτε την απάντηση σας με την ενσωματωμένη εντολή hilb.

4.11 Γράψτε ένα function m-file με το όνομα mhilb.m το οποίο θα κατασκευάζει τον $n\times n$ πίνακα με γενικό στοιχείο το

$$a_{ij}=\frac{1}{i+j+1}$$

4.12 Γράψτε ένα function m-file με το όνομα fibon.m το οποίο θα έχει ως μεταβλητές εισόδου τα a_0, a_1 και n και θα επιστρέφει το διάνυσμα των πρώτων n όρων της ακολουθίας Fibonacci, η οποία ορίζεται από την αναδρομική σχέση $a_i=a_{i-1}+a_{i-2}$.

4.13 Γράψτε ένα function m-file με το όνομα sharm1.m για τον υπολογισμό του αθροίσματος

$$\sum_{k=1}^{n}\frac{1}{k}$$

με τη χρήση βρόχου for. Στη συνέχεια, υπολογίστε το άθροισμα για n=1:100, και κατασκευάστε το γράφημά του συναρτήσει του n. Σχολιάστε το γράφημα που πήρατε.
Σημείωση: Το άθροισμα μπορεί να υπολογιστεί εύκολα χωρίς τη χρήση βρόχου for ως εξής: **sum(1./(1:n)).**

4.14 Γράψτε ένα function m-file με το όνομα sharm2.m για τον υπολογισμό του αθροίσματος

$$\sum_{k=1}^{n}\frac{1}{k^2}$$

με τη χρήση βρόχου for. Στη συνέχεια, υπολογίστε το άθροισμα για n=1:100 και κατασκευάστε το γράφημά του συναρτήσει του n. Σχολιάστε το γράφημα που πήρατε.
Σημείωση: Το άθροισμα μπορεί να υπολογιστεί εύκολα χωρίς τη χρήση βρόχου for ως εξής: **sum(1./(1:n).^2).**

4.15 Γράψτε ένα function m-file με το όνομα vecprod.m και δεδομένα εισόδου τα διανύσματα x και y για τον υπολογισμό του γινομένου

$$\prod_{i=1}^{n}|x_i - y_i|$$

με τη χρήση βρόχου for. Πως μπορεί πιο απλά να υπολογιστεί το γινόμενο;

4.16 Γράψτε ένα function m-file με το όνομα nameij.m που θα κατασκευάζει τον $m\times n$ πίνακα με γενικό στοιχείο το

$$a_{ij}=\frac{1+i}{1+j}$$

Ελέγξτε τα αποτελέσματα για τις περιπτώσεις 2×2, 3×3, 4×4, 1×3, και 3×2.

4.17 Γράψτε ένα function m-file με το όνομα upperm.m που θα κατασκευάζει τον n×n άνω τριγωνικό πίνακα που ορίζεται από την

$$a_{ij}=\begin{cases}\dfrac{i+j}{2}, & j\geq i\\ 0, & j<i\end{cases}$$

Ελέγξτε τα αποτελέσματα για τις περιπτώσεις n = 1, 2, 3, 4 και 5.

4.18 Γράψτε ένα function m-file με το όνομα lowerm.m που θα κατασκευάζει τον n×n κάτω τριγωνικό πίνακα που ορίζεται από την

$$a_{ij}=\begin{cases}\dfrac{i+2j}{2}, & j\leq i\\ 0, & j>i\end{cases}$$

Ελέγξτε τα αποτελέσματα για τις περιπτώσεις n = 1, 2, 3, 4 και 5.

4.19 Έστω τα πολυώνυμα Maclaurin για τη συνάρτηση e^x:

$$p_n(x)=1+x+\frac{x^2}{2!}+\cdots+\frac{x^n}{n!}=\sum_{k=0}^{n}\frac{x^k}{k!}$$

Γράψτε ένα function m-file με όνομα appr.m και μεταβλητές εισόδου το n και το *διάνυσμα* x που υπολογίζει το πολυώνυμο p_n. Στη συνέχεια υπολογίστε και σχεδιάστε το σφάλμα προσέγγισης

$$\varepsilon_n = \left| p_n - e^x \right|$$

στο διάστημα [–2, 2] για n = 2 και 3.

4.20 Γράψτε ένα function m-file με το όνομα diagmsl.m που θα επιλύει το γραμμικό σύστημα Ax = b όταν ο A είναι διαγώνιος πίνακας:

$$x_i = b_i / a_{ii}, \qquad i = 1, \cdots, n$$

Το m-file πρέπει να ελέγχει τα εξής:

- αν ο A είναι τετραγωνικός και αν όχι να επιστρέφει σχετικό μήνυμα λάθους.
- αν ο A είναι διαγώνιος και αν όχι να επιστρέφει σχετικό μήνυμα λάθους.
- αν οι A και b έχουν το ίδιο πλήθος στηλών και αν όχι να επιστρέφει σχετικό μήνυμα λάθους.
- αν ο A είναι αντιστρέψιμος και αν όχι να επιστρέφει σχετικό μήνυμα λάθους.

4.21 Γράψτε ένα function m-file με το όνομα uppersol.m που θα επιλύει το γραμμικό σύστημα Ax = b με **πίσω αντικατάσταση** (back substitution) όταν ο A είναι άνω τριγωνικός πίνακας:

$$x_n = b_n / a_{nn}$$

$$x_i = \left(b_i - \sum_{k=i+1}^{n} a_{ik} x_k \right) / a_{ii}, \qquad i = n-1, \cdots, 1$$

Το m-file πρέπει να ελέγχει τα εξής:

- αν ο A είναι τετραγωνικός και αν όχι να επιστρέφει σχετικό μήνυμα λάθους.
- αν ο A είναι άνω τριγωνικός και αν όχι να επιστρέφει σχετικό μήνυμα λάθους.
- αν οι A και b έχουν το ίδιο πλήθος στηλών και αν όχι να επιστρέφει σχετικό μήνυμα λάθους.
- αν ο A είναι αντιστρέψιμος και αν όχι να επιστρέφει σχετικό μήνυμα λάθους.

Στη συνέχεια επιλύστε τα πιο κάτω συστήματα:

(α)
$$\begin{aligned} 5x_1 + 4x_2 + 3x_3 + 2x_4 + x_5 &= 35 \\ 2x_2 + x_3 + x_4 + x_5 &= 16 \\ 3x_3 - x_4 - x_5 &= 0 \\ 3x_4 + x_5 &= 17 \\ 3x_5 &= 15 \end{aligned}$$

(β) Το τυχαίο άνω τριγωνικό σύστημα με b = rand(5,1) και A = triu(rand(5)).

4.22 Γράψτε ένα function m-file με το όνομα lowersol.m που θα επιλύει το γραμμικό σύστημα Ax = b με **εμπρός αντικατάσταση** όταν ο A είναι κάτω τριγωνικός πίνακας.

$$x_1 = b_1 / a_{11}$$

$$x_i = \left(b_i - \sum_{k=1}^{i-1} a_{ik} x_k \right) / a_{ii}, \qquad i = 2, \cdots, n$$

Το m-file πρέπει να ελέγχει τα εξής:

- αν ο A είναι τετραγωνικός και αν όχι να επιστρέφει σχετικό μήνυμα λάθους.
- αν ο A είναι κάτω τριγωνικός και αν όχι να επιστρέφει σχετικό μήνυμα λάθους.
- αν οι A και b έχουν το ίδιο πλήθος στηλών και αν όχι να επιστρέφει σχετικό μήνυμα λάθους.
- αν ο A είναι αντιστρέψιμος και αν όχι να επιστρέφει σχετικό μήνυμα λάθους.

Στη συνέχεια επιλύστε τα πιο κάτω συστήματα:

(α)
$$\begin{aligned}
5x_1 &&&&&&= 35\\
2x_1 &+x_2 &&&&&= 15\\
x_1 &-x_2 &-3x_3 &&&&= 0\\
4x_1 &-2x_2 &-10x_3 &-x_4 &&&= 1\\
x_1 &+x_2 &+x_3 &+x_4 &-3x_5 &&= 3
\end{aligned}$$

(β) Το τυχαίο άνω τριγωνικό σύστημα με b = rand(6,1) και A = tril(rand(6)).

4.23 Γράψτε ένα function m-file με όνομα issquare που θα ελέγχει αν ένας πίνακας είναι τετραγωνικός ή όχι και θα επιστρέφει τις λογικές τιμές 1 και 0, αντίστοιχα.

4.24 Γράψτε ένα function m-file με όνομα isnormalized που θα ελέγχει αν ένα διάνυσμα είναι μοναδιαίο ή όχι και θα επιστρέφει τις λογικές τιμές 1 και 0, αντίστοιχα. Επιπλέον, αν η μεταβλητή εισόδου δεν είναι διάνυσμα θα πρέπει να τυπώνεται στο παράθυρο εργασίας μήνυμα λάθους.

4.25 Γράψτε ένα function m-file με όνομα issingular που θα ελέγχει αν ένας τετραγωνικός πίνακας είναι ιδιάζων ή όχι και θα επιστρέφει τις λογικές τιμές 1 και 0, αντίστοιχα. Επιπλέον, αν η μεταβλητή εισόδου δεν είναι τετραγωνικός πίνακας θα πρέπει να τυπώνεται στο παράθυρο εργασίας μήνυμα λάθους.

4.26 Γράψτε ένα function m-file με όνομα arecompatible που θα ελέγχει αν δύο τετραγωνικοί πίνακες είναι συμβιβαστοί ως προς τον πολλαπλασιασμό ή όχι και θα επιστρέφει τις λογικές τιμές 1 και 0, αντίστοιχα.

4.27 Γράψτε το m-file ticketp.m του παραδείγματος 4.6.3 χρησιμοποιώντας την εντολή if αντί της switch.

5 ΓΡΑΦΙΚΑ

Η MATLAB έχει εξαιρετικές δυνατότητες για γραφικά και είναι εφοδιασμένη με αρκετές συναρτήσεις για εύκολο και ευέλικτο σχεδιασμό επίπεδων καμπυλών, τρισδιάστατων επιφανειών, ισοϋψών, παραμετρικών δισδιάστατων αλλά και τρισδιάστατων καμπυλών κα. Το κεφάλαιο αυτό είναι μια εισαγωγή στις σημαντικότερες γραφικές συναρτήσεις της MATLAB.

5.1 Η εντολή plot

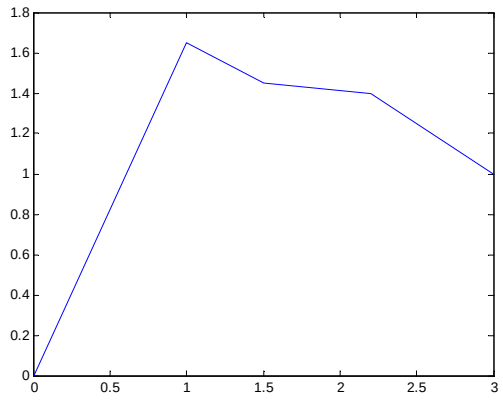
Η συνάρτηση **plot** χρησιμοποιείται για την κατασκευή του γραφήματος μιας **επίπεδης καμπύλης** τα σημεία της οποίας είναι αποθηκευμένα στα **ισομήκη διανύσματα** *x* και *y*. Αν για παράδειγμα, έχουμε τα διανύσματα

```
>> x=[0 1 1.5 2.2 3.]
x =
         0    1.0000    1.5000    2.2000    3.0000
>> y=[0 1.65 1.45 1.4 1]
y =
         0    1.6500    1.4500    1.4000    1.0000
```

με την εντολή

```
>> plot(x,y)
```

εμφανίζεται αυτόματα ένα παράθυρο γραφικών με το πιο κάτω γράφημα:



Παρατηρούμε ότι η plot ενώνει τα 5 σημεία που ορίζουν τα *x* και *y* με συνεχείς γραμμές. Υπάρχουν φυσικά και άλλες επιλογές τις οποίες θα δούμε στη συνέχεια.

Αν τώρα θέλουμε να κατασκευάσουμε το γράφημα μιας συνάρτησης $y = f(x)$ κατασκευάζουμε πρώτα το διάνυσμα *x* στο διάστημα που μας ενδιαφέρει με

ομοιόμορφα ή λογαριθμικά κατανεμημένα σημεία και με μικρό βήμα. Στη συνέχεια υπολογίζουμε το y και γράφουμε την εντολή

```
>> plot(x,y)
```

Τα διανύσματα x και y πρέπει να έχουν το ίδιο μήκος. Αυτή είναι η πιο απλή εκδοχή της εντολής. Στη συνέχεια θα δούμε πως χρησιμοποιούνται άλλες δυνατότητες όπως η επιλογή χρώματος και είδους γραμμής, η κατασκευή πολλαπλών γραφημάτων κα. Μπορείτε βέβαια να μάθετε περισσότερα με την εντολή **help plot**.

Παράδειγμα 5.1.1

Θα κατασκευάσουμε το γράφημα της $y = \cos(x)$ στο διάστημα $[-\pi, \pi]$. Ακολουθούμε τα εξής βήματα:

(α) Διαμερίζουμε το διάστημα $[-\pi, \pi]$ σε υποδιαστήματα μήκους 0.01 και αποθηκεύουμε τα σημεία x_i στο διάνυσμα x:

```
>> x=-pi:0.01:pi;
```

Αν αντί για το μήκος του κάθε υποδιαστήματος μας ενδιαφέρει το πλήθος των σημείων x_i στο διάνυσμα x, μπορούμε να χρησιμοποιήσουμε τη συνάρτηση **linspace**. Αν n είναι το επιθυμητό πλήθος σημείων γράφουμε

```
>> x=linspace(-pi,pi, n);
```

Αν παραλείψουμε το n,

```
>> x=linspace(-pi,pi);
```

η linspace θεωρεί ότι n = 101 και μας δίνει αυτόματα 101 σημεία (δηλ. χωρίζει το διάστημα σε 100 υποδιαστήματα).

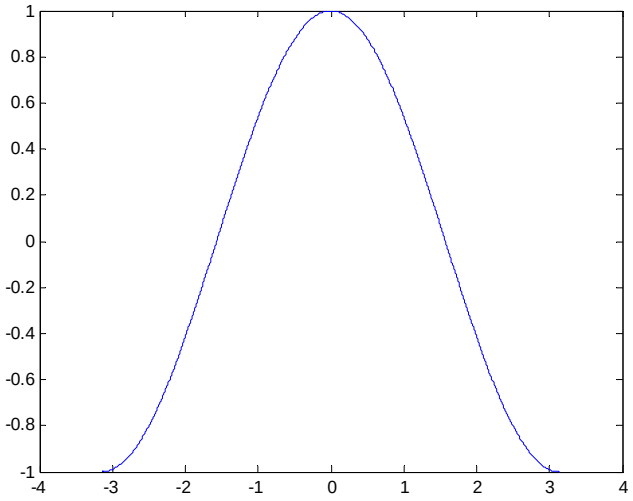
(β) Ορίζουμε το διάνυσμα y με τιμές τα $y_i = \cos(x_i)$:

```
>> y=cos(x);
```

(γ) Κατασκευάζουμε το γράφημα της $y = \cos(x)$ στο διάστημα $[-\pi, \pi]$:

```
>> plot(x,y)
```

Με αυτή την εντολή **εμφανίζεται ένα νέο παράθυρο με ένα πλούσιο μενού επιλογών και με το γράφημα της $y = \cos(x)$:**



Παρατήρηση 1

Συνοψίζοντας, μπορούμε να πάρουμε το πιο πάνω γράφημα είτε με τις εντολές

```
>> x=-pi:0.01:pi; y=cos(x); plot(x,y)
```

είτε με τις

```
>> x=linspace(-pi,pi,301); y=cos(x); plot(x,y)
```

Θα μπορούσαμε πάντως να γράψουμε απευθείας

```
>> x=-pi:0.01:pi; plot(x,cos(x))
```

ή σε μια εντολή

```
>> plot(-pi:0.01:pi,cos(-pi:0.01:pi))
```

ή ακόμα

```
>> plot(linspace(-pi,pi,301),cos(linspace(-pi,pi,301)))
```

Παρατήρηση 2

Μια συνάρτηση παρόμοια με τη linspace είναι η **logspace** η οποία παράγει διανύσματα με συνιστώσες που ισαπέχουν λογαριθμικά. Για παράδειγμα,

```
>> logspace(0,2,3)
ans =
     1    10   100
>> logspace(0,2,11)
ans =
  Columns 1 through 5
    1.0000    1.5849    2.5119    3.9811    6.3096
  Columns 6 through 10
   10.0000   15.8489   25.1189   39.8107   63.0957
  Column 11
  100.0000
```

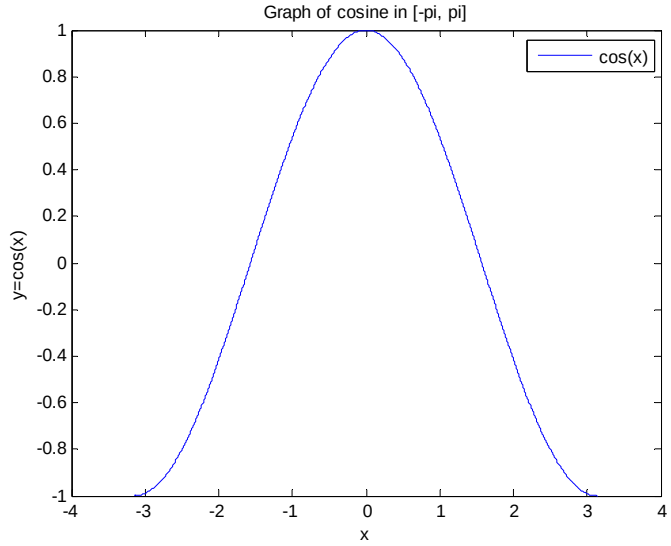
5.1.1 Χρήσιμες συναρτήσεις για γραφικά

Το γράφημα που πήραμε πιο πάνω είναι απλό αφού δεν έχει ετικέτες στους άξονες, τίτλο και λεζάντα. Όλα αυτά μπορούν να προστεθούν και να τροποποιηθούν είτε απευθείας με το μενού του παραθύρου γραφικών είτε με συναρτήσεις βιβλιοθήκης στο παράθυρο εντολών.

Με τις πιο κάτω εντολές:

```
>> xlabel('x')
>> ylabel('y=cos(x)')
>> title('Graph of cosine in [-pi, pi]')
>> legend('cos(x)')
```

το γράφημα που παίρνουμε τώρα είναι το πιο κάτω:

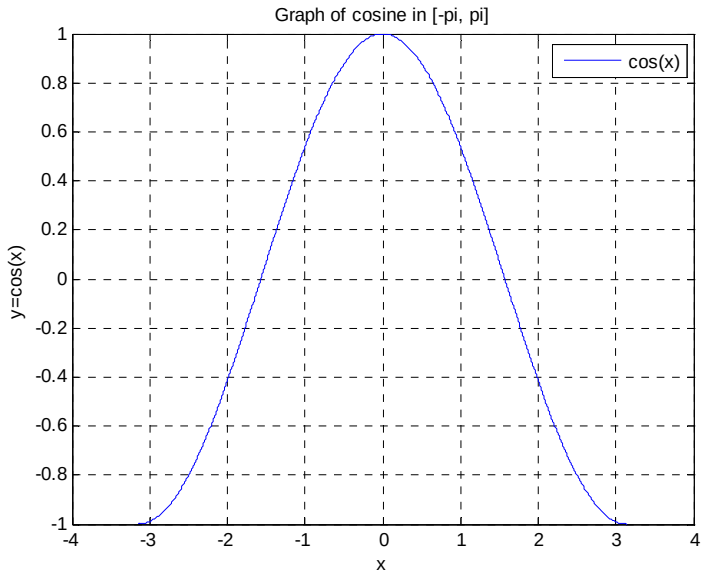


Όπως ήδη αναφέραμε, οι ετικέτες των αξόνων, ο τίτλος και η λεζάντα μπορούν να εισαχθούν **απευθείας στο παράθυρο του γραφήματος**. Κάντε απλώς κλικ στην επιλογή Insert και θα δείτε το σχετικό μενού. Αυτό περιλαμβάνει και άλλες επιλογές όπως την εισαγωγή κειμένου (text) ή βέλους (arrow) κα.

Κάνοντας διπλό κλικ στους άξονες εμφανίζεται παράθυρο επιλογών για τα χαρακτηριστικά τους τα οποία μπορούμε να αλλάξουμε. Για παράδειγμα μπορούμε να αλλάξουμε τα όρια του άξονα των y σε [–1.2, 1.2]. Μπορούμε επίσης να δημιουργήσουμε πλέγμα με την εντολή:

```
>> grid
```

Παίρνουμε τότε το γράφημα:



Οι κυριότερες συναρτήσεις για γραφικά είναι συγκεντρωμένες στον ακόλουθο πίνακα:

Εντολή	Περιγραφή	Παράδειγμα
plot	Δημιουργεί το γράφημα του y συναρτήσει του x	plot(x,y)
title	Προσθήκη τίτλου	title('Titlos')
xlabel	Προσθήκη ετικέτας στον οριζόντιο άξονα	xlabel('Xronos, t')
ylabel	Προσθήκη ετικέτας στον κατακόρυφο άξονα	ylabel('Taxuthta, cm/s')
legend	Προσθήκη λεζάντας	legend('First', 'Second')
text	Προσθήκη κειμένου στη θέση (x_i, y_i)	text(xi, yi, 'string')
grid	Δημιουργία πλέγματος	grid grid on grid off
figure	Άνοιγμα (άλλου) παραθύρου γραφικών	figure(2)
hold	Πάγωμα του τρέχοντος παραθύρου γραφικών για το σχεδιασμό και άλλων καμπυλών	hold on/hold off
axis	Κλείδωμα/ξεκλείδωμα αξόνων Ίσες μονάδες αξόνων Διαγραφή αξόνων Όρια αξόνων.	axis axis equal axis off axis([xmin, xmax,ymin,ymax])

Στις ετικέτες, τον τίτλο και τη λεζάντα δεν μπορούμε να έχουμε ελληνικά με απλό τρόπο. Για όσους είναι εξοικειωμένοι με το LaTeX είναι εύκολο να αντιληφθούν τις εντολές:

```
>> title('Graph of cosine in [-\pi, \pi]')
>> legend('e^x')
```

Στο LaTeX τα ελληνικά γράμματα τα παίρνουμε γράφοντας '\' και μετά το γράμμα ολογράφως, π.χ. \alpha.

Η εντολή **figure** μας επιτρέπει να ανοίξουμε ένα νέο παράθυρο γραφικών εκτός από το προεπιλεγμένο με όνομα Figure 1. Αν έχουμε ήδη δημιουργήσει το Figure 1 και δεν θέλουμε να το διαγράψουμε, γράφουμε

```
>> figure(2)
```

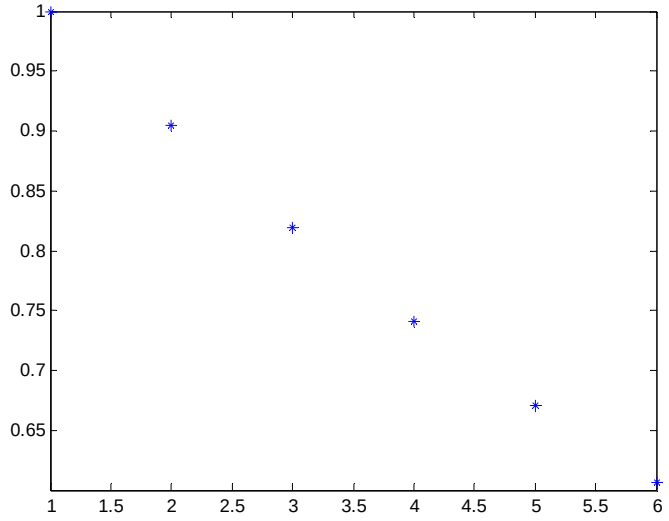
Την επόμενη φορά που θα δημιουργήσουμε γραφικά, αυτά θα εμφανιστούν στο παράθυρο με όνομα Figure 2. Αυτό θα είναι το παράθυρο εργασίας για όλες τις γραφικές παραστάσεις μέχρι να χρησιμοποιήσουμε εκ νέου την εντολή figure.

Η εντολή **hold** χρησιμοποιείται όταν θέλουμε να σχεδιάσουμε επιπλέον καμπύλες στο ίδιο γράφημα. Θα τη συζητήσουμε στη σχετική παράγραφο.

Αν παραλείψουμε το x, η εντολή plot(y) σχεδιάζει τα στοιχεία y_i συναρτήσει των δεικτών i. Έτσι οι εντολές

```
>> x=0:-0.1:-0.5;
>> plot(exp(x), '*')
```

δίνουν το ακόλουθο γράφημα:

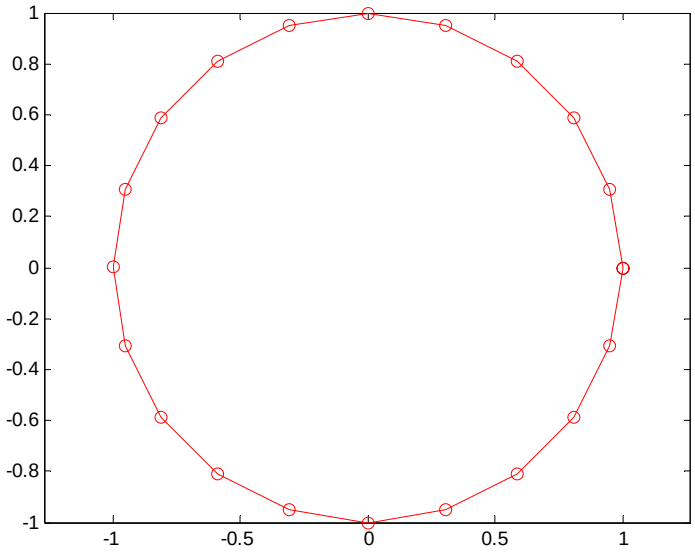


Επειδή το διάνυσμα έχει μήκος 6 στον άξονα των *i* οι τιμές κυμαίνονται από 1 έως 6.

Η εντολή **plot** δουλεύει ακόμα και όταν τα δεδομένα εισόδου είναι μιγαδικοί αριθμοί. Σε αυτή την περίπτωση, το μιγαδικό μέρος των αριθμών αγνοείται, εκτός αν χρησιμοποιήσουμε μόνο ένα δεδομένο εισόδου οπότε το γράφημα που παίρνουμε αντιστοιχεί στη γραφική παράσταση του μιγαδικού μέρους έναντι του πραγματικού. Συγκεκριμένα, η εντολή **plot(Z)** όταν το *Z* είναι μιγαδικό διάνυσμα (ή πίνακας) είναι ισοδύναμη με την **plot(real(Z), imag(Z))**. Για παράδειγμα, οι εντολές

```
>> t=0:pi/10:2*pi;
>> plot(exp(i*t),'-o')
>> axis equal
```

παράγουν το ακόλουθο γράφημα



Η εντολή **axis equal** αναγκάζει τους άξονες να έχουν το ίδιο μέγεθος, έτσι ώστε η γραφική παράσταση να είναι κυκλική.

5.2 Η εντολή ezplot

Η συνάρτηση **ezplot** καθώς και η συνάρτηση **fplot** που θα συζητήσουμε στην επόμενη παράγραφο παράγουν πιο εύκολα απ' ότι η plot τη γραφική παράσταση της $y = f(x)$, ειδικά αν η f είναι συνάρτηση βιβλιοθήκης ή ανώνυμη συνάρτηση ή έχει οριστεί μέσω της εντολής inline.

Η συνάρτηση ezplot (easy plot) έχει σημαντικά πλεονεκτήματα σε σχέση με τις plot και fplot:

- βρίσκει αυτόματα τα διαστήματα των αξόνων και δίνει επίσης τη δυνατότητα στο χρήστη να τα επιλέξει ο ίδιος,
- μπορεί να κάνει το γράφημα πεπλεγμένης συνάρτησης, και
- μπορεί να κάνει το γράφημα παραμετρικής καμπύλης.

Αν $f(x)$ είναι μια συνάρτηση βιβλιοθήκης ή συνάρτηση ορισμένη από το χρήστη, τότε

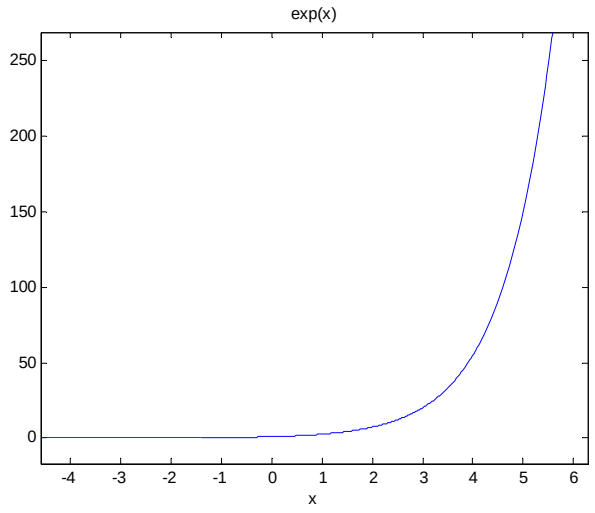
- η **ezplot(f)** παράγει το γράφημα της f επιλέγοντας τα διαστήματα των αξόνων των x και y. Σύμφωνα με το εγχειρίδιο της MATLAB η προεπιλογή για το x είναι $-2\pi < x < 2\pi$.
- η **ezplot(f, a, b)** ή η **ezplot(f, [a, b])** παράγει το γράφημα της f έτσι ώστε οι τιμές του x να βρίσκονται στο $[a, b]$.

Μερικά παραδείγματα χρήσης της συνάρτησης φαίνονται πιο κάτω.

Με την εντολή

```
>> ezplot('exp(x)')
```

παίρνουμε το γράφημα

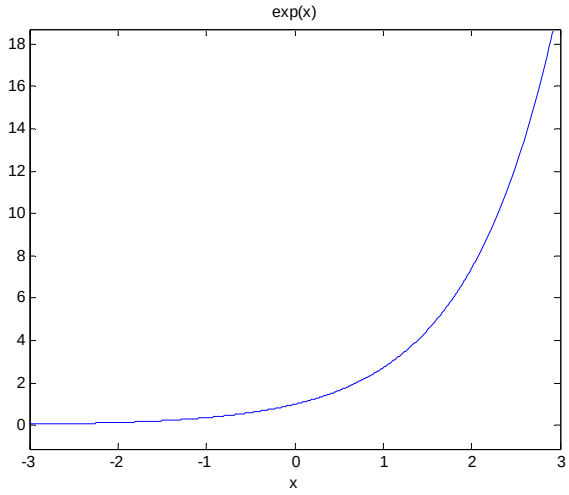


Παρατηρούμε ότι η η συνάρτηση προσθέτει επίσης και τίτλο στο γράφημα καθώς και ετικέτα στον άξονα των x.

Αν επιλέξουμε τώρα το διάστημα $[-3, 3]$ για τις τιμές του x,

```
>> ezplot('exp(x)', -3, 3 )
```

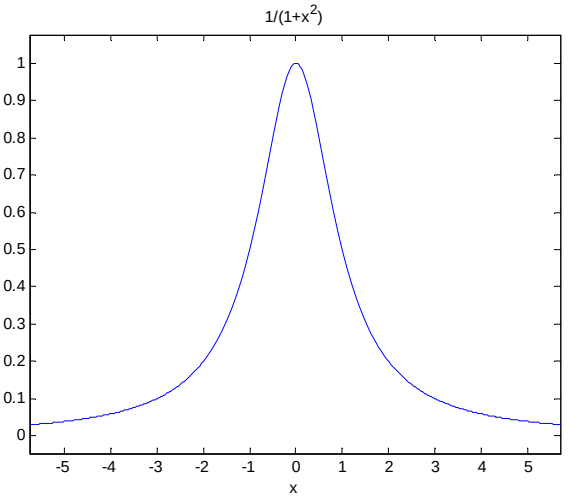
παίρνουμε το γράφημα



Ας δοκιμάσουμε τώρα τη συνάρτηση

$$f(x)=\frac{1}{1+x^2}$$

```
>> ezplot('1./(1+x.^2)')
```



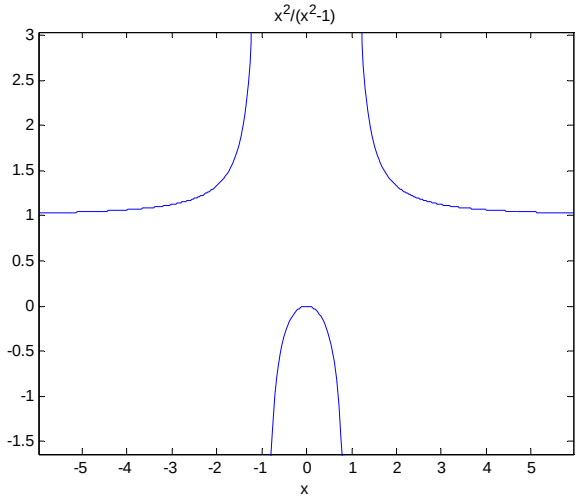
Η συνάρτηση *f* μπορεί να οριστεί εξωτερικά σαν ανώνυμη συνάρτηση ή με την εντολή inline. Για παράδειγμα, με τις εντολές

```
>> f = @(x) x.^2./(x.^2-1);
>> ezplot(f)
```

παίρνουμε τη γραφική παράσταση της

$$f(x)=\frac{x^2}{x^2-1},$$

σε διαστήματα των *x* και *y* που επιλέγει η MATLAB έτσι ώστε το γράφημα περιέχει όλες τις σημαντικές πληροφορίες της συνάρτησης και να είναι ευπαρουσίαστο.



Με την ezplot μπορούμε να σχεδιάσουμε εύκολα τα γραφήματα **πεπλεγμένων συναρτήσεων** (implicit functions) της μορφής

$$f(x, y) = 0.$$

Χαρακτηριστικά παραδείγματα είναι τα εξής:

- η **ezplot(f)** παράγει το γράφημα της f στο προεπιλεγμένο χωρίο που ορίζεται από τις $-2\pi < x < 2\pi$ και $-2\pi < y < 2\pi$.
- η **ezplot(f, [a, b])** παράγει το γράφημα της f με $a < x < b$ και $a < y < b$.
- η **ezplot(f, [xmin, xmax, ymin, ymax])** παράγει το γράφημα της f στο χωρίο που ορίζεται από τις xmin $< x <$ xmax και ymin $< y <$ ymax.

Παράδειγμα 5.2.1

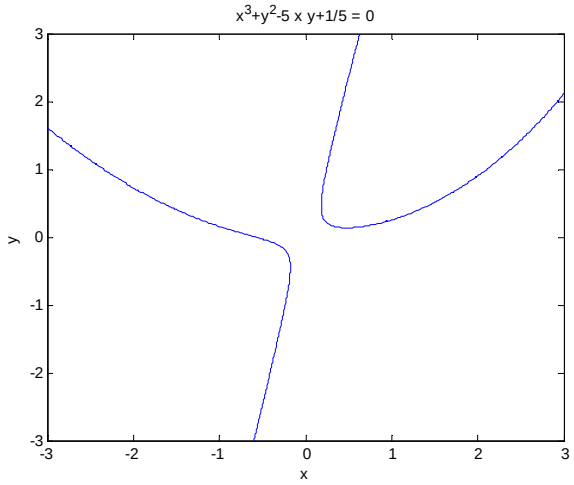
Για να κατασκευάσουμε το γράφημα της πεπλεγμένης συνάρτησης

$$x^3 + y^2 - 5xy + \frac{1}{5} = 0$$

γράφουμε

```
>> ezplot('x^3+y^2-5*x*y+1/5',[-3,3])
```

Παίρνουμε έτσι το γράφημα:



Παράδειγμα 5.2.2

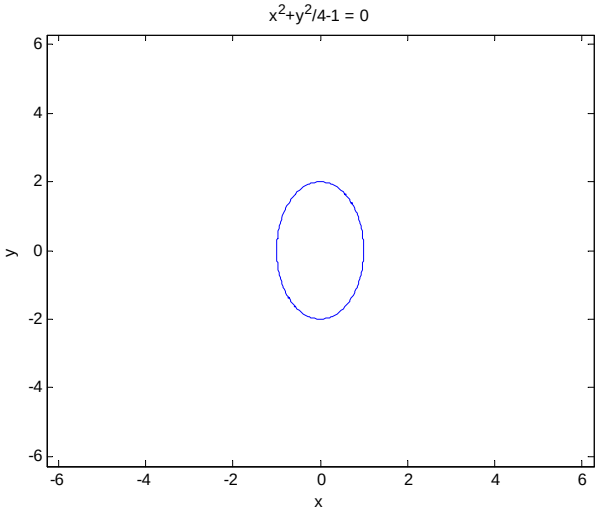
Για να κατασκευάσουμε το γράφημα της έλλειψης

$$x^2 + \frac{y^2}{4} - 1 = 0$$

γράφουμε

```
>> ezplot('x^2+y^2/4-1')
```

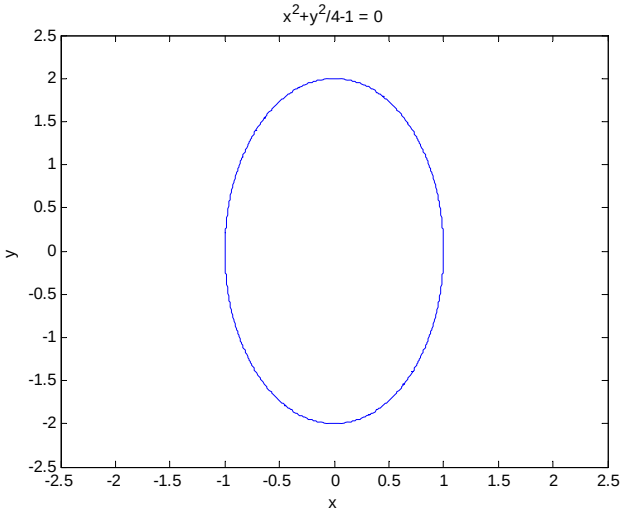
Παίρνουμε έτσι το γράφημα:



Αν γράψουμε

```
>> ezplot('x^2+y^2/4-1',[-2.5,2.5])
```

παίρνουμε το εξής:



Η ezplot μπορεί επίσης να σχεδιάσει καμπύλες που ορίζονται παραμετρικά, δηλ.:

$$x = x(t), y = y(t)$$

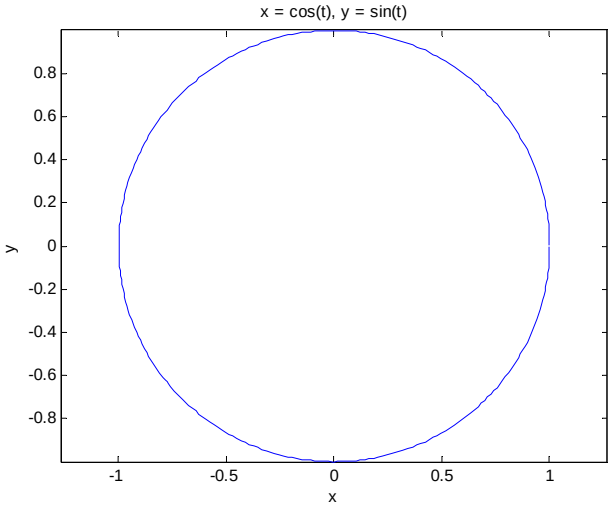
όπου η παράμετρος t παίρνει τιμές σε κάποιο διάστημα $[a, b]$. Στη περίπτωση αυτή γράφουμε

ezplot('x(t)', 'y(t)', [a,b])

Για παράδειγμα, με την εντολή

```
>> ezplot('cos(t)', 'sin(t)',[0,2*pi])
```

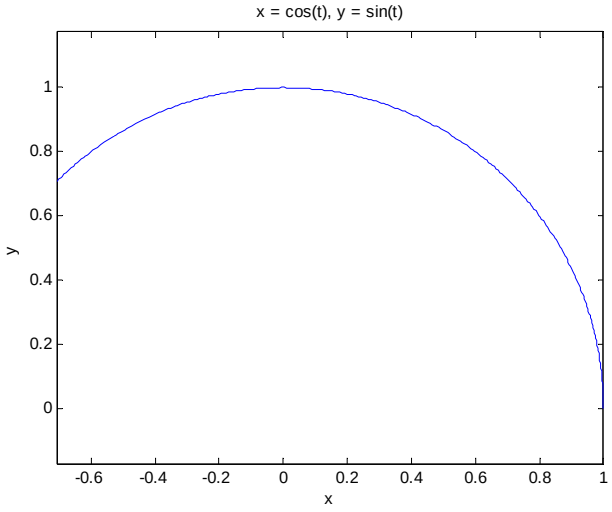
παίρνουμε τον κύκλο



ενώ για τιμές της παραμέτρου *t* στο διάστημα [0, 3π/4],

```
>> ezplot('cos(t)', 'sin(t)',[0,0.75*pi])
```

παίρνουμε το πιο κάτω γράφημα:



Σημειώνουμε ότι μπορούμε να σχεδιάσουμε την παραμετρική καμπύλη με τη συνάρτηση plot ως εξής:

```
>> t=0:3*pi/400:3*pi/4;
>> x=cos(t); y=sin(t);
>> plot(x,y)
```

5.2.1 Η εντολή comet

Η εντολή **comet** μας δίνει ένα κινούμενο γράφημα μιας παραμετρικής καμπύλης. Ένας κύκλος που αποτελεί την κεφαλή του κομήτη διαγράφει την τροχιά που αντιστοιχεί στα σημεία της καμπύλης. Η πιο απλή της μορφή είναι η

comet(x,y)

Ο αναγνώστης μπορεί να δοκιμάσει το πιο κάτω παράδειγμα:

```
>> t=0:pi/1000:8*pi;
>> x=cos(t);
>> y=sin(t);
>> comet(x,y)
```

Θα πρέπει να σημειώσουμε ότι δεν μπορούμε να τυπώσουμε ή να αποθηκεύσουμε το γράφημα της καμπύλης. Το μόνο που τυπώνεται είναι η κεφαλή του κομήτη. Αυτός είναι και ο λόγος που δεν δείχνουμε το γράφημα εδώ.

5.3. Η εντολή fplot

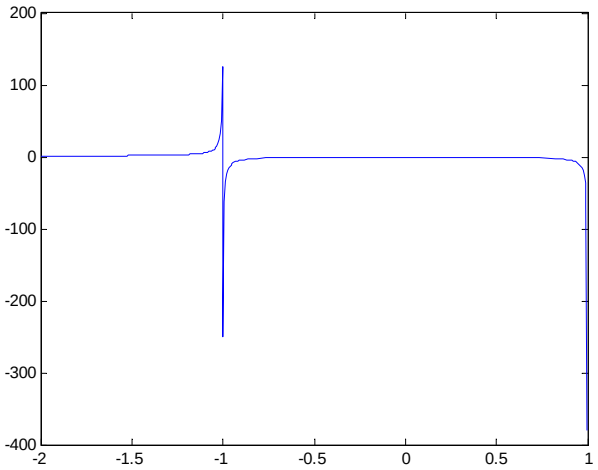
Σε αντίθεση με την ezplot, στη συνάρτηση fplot είναι υποχρεωτική η εισαγωγή του πεδίου σχεδίασης του γραφήματος μιας συνάρτησης. Δύο χαρακτηριστικές περιπτώσεις είναι οι εξής:

- η **fplot(f, [xmin, xmax])** παράγει το γράφημα της f με xmin < x < xmax
- η **fplot(f, [xmin, xmax, ymin, ymax])** παράγει το γράφημα της f στο χωρίο που ορίζεται από τις xmin < x < xmax και ymin < y <ymax.

Για παράδειγμα, με τις εντολές

```
>> f = @(x) x.^2./(x.^2-1);
>> fplot(f, [-2,1])
```

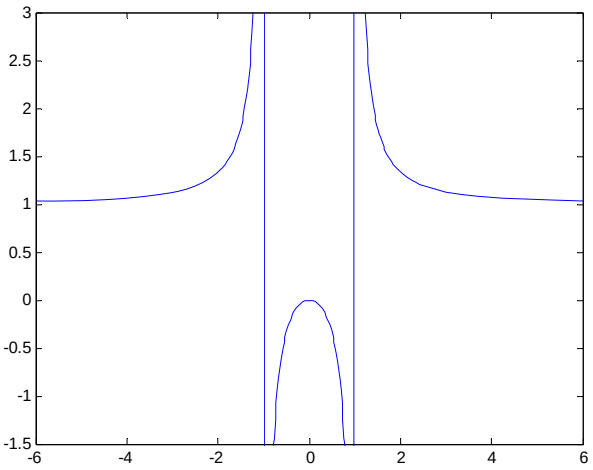
παίρνουμε την όχι και τόσο ευπαρουσίαστη γραφική παράσταση της $f(x)=\frac{x^2}{x^2-1}$:



Αν ορίσουμε το χωρίο του γραφήματος με την εντολή

```
>> fplot(f,[-6 6 -1.5 3])
```

παίρνουμε το εξής γράφημα



Το γράφημα αυτό διαφέρει από αυτό που παίρνουμε με τη συνάρτηση ezplot. Ποιες είναι οι διαφορές και πως μπορούμε να τις εξηγήσουμε;

Για άλλες δυνατότητες που μας δίνει η συνάρτηση fplot, δοκιμάστε την εντολή **help fplot**. Θα συζητήσουμε κάποιες από αυτές στη συνέχεια.

Όπως και στην ezplot, μπορούμε να ορίσουμε απευθείας τη συνάρτησή μας στην fplot. Για παράδειγμα, δοκιμάστε την εντολή

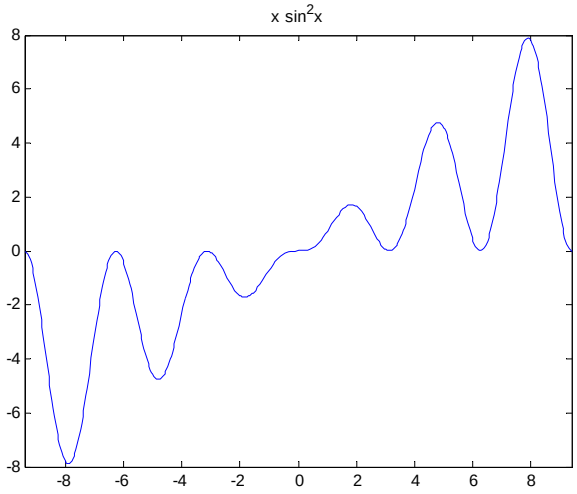
```
>> fplot('sin(x)',[-2*pi, 2*pi])
```

Παράδειγμα 5.3.1

Για να σχεδιάσουμε το γράφημα της συνάρτησης $f(x) = x \sin^2 x$ στο $[-3\pi,3\pi]$ χρησιμοποιούμε την εντολή:

```
>> fplot('x*sin(x)^2',[-3*pi, 3*pi]), title('x sin^2x')
```

Το γράφημα που πήραμε είναι το ακόλουθο:



5.4 Χρώματα, σύμβολα και γραμμές

Η εντολή plot παρέχει μια ευρεία επιλογή χρωμάτων, συμβόλων και τύπων γραμμών. Ενώ η εντολή

```
>> plot(x,y)
```

μας δίνει γράφημα με μπλε συνεχή γραμμή, η εντολή

```
>> plot(x,y, ' [color][stype][ltype]' )
```

μας επιτρέπει να επιλέξουμε το χρώμα του γραφήματος και τους τύπους συμβόλου και γραμμής.

Το χρώμα **[color]** ορίζεται με ένα από τα πιο κάτω γράμματα:

[color]	Color	Χρώμα
b	blue	**μπλε**
g	green	**πράσινο**
r	red	**κόκκινο**
c	cyan	**κυανό**
m	magenta	**μοβ**
y	yellow	**κίτρινο**
k	black	**μαύρο**
w	white	**άσπρο**

Για τον τύπο **[stype]** του συμβόλου έχουμε τις εξής επιλογές:

[stype]	Symbol	Σύμβολο
.	point	τελεία
o	circle	κύκλος
x	x-mark	χι
+	plus	συν
*	star	αστερίσκος
s	square	τετράγωνο
d	diamond	ρόμβος
v	triangle (down)	κάτω τρίγωνο
^	triangle (up)	άνω τρίγωνο
<	triangle (left)	αριστερό τρίγωνο
>	triangle (right)	δεξιό τρίγωνο
p	pentagram	πεντάλφα
h	hexagram	εξάλφα

Για τον τύπο **[ltype]** της γραμμής έχουμε τις εξής επιλογές:

[ltype]	Line type	Τύπος γραμμής
-	solid	συνεχής
:	dotted	λεπτή διακεκομμένη
--	dashed	αδρή διακεκομμένη
-.	dashdot	διακεκομμένη-τελείες

Για παράδειγμα, η εντολή

```
>> plot (x, y, 'g--')
```

μας δίνει πράσινη αδρή διακεκομμένη γραμμή, ενώ η εντολή

```
>> plot (x, y, 'm:')
```

μας δίνει μοβ λεπτή διακεκομμένη γραμμή.

Η εντολή

```
>> plot (x, y, 'c+:')
```

σχεδιάζει μια κυανή λεπτή διακεκομμένη γραμμή και το σύμβολο + σε κάθε σημείο.

Δεν είναι απαραίτητο να προσδιορίσουμε το χρώμα και τον τύπο γραμμής. Σ' αυτή την περίπτωση η MATLAB χρησιμοποιεί τις **προεπιλογές** (defaults):

- Για το χρώμα [color] το b (μπλε).
- Για τον τύπο [type] το – (συνεχής γραμμή).

Έτσι η εντολή

```
>> plot (x, y, 'g')
```

μας δίνει πράσινη συνεχή γραμμή ενώ η εντολή

```
>> plot (x, y, 'o')
```

μας δίνει μπλε κύκλους σε κάθε σημείο.

Παράδειγμα 5.4.1

Ορίζουμε τα πιο κάτω σημεία και τις αντίστοιχες τιμές της exp(x):

```
>> x = -1:0.2:1;
>> y=exp(x);
```

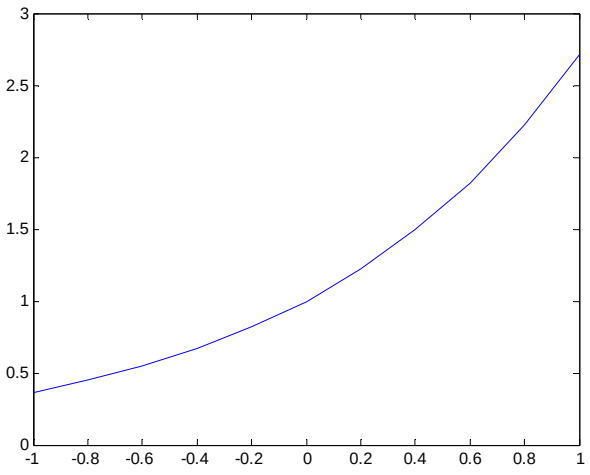
Οι εντολές

```
>> plot(x, y)
```

ή

```
>>plot(x, y, 'b- ')
```

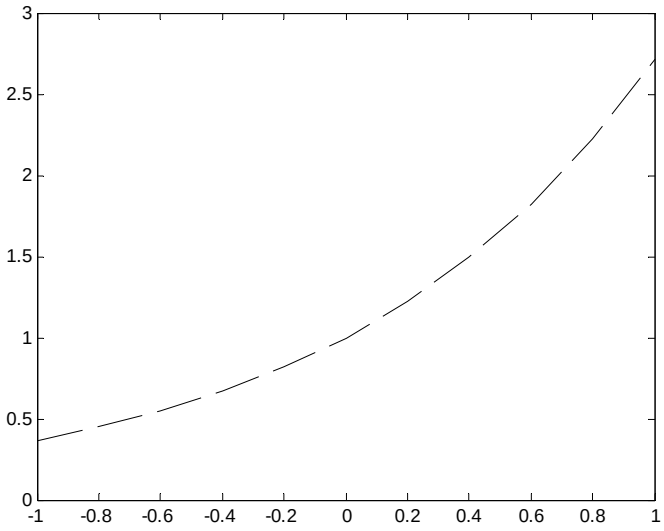
μας δίνουν το ίδιο γράφημα (μπλε συνεχής γραμμή):



Η εντολή

```
>> plot(x, y, 'k--')
```

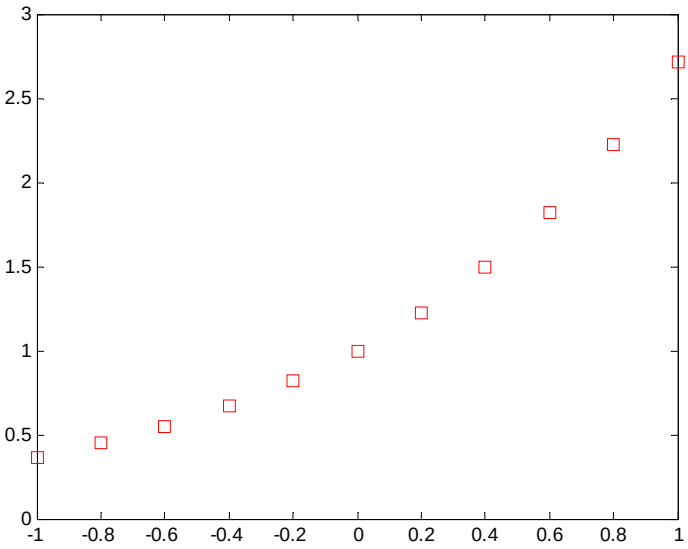
μας δίνει μαύρη αδρή διακεκομμένη γραμμή:



Η εντολή

```
>> plot( x, y, 'rs')
```

μας δίνει κόκκινα τετράγωνα:



5.5 Πολλαπλά γραφήματα

Με την εντολή plot μπορούμε να έχουμε διαφορετικές καμπύλες στο ίδιο γράφημα. Για παράδειγμα, αν

y1 = f1(x), y2 = f2(x), …

μπορούμε να δώσουμε την εντολή

>> plot(x1, y1, ' [colour][stype][ltype]', x2, y2, ' [colour][stype][ltype]', …….)

Στην περίπτωση πολλαπλών γραφημάτων η λεζάντα (legend) ορίζεται με ανάλογο τρόπο:

>> legend ('legend y1', 'legend y2', ……)

όπου το 'legend y1' είναι η λεζάντα για την y1 κοκ.

Παράδειγμα 5.5.1

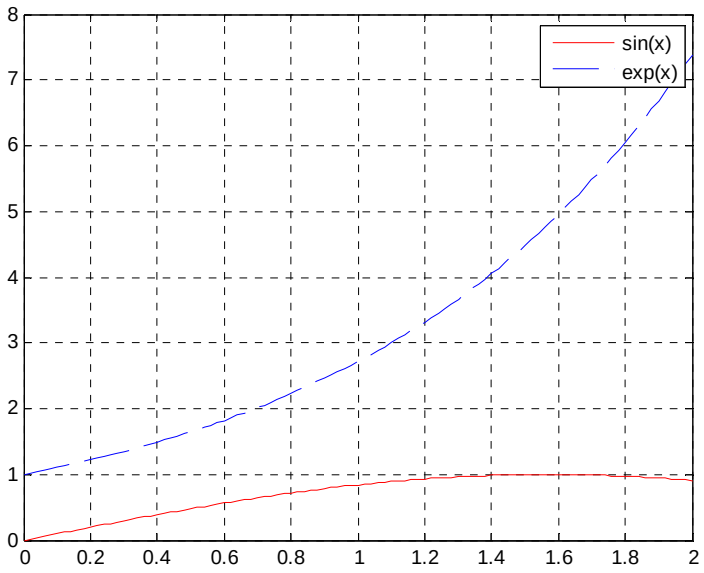
Θα σχεδιάσουμε στο διάστημα [0, 2] τις εξής καμπύλες:

- την καμπύλη $y = \sin(x)$ με κόκκινη συνεχή γραμμή
- την καμπύλη $z = \exp(x)$ με μπλε αδρή διακεκομμένη γραμμή.

Θα δημιουργήσουμε επίσης πλέγμα και θα έχουμε λεζάντα. Ακολουθούν οι σχετικές εντολές:

```
>> x=0:0.02:2;
>> y=sin(x);
>> z=exp(x);
>> plot( x, y, 'r', x, z, '--')
>> grid
>> legend ( 'sin(x)', 'exp(x)' )
```

Παίρνουμε με τις εντολές αυτές το πιο κάτω γράφημα:



Μπορούμε επίσης να προσθέσουμε μια καινούργια καμπύλη σε ένα υφιστάμενο γράφημα με την εντολή **hold on**.

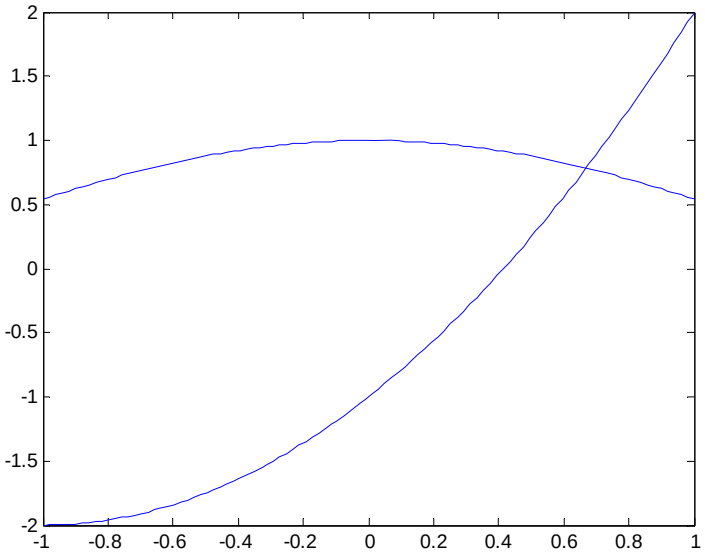
Για παράδειγμα, οι εντολές

```
>> t=linspace(-1,1);
>> y=t.^2 + 2*t -1;
>> plot(t,y)
```

μας δίνουν τη γραφική παράσταση του πολυωνύμου $t^2 + 2t - 1$. Αν τώρα γράψουμε

```
>> hold on
>> z=cos(t)
>> plot(t,z)
```

η γραφική παράσταση του cos(t) θα εμφανιστεί στο ίδιο γράφημα (με το ίδιο χρώμα):



Η εντολή αυτή *δεσμεύει* το υφιστάμενο γράφημα, έτσι όταν φτιάξουμε επιπρόσθετα γραφήματα τότε αυτά θα προστεθούν σε αυτό που ήδη έχουμε. Για να *ελευθερώσουμε* το γράφημα, χρησιμοποιούμε την εντολή **hold off**.

Παράδειγμα 5.5.2

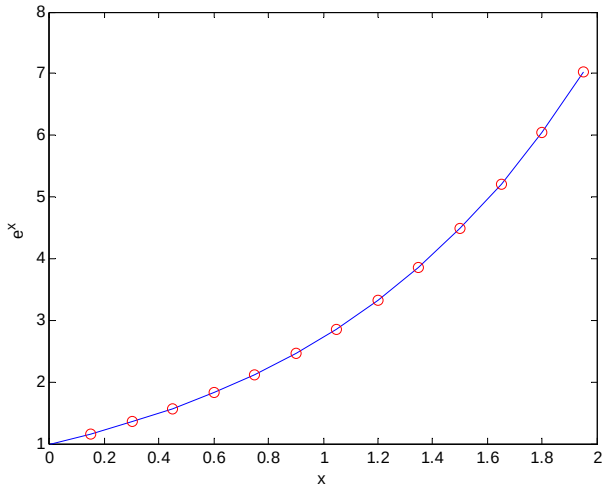
Η εντολή

```
>> plot( x, y, 'b', x, y, 'ro')
```

σχεδιάζει τα ίδια δεδομένα δύο φορές. Την πρώτη με μπλε συνεχή γραμμή και τη δεύτερη με (ασύνδετους) κόκκινους κύκλους στα δεδομένα σημεία. Με τις εντολές

```
>> x=0:0.15:2;
>> y=exp(x);
>> plot( x, y, 'b', x, y, 'ro')
>> xlabel('x')
>> ylabel('e^x')
```

παίρνουμε το ακόλουθο γράφημα:



Στην περίπτωση που το διάνυσμα της ανεξάρτητης μεταβλητής είναι κοινό για όλες τις καμπύλες, αντί της εντολής

plot(x,y1, x,y2, x,y3,…),

μπορούμε επίσης να γράψουμε

plot(x, Y)

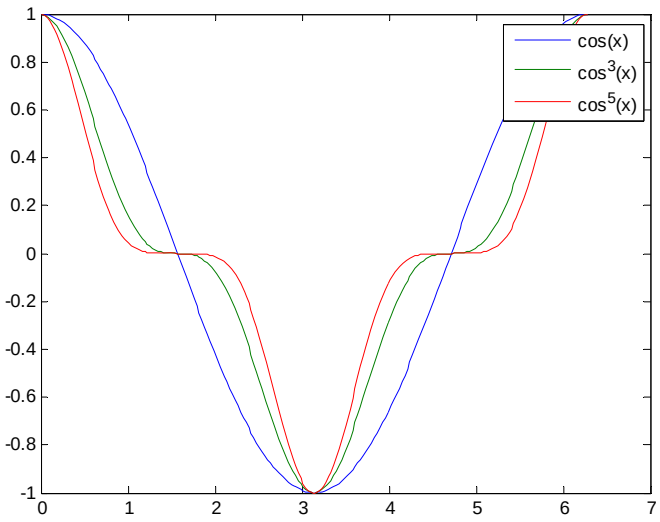
όπου Y = [y1; y2; y3; …].

Παράδειγμα 5.5.3

Θα σχεδιάσουμε τα γραφήματα των $\cos(x)$, $\cos^3(x)$ και $\cos^5(x)$ και θα προσθέσουμε σχετική λεζάντα:

```
>> x=0:pi/100:2*pi;
>> Y=[cos(x); cos(x).^3; cos(x).^5];
>> plot(x,Y)
>> legend('cos(x)','cos^3(x)', 'cos^5(x)')
```

Με τις εντολές αυτές παίρνουμε το γράφημα που ακολουθεί:



Πολλαπλά γραφήματα με την fplot

Με την fplot μπορούμε εύκολα να σχεδιάσουμε διάφορες καμπύλες στο ίδιο γράφημα γράφοντας

fplot('[f1(x), f2(x), ….]', [xmin, xmax])

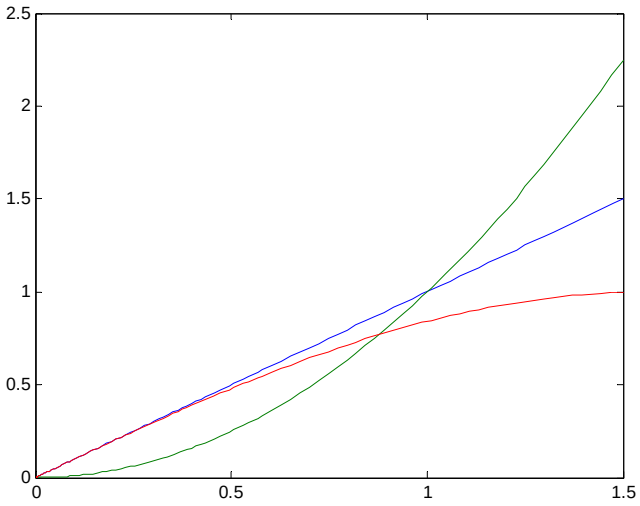
ή

fplot('[f1(x), f2(x), ….]', [xmin, xmax, ymin, ymax])

Για παράδειγμα, η εντολή

```
>> fplot('[x, x^2, sin(x)]', [0, 1.5])
```

παράγει το πιο κάτω γράφημα



5.6 Άλλες χρήσιμες εντολές για γραφικά

5.6.1 Λογαριθμικοί άξονες

Οι εντολές του πίνακα που ακολουθεί χρησιμοποιούνται για την κατασκευή λογαριθμικών ή ημιλογαριθμικών γραφημάτων (συμπεριλάβαμε και την plot για λόγους πληρότητας).

Εντολή	Περιγραφή
plot(x,y)	Κανονικό ή γραμμικό γράφημα. Οι άξονες των *x* και *y* είναι γραμμικοί.
loglog(x,y)	Λογαριθμικό γράφημα. Οι άξονες των *x* και *y* είναι λογαριθμικοί.
semilogx(x,y)	Ημιλογαριθμικό γράφημα. Ο άξονας των *x* είναι λογαριθμικός και των *y* γραμμικός.
semilogy(x,y)	Ημιλογαριθμικό γράφημα. Ο άξονας των *x* είναι γραμμικός και των *y* λογαριθμικός.

Θα πρέπει να έχουμε υπόψη μας ότι μπορούμε να αλλάξουμε ένα άξονα από γραμμικό σε λογαριθμικό ή αντίστροφα χρησιμοποιώντας τις επιλογές στο παράθυρο του γραφήματος. Τις επιλογές αυτές μπορούμε να τις πάρουμε επίσης με διπλό κλικ στον άξονα.

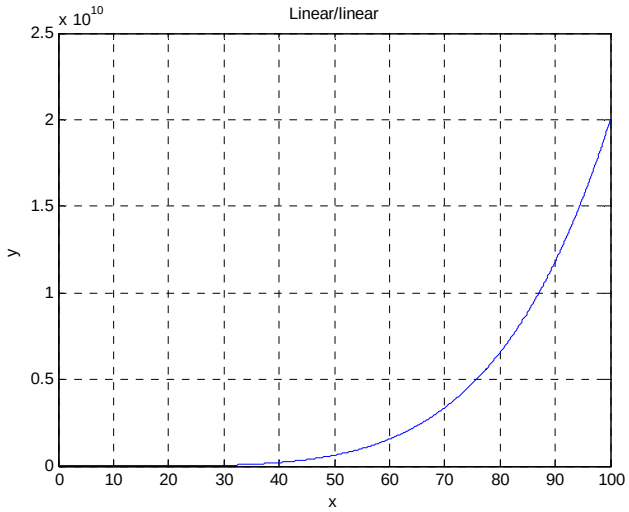
Παράδειγμα 5.6.1

Θα σχεδιάσουμε το γράφημα της $y = 1 + 2x^5$ με τέσσερις διαφορετικούς τρόπους. Ορίζουμε πρώτα τα διανύσματα x και y στο διάστημα [0, 100]:

```
>> x=0:0.1:100;
>> y=1+2*x.^5;
```

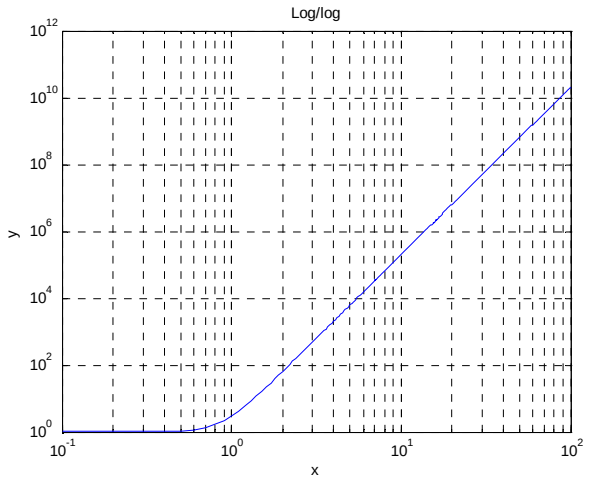
(α) Κανονικό γράφημα

```
>> plot(x,y), grid, xlabel('x'), ylabel('y')
>> title('Linear/linear')
```



(β) Λογαριθμικό γράφημα

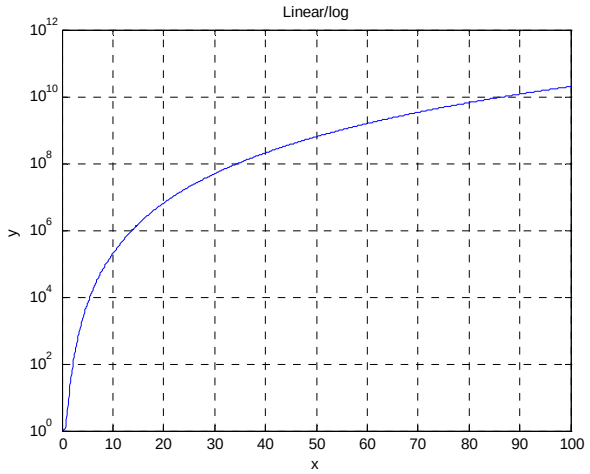
```
>> loglog(x,y), grid, xlabel('x'), ylabel('y')
>> title('Log/log')
```



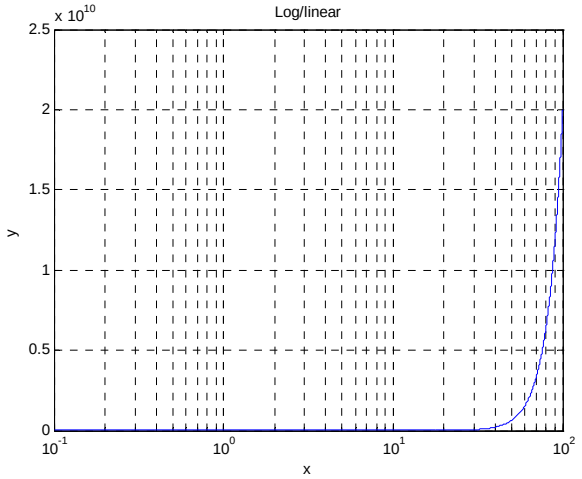
(γ) Ημιλογαριθμικά γραφήματα

```
>> semilogy(x,y), grid, xlabel('x'), ylabel('y')
```

```
>> title('Linear/log')
```




```
>> semilogx(x,y), grid, xlabel('x'), ylabel('y')
>> title('Log/linear')
```

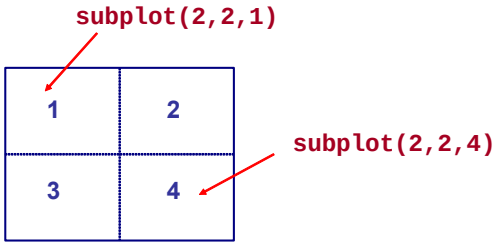


5.6.2 Πολλαπλά γραφήματα στο ίδιο παράθυρο

Η συνάρτηση **subplot** μας επιτρέπει να βάλουμε πολλά γραφήματα στο ίδιο παράθυρο γραφικών. Η εντολή

subplot(m,n,p)

διαμερίζει το παράθυρο γραφικών σε $m \times n$ υποπαράθυρα και τοποθετεί το επόμενο γράφημα στη θέση p. Η αρίθμηση των γραφημάτων γίνεται κατά γραμμές, π.χ.

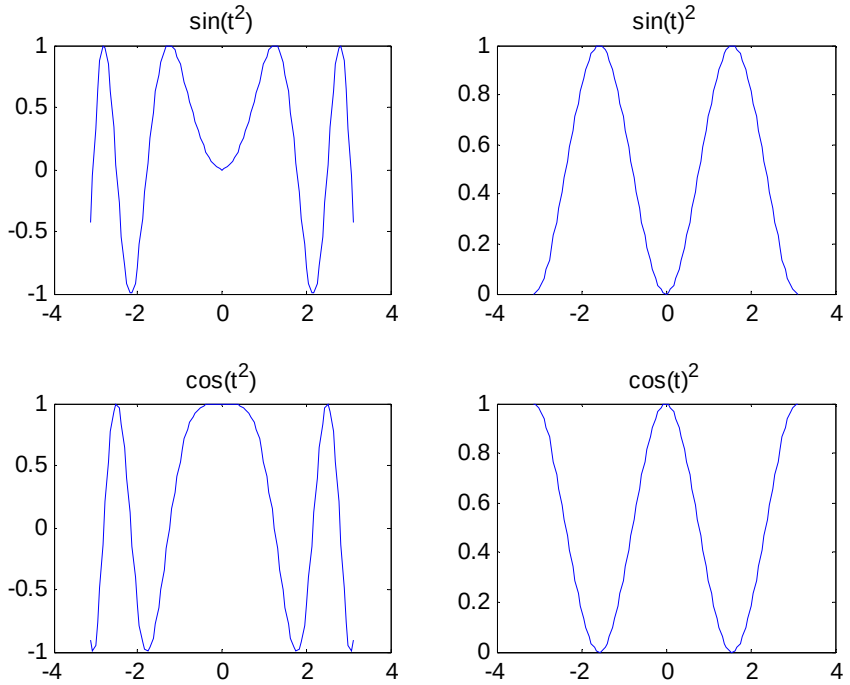


Για παράδειγμα, οι εντολές

```
>>t = -pi:2*pi/100:pi;
```

```
>>f1=sin(t.^2);
>>f2=(sin(t)).^2;
>>f3=cos(t.^2);
>>f4=(cos(t)).^2;
>>subplot(2,2,1);plot(t,f1);
>>title('sin(t^2)')
>>subplot(2,2,2);plot(t,f2);
>>title('sin(t)^2')
>>subplot(2,2,3);plot(t,f3);
>>title('cos(t^2)')
>>subplot(2,2,4);plot(t,f4);
>>title('cos(t)^2')
```

δίνουν το ακόλουθο πολλαπλό γράφημα (στο ίδιο παράθυρο):

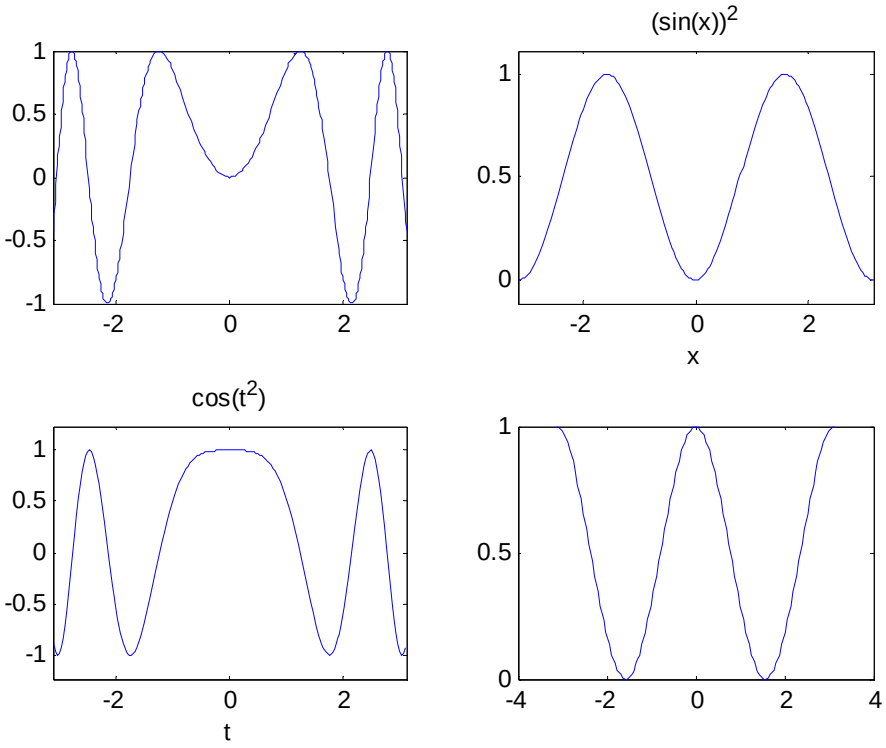


Παράδειγμα 5.6.2

Θα κατασκευάσουμε το πολλαπλό γράφημα που είδαμε πιο πάνω χρησιμοποιώντας ανάμεικτα τις συναρτήσεις fplot, ezplot και plot. Με τις εντολές

```
>> x=-pi:2*pi/100:pi;
>> f2=@(x) (sin(x))^2; ezplot(f2)
f2 =
    @(x) (sin(x))^2
>> f3=@(t) cos(t^2);
f3 =
    @(t) cos(t^2)
>> subplot(2,2,1), fplot('sin(t*t)',[-pi,pi])
>> subplot(2,2,2), ezplot(f2,[-pi,pi])
>> subplot(2,2,3), ezplot(f3,[-pi,pi])
>> subplot(2,2,4), plot(x,(cos(x)).^2)
```

παίρνουμε:



5.6.3 Γραφήματα σε πολικές συντεταγμένες

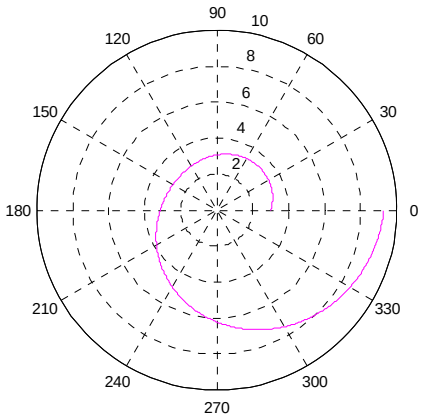
Η εντολή **polar** μας επιτρέπει να πάρουμε γραφικές παραστάσεις συναρτήσεων σε πολικές συντεταγμένες. Αν $r = r(\theta)$ είναι η καμπύλη που θέλουμε να σχεδιάσουμε, γράφουμε την εντολή

polar(theta,r)

Για παράδειγμα, για την $r = 3\cos^2(\theta/2)$, $0 \le \theta \le 2\pi$, με τις εντολές

```
>> t=0:0.01:2*pi;
>> r=3*cos(t/2).^2+t;
>> polar(t,r 'm')
```

παίρνουμε το γράφημα



Παράδειγμα 5.6.3

Θα σχεδιάσουμε στο ίδιο γράφημα τις καμπύλες

$$r_1(\theta)=\cos\theta,\quad r_2(\theta)=\cos^2\theta+1,\quad r_3(\theta)=\cos^3\theta,\quad r_4(\theta)=\cos^3\theta+1$$

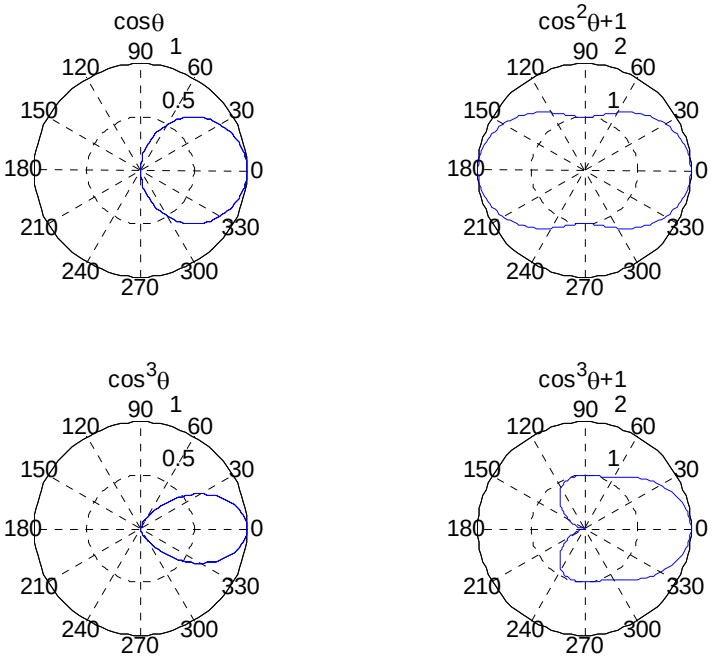
Ορίζουμε πρώτα τα διανύσματα theta, r_1, r_2, r_3 και r_4:

```
>> theta=0:pi/100:2*pi;
>> r1=cos(theta);
>> r2=cos(theta).^2+1;
>> r3=cos(theta).^3;
>> r4=cos(theta).^3+1;
```

Κατασκευάζουμε μετά ένα πολλαπλό 2×2 γράφημα:

```
>> subplot(2,2,1), polar(theta,r1)
>> title('cos(\theta)')
>> subplot(2,2,2), polar(theta,r2)
>> title('cos^2(\theta)+1')
>> subplot(2,2,1), polar(theta,r1)
>> title('cos\theta')
>> subplot(2,2,2), polar(theta,r2)
>> title('cos^2\theta+1')
>> subplot(2,2,3), polar(theta,r3)
>> title('cos^3\theta')
>> subplot(2,2,4), polar(theta,r4)
>> title('cos^3\theta+1')
```

Παίρνουμε το εξής γράφημα:



Παρατηρούμε ότι στις πολικές συντεταγμένες το γράφημα του cosθ είναι ένας κύκλος.

5.6.4 Ραβδοδιαγράμματα και εμβαδογράμματα

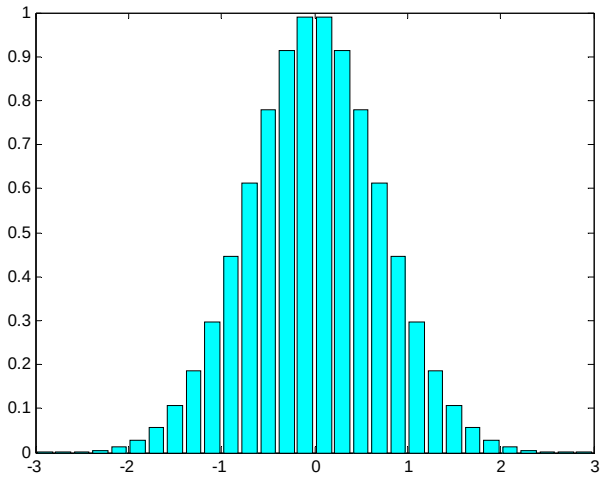
Τα **ραβδοδιαγράμματα** και **εμβαδογράμματα** (bar and area graphs) παρουσιάζουν δεδομένα διανυσμάτων και πινάκων, και χρησιμοποιούνται στη Στατιστική. Παραθέτουμε στον πίνακα πέντε σχετικές συναρτήσεις της MATLAB:

Εντολή	Περιγραφή
bar(x)	Κάθετο διδιάστατο ραβδοδιάγραμμα. Εμφανίζει τις στήλες του $m \times n$ πίνακα x σε m ομάδες από n κατακόρυφες ράβδους.
barh(x)	Οριζόντιο διδιάστατο ραβδοδιάγραμμα. Εμφανίζει τις στήλες του $m \times n$ πίνακα x σε m ομάδες από n οριζόντιες ράβδους.
bar3(x)	Κάθετο τρισδιάστατο ραβδοδιάγραμμα.
bar3h(x)	Οριζόντιο τρισδιάστατο ραβδοδιάγραμμα.
area(x)	Εμβαδόγραμμα του διανύσματος x.

Για παράδειγμα, οι εντολές

```
>> x = -2.9:0.2:2.9;
>> y=exp(-x.^2);
>> bar(x,y)
>> colormap cool
```

μας δίνουν το πιο κάτω ραβδοδιάγραμμα. (Η εντολή `colormap cool` αλλάζει το χρώμα του γραφήματος – δοκιμάστε την `help colormap` για περισσότερες πληροφορίες.)



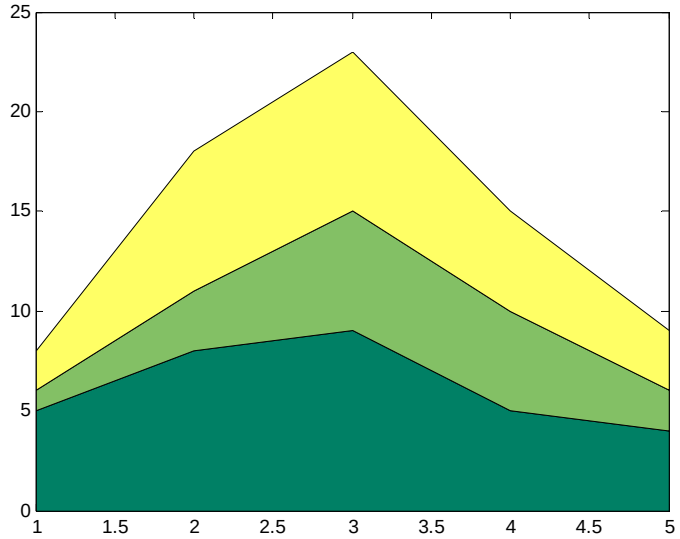
Επίσης, αν ορίσουμε τον πίνακα

```
>> Y = [5 1 2; 8 3 7; 9 6 8; 5 5 5; 4 2 3];
```

τότε

```
>> area(Y)
>> colormap summer
```

μας δίνει το εξής εμβαδόγραμμα:

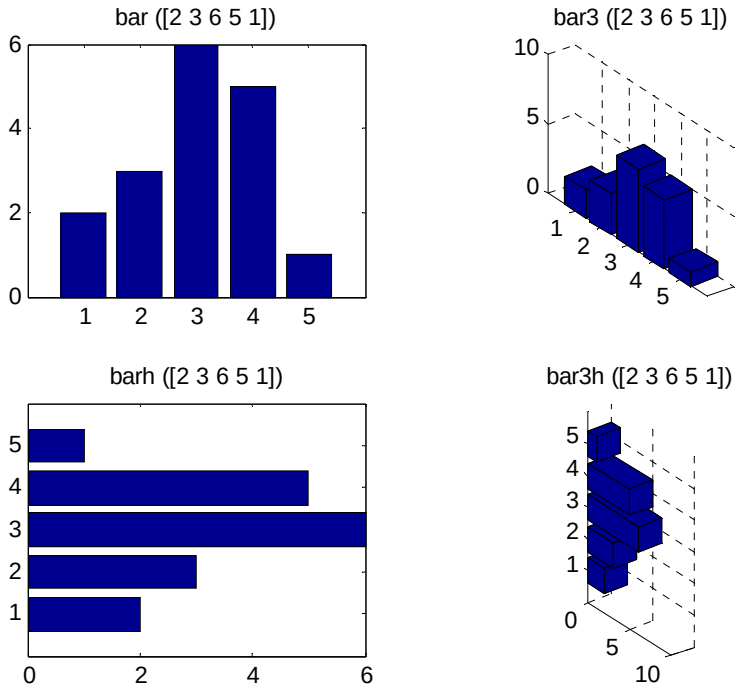


Παράδειγμα 5.6.4

Θεωρούμε το διάνυσμα x = (2, 3, 6, 5, 1). Θα κατασκευάσουμε όλα τα ραβδοδιαγράμματα που μπορούμε να κάνουμε με τη MATLAB:

```
>> x=[2 3 6 5 1];
>> subplot(2,2,1), bar(x), title('bar ([2 3 6 5 1])')
>> subplot(2,2,2), bar3(x), title('bar3 ([2 3 6 5 1])')
>> subplot(2,2,3), barh(x), title('barh ([2 3 6 5 1])')
>> subplot(2,2,4), bar3h(x), title('bar3h ([2 3 6 5 1])')
```

Παίρνουμε το πιο κάτω γράφημα.



Παράδειγμα 5.6.5

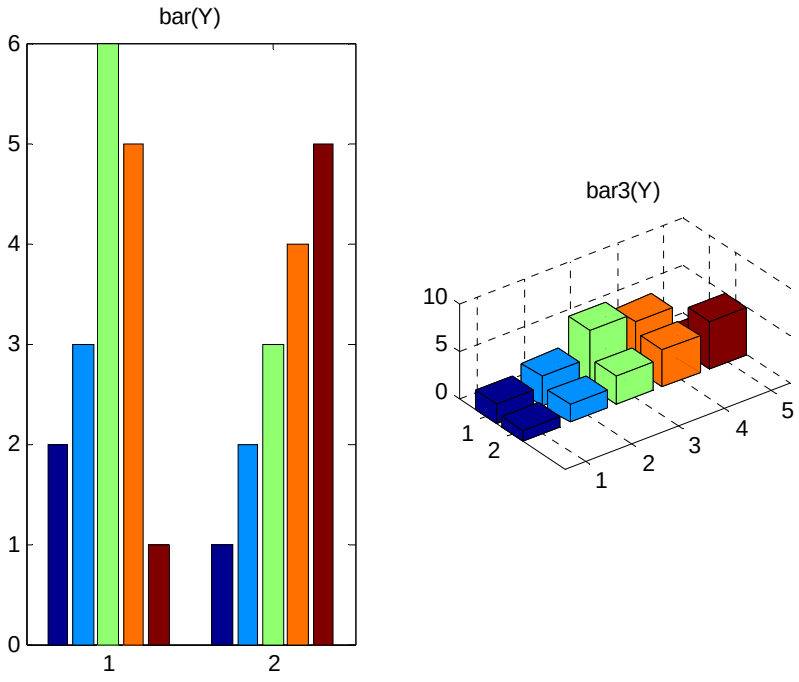
Στο παράδειγμα αυτό μπορούμε να δούμε πως όταν το όρισμα τους είναι πίνακας, οι συναρτήσεις ραβδοδιαγραμμάτων ομαδοποιούν τα δεδομένα κατά γραμμές. Θεωρούμε τον πίνακα Y του οποίου η πρώτη γραμμή είναι το διάνυσμα *x* του προηγούμενου παραδείγματος.

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

Θα κατασκευάσουμε ραβδοδιαγράμματα με τις εντολές bar και bar3:

```
>> subplot(1,2,1), bar(Y), title('bar(Y)')
>> subplot(1,2,2), bar3(Y), title('bar3(Y)')
```

Παίρνουμε έτσι το πιο κάτω γράφημα.

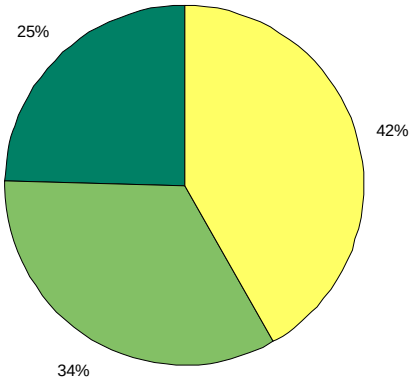


5.6.5 Τομεογράμματα

Οι εντολές **pie** και **pie3** μας δίνουν **τομεογράμματα** (pie charts) στις 2 και 3 διαστάσεις, αντίστοιχα. Για παράδειγμα, οι εντολές

```
>> x=[194.8,266.5,330.9];
>> pie(x)
```

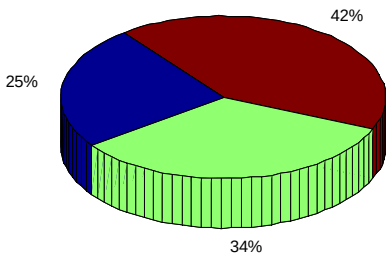
μας δίνουν το εξής τομεόγραμμα:



ενώ με την εντολή

```
>> pie3(x)
```

παίρνουμε το τρισδιάστατο γράφημα:



5.6.6 Ιστογράμματα

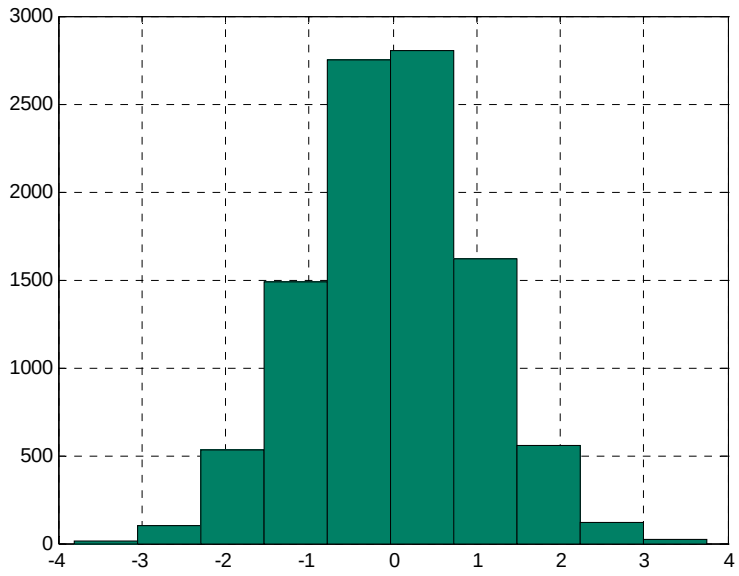
Τα **ιστογράμματα** (histograms) χρησιμοποιούνται εκτεταμένα στην Στατιστική. Στη MATLAB αυτά παράγονται με τη συνάρτηση **hist**. Αν *y* είναι ένα διάνυσμα, π.χ.,

```
>> y=randn(10000,1);
```

τότε,

```
>> hist(y)
>> grid
```

μας δίνει το πιο κάτω ιστόγραμμα (με πλέγμα).



Δοκιμάστε επίσης την εντολή **rose**, που κάνει την ίδια δουλειά σε πολικές συντεταγμένες.

5.7 Γραφήματα στις 3 διαστάσεις

Σ' αυτή την παράγραφο θα ασχοληθούμε με εντολές που σχεδιάζουν τη γραφική παράσταση μιας συνάρτησης δύο μεταβλητών:

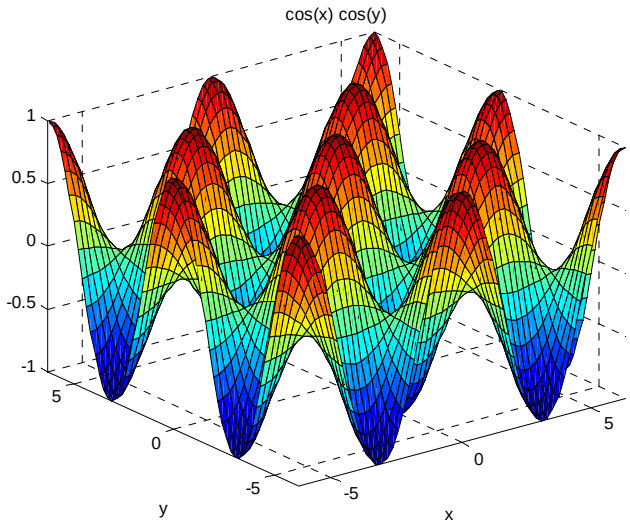
$$z = f(x, y)$$

Είναι γνωστό ότι η γραφική παράσταση της πιο πάνω συνάρτησης είναι μια επιφάνεια στον τρισδιάστατο χώρο.

Αν η $f(x, y)$ έχει οριστεί σαν ανώνυμη συνάρτηση (ή μέσω της εντολής inline), τότε ο πιο εύκολος τρόπος για να πάρουμε την γραφική της παράσταση είναι με την εντολή **ezsurf** (που δουλεύει με ανάλογο τρόπο όπως η ezplot). Για παράδειγμα,

```
>> z = @(x,y) cos(x).*cos(y);
>> ezsurf(z)
```

δίνει το πιο κάτω γράφημα:



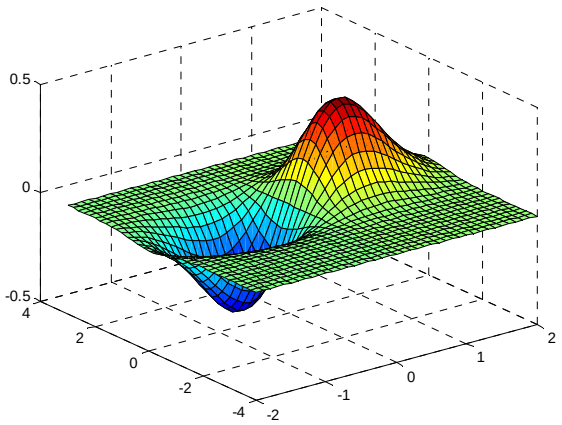
Αν, όμως, η $f(x, y)$ έχει οριστεί από ένα m-file, ή αν θέλουμε να επιλέξουμε τα σημεία στους άξονες των x και y, τότε πρέπει πρώτα να κατασκευάσουμε ένα *πλέγμα* στο επίπεδο xy, με την εντολή **meshgrid**. Για παράδειγμα,

```
>> [x,y] = meshgrid(-2:0.1:2, -4:0.2:3);
```

κατασκευάζει ένα τέτοιο πλέγμα στο ορθογώνιο $[-2, 2]\times[-4, 3]$ με βήμα 0.1 στο άξονα των x, και με βήμα 0.2 στο άξονα των y, και αποθηκεύει τις συντεταγμένες των κόμβων στα διανύσματα x και y. Στη συνέχεια, παίρνουμε τις τιμές του $z = f(x, y)$ τις οποίες μπορούμε να χρησιμοποιήσουμε για να πάρουμε την γραφική παράσταση της συνάρτησης. Για παράδειγμα, οι εντολές

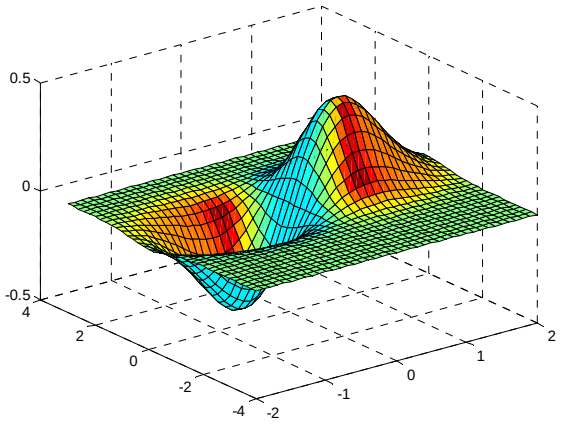
```
>> z = x .* exp(-x.^2 - y.^2);
>> surf(x,y,z)
```

δίνουν τη γραφική παράσταση της συνάρτησης $z = f(x, y) = xe^{-x^2-y^2}$:

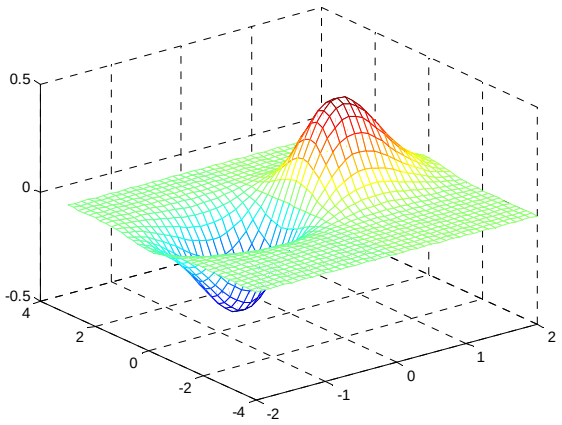


Η εντολή **surf**, που είδαμε στο πιο πάνω παράδειγμα, δίνει την γραφική παράσταση μιας επιφάνειας, ενώ οι εντολές **surfl** και **mesh** κάνουν την ίδια δουλειά, αλλά το γράφημα διαφέρει στο φωτισμό και στη σκίαση, αντίστοιχα, όπως φαίνεται πιο κάτω:

```
>> surfl(x,y,z)
```




```
>> mesh(x,y,z)
```



Οι εντολές **xlabel**, **ylabel**, **zlabel**, **title** και **legend** δουλεύουν όπως και πριν και μας επιτρέπουν να προσθέσουμε κείμενο στους άξονες, τίτλο στο γράφημα, και λεζάντα,

αντίστοιχα. Επίσης, υπάρχει η δυνατότητα να αλλάξουμε τα χρώματα του γραφήματος (π.χ. colormap cool, colormap summer κοκ). Για να πάρετε μια ιδέα αυτών των δυνατοτήτων, γράψετε

```
>> graf3d
```

και ασχοληθείτε με το demo αυτό.

Παράδειγμα 5.7.1

Θα κατασκευάσουμε τις γραφικές παραστάσεις των επιπέδων $z_1 = 0.2x + 3y$ και $z_2 = 0.6x + 0.5y$, για $-4 \le x \le 4, -4 \le y \le 4$, στο ίδιο γράφημα. Πρώτα κατασκευάζουμε το πλέγμα, με βήμα 0.2:

```
>> [x,y] = meshgrid(-4:0.2:4, -4:0.2:4);
```

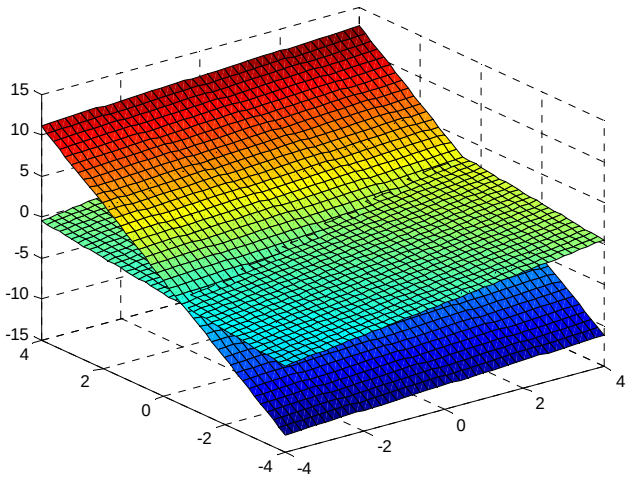
Έπειτα, ορίζουμε τις δύο συναρτήσεις:

```
>> z1 = 0.2*x+3*y;
>> z2 = 0.6*x+0.5*y;
```

Κατασκευάζουμε πρώτα την πρώτη γραφική παράσταση, γράφουμε την εντολή hold on (για να κρατηθεί το παράθυρο) και μετά κατασκευάζουμε τη δεύτερη:

```
>> surf(x,y,z1)
>> hold on
>> surf(x,y,z2)
```

Το αποτέλεσμα είναι το εξής:

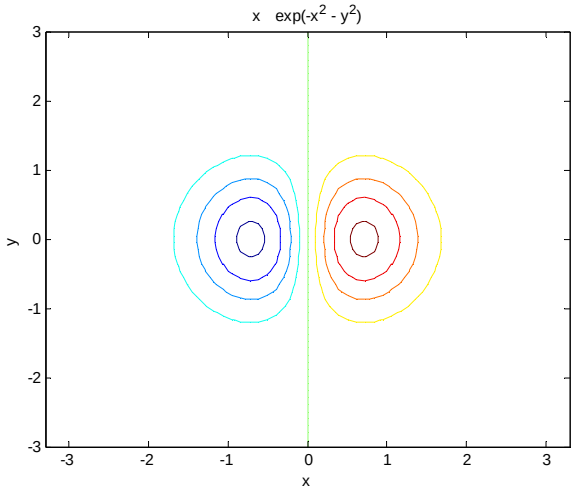


5.7.1 Ισοϋψείς καμπύλες

Η MATLAB έχει την δυνατότητα να κατασκευάζει *διαγράμματα ισοϋψών* (contour plots) μιας συνάρτησης $z(x, y)$ που έχει οριστεί σαν ανώνυμη συνάρτηση (ή μέσω της εντολής inline) με τις εντολές **ezcontour** και **ezcontourf**. Για παράδειγμα,

```
>> z = @(x,y) x .* exp(-x.^2 - y.^2);
>> ezcontour(z)
```

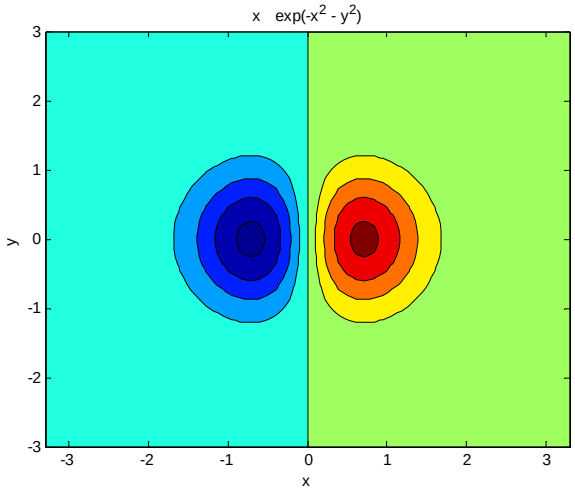
μας δίνει το εξής διάγραμμα ισοϋψών:



και

```
>> ezcontourf(z)
```

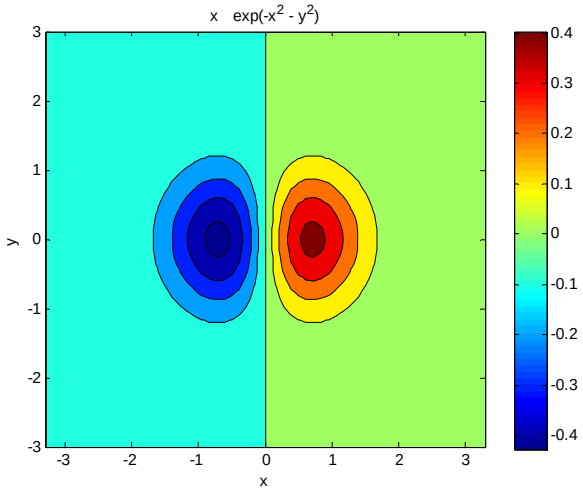
μας δίνει ίδιο διάγραμμα, αλλά με χρώματα:



Η εντολή

```
>> colorbar
```

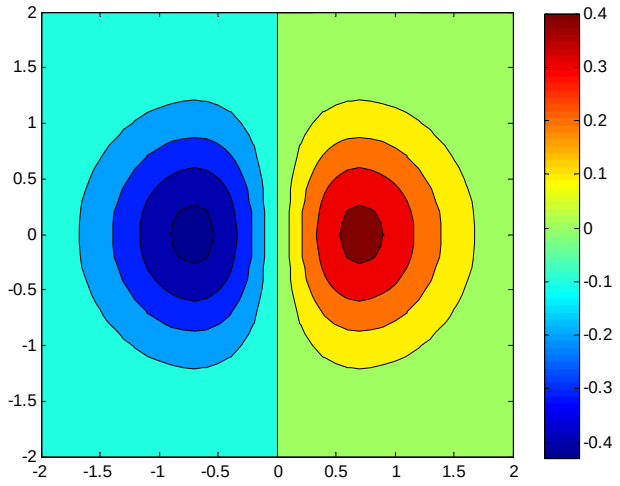
προσθέτει λεζάντα για το κάθε χρώμα, όπως φαίνεται πιο κάτω:



Στην περίπτωση που η συνάρτηση δεν έχει οριστεί ανώνυμα, τότε οι πιο πάνω εντολές δεν φέρουν τα γράμματα **ez** πριν από το όνομά τους. Για παράδειγμα, οι εντολές

```
>> [x,y] = meshgrid(-2:0.1:2, -2:0.1:2);
>> z = x .* exp(-x.^2 - y.^2);
>> contour(x,y,z)
>> contourf(x,y,z)
>> colorbar
```

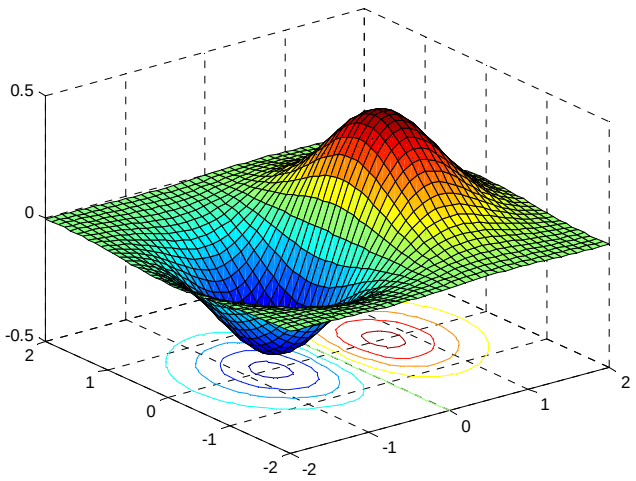
μας δίνουν το πιο κάτω διάγραμμα ισοϋψών (με λεζάντα για τα χρώματα):



Αν θέλουμε, μπορούμε να συνδυάσουμε τις εντολές **surf** και **contour** για να πάρουμε την γραφική παράσταση μιας συνάρτησης δύο μεταβλητών με το διάγραμμα ισοϋψών στο επίπεδο *xy*, χρησιμοποιώντας την εντολή **surfc**. Για παράδειγμα,

```
>> surfc(x,y,z)
```

μας δίνει το πιο κάτω γράφημα:



Παράδειγμα 5.7.2

Θεωρούμε το **υπερβολικό παραβολοειδές**

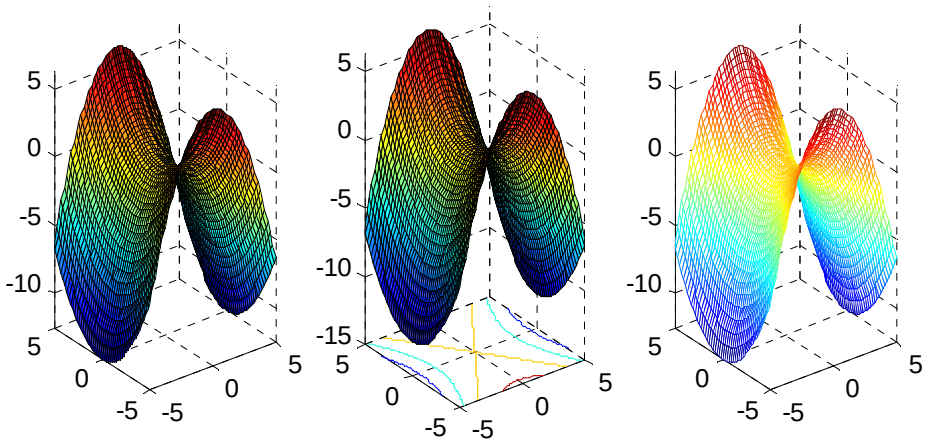
$$z = \frac{y^2}{4} - \frac{x^2}{2}$$

Θα σχεδιάσουμε το γράφημά του με τρεις διαφορετικούς τρόπους, δηλ. με τις εντολές surf, surfc και mesh. Για εξοικονόμηση χώρου, θα τοποθετήσουμε τα τρία γραφήματα οριζόντια στο ίδιο πολλαπλό γράφημα.

Με τις εντολές

```
>> [x,y]=meshgrid(-5:0.2:5, -5:0.2:5);
>> z=y.^2/4-x.^2/2;
>> subplot(1,3,1), surf(x,y,z), axis equal
>> subplot(1,3,2), surfc(x,y,z), axis equal
>> subplot(1,3,3), mesh(x,y,z), axis equal
```

παίρνουμε το πιο κάτω γράφημα:



Παράδειγμα 5.7.3

Η επιφάνεια που ορίζεται από την

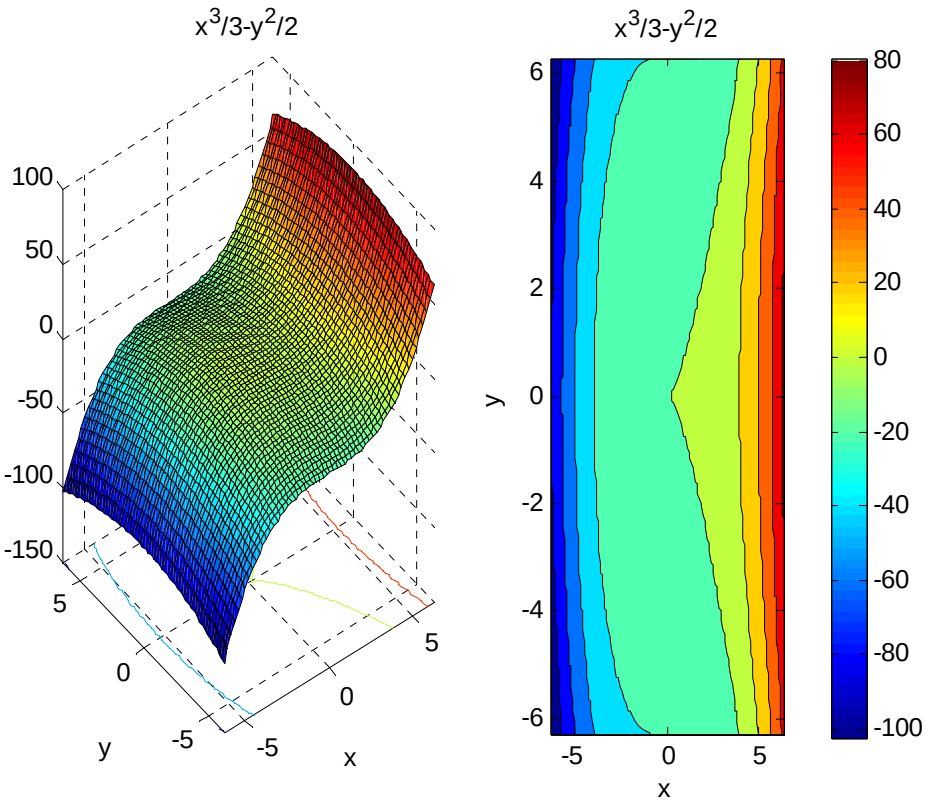
$$z = \frac{x^3}{3} - \frac{y^2}{2}$$

είναι γνωστή ως **επιφάνεια παπουτσιού** (shoe surface). Θα σχεδιάσουμε το γράφημά της καθώς και τις ισοϋψείς της χρησιμοποιώντας τις εντολές ezsurfc και ezcontourf. Για το σκοπό αυτό πρέπει να ορίσουμε την $z = f(x, y)$ σαν ανώνυμη συνάρτηση. Θα τοποθετήσουμε τα δύο γραφήματα οριζόντια στο ίδιο πολλαπλό γράφημα.

Με τις εντολές

```
>> fshoe=@(x,y) x.^3/3-y.^2/2
fshoe =
    @(x,y) x.^3/3-y.^2/2
>> subplot(1,2,1), ezsurfc(fshoe), axis equal
>> subplot(1,2,2), ezcontourf(fshoe)
```

παίρνουμε το πιο κάτω γράφημα:



Παράδειγμα 5.7.4

Θεωρούμε τη συνάρτηση

$$z = f(x,y) = \frac{xy\left(x^2 - y^2\right)}{x^2 + y^2}$$

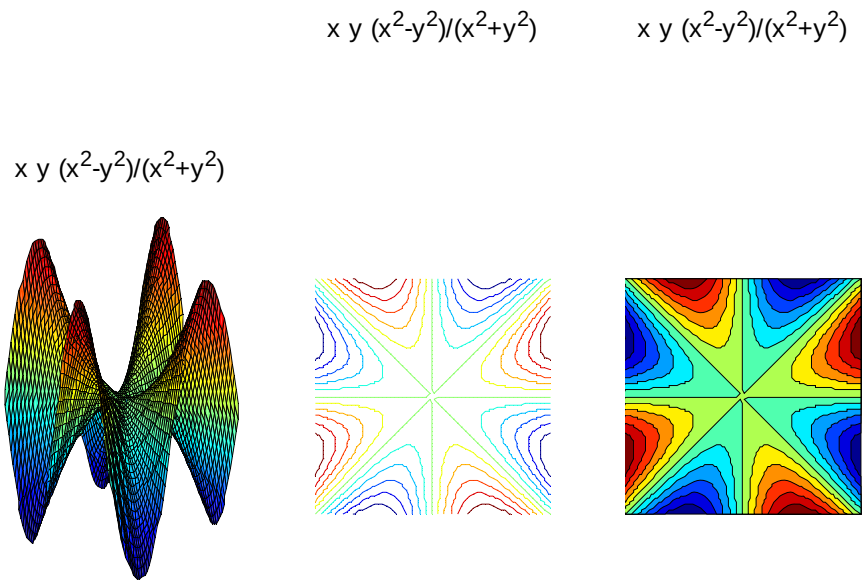
Θα σχεδιάσουμε το γράφημά της καθώς και τις ισοϋψείς της χρησιμοποιώντας τις εντολές ezsurf, ezcontour και ezcontourf. Για το σκοπό αυτό πρέπει να ορίσουμε την $z = f(x, y)$ σαν ανώνυμη συνάρτηση. Θα τοποθετήσουμε τα τρία γραφήματα οριζόντια στο ίδιο πολλαπλό γράφημα.

Με τις εντολές

```
>> fex1=@(x,y) x.*y.*(x.^2-y.^2)./(x.^2+y.^2)
fex1 =
    @(x,y) x.*y.*(x.^2-y.^2)./(x.^2+y.^2)
>> subplot(1,3,1), ezsurf(fex1), axis equal
>> subplot(1,3,2), ezcontour(fex1), axis equal
```

```
>> subplot(1,3,3), ezcontourf(fex1), axis equal
```

παίρνουμε το πιο κάτω γράφημα:



5.8 Τρισδιάστατες καμπύλες

Στη MATLAB μπορούμε να σχεδιάσουμε τρισδιάστατες καμπύλες με την εντολή **plot3** η οποία είναι το τρισδιάστατο ανάλογο της plot. Η γενική μορφή της plot3 είναι

plot3(x, y, z)

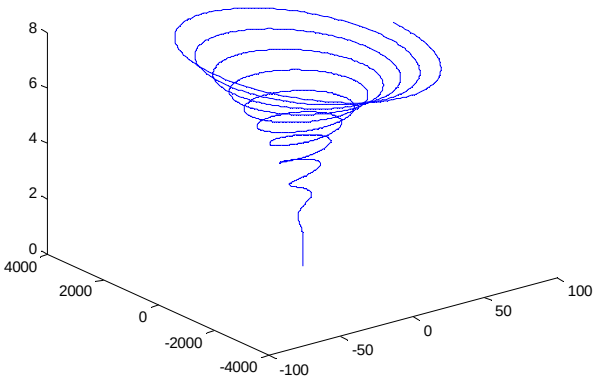
Για να σχεδιάσουμε την τρισδιάστατη καμπύλη

$$x(t) = t\cos t, \quad y(t) = t^2 \sin t, \quad z(t) = \sqrt{t}, \quad t \in [0, 20\pi]$$

χρησιμοποιούμε τις εντολές

```
>> t=0:pi/100:20*pi;
>> x=t.*cos(t);
>> y=t.^2.*sin(t);
>> z=sqrt(t);
>> plot3(x,y,z)
```

οι οποίες παράγουν το πιο κάτω γράφημα



Η εντολή **comet3** είναι το τρισδιάστατο ανάλογο της comet. Δοκιμάστε τις εντολές:

```
>> t=0:pi/100:20*pi;
>> x=t.*cos(t);
>> y=t.^2.*sin(t);
>> z=sqrt(t);
>> comet3(x,y,z)
```

5.9 Ασκήσεις

5.1 Να κάνετε τις γραφικές παραστάσεις των $y=\sin(x), z=\cos(x)$ στους ίδιους άξονες, για $x\in[-2,2]$ με βήμα 0.01. Να βάλετε ετικέτες στους άξονες, τίτλο και λεζάντα.

5.2 Να γράψετε ένα m-file, που να καλείται `ask5_2.m`, το οποίο να παίρνει σαν δεδομένα εισόδου τα άκρα *a*, *b* ενός διαστήματος $[a,b]$ και ένα ακέραιο *n*, και να κατασκευάζει την γραφική παράσταση των συναρτήσεων $y=\sin(x), z=\cos(x)$ στους ίδιους άξονες, για $x\in[a,b]$ χρησιμοποιώντας *n* κομβικά σημεία. Η γραφική παράσταση πρέπει να έχει ετικέτες στους άξονες, τίτλο και λεζάντα. (Σημείωση: Το m-file αυτό δεν θα έχει δεδομένα εξόδου.)

5.3 Να κάνετε τις γραφικές παραστάσεις των $f(x)=x^2, g(x)=x^3$ στους ίδιους άξονες, για $x\in[-1,1]$ χρησιμοποιώντας 101 κομβικά σημεία. Να βάλετε ετικέτες στους άξονες, τίτλο και λεζάντα.

5.4 Να κάνετε την γραφική παράσταση της πιο κάτω συνάρτησης για $x\in[-1,1]$:

$$h(x)=\begin{cases} x^2+1 & ,x<0 \\ x & ,x\geq 0 \end{cases}$$

5.5 Έστω $f(x)=3x^3-26x+10$. Να κάνετε την γραφική παράσταση των f, f', f'' για $x\in[-2,4]$, στους ίδιους άξονες, χρησιμοποιώντας διαφορετικού είδους καμπύλες για την κάθε μια. Να βάλετε ετικέτες στους άξονες, τίτλο και λεζάντα.

5.6 Να κάνετε την γραφική παράσταση της $h(x)=2^{-0.2x+10}$ για $x\in[0.1,60]$ χρησιμοποιώντας γραμμική, λογαριθμική και ημιλογαριθμικές κλίμακες. Να βάλετε ετικέτες στους άξονες, τίτλο και λεζάντα.

5.7 Να κάνετε την γραφική παράσταση της $z(x,y)=\dfrac{xy^2}{x^2+y^2}$ για $x\in[-1,3], y\in[1,4]$. Να βάλετε ετικέτες στους άξονες, τίτλο και λεζάντα.

5.8 Στο τέλος της βοήθειας help ezplot δίνονται αρκετά παραδείγματα χρήσης της εντολής. Δοκιμάστε όλα τα παραδείγματα στη MATLAB και περιγράψτε σε κάθε περίπτωση τη συνάρτηση που σχεδιάζετε.

5.9 Έστω η συνάρτηση

$$y=\tan(\sin x)-\sin(\tan x)$$

(α) Σχεδιάστε το γράφημα της συνάρτησης στο διάστημα [–π, π] με την εντολή plot χρησιμοποιώντας βήμα π/2000.

(β) Σχεδιάστε το γράφημα της συνάρτησης στο διάστημα [–π, π] με την εντολή ezplot χρησιμοποιώντας βήμα π/2000.

(γ) Σχεδιάστε την καμπύλη στο διάστημα [–π, π] με την εντολή comet χρησιμοποιώντας βήμα π/2000.

5.10 Σχεδιάστε την καμπύλη που ορίζεται από την πεπλεγμένη συνάρτηση

$$x^3 + 2x^2 - 3x + 5 - y^2 = 0$$

5.11 Σχεδιάστε την καμπύλη που ορίζεται από την πεπλεγμένη συνάρτηση

$$y^2 - x^2 - 1 = 0$$

με $-3 < x < 2$ και $-2 < y < 3$.

5.12 Σχεδιάστε την παραμετρική καμπύλη

$$x(t) = t\cos t, \quad y(t) = t\sin t, \quad 0 \le t \le 4\pi$$

5.13 Σχεδιάστε την παραμετρική καμπύλη

$$x(t) = \cos t, \quad y(t) = \sin t^2, \quad 0 \le t \le \pi$$

πρώτα με την εντολή ezplot και μετά με τη εντολή comet.

5.14 Σχεδιάστε την παραμετρική καμπύλη

$$x(t) = \cos(2t)\cos^2 t, \quad y(t) = \sin(2t)\sin^2 t, \quad 0 \le t \le 2\pi$$

πρώτα με την εντολή ezplot και μετά με τη εντολή comet.

5.15 Σχεδιάστε την παραμετρική καμπύλη

$$x(t) = t\sin t, \quad y(t) = 2\cos t, \quad 0 \le t \le 10\pi$$

πρώτα με την εντολή ezplot και μετά με τη εντολή comet.

5.16 Σχεδιάστε στο ίδιο παράθυρο τα γραφήματα της $y = e^x$ (αριστερά) και της $y = \sqrt{x}$ (δεξιά).

5.17 Σχεδιάστε στο ίδιο παράθυρο τα γραφήματα των συναρτήσεων Bessel $J_0(x)$ (πάνω) και της $J_1(x)$ (κάτω). Για το σκοπό αυτό χρησιμοποιείστε τη συνάρτηση βιβλιοθήκης **besselj**.

5.18 Σχεδιάστε στο ίδιο παράθυρο τα γραφήματα των συναρτήσεων $\sin(x)$, $\cos(x)$, $\tan(x)$ και $\cot(x)$ ως εξής:

sinx	**cosx**
tanx	**cotx**

5.19 Κάνετε το ημιλογαριθμικό γράφημα των πρώτων 18 όρων της ακολουθίας Fibonacci με την εντολή

```
semilogy(fibonacci(18),'-o')
```

Το m-file Fibonacci.m φαίνεται πιο κάτω:

```
function f = fibonacci(n)
%FIBONACCI  Fibonacci sequence
%   f = FIBONACCI(n) generates the first n Fibonacci numbers.
f = zeros(n,1);
```

```
f(1) = 1;
f(2) = 2;
for k = 3:n
   f(k) = f(k-1) + f(k-2);
end
```

Το γράφημα προσεγγίζει μια ευθεία. Ποια είναι η κλίση της ευθείας αυτής;

5.20 Σχεδιάστε τα γραφήματα της $f(x) = 5x^4$ σε πολλαπλό γράφημα όπως φαίνεται στο σχήμα:

Γραμμικό γράφημα **plot**	**Λογαριθμικό γράφημα** **loglog**
Ημιλογαριθμικό γράφημα **semilogx**	**Ημιλογαριθμικό γράφημα** **semilogy**

5.21 Σχεδιάστε σε πολικές συντεταγμένες τα γραφήματα των $r = r(\theta)$ με $\theta \in [0, 2\pi]$:

(α) $r = 5\cos(4\theta)$
(β) $r = 3\cos(6\theta)$
(γ) $r = 3 - 3\sin(\theta)$
(δ) $r = \sqrt{\cos(2\theta)}$

5.22 Σχεδιάστε τα γραφήματα των πιο κάτω συναρτήσεων χρησιμοποιώντας την εντολή **ezsurf**:

(α) $z = x^2 - y^2$

(β) $z = \dfrac{\sin\left(2x^2 + 3y^2\right)}{x^2 + y^2}$

(γ) $z = \left(x^2 + 3y^2\right)e^{1-x^2-y^2}$

(δ) $z = 20\left(x^2 + y^2\right)$

(ε) $z = 2\left(x^2 + 4y^2\right)$

(στ) $z = \dfrac{1}{2}\ln\left(x^2 + y^2\right)$

5.23 (α) Σχεδιάστε το γράφημα της συνάρτησης $z = \cos(xy)$ χρησιμοποιώντας την εντολή **ezsurf.** Τι παρατηρείτε;

(β) Κατασκευάστε με την εντολή **meshgrid** ένα πλέγμα του τετραγώνου [–4, 4]×[–4, 4] και σχεδιάστε το γράφημα της συνάρτησης με την εντολή **surf** καθώς και τις ισοϋψείς της με τις εντολές **contour** και **contourf**.

(γ) Βάλτε μαζί και κατακόρυφα σ'ένα πολλαπλό γράφημα τα τρία γραφήματα που ζητούνται στο (β).

5.24 (α) Με τη συνάρτηση ezsurf είναι εύκολο να σχεδιάσουμε το γράφημα της

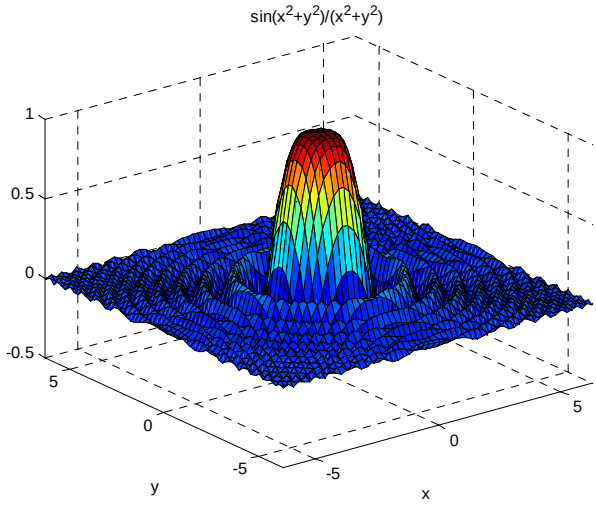
συνάρτησης

$$z = \frac{\sin\left(x^2 + y^2\right)}{x^2 + y^2}$$

Επαληθεύσατε ότι με τις εντολές

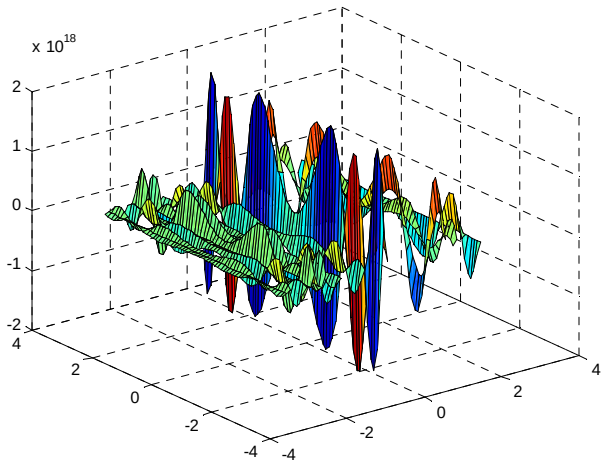
```
>> fsinx=@(x,y) sin(x.^2+y.^2)/(x.^2+y.^2)
fsinx =
    @(x,y) sin(x.^2+y.^2)/(x.^2+y.^2)
>> ezsurf(fsinx)
```

παίρνουμε το πιο κάτω γράφημα.



(β) Σχεδιάστε τώρα το γράφημα της συνάρτησης με τις εντολές:

```
>> [x,y]=meshgrid(-3:0.1:3, -3:0.1:3);
>> z=fsinx(x,y);
Warning: Matrix is singular to working precision.
> In @(x,y) sin(x.^2+y.^2)/(x.^2+y.^2)
>> surf(x,y,z)
```



Εξηγήστε γιατί τα δύο γραφήματα διαφέρουν τόσο. Πως μπορούμε να

βελτιώσουμε το γράφημα που παίρνουμε με την εντολή surf;

5.25 Η επιφάνεια που ορίζεται από την $z = x^4 + x^2 y - y^2$ είναι γνωστή σαν **επιφάνεια του Menn**. Σχεδιάστε το γράφημά της με τις εντολές surf, surfc και mesh. Για εξοικονόμηση χώρου, τοποθετείστε τα τρία γραφήματα οριζόντια στο ίδιο πολλαπλό γράφημα.

5.26 Η επιφάνεια που ορίζεται από την $z = x(x^2 - 3y^2)$ είναι γνωστή σαν **επιφάνεια** ή **σαμάρι του πιθήκου** (monkey saddle). Σχεδιάστε το γράφημά της με τις εντολές ezsurf, ezsurfc και contour. Για λόγους σύγκρισης τοποθετείστε τα τρία γραφήματα οριζόντια στο ίδιο πολλαπλό γράφημα.

5.27 Η **επιφάνεια του Peano** ορίζεται από την

$$z = \frac{2x^2 - y}{y - x^2}$$

(α) Σχεδιάστε το γράφημά της με τις εντολές surf, surfl, surfc και mesh.
(β) Σχεδιάστε το γράφημά της με τις εντολές ezsurf και ezsurfc.
(γ) Σχεδιάστε τις ισοϋψείς της με τις εντολές contour και contourf.
(δ) Σχεδιάστε τις ισοϋψείς της με τις εντολές ezcontour και ezcontourf.

5.28 Σχεδιάστε την τρισδιάστατη καμπύλη

$$x(t) = \cos t, \quad y(t) = \sin t, \quad z(t) = \sqrt{t}, \quad t \in [0, 8\pi]$$

με τις εντολές plot3 και comet3.

5.29 Σχεδιάστε την τρισδιάστατη καμπύλη

$$x(t) = t, \quad y(t) = \sin t, \quad z(t) = e^{\sqrt{t}/(1+t)}, \quad t \in [0, 12\pi]$$

με τις εντολές plot3 και comet3.

5.30 (α) Η **συνάρτηση σφάλματος** (error function) που ορίζεται από την

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

υπολογίζεται στη MATLAB με τη συνάρτηση βιβλιοθήκης **erf**. Σχεδιάστε το γράφημα της συνάρτησης για $0 \le x \le 1$.
(β) Η συνάρτηση

$$u(y,t) = 1 - erf\left(\frac{y}{2\sqrt{t}}\right)$$

είναι λύση της μερικής διαφορικής εξίσωσης

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial t^2}$$

με συνοριακές συνθήκες τις

$$u(0,t) = 1, \quad t > 0$$
$$u(\infty,t) = 0, \quad t \ge 0$$

και αρχική συνθήκη την

$$u(y,0) = 0, \quad 0 \le y < \infty$$

Γράψτε ένα function m-file με όνομα erfile.m που να υπολογίζει τη συνάρτηση $u(y, t)$ για οποιοδήποτε διάνυσμα y και ορισμένη τιμή του t (δηλ. με δεδομένα εισόδου τα y και t).
(γ) Σχεδιάστε στο ίδιο γράφημα τις καμπύλες $u(y, t_i)$ με $y = 0{:}0.01{:}1$ και $t_1 = 0$, $t_2 = 0.0001$, $t_3 = 0.001$, $t_4 = 0.01$, $t_5 = 0.1$ και $t_6 = 1$. Στο γράφημα τοποθετείστε ετικέτες για τους άξονες, τίτλο και λεζάντα.
Παραδοτέα: τα γραφήματα στα (α) και (γ) και το m-file στο (β).

5.31 Η συνάρτηση

$$u(y,t)=1-y-\frac{2}{\pi}\sum_{k=1}^{\infty}\frac{1}{k}\sin(k\pi y)\,e^{-k^2\pi^2 t}$$

είναι λύση της μερικής διαφορικής εξίσωσης

$$\frac{\partial u}{\partial t}=\frac{\partial^2 u}{\partial t^2}$$

με συνοριακές συνθήκες τις

$$u(0,t)=1,\quad t>0$$
$$u(1,t)=0,\quad t\ge 0$$

και αρχική συνθήκη την

$$u(y,0)=0,\quad 0\le y<1$$

(α) Γράψτε ένα function m-file με όνομα plcouef.m που υπολογίζει την προσεγγιστική συνάρτηση

$$\overline{u}(y,t)=1-y-\frac{2}{\pi}\sum_{k=1}^{N}\frac{1}{k}\sin(k\pi y)\,e^{-k^2\pi^2 t}$$

για οποιοδήποτε διάνυσμα y και ορισμένη τιμή του t (δηλ. με δεδομένα εισόδου τα y, t και N).
(β) Σχεδιάστε στο ίδιο γράφημα τις καμπύλες $\overline{u}(y,t_i)$ με $N = 100$ και $y = 0{:}0.01{:}1$ και $t_1 = 0$, $t_2 = 0.0001$, $t_3 = 0.001$, $t_4 = 0.01$, $t_5 = 0.1$ και $t_6 = 1$. Στο γράφημα τοποθετείστε ετικέτες για τους άξονες, τίτλο και λεζάντα.
(γ) Σχεδιάστε στο ίδιο γράφημα τις καμπύλες $\overline{u}(y,t_i)$ με $N = 1000$ και $y = 0{:}0.01{:}1$ και $t_1 = 0$, $t_2 = 0.0001$, $t_3 = 0.001$, $t_4 = 0.01$, $t_5 = 0.1$, $t_6 = 1$. Στο γράφημα τοποθετείστε ετικέτες για τους άξονες, τίτλο και λεζάντα.
(δ) Σχεδιάστε στο ίδιο γράφημα τις καμπύλες του (β) με διακεκομμένες και τις καμπύλες του (γ) με συνεχείς γραμμές.
Παραδοτέα: το m-file στο (α) και τα γραφήματα στα (β)–(δ).

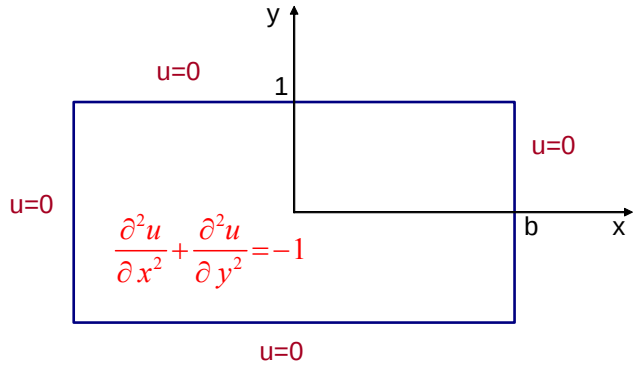
5.32 Η συνάρτηση

$$u(y,z)=\frac{1}{2}\left[1-z^2+4\sum_{k=1}^{\infty}\frac{(-1)^k\cosh(\alpha_k y)}{\alpha_k^3\cosh(\alpha_k b)}\cos(a_k z)\right]$$

όπου

$$\alpha_k=(2k-1)\frac{\pi}{2},\qquad k=1,2,\cdots$$

είναι η λύση του προβλήματος συνοριακών τιμών που φαίνεται στο πιο κάτω σχήμα:



(α) Γράψτε ένα function m-file με όνομα poisrect.m που υπολογίζει την προσεγγιστική συνάρτηση

$$\bar{u}(y,z)=\frac{1}{2}\left[1-z^2+4\sum_{k=1}^{N}\frac{(-1)^k\cosh(\alpha_k y)}{\alpha_k^3\cosh(\alpha_k b)}\cos(a_k z)\right]$$

για οποιαδήποτε διανύσματα y και z (δηλ. με δεδομένα εισόδου τα y, z, b και N).

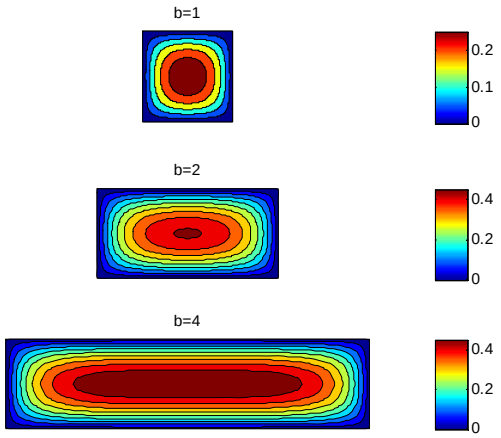
(β) Για b =1, 2 και 4 κατασκευάστε τα πιο κάτω πλέγματα:

```
>>[y1,z1]=meshgrid(-1:0.05:1,-1:0.05:1);
>>[y2,z2]=meshgrid(-2:0.05:2,-1:0.05:1);
>>[y3,z3]=meshgrid(-4:0.05:4,-1:0.05:1);
```

και υπολογίστε τις αντίστοιχες λύσεις για N = 20:

```
>> u1= poisrect(1,y1,z1,20);
>> u2= poisrect(2,y2,z2,20);
>> u3= poisrect(4,y3,z3,20);
```

(γ) Κατασκευάστε το πιο κάτω πολλαπλό γράφημα με τις ισοϋψείς της $u(y, z)$ για b =1, 2 και 4:



(δ) Κατασκευάστε τα τρισδιάστατα γραφήματα της $u(y, z)$ για b = 1, 2 και 4 με την εντολή surfc.

Παραδοτέα: το m-file στο (α) και τα γραφήματα στα (γ) και (δ).

5.33 Σύμφωνα με το θεώρημα Taylor ισχύει

$$e^x = p_n(x) + R_n(x)$$

όπου $p_n(x)$ το πολυώνυμο Taylor

$$p_n(x) = 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \cdots + \frac{1}{n!}x^n$$

και $R_n(x)$ το υπόλοιπο (remainder)

$$R_n(x) = \frac{1}{(n+1)!}x^{n+1}e^{\xi_x}$$

όπου ξ_x αριθμός μεταξύ των x και 0. Αν προσεγγίσουμε στην περιοχή του 0 τη συνάρτηση e^x με το πολυώνυμο $p_n(x)$, το **απόλυτο σφάλμα** (absolute error) της προσέγγισης δίνεται από την

$$e_n(x) = \left| e^x - p_n(x) \right| = \left| R_n(x) \right|$$

(α) Δημιουργήστε function m-files τα οποία να υπολογίζουν τα πολυώνυμα $p_2(x)$, $p_3(x)$ και $p_4(x)$. Στην αρχή τους να γράφετε σαν σχόλιο το ονοματεπώνυμό σας και τον ΑΜ.

(β) Κατασκευάστε στην ίδια γραφική παράσταση τα γραφήματα των e^x, $p_2(x)$, $p_3(x)$ και $p_4(x)$ με συνεχή, αδρή διακεκομμένη, διακεκομμένη-τελείες και λεπτή διακεκομμένη γραμμή αντίστοιχα στο διάστημα [–1.5, 1.5]. Το γράφημα πρέπει να έχει λεζάντα και ετικέτες για τους άξονες. Σαν τίτλο βάλτε τον ΑΜ.

(γ) Κατασκευάστε σε άλλη γραφική παράσταση τα γραφήματα των σφαλμάτων με τους πιο πάνω τύπους γραμμής. Το γράφημα πρέπει να έχει λεζάντα και ετικέτες για τους άξονες. Σαν τίτλο βάλτε τον ΑΜ.

Παραδοτέα: Τα τρία function m-files στο (α) και τα γραφήματα στα (β) και (γ).

6 ΠΟΛΥΩΝΥΜΑ

6.1 Γενικά περί πολυωνύμων

Στη MATLAB τα πολυώνυμα αναπαριστώνται από διανύσματα που περιέχουν τους συντελεστές τους σε *κατιούσα διάταξη.*

Για παράδειγμα το πολυώνυμο

$$p(x) = x^2 - 3x + 5$$

αναπαριστάται από το διάνυσμα

$$\text{p} = [1, -3, 5\ ]$$

ενώ το διάνυσμα

$$\text{q} = [\ 1, 0, 7, -1, 0\ ]$$

παριστάνει το πολυώνυμο

$$q(x) = x^4 + 7x^2 - x\,.$$

Συναντήσαμε ήδη την εντολή **length** η οποία μας δίνει το μήκος ενός διανύσματος. Έτσι αν το διάνυσμα s παριστάνει το πολυώνυμο $s(x)$, ο βαθμός του $s(x)$ είναι

length(s) – 1

Παράδειγμα 6.1.1

Είναι εύκολο να βρούμε το βαθμωτό πολλαπλάσιο ενός πολυωνύμου αφού αυτό αναπαριστάται από το βαθμωτό πολλαπλάσιο του αντίστοιχου διανύσματος.

```
>> p=[ 1, -3, 5];
>> q=[ 1, 0, 7, -1, 0];
>> 3*p

ans =
     3    -9    15

>> -4*q

ans =
    -4     0   -28     4     0
```

Είναι επίσης απλό να προσθέσουμε πολυώνυμα του ίδιου βαθμού:

```
>> s=[2, 3, -4];
>> s+p

ans =
     3     0     1
```

Παράδειγμα 6.1.2

Για την πρόσθεση πολυωνύμων διαφορετικού βαθμού πρέπει να γράψουμε το διάνυσμα που αντιστοιχεί στο πολυώνυμο με το μικρότερο βαθμό σαν διάνυσμα ίσου μήκους με το διάνυσμα που αντιστοιχεί στο πολυώνυμο με τον μεγαλύτερο βαθμό. Για παράδειγμα, για να προσθέσουμε τα πολυώνυμα που αναπαριστώνται από τα διανύσματα

$$p = [\ 1, 2, 3, 4, 5] \quad \text{και} \quad q = [\ -2, 0, 1]$$

γράφουμε το δεύτερο σαν

$$qnew = [\ 0, 0, -2, 0, 1]$$

και υπολογίζουμε το άθροισμα p+qnew.

Το ακόλουθο function m-file με το όνομα **polyadd.m** κάνει ακριβώς αυτή τη δουλειά για μας.

```
function pplusq=polyadd(p,q)
%
% pplusq=polyadd(p,q)
%
% Briskei to a0roisma twn dianusmatwn p kai q
% pou avtiproswpeuouv polywvuma.
%
degp1 = length(p);
degq1 = length(q);
if degp1==degq1
    pplusq =p + q;
elseif degp1>degq1
    pplusq = p + [zeros(1, degp1-degq1) q];
else
    pplusq = q + [zeros(1, degq1-degp1) p];
end
% End of POLYADD
```

Ακολουθούν αποτελέσματα που πήραμε με το polyadd.m:

```
>> p = [ 1, 2, 3, 4, 5];
>> q = [-2, 0, 1];

>> polyadd(p,q)

ans =
     1     2     1     4     6

>> p=[1, -3, 5];
>> q=[1, 0, 7, -1, 0];

>> polyadd(p,q)

ans =
     1     0     8    -4     5
```

6.2 Χρήσιμες συναρτήσεις για πολυώνυμα

Οι σημαντικότερες εντολές για πολυώνυμα συνοψίζονται στον πιο κάτω πίνακα:

polyval	**Βρίσκει την τιμή ενός πολυωνύμου *p*(*x*) σε ένα σημείο *x*.** **Δομή:** **polyval(p, x)** **p: το διάνυσμα που αντιστοιχεί στο πολυώνυμο *p*(*x*)** **x: το σημείο όπου υπολογίζεται το πολυώνυμο**
roots	**Βρίσκει τις ρίζες ενός πολυωνύμου *p*(*x*).** **Δομή:** **roots(p)** **p: το διάνυσμα που αντιστοιχεί στο πολυώνυμο *p*(*x*)**
conv	**Βρίσκει τη συνέλιξη (γινόμενο) δύο πολυωνύμων *p*(*x*) και *q*(*x*).** **Δομή:** **conv(p, q)** **p: το διάνυσμα που αντιστοιχεί στο πολυώνυμο *p*(*x*)** **q: το διάνυσμα που αντιστοιχεί στο πολυώνυμο *q*(*x*)**
deconv	**Βρίσκει τη αποσυνέλιξη (διαίρεση) δύο πολυωνύμων p(x) και *q*(*x*).** **Δομή:** **[s, r] = deconv(p, q)** **p: το διάνυσμα που αντιστοιχεί στο πολυώνυμο *p*(*x*)** **q: το διάνυσμα που αντιστοιχεί στο πολυώνυμο q(x)** **s: το πηλίκο της διαίρεσης** **r: το υπόλοιπο της διαίρεσης**
polyder	**Βρίσκει την παράγωγο ενός πολυωνύμου p(x).** **Δομή:** **polyder(p)** **p: το διάνυσμα που αντιστοιχεί στο πολυώνυμο *p*(*x*)**

Παράδειγμα 6.2.1
Θα βρούμε τις τιμές του πολυωνύμου

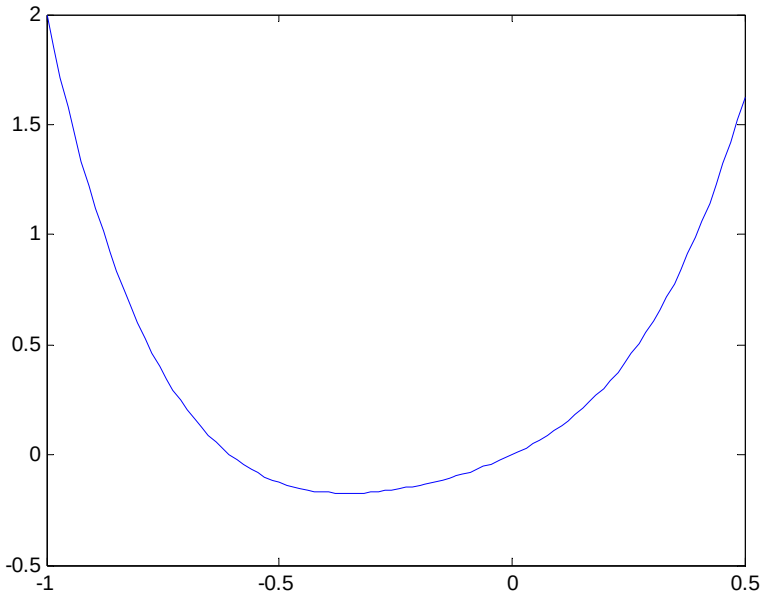
$$p(x) = 4x^4 + 3x^3 + 2x^2 + x$$

σε διάφορα σημεία:

```
>> p=[4, 3, 2, 1, 0];
>> polyval(p,0)

ans =
     0

>> polyval(p,1)

ans =
    10

>> polyval(p,-1)

ans =
     2
```

Άρα, για να πάρουμε τη γραφική παράσταση του $p(x)$ στο διάστημα [–1, ½] γράφουμε

```
>> x=linspace(-1,0.5);
>> plot(x,polyval(p,x))
```



Μπορούμε επίσης να βρούμε τις ρίζες του πολυωνύμου με την εντολή **roots**:

```
>> roots(p)

ans =
         0
   -0.6058
   -0.0721 + 0.6383i
   -0.0721 - 0.6383i
```

Παρατηρούμε ότι η MATLAB βρίσκει τόσο τις πραγματικές όσο και τις μιγαδικές ρίζες. Ας δοκιμάσουμε τώρα το εξής:

```
>> q=[1, 0, 0, 0, -1];
>> roots(q)

ans =
  -1.0000
   0.0000 + 1.0000i
   0.0000 - 1.0000i
   1.0000
```

(Βρήκαμε τις ρίζες του $x^4 - 1=0$.)

Παράδειγμα 6.2.2

Θα πολλαπλασιάσουμε τα πολυώνυμα $p(x) = x + 2$ και $q(x) = x^2 + 4x + 8$:

```
>> p=[1, 2];
>> q=[1, 4, 8];
>> z=conv(p,q)

z =
     1     6    16    16
```

Βρήκαμε, δηλαδή, ότι

$$z(x) = p(x)\, q(x) = x^3 + 6x^2 + 16x + 16$$

Θα διαιρέσουμε τώρα το $z(x)$ με το $q(x)$

$$\frac{z(x)}{q(x)}$$

```
>> [s, r] = deconv(z,q)

s =
     1     2

r =
     0     0     0     0
```

(Η διαίρεση είναι φυσικά ακριβής.)

Μπορούμε φυσικά να δοκιμάσουμε πιο εξεζητημένα παραδείγματα:

```
>> p=[1, -4, 0, 0, -2, 3];
>> q=[3, 2, 1];
>> s=conv(p,q)
s =
     3   -10    -7    -4    -6     5     4     3

>> [z,r]=deconv(p,q)

z =
    0.3333   -1.5556    0.9259   -0.0988

r =
  Columns 1 through 5

         0   -0.0000    0.0000    0.0000   -2.7284
  Column 6

    3.0988
```

Παράδειγμα 6.2.3

Η παράγωγος του πολυωνύμου

$$p(x) = 3x^3 - 4x^2 - x + 2$$

υπολογίζεται εύκολα ως εξής:

```
>> p=[3, -4, -1, 2];
>> polyder(p)

ans =
     9    -8    -1
```

Μπορούμε εύκολα να βρούμε τα **στάσιμα σημεία** του $p(x)$:

```
>> roots(polyder(p))

ans =
    1.0000
   -0.1111
```

Παράδειγμα 6.2.4

Η εντολή **poly** βρίσκει το **χαρακτηριστικό πολυώνυμο** ενός τετραγωνικού πίνακα. Για παράδειγμα:

```
>> A=[1, 2, 3; 0, 2, 4; 0 0 5]

A =
     1     2     3
     0     2     4
     0     0     5

>> p=poly(A)

p =
     1    -8    17   -10
```

Μπορούμε να βρούμε τις ρίζες του χαρακτηριστικού πολυωνύμου (δηλ. τις **ιδιοτιμές** του A) με την

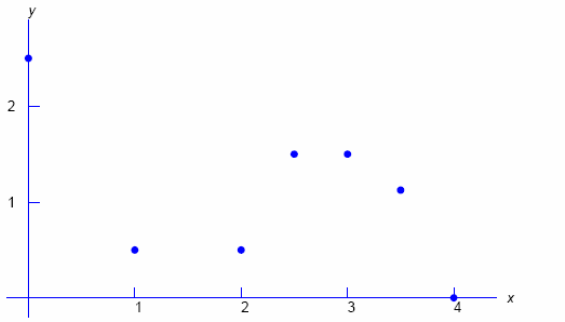
```
>> roots(p)

ans =
    5.0000
    2.0000
    1.0000
```

ή πιο απλά με την εντολή **eig**:

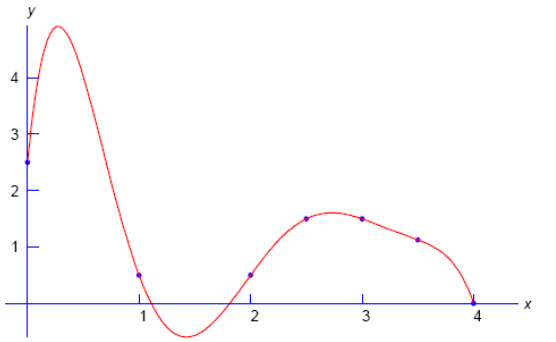
```
>> eig(A)

ans =
     1
     2
     5
```

6.3 Προσαρμογή δεδομένων και η εντολή polyfit

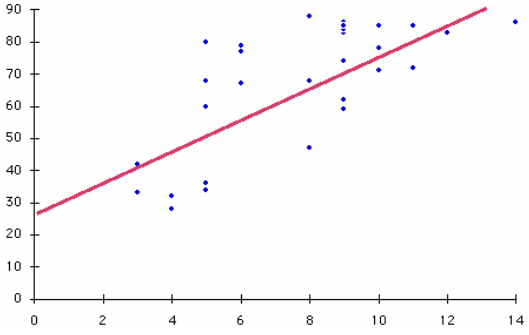
Μια από τις πιο κύριες εφαρμογές των πολυωνύμων είναι η **προσαρμογή δεδομένων** (curve fitting), που είναι επίσης γνωστή ως **παλινδρόμηση** (regression). Σε αυτή τη διαδικασία, έχουμε σαν δεδομένα ένα πεπερασμένο αριθμό σημείων (σαν αυτά που φαίνονται στο πιο κάτω γράφημα), και θέλουμε να βρούμε μια συνάρτηση, συνήθως πολυώνυμο μικρού βαθμού, που να αντιπροσωπεύει τα δεδομένα, δηλ. να προσαρμόσουμε το πολυώνυμο στα δεδομένα.

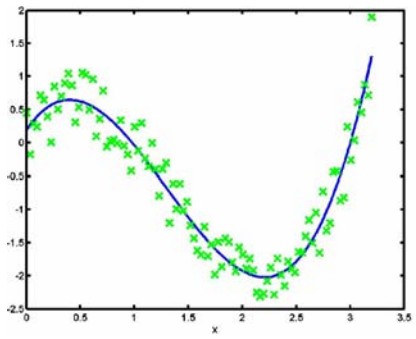


Αν απαιτήσουμε το πολυώνυμο να περνά από όλα τα σημεία (όπως στο πιο κάτω γράφημα), τότε παίρνουμε το λεγόμενο ***πολυώνυμο παρεμβολής*** (interpolant), του οποίου ο βαθμός είναι ίσος με τον αριθμό των σημείων πλην ένα.



Αν δεν απαιτήσουμε το πολυώνυμο να περνά από όλα τα σημεία, αλλά να έχει βαθμό μικρότερο από τον αριθμό των σημείων πλην ένα, τότε αυτό μπορεί να γίνει με διάφορες μεθόδους, η πιο συνηθισμένη των οποίων είναι η ***μέθοδος των ελάχιστων τετραγώνων*** (least squares method). Στα πιο κάτω γραφήματα φαίνονται δύο τέτοιες περιπτώσεις.





Στη MATLAB, τα πιο πάνω μπορούν να επιτευχθούν με την εντολή **polyfit**, η οποία δουλεύει ως εξής: Έστω ότι τα $(x_i, y_i), i = 1,..., N+1$ είναι δεδομένα και έχουν αποθηκευτεί σε δύο διανύσματα x και y, αντίστοιχα. Τότε, η εντολή

p = polyfit(x, y, M)

μας δίνει τους συντελεστές του πολυωνύμου p που έχει βαθμό M και που αντιπροσωπεύει τα δεδομένα. Σημειώνουμε ότι αν $M = N$, τότε παίρνουμε τους συντελεστές του πολυώνυμου παρεμβολής που περνά από όλα τα σημεία (x_i, y_i), ενώ αν $M < N$ τότε παίρνουμε τους συντελεστές ενός πολυωνύμου που προσαρμόζει τα δεδομένα με την μέθοδο των ελάχιστων τετραγώνων. Αν $M > N$, τότε το πολυώνυμο δεν είναι μοναδικό. Όμως αυτή η περίπτωση δεν παρουσιάζει ενδιαφέρον.

Παράδειγμα 6.3.1

Έστω ότι έχουμε τα πιο κάτω δεδομένα:

```
>> x = [1 2 3 4 7];
>> y = [4 6 9 11 20];
```

Τότε,

```
>> p4 = polyfit(x,y,4)

p4 =
    0.0806   -1.1389    5.3194   -7.1944    6.9333
```

το οποίο σημαίνει ότι το πολυώνυμο

$$p_4(x) = 0.0806x^4 - 1.1389x^3 + 5.3194x^2 - 7.1944x + 6.9333$$

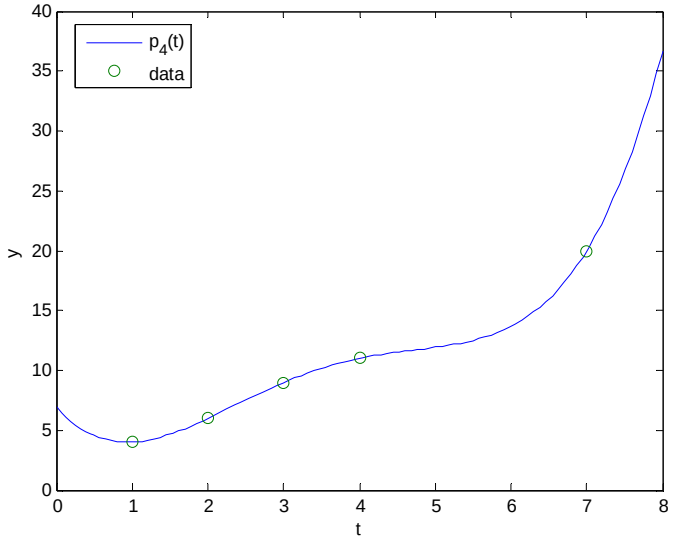
περνά από όλα τα δοθέντα σημεία, μια και έχουμε 5 σημεία και το πολυώνυμο που κατασκευάσαμε έχει βαθμό 4.

Για να βρούμε τη τιμή του $p_4(x)$ σε κάποιο σημείο, π.χ. $x = 1.574$, χρησιμοποιούμε την εντολή `polyval`, που έχουμε ήδη δει:

```
>> polyval(p4,1.574)

ans =
    4.8414
```

Για να κάνουμε τη γραφική παράσταση του $p_4(t)$, $t \in [0, 8]$, και να δείξουμε τα δεδομένα σημεία στους ίδιους άξονες, προχωρούμε ως εξής:

```
>> t = linspace(1,7);
>> P4 = polyval(p4,t);
>> plot(t,P4,x,y,'o')
>> xlabel('x')
>> ylabel('y')
>> legend('p_4(x)','data')
```



Σημειώνουμε ότι χρησιμοποιήσαμε p4, με p μικρό, για τους συντελεστές του πολυωνύμου, και P4, με P κεφαλαίο, για τις τιμές του πολυωνύμου, έτσι ώστε να μην συγχύσουμε τα δύο.

Ας δούμε τώρα τι παίρνουμε αν ζητήσουμε πολυώνυμα που έχουν βαθμό 1, 2 και 3.

```
>> p1 = polyfit(x,y,1)
p1 =
    2.6887    0.8585

>> p2 = polyfit(x,y,2)
p2 =
    0.0845    1.9935    1.8870

>> p3 = polyfit(x,y,3)
p3 =
    0.0119   -0.0532    2.4254    1.5400
```

Τα πιο πάνω σημαίνουν ότι τα πολυώνυμα

$$p_1(x) = 2.6887x + 0.8585$$

$$p_2(x) = 0.0845x^2 + 1.9935x + 1.8870$$

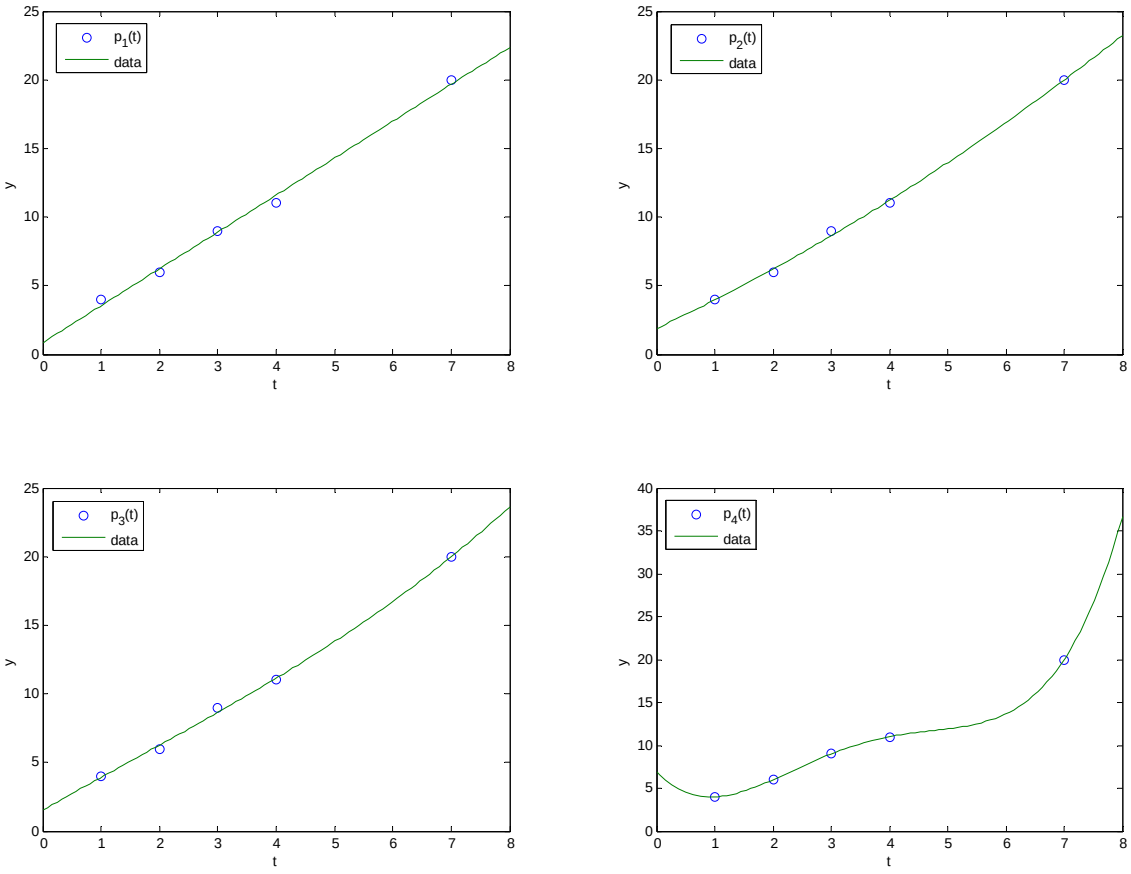
$$p_3(x) = 0.0119x^3 - 0.05324x^2 + 2.4254x + 1.54$$

προσαρμόζουν τα δεδομένα, χωρίς κατά ανάγκη να περνούν από όλα τα σημεία – ίσως να μην περνούν και από κανένα από αυτά.

Παίρνουμε την γραφική παράσταση όλων των πολυωνύμων που έχουμε κατασκευάσει, στο ίδιο παράθυρο με την εντολή **subplot**, ως εξής:

```
>> P1 = polyval(p1,t);
>> P2 = polyval(p2,t);
>> P3 = polyval(p3,t);

>> subplot(2,2,1)
>> plot(x,y,'o',t,P1)
```

```
>> xlabel('t')
>> ylabel('y')
>> legend('p_1(t)','data')

>> subplot(2,2,2)
>> plot(x,y,'o',t,P2)
>> xlabel('t')
>> ylabel('y')
>> legend('p_2(t)','data')

>> subplot(2,2,3)
>> plot(x,y,'o',t,P3)
>> xlabel('t')
>> ylabel('y')
>> legend('p_3(t)','data')

>> subplot(2,2,4)
>> plot(x,y,'o',t,P4)
>> xlabel('t')
>> ylabel('y')
>> legend('p_4(t)','data')
```



Παράδειγμα 6.3.2

Θα κατασκευάσουμε πολυώνυμα που προσαρμόζουν τα δεδομένα (0.9, 0.9), (1.5, 1.5), (3, 2.5), (4, 5.1), (6, 4.5), (8, 4.9) και (9.5, 6.3).

Ορίζουμε τα διανύσματα *x* και *y* ως

```
>> x = [0.9, 1.5, 3, 4, 6, 8, 9.5];
>> y = [0.9, 1.5, 2.5, 5.1, 4.5, 4.9, 6.3];
```

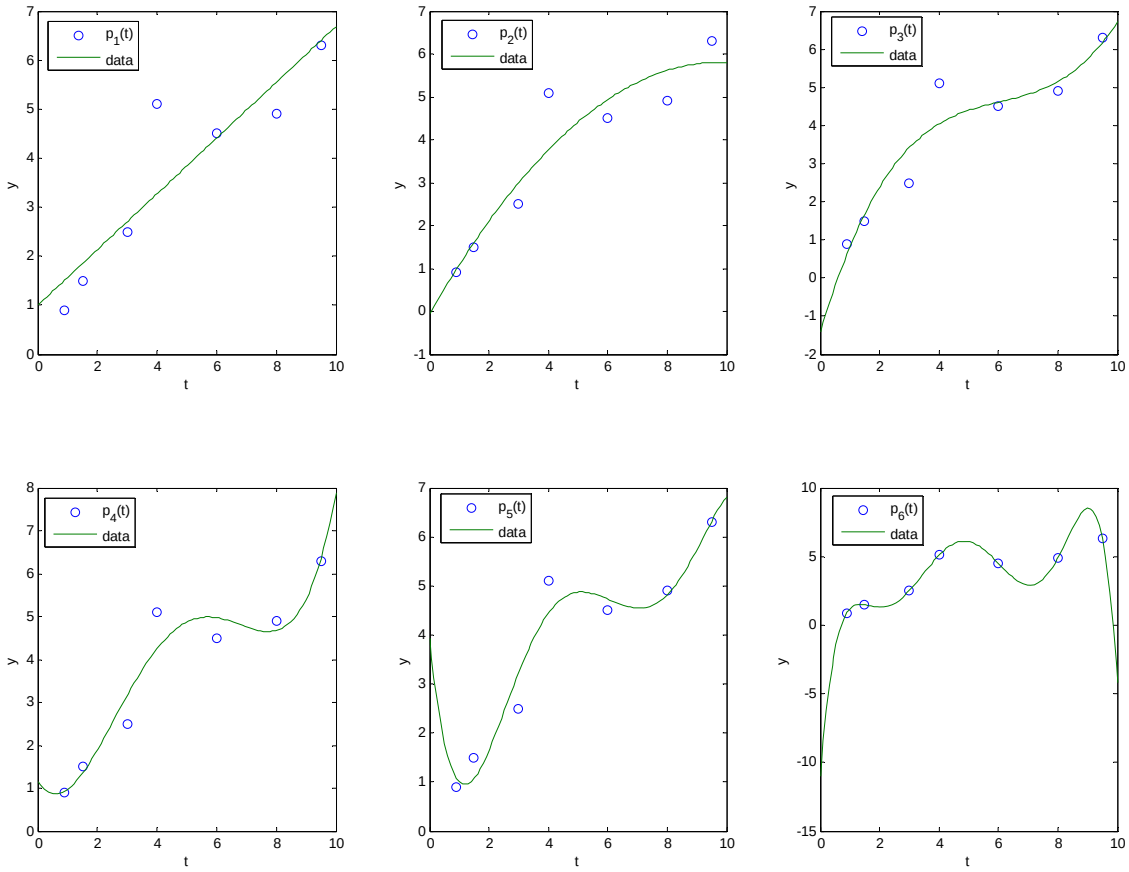
και με την εντολή polyfit παίρνουμε τους συντελεστές των πολυωνύμων βαθμού 1, 2, …, 6 (μια και έχουμε 7 σημεία) ως εξής:

```
>> p1 = polyfit(x,y,1);
>> p2 = polyfit(x,y,2);
>> p3 = polyfit(x,y,3);
>> p4 = polyfit(x,y,4);
>> p5 = polyfit(x,y,5);
>> p6 = polyfit(x,y,6);
```

Παίρνουμε την γραφική παράσταση των πιο πάνω πολυωνύμων, στο διάστημα [0, 10] ως εξής:

```
>> t = linspace(0,10);

>> P1 = polyval(p1,t);
>> P2 = polyval(p2,t);
>> P3 = polyval(p3,t);
>> P4 = polyval(p4,t);
>> P5 = polyval(p5,t);
>> P6 = polyval(p6,t);

>> subplot(2,3,1)
>> plot(x,y,'o',t,P1)
>> xlabel('t')
>> ylabel('p_1(t)')
>> legend('data','p_1(t)')

>> subplot(2,3,2)
>> plot(x,y,'o',t,P2)
>> xlabel('t')
>> ylabel('p_2(t)')
>> legend('data','p_2(t)')

>> subplot(2,3,3)
>> plot(x,y,'o',t,P3)
>> xlabel('t')
>> ylabel('p_3(t)')
>> legend('data','p_3(t)')

>> subplot(2,3,4)
>> plot(x,y,'o',t,P4)
>> xlabel('t')
>> ylabel('p_4(t)')
>> legend('data','p_4(t)')

>> subplot(2,3,5)
>> plot(x,y,'o',t,P5)
>> xlabel('t')
>> ylabel('p_5(t)')
>> legend('data','p_5(t)')

>> subplot(2,3,6)
>> plot(x,y,'o',t,P6)
>> xlabel('t')
>> ylabel('p_6(t)')
```

```
>> legend('data','p_6(t)')
```



Προσαρμογή δεδομένων με μη-πολυωνυμικές συναρτήσεις

Υπάρχουν περιπτώσεις για της οποίες δεν θέλουμε να προσαρμόσουμε πολυώνυμο στα δεδομένα, αλλά κάποια άλλη συνάρτηση που θα τα αντιπροσωπεύει καλύτερα. Αν, για παράδειγμα, πιστεύουμε ότι τα δεδομένα x και y σχετίζονται με εκθετικό τρόπο, δηλ. θέλουμε να βρούμε σταθερές b και m, τέτοιες ώστε η συνάρτηση

$$y = be^{mx}$$

να προσαρμόζει τα δεδομένα, τότε πάλι μπορούμε να χρησιμοποιήσουμε την εντολή polyfit. Αυτή τη φορά δεν θα δώσουμε σαν δεδομένα εισόδου τα διανύσματα x και y, αλλά τα διανύσματα x και $\ln y$, και θα κάνουμε γραμμική προσαρμογή ως εξής:

polyfit(x, log(y), 1)

Ο λόγος είναι απλός: Αν $y = be^{mx}$ τότε,

$$\ln y = \ln be^{mx} \Rightarrow \underbrace{\ln y}_{Y} = \underbrace{\ln b}_{B} + mx \Leftrightarrow Y = mx + B$$

που μας λέει ότι θέλουμε να βρούμε ένα πολυώνυμο $1^{ου}$ βαθμού για τα δεδομένα $(x, Y) = (x, \ln y)$.

Παράδειγμα 6.3.3

Έστω ότι έχουμε τα εξής δεδομένα

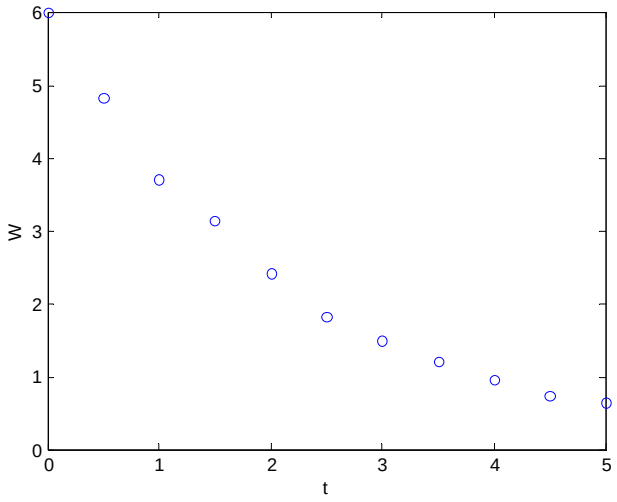
t	0.0	0.5	1	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
W	6.00	4.83	3.70	3.15	2.41	1.83	1.49	1.21	0.96	0.73	0.64

και θέλουμε να βρούμε μια κατάλληλη συνάρτηση $y = f(t)$ που τα αντιπροσωπεύει. Ορίζουμε τα διανύσματα

```
>> t=[0.0 0.5 1 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0];
>> W=[6.00 4.83 3.70 3.15 2.41 1.83 1.49 1.21 0.96 0.73 0.64];
```

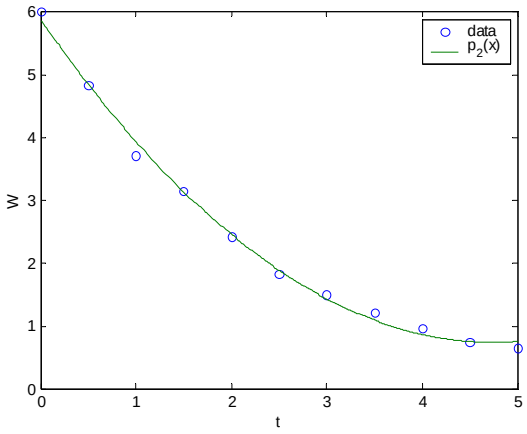
και κάνουμε την γραφική τους παράσταση:

```
>> plot(t,W,'o')
>> xlabel('t')
>> ylabel('W')
```

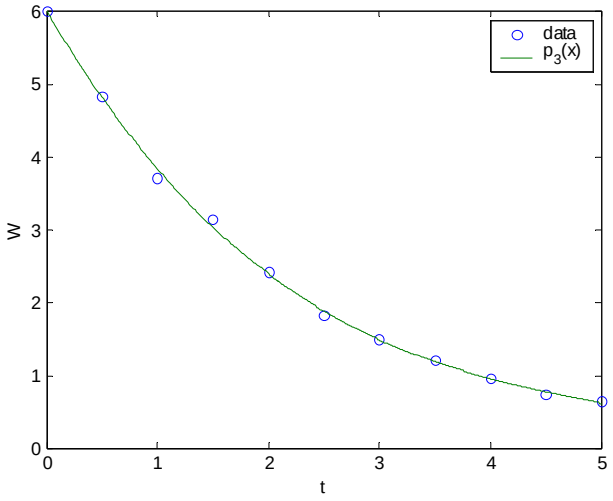


Προφανώς, δεν θα ψάξουμε για πολυώνυμο $1^{ου}$ βαθμού, μια και τα δεδομένα δεν φαίνεται να βρίσκονται πάνω σε μια ευθεία. Ας δούμε τι παίρνουμε αν χρησιμοποιήσουμε πολυώνυμα $2^{ου}$, $3^{ου}$ και $4^{ου}$ βαθμού.

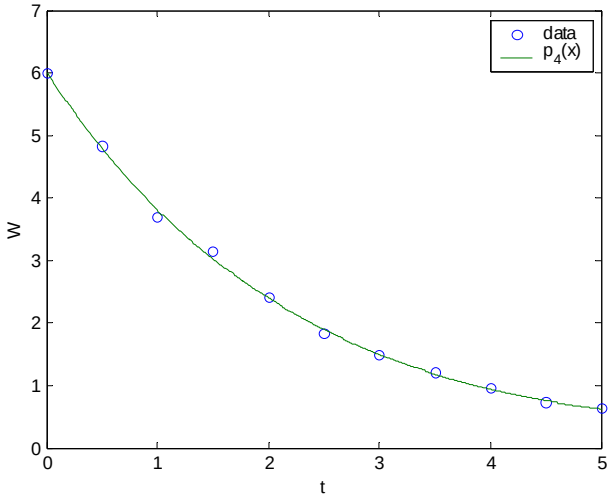
```
>> p=polyfit(t,W,2);
>> tt=0:0.01:5;
>> p2=polyval(p,tt);
>> plot(t,W,'o',tt,p2)
>> xlabel('t')
>> ylabel('W')
>> legend('data','p_2(x)')
```



```
>> p=polyfit(t,W,3);
>> tt=0:0.01:5;
>> p3=polyval(p,tt);
>> plot(t,W,'o',tt,p3)
>> xlabel('t')
>> ylabel('W')
>> legend('data','p_3(x)')
```




```
>> p=polyfit(t,W,4);
>> tt=0:0.01:5;
>> p4=polyval(p,tt);
>> plot(t,W,'o',tt,p4)
>> xlabel('t')
>> ylabel('W')
>> legend('data','p_4(x)')
```



Τα πιο πάνω δίνουν ικανοποιητικά αποτελέσματα, αλλά ας δοκιμάσουμε και μια εκθετική συνάρτηση:

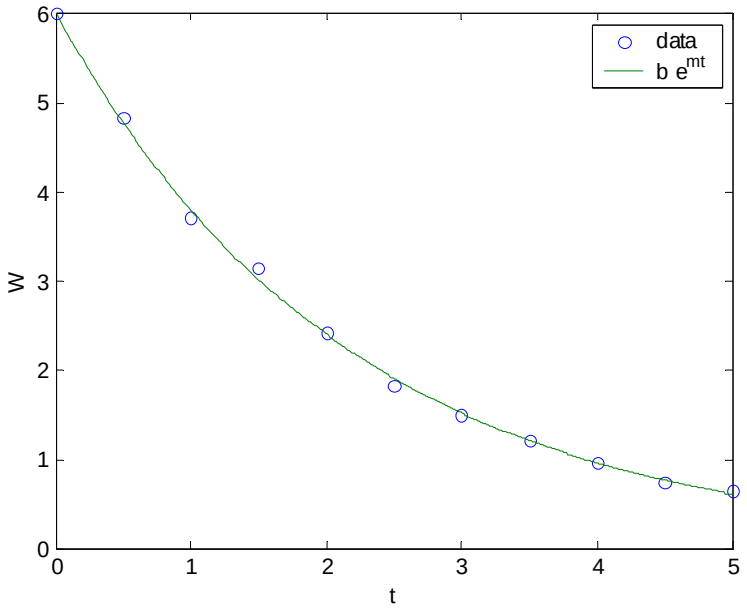
```
>> p=polyfit(t,log(W),1)
p =
   -0.4580    1.7899
```

Αυτό σημαίνει ότι η συνάρτηση $y_{\exp}(t) = be^{mt}$, όπου

```
>> m=p(1);
>> b=exp(p(2))
b =
    5.9889
```

προσαρμόζει τα δεδομένα. Ας δούμε την γραφική της παράσταση:

```
>> y_exp=b*exp(m*tt);
>> plot(t,W,'o',tt,y_exp)
>> xlabel('t')
>> ylabel('W')
>> legend('data','b e^{mt}')
```



Από ότι φαίνεται, τα δεδομένα σχετίζονται με εκθετικό τρόπο αφού το πιο πάνω αποτέλεσμα είναι το ‘καλύτερο’.

6.4 Ασκήσεις

6.1 Τροποποιείστε το m-file `polyadd.m` του παραδείγματος 6.1.2 έτσι ώστε όταν το άθροισμά των πολυωνύμων p και q έχει βαθμό μικρότερο από το max{degp, degq} να μην εμφανίζονται περιττά μηδενικά στην αρχή του διανύσματος που αντιστοιχεί στο $p+q$.

6.2 Δημιουργήστε ένα function m-file με όνομα `pconv.m` και μεταβλητές εισόδου το πολυώνυμο p και ένα φυσικό αριθμό n που να βρίσκει το πολυώνυμο p^n.

6.3 Χρησιμοποιώντας της εντολές πολυωνύμων βιβλιοθήκης της MATLAB, να κάνετε την γραφική παράσταση του $y = 1.5x^4 - 5x^2 + x + 2$ για $x \in [-2, 2]$.

6.4 Χρησιμοποιώντας της εντολές πολυωνύμων βιβλιοθήκης της MATLAB, να διαιρέσετε το πολυώνυμο $15x^5 + 35x^4 - 37x^3 - 19x^2 + 41x - 15$ δια $x^2 + 4x + 2$. Να γράψετε την απάντηση που σας δίνει η MATLAB σε μαθηματική μορφή.

6.5 Έστω $f(x) = 4x^4 + 6x^3 - 2x^2 - 5x + 3$ και $g(x) = x^2 + 4x + 2$. Χρησιμοποιώντας της εντολές πολυωνύμων βιβλιοθήκης της MATLAB, να:

(α) Βρείτε τις ρίζες της συνάρτησης $5f(x) - 3g(x)$ και να τις εντοπίσετε στην γραφική της παράσταση (με βέλη και κείμενο).

(β) Βρείτε μια παράσταση για την συνάρτηση $\left(f(x)g(x)\right)'$.

(γ) Βρείτε μια παράσταση για την συνάρτηση $\left(\dfrac{g(x)}{f(x)}\right)$.

6.6 Γράψτε ένα function m-file με όνομα `polyint.m`, το οποίο να υπολογίζει το ολοκλήρωμα ενός πολυωνύμου p θέτοντας τη σταθερά ολοκλήρωσης ίση με 0. Για παράδειγμα για το `p = [1 1 1]` το m-file πρέπει να δίνει

```
polyint(p) = [0.33333 0.50000 1.00000 0.00000]
```

Σε τι διαφέρουν οι `polyder(polyint(p))` και `polyint(polyder(p))`;

6.7 Έστω $f(x) = 3x^4 - 16x^3 + 18x^2$. Χρησιμοποιώντας της εντολές πολυωνύμων, καθώς και άλλες εντολές βιβλιοθήκης της MATLAB, να βρείτε τα μέγιστα και ελάχιστα σημεία της συνάρτησης (αν υπάρχουν), όπως και τα σημεία καμπής (αν υπάρχουν). Να βρείτε, επίσης, τις ρίζες της συνάρτησης (δηλ. τα σημεία που περνά από τον από τον άξονα των x) και να εντοπίσετε όλες αυτές τις πληροφορίες στη γραφική παράσταση της $f(x)$ με βέλη και κείμενο.

6.8 Έστω ότι έχουμε τα εξής δεδομένα:

x	-5	-4	-2.2	-1	0	1	2.2	4	5	6	7
y	0.1	0.2	0.8	2.6	3.9	5.4	3.6	2.2	3.3	6.7	8.9

Να προσαρμόσετε τα πιο πάνω με πολυώνυμα βαθμού 1, 3, 4 και 10. Για το καθένα, να κατασκευάσετε ένα γράφημα που να δείχνει το πολυώνυμο μαζί με τα σημεία (στο ίδιο γράφημα). Τι παρατηρείτε όταν το πολυώνυμο έχει βαθμό 10;

6.9 Να γράψετε ένα m-file, που να καλείται `powerfit.m`, το οποίο να παίρνει σαν δεδομένα εισόδου δυο διανύσματα X και Y (του ιδίου μεγέθους) και να δίνει σαν

δεδομένα εξόδου τα b και m, έτσι ώστε η συνάρτηση $y = bx^m$ να προσαρμόζει τα δεδομένα. Να χρησιμοποιήσετε το m-file σας για τα πιο κάτω δεδομένα και να να κατασκευάσετε ένα γράφημα που να δείχνει τη συνάρτηση μαζί με τα δοθέντα σημεία (στο ίδιο γράφημα).

X	0.5	1.9	3.3	4.7	6.1	7.5
Y	0.8	10.1	25.7	59.2	105	122

6.10 Έστω τα δεδομένα:

x	0.2	0.5	1	1.2	1.5	1.7	2
y	0.21	0.49	0.95	1.05	0.95	0.79	0.65

(α) Να προσαρμόσετε τα πιο πάνω με πολυώνυμα βαθμού 1, 2, 3, 4, 5 και 6. Για το καθένα, να κατασκευάσετε ένα γράφημα που να δείχνει το πολυώνυμο μαζί με τα σημεία (στο ίδιο γράφημα). Τι παρατηρείτε όταν το πολυώνυμο έχει βαθμό 6;

(β) Για κάθε πολυώνυμο παρεμβολής p_i υπολογίσετε τις τιμές Y_i στα πιο πάνω σημεία καθώς και τη νόρμα του διανύσματος $Y_i - y$. Πως μεταβάλλεται η νόρμα αυτή με το βαθμό του πολυωνύμου παρεμβολής;

7 ΕΚΤΥΠΩΣΗ ΚΑΙ ΑΝΑΓΝΩΣΗ ΔΕΔΟΜΕΝΩΝ

Στο κεφάλαιο αυτό θα δούμε πως γίνεται η εισαγωγή δεδομένων από τον χρήστη, η εκτύπωση πληροφοριών στην οθόνη και η ανάγνωση δεδομένων από αρχεία καθώς και η αποθήκευση πληροφοριών σε αρχεία.

7.1 Εισαγωγή δεδομένων από τον χρήστη

Γνωρίζουμε ήδη ότι ένας απλός τρόπος για να εισαγάγουμε την τιμή μιας νέας μεταβλητής είτε στο παράθυρο εντολών είτε μέσω κάποιου m-file είναι με την εντολή **input**. Ακολουθούν μερικά παραδείγματα:

```
>> x=input('Enter initial point: ')
Enter initial point: 1.5
x =
    1.5000

>> A=input('Enter 2X3 matrix: ')
Enter 2X3 matrix: rand(2,3)
A =
    0.9501    0.6068    0.8913
    0.2311    0.4860    0.7621
```

Είναι επίσης γνωστό ότι μπορούμε να εισαγάγουμε μια αλφαριθμητική τιμή είτε μέσα σε τόνους,

```
>> str1=input('Enter a string: ')
Enter a string: 'First string'
str1 =
First string
```

είτε χωρίς, εφόσον δώσουμε ως όρισμα της input το ‘s’:

```
>> str1=input('Enter another string: ','s')
Enter another string: Second string
str1 =
Second string
```

Αν βάλουμε ερωτηματικό στο τέλος της εντολής input τότε η εισαγόμενη τιμή δεν τυπώνεται στην οθόνη. Για παράδειγμα,

```
>> str1=input('Enter another string: ','s');
Enter another string: Second string
```

7.1.1 Η εντολή pause

Η εντολή **pause** χρησιμοποιείται συνήθως σε function m-files για να δώσει χρόνο στον χρήστη να παρατηρήσει ένα γράφημα πριν τη δημιουργία ενός άλλου ή να

ετοιμάσει κάποια δεδομένα που θα χρειαστούν στη συνέχεια. Οι πιθανοί τρόποι κλήσης της εντολής αυτής είναι οι εξής:

- **pause**: αναστολή της εκτέλεσης του προγράμματος μέχρι να πατήσει ο χρήστης οποιοδήποτε πλήκτρο.
- **pause(n)**: αναστολή της εκτέλεσης του προγράμματος για n δευτερόλεπτα.
- **pause off**: απενεργοποίηση όλων των επόμενων εντολών pause και pause(n). Με αυτή την επιλογή μπορούμε να τρέξουμε διαδραστικά m-files γρήγορα αποφεύγοντας το πάτημα πλήκτρων ή διαδοχικές αναμονές.
- **pause on**: ενεργοποίηση των εντολών pause και pause(n).

Παράδειγμα 7.1.1

Οι συναρτήσεις Bessel πρώτου είδους υπολογίζονται από τη συνάρτηση βιβλιοθήκης

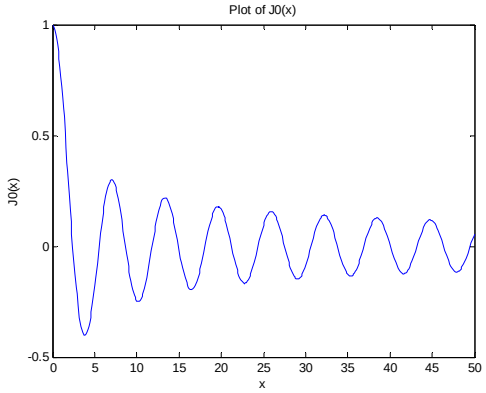
besselj(n,x)

όπου *n* η τάξη της συνάρτησης Bessel. Για παράδειγμα, η συνάρτηση Bessel πρώτου είδους και μηδενικής τάξης που συμβολίζεται με $J_0(x)$ υπολογίζεται με την besselj(0,x), η $J_1(x)$ με την besselj(1,x) κοκ. Μπορείτε να μάθετε περισσότερα για τη συνάρτηση besselj καθώς και για τις άλλες συναρτήσεις **bessely**, **besseli** και **besselk** με την εντολή help.

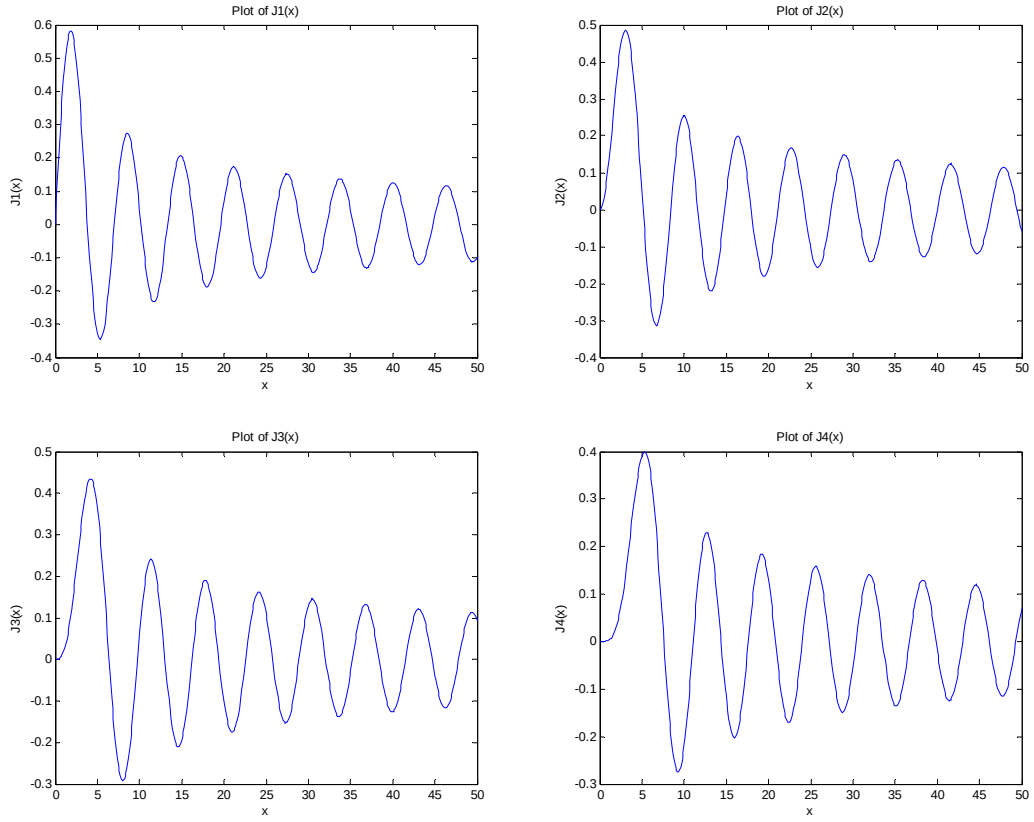
Το πιο κάτω function m-file κάνει διαδοχικά τα γραφήματα των J_0, J_1, J_2, J_3 και J_4 στο διάστημα [0, *b*] και αναμένει από τον χρήστη να πατήσει ένα πλήκτρο για να προχωρήσει από το ένα γράφημα στο άλλο.

```
function []=drawjn(b)

% DRAWJN
% Draws the plots of the Bessel functions J0-J4
% in the interval [0,b].
% The user should press any key to proceed from one
% plot to the next one

J0=@(x) besselj(0,x);
fplot(J0,[0,b]);  xlabel('x');ylabel('J0(x)');  title('Plot of
J0(x)'); pause
J1=@(x) besselj(1,x);
fplot(J1,[0,b]);  xlabel('x');ylabel('J1(x)');  title('Plot of
J1(x)'); pause
J2=@(x) besselj(2,x);
fplot(J2,[0,b]);  xlabel('x');ylabel('J2(x)');  title('Plot of
J2(x)'); pause
J3=@(x) besselj(3,x);
fplot(J3,[0,b]);  xlabel('x');ylabel('J3(x)');  title('Plot of
J3(x)'); pause
J4=@(x) besselj(4,x);
fplot(J4,[0,b]);  xlabel('x');ylabel('J4(x)');  title('Plot of
J4(x)'); pause
close

%End of DRAWJN
```

Όταν καλέσουμε το drawjn ανοίγει το παράθυρο γραφικών με το γράφημα της J_0:



Στη συνέχεια πατώντας οποιοδήποτε πλήκτρο παίρνουμε διαδοχικά τα υπόλοιπα γραφήματα:



7.1.2 Η εντολή ginput

Η συνάρτηση **ginput** είναι μια πολύ ενδιαφέρουσα συνάρτηση για την εισαγωγή δεδομένων μέσω του ποντικιού (mouse). Η συνάρτηση χρησιμοποιείται αφού πρώτα έχουμε ανοίξει ένα παράθυρο γραφικών. Η εντολή

[x, y] = ginput(n)

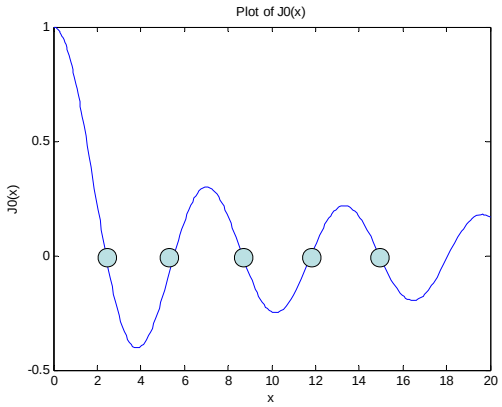
αποθηκεύει στα διανύσματα *x* και *y* τις συντεταγμένες των επόμενων *n* κλικ που θα κάνουμε στο παράθυρο των γραφικών. Τα σημεία αυτά μπορούν να επιλεγούν επίσης πατώντας οποιοδήποτε πλήκτρο εκτός από το (Carriage) Return που αν πατηθεί διακόπτει τη διαδικασία πριν την εισαγωγή των *n* σημείων.

Παράδειγμα 7.1.2

Κατασκευάζουμε πρώτα το γράφημα της $J_0(x)$.

```
>> J0=@(x) besselj(0,x)
J0 =
    @(x) besselj(0,x)

>> fplot(J0,[0,20])
>> xlabel('x');ylabel('J0(x)');  title('Plot of J0(x)');
```

Θα βρούμε πρώτα αδρές προσεγγίσεις των πρώτων 5 ριζών της J_0 με τη βοήθεια της εντολής ginput κάνοντας κλικ στα σημεία που φαίνονται στο γράφημα της συνάρτησής μας:



Παίρνουμε έτσι:

```
>> [x,y]=ginput(5)
x =
   2.419354838709677
   5.460829493087557
   8.732718894009215
  11.728110599078340
  14.953917050691242
y =
  -0.002192982456140
  -0.006578947368421
  -0.006578947368421
  -0.006578947368421
  -0.002192982456140
```

Μπορούμε να βρούμε τις ρίζες με μεγαλύτερη ακρίβεια αξιοποιώντας τις αδρές τιμές που έχουμε ήδη βρει. Η συνάρτηση **fzero** της MATLAB αναζητεί μια ρίζα μιας συνάρτησης fun κοντά στο αρχικό σημείο x_0. Η αντίστοιχη εντολή είναι

fzero(fun, x0)

Θα βρούμε τις 5 πρώτες 'ακριβείς' ρίζες της J_0 με ένα βρόχο for:

```
>> for i=1:5, xexact(i)=fzero(J0,x(i)); end
>> xexact'
ans =
2.404825557695773
5.520078110286312
8.653727912911013
11.791534439014280
14.930917708487785
```

7.2 Εκτύπωση δεδομένων στην οθόνη

Ο πιο απλός τρόπος για να τυπώσουμε τα αποτελέσματα της MATLAB στο παράθυρο εντολών είναι να γράψουμε απλώς το όνομα της μεταβλητής ή ακόμα να μη χρησιμοποιήσουμε ερωτηματικό στο τέλος της εντολής που την υπολογίζει. Η MATLAB γράφει τότε το όνομα, ένα '=' και μετά την τιμή της μεταβλητής.

Η συνάρτηση **disp** παραλείπει το όνομα της μεταβλητής και το '=' και εκτυπώνει την τιμή της συνάρτησης σύμφωνα με την τρέχουσα format:

```
>> disp(1/eps), disp(pi)

    4.5036e+015
    3.1416

>> format long

>> disp(1/eps), disp(pi)

   4.503599627370496e+015
   3.141592653589793
```

Ένα πλεονέκτημα της disp είναι το ότι μας επιτρέπει την εκτύπωση χρήσιμων ενημερωτικών αλφαριθμητικών. Για παράδειγμα

```
>> disp('Here is a 3x2 random matrix:'), rand(2,3)

Here is a 3x2 random matrix:

ans =
    0.9218    0.1763    0.9355
    0.7382    0.4057    0.9169
```

Κάτι που δεν μπορούμε να κάνουμε με τη disp είναι να τυπώσουμε διάφορες μεταβλητές σε μια γραμμή:

```
>> disp('Calculated speed:'), disp(121), disp('km/h')

Calculated speed:
121
km/h
```

Αυτό το πρόβλημα λύνεται με τις εντολές **sprintf** και **fprintf** με τις οποίες θα ασχοληθούμε στη συνέχεια.

Ας δούμε τώρα ένα άλλο παράδειγμα όπου έχουμε ένα διάνυσμα *u*:

```
>> u=1:6
u =
     1     2     3     4     5     6
```

Με τις εντολές

```
>> disp('The values in u are: '), disp(u)

The values in u are:
     1     2     3     4     5     6
```

παρατηρούμε ξανά ότι τα αποτελέσματα τυπώνονται σε διαφορετικές γραμμές. Ένας άλλος τρόπος για να αποφύγουμε αυτό το πρόβλημα είναι να δημιουργήσουμε ένα σύνθετο πίνακα με τα δύο αποτελέσματα στην ίδια γραμμή. Δεδομένου ότι το *u* είναι διάνυσμα, χρησιμοποιούμε την εντολή **num2str** (number to string) η οποία

μετατρέπει αριθμούς και διανύσματα σε αλφαριθμητικά ή ακόμα πίνακες αριθμών σε πίνακες αλφαριθμητικών. Έτσι με την εντολή

```
>> disp(['The values in u are: ' num2str(u)])
```

παίρνουμε:

```
The values in u are: 1  2  3  4  5  6
```

Σημειώνουμε ότι απλώς τυπώσαμε στην οθόνη το αλφαριθμητικό που δημιουργήσαμε:

```
>> str=['The values in u are: ' num2str(u)]
str =
The values in u are: 1  2  3  4  5  6
```

Οι δυνατότητες της num2str δεν εξαντλούνται στο πιο πάνω παράδειγμα. Ας πάρουμε για παράδειγμα τον πίνακα

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

>> disp('The values in A are: '), disp(A)

The values in A are:
     1     2     3
     4     5     6
```

Αν χρησιμοποιήσουμε τη num2str για τον A παίρνουμε ένα αλφαριθμητικό 2×7 πίνακα:

```
>> B=num2str(A)
B =
1  2  3
4  5  6

>> whos
  Name      Size            Bytes  Class     Attributes
  A         2x3                48  double
  B         2x7                28  char
```

Έτσι ο B δεν είναι πίνακας αριθμών αλλά αλφαριθμητικών. Μπορούμε λοιπόν να γράψουμε:

```
>> disp([['The first line in A is:  ';'The second line in A is:
']  num2str(A)])
The first line in A is:  1  2  3
The second line in A is: 4  5  6
```

7.2.1 Η εντολή sprintf

Η εντολή **sprintf** καθώς επίσης και η συγγενική της **fprintf** που θα δούμε αργότερα χρησιμοποιείται αντί της disp στην περίπτωση που είναι επιθυμητή η εκτύπωση δεδομένων σε συγκεκριμένη **μορφή** (format). Στην πραγματικότητα η sprintf εκτυπώνει τις μεταβλητές σ'ένα αλφαριθμητικό s το οποίο τυπώνεται στην οθόνη εφόσον δεν βάλουμε ερωτηματικό στο τέλος της εντολής. Η sprintf είναι ιδιαίτερα χρήσιμη για τη δημιουργία ετικετών για γραφικά.

Η σύνταξη της sprintf στη γενική περίπτωση έχει ως εξής:

[s, errmsg] = sprintf(format, A, B, ….)

όπου οι πίνακες A, B, …. τυπώνονται στο αλφαριθμητικό s με τη μορφή που καθορίζεται στο αλφαριθμητικό format, όπως θα δούμε πιο κάτω. Το errmsg είναι ένα προαιρετικό όρισμα εισόδου και επιστρέφει ένα αλφαριθμητικό μήνυμα σφάλματος αν έχει συμβεί σφάλμα ή ένα κενό αλφαριθμητικό διαφορετικά. Άρα η πιο απλή εκδοχή της εντολής είναι η

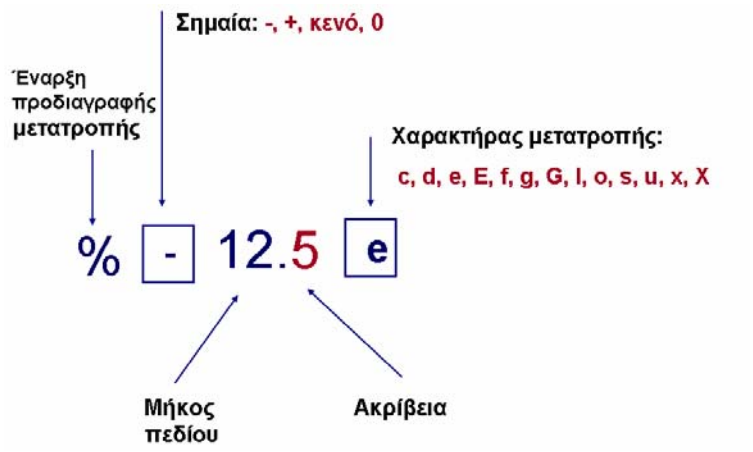
s = sprintf(format, A, B, ….)

Αν αυτό που μας ενδιαφέρει είναι η εκτύπωση του s στην οθόνη μπορούμε να γράψουμε απευθείας

disp(sprintf(format, A, B, ….))

Το όρισμα format είναι ένα αλφαριθμητικό με συνήθεις χαρακτήρες ή/και **προδιαγραφές μετατροπής** (conversion specification) της γλώσσας C. Μια προδιαγραφή μετατροπής ελέγχει τον συμβολισμό, τη στοίχιση, το πλήθος των σημαντικών ψηφίων, το μήκος πεδίου, και άλλα χαρακτηριστικά της μορφής της εκτύπωσης.

Οι προδιαγραφές μετατροπής ξεκινούν πάντα με το σύμβολο '%' και έχουν την πιο κάτω μορφή:



Σημαίες (flags)

Η ευθυγράμμιση της εκτύπωσης ελέγχεται με τη χρήση των πιο κάτω προαιρετικών σημαίων:

Χαρακτήρας	Περιγραφή	Παράδειγμα
-	Αριστερή στοίχιση	%-7.2f
+	Εκτύπωση προσήμου (+ ή -)	%+7.2f
κενό	Εισαγωγή κενού πριν από την τιμή	% 7.2f
0	Pads με μηδενικά αντί με κενά	%07.2f

Ο χαρακτήρας μετατροπής f δηλώνει συμβολισμό σταθερής υποδιαστολής ενώ ο d δηλώνει (προσημασμένο) δεκαδικό συμβολισμό.

Παράδειγμα 7.2.1

Θεωρούμε τον αριθμό π. Ας δούμε πως τυπώνεται με όλες τις πιθανές σημαίες:

```
>> x=pi
x =
    3.1416

>> sprintf('Oi 360 moires antistoixoun se %9.2f aktinia',x)
ans =
Oi 360 moires antistoixoun se      3.14 aktinia

>> sprintf('Oi 360 moires antistoixoun se %-9.2f aktinia',x)
ans =
Oi 360 moires antistoixoun se 3.14      aktinia

>> sprintf('Oi 360 moires antistoixoun se %+9.2f aktinia',x)
ans =
Oi 360 moires antistoixoun se     +3.14 aktinia

>> sprintf('Oi 360 moires antistoixoun se % 9.2f aktinia',x)
ans =
Oi 360 moires antistoixoun se      3.14 aktinia

>> sprintf('Oi 360 moires antistoixoun se %09.2f aktinia',x)
ans =
Oi 360 moires antistoixoun se 000003.14 aktinia
```

Παρατηρούμε ότι πριν από την προδιαγραφή μετατροπής μπορούμε να έχουμε αλφαριθμητικά της επιλογής μας.

Παράδειγμα 7.2.2

Θεωρούμε τον αριθμό $-e^3$. Ας δούμε πως τυπώνεται για διαφορετικές σημαίες:

```
>> sprintf('O -e^{-3} einai: %14.4d ',-exp(-3))
ans =
O -e^{-3} einai:   -4.9787e-002

>> sprintf('O -e^{-3} einai: %-14.4d ',-exp(-3))
ans =
O -e^{-3} einai: -4.9787e-002

>> sprintf('O -e^{-3} einai: %+14.4d ',-exp(-3))
ans =
O -e^{-3} einai:   -4.9787e-002

>> sprintf('O -e^{-3} einai: % 14.4d ',-exp(-3))
ans =
O -e^{-3} einai:   -4.9787e-002

>> sprintf('O -e^{-3} einai: %014.4d ',-exp(-3))
ans =
O -e^{-3} einai: -004.9787e-002
```

Στα πιο πάνω παραδείγματα είδαμε πως μπορούμε να ελέγξουμε το **μήκος πεδίου** και την **ακρίβεια** (precision) της εκτύπωσης. Το μήκος πεδίου είναι ο αριθμός που ακολουθεί τη σημαία και προσδιορίζει τον ελάχιστο αριθμό ψηφίων που θα εκτυπωθούν. Η ακρίβεια είναι ο αριθμός που ακολουθεί την τελεία (υποδιαστολή)

και προσδιορίζει το πλήθος των ψηφίων που θα τυπωθούν δεξιά της υποδιαστολής. Αν δεν δηλωθεί η ακρίβεια, η προεπιλογή της MATLAB είναι τα 6 δεκαδικά ψηφία.

Παράδειγμα 7.2.3

Θα τυπώσουμε τον αριθμό π με χαρακτήρα μετατροπής f και διάφορα μήκη πεδίου και ακρίβειας:

```
>> sprintf('O \\pi me format %%6f grafetai: %6f', pi)
ans =
O \pi me format %6f grafetai: 3.141593

>> sprintf('O \\pi me format %%6.2f grafetai: %6.2f', pi)
ans =
O \pi me format %6.2f grafetai:   3.14

>> sprintf('O \\pi me format %%6.4f grafetai: %6.4f', pi)
ans =
O \pi me format %6.4f grafetai: 3.1416

>> sprintf('O \\pi me format %%9.2f grafetai: %9.2f', pi)
ans =
O \pi me format %9.2f grafetai:      3.14

>> sprintf('O \\pi me format %%9.5f grafetai: %9.5f', pi)
ans =
O \pi me format %9.5f grafetai:   3.14159
```

Θα πρέπει να σημειώσουμε ότι πιο πάνω χρησιμοποιήσαμε **χαρακτήρες απόδρασης**. Το διπλό \\ δηλώνει το απλό σύμβολο \ (και ότι δεν ακολουθεί χαρακτήρας απόδρασης – escape character) ενώ το διπλό %% δηλώνει το σύμβολο % (και ότι δεν ακολουθεί προδιαγραφή μετατροπής).

Παράδειγμα 7.2.4

Ο χαρακτήρας μετατροπής e δηλώνει τον εκθετικό συμβολισμό. Θα τυπώσουμε τον αριθμό 1/1234 με διάφορα μήκη πεδίου και ακρίβειας:

```
>> x=1/1234;

>> disp(sprintf('%9e', x))
8.103728e-004

>> disp(sprintf('%9.0e', x))
   8e-004

>> disp(sprintf('%9.4e', x))
8.1037e-004
```

Εδώ χρησιμοποιήσαμε τη disp μαζί με την sprintf για να αποφύγουμε την εκτύπωση του ονόματος της προεπιλεγμένης πρόχειρης μεταβλητής ans στην οθόνη.

Παράδειγμα 7.2.5

Θα τυπώσουμε τον αριθμό π με τρεις τρόπους χωρίς να δηλώσουμε την ακρίβεια:

```
>> disp(sprintf('%d  %e  %f', pi, pi, pi))
3.141593e+000  3.141593e+000  3.141593
```

Ας δηλώσουμε τώρα ακρίβεια δύο δεκαδικών ψηφίων και στις τρεις περιπτώσεις, παίρνουμε:

```
>> disp(sprintf('%6.2d  %6.2e  %6.2f', pi, pi, pi))
3.14e+000  3.14e+000    3.14
```

Χαρακτήρες μετατροπής (conversion characters)

Οι χαρακτήρες μετατροπής καθορίζουν το συμβολισμό της εκτύπωσης όπως φαίνεται στον πίνακα που ακολουθεί. Στο κεφάλαιο αυτό μας ενδιαφέρουν κυρίως οι πρώτοι έξι χαρακτήρες.

Χαρακτήρας	Περιγραφή
%d	(Προσημασμένος) δεκαδικός συμβολισμός
%e	Εκθετικός συμβολισμός (με μικρό e, όπως στον 3.1415e+00)
%f	Συμβολισμός σταθερής υποδιαστολής
%g	Η πιο συμπαγής μορφή από τις %e και %f με παράλειψη αχρείαστων μηδενικών.
%c	Απλός χαρακτήρας
%s	Αλφαριθμητικό (ακολουθία χαρακτήρων)
%E	Εκθετικός συμβολισμός (με κεφαλαίο E, όπως στον 3.1415E+00)
%G	Ίδια με την %g αλλά με τη χρήση κεφαλαίου E.
%i	(Προσημασμένος) δεκαδικός συμβολισμός
%o	(Μη προσημασμένος) οκταδικός συμβολισμός
%u	(Μη προσημασμένος) δεκαδικός συμβολισμός
%x	Δεκαεξαδικός συμβολισμός (με μικρά γράμματα a-f)
%X	Δεκαεξαδικός συμβολισμός (με κεφαλαία γράμματα A-F)

Αξίζει να σημειωθούν τα πιο κάτω:

- Η MATLAB επεκτείνει το μήκος πεδίου όποτε αυτό είναι αναγκαίο. Για παράδειγμα,

```
>> x=1.234567;
>> disp(sprintf('%6.2f',x))
  1.23
>> x=123456.789;
>> disp(sprintf('%6.2f',x))
123456.79
```

- Ο συμβολισμός σταθερής υποδιαστολής είναι κατάλληλος για την εκτύπωση ακεραίων (χρησιμοποιώντας %n.0f) ή αριθμών με δεδομένο αριθμό δεκαδικών ψηφίων μετά την υποδιαστολή (π.χ. %7.2f για λίρες ή %4.1f για βαθμούς). Για παράδειγμα,

```
>> for iter=1:5
     disp(sprintf('Iteration #: %3.0f',iter))
   end
Iteration #:   1
Iteration #:   2
Iteration #:   3
Iteration #:   4
Iteration #:   5
```

- Μπορούμε να τυπώσουμε αλφαριθμητικά με τον χαρακτήρα μετατροπής s.

Παράδειγμα 7.2.6

Θα τυπώσουμε τον αριθμό 1234.567890 με διαφορετικούς τρόπους:

```
>> x=1234.567890;
>> disp(sprintf('%c %6.2d %6.2e %6.2E', x, x, x, x))
1.234568e+003 1.23e+003 1.23e+003 1.23E+003
>> disp(sprintf('%7.3f %7.3g %7.3G %7.3i %7.3u', x, x, x, x, x))
1234.568 1.23e+003 1.23E+003 1.235e+003 1.235e+003
```

Παράδειγμα 7.2.7

Θα τυπώσουμε σε στήλες τις τιμές του ημιτόνου για διάφορες γωνίες:

```
>> x=0:30:360;
>> xr=x*pi/180;
>> y=sin(xr);
>> for i=1:length(x)
     disp(sprintf('%6.2f %12.8f', x(i), y(i)))
   end
  0.00   0.00000000
 30.00   0.50000000
 60.00   0.86602540
 90.00   1.00000000
120.00   0.86602540
150.00   0.50000000
180.00   0.00000000
210.00  -0.50000000
240.00  -0.86602540
270.00  -1.00000000
300.00  -0.86602540
330.00  -0.50000000
360.00  -0.00000000
```

Μπορούμε να αποφύγουμε τον βρόχο for ως εξής:

```
>> A=[x;y];
>> disp(sprintf('%6.2f %12.8f \n', A))
  0.00   0.00000000
 30.00   0.50000000
 60.00   0.86602540
 90.00   1.00000000
120.00   0.86602540
150.00   0.50000000
180.00   0.00000000
210.00  -0.50000000
240.00  -0.86602540
270.00  -1.00000000
300.00  -0.86602540
330.00  -0.50000000
360.00  -0.00000000
```

Παράδειγμα 7.2.8

Ο χαρακτήρας μετατροπής s χρησιμοποιείται για την εκτύπωση αλφαριθμητικών:

```
>> xmin=1.23e-3;
>> xmax=4.123e4;

>> sprintf('The value of %s is %7.2e', 'xmin', xmin)
ans =
The value of xmin is 1.23e-003

>> sprintf('The value of %s is %7.2e', 'xmax', xmax)
ans =
The value of xmax is 4.12e+004
```

Παράδειγμα 7.2.9

Θα τυπώσουμε τους βαθμούς μιας τάξης. Χρειάζεται να τυπώσουμε λοιπόν τα ονόματα (αλφαριθμητικά) και τους βαθμούς (αριθμοί) με ένα δεκαδικό ψηφίο. Όταν δηλώνουμε τα ονόματα σαν γραμμές ενός (αλφαριθμητικού) πίνακα μεριμνούμε έτσι ώστε όλες οι γραμμές να έχουν το ίδιο μήκος.

```
>> class=[
'Ioannis Ioannou   ';
'Kyriakos Kyriakou ';
'Omiros Omirou     ']

class =
Ioannis Ioannou
Kyriakos Kyriakou
Omiros Omirou

>> grades=[5.5;9.5;7.0]

grades =
    5.5000
    9.5000
    7.0000

>> for i=1:3
     disp(sprintf('%18s %4.1f',class(i,:),grades(i)))
   end

Ioannis Ioannou     5.5
Kyriakos Kyriakou   9.5
Omiros Omirou       7.0
```

Παράδειγμα 7.2.10

Ο πίνακας A είναι άγνωστων διαστάσεων. Θα τις βρούμε και θα τις τυπώσουμε ως εξής:

```
>> [m,n]=size(A);
>> disp(sprintf('A is a %dX%d matrix.', m,n))
A is a 5X9 matrix.
```

Χαρακτήρες απόδρασης (escape characters)

Οι χαρακτήρες απόδρασης καθορίζουν μη εκτυπωτικούς χαρακτήρες (nonprinting characters) σε μια προδιαγραφή μορφής:

Χαρακτήρας	Περιγραφή
\b	πίσω διάστημα (backspace)
\f	νέα τροφοδοσία (form feed)
\n	Νέα γραμμή
\r	Carriage return, παρόμοιο με τη νέα γραμμή
\t	Οριζόντια στοίχιση (Horizontal tab)
****	\
\' ' ή **''**	' (μονό εισαγωγικό)
%%	(το σύμβολο) %

Ο χαρακτήρας \t είναι χρήσιμος στη δημιουργία πινάκων αφού εξασφαλίζει τη στοίχιση των μεταβλητών εξόδου. Μπορούμε να συγκρίνουμε την εντολή

```
>> sprintf('%4.2f %4.2f %4.2f', 144.1, 111.213, 34.211)
ans =

144.10 111.21 34.21
```

με την

```
>> sprintf('%4.2f \t %4.2f \t %4.2f', 144.1, 111.213, 34.211)
ans =
144.10   111.21   34.21
```

Παράδειγμα 7.2.11

Έστω ο πίνακας

$$W = \begin{bmatrix} 3.330 & 3.200 & 3.080 & 3.130 \\ 3330 & 3200 & 3080 & 3130 \end{bmatrix}$$

που δείχνει το βάρος ενός νεογέννητου τις 4 πρώτες μέρες της ζωής του. Η πρώτη γραμμή είναι το βάρος σε κιλά ενώ η δεύτερη είναι το βάρος σε γραμμάρια. Θα τυπώσουμε τον W με διαφορετικούς τρόπους. Με την εντολή

```
>> sprintf('%6.3f %6.3f',W)
ans =
 3.330 3330.000 3.200 3200.000 3.080 3080.000 3.130 3130.000
```

τα στοιχεία του W τυπώνονται κατά στήλες σε μια γραμμή. Επειδή δεν είναι απαραίτητα τα δεκαδικά στην περίπτωση που το βάρος δίνεται σε γραμμάρια γράφουμε επίσης:

```
>> sprintf('%6.3f %5.0f',W)
ans =
 3.330  3330 3.200  3200 3.080  3080 3.130  3130
```

Για να γράψουμε τα στοιχεία του σε διαφορετικές στήλες (το βάρος σε κιλά στην $1^η$ και το βάρος σε γραμμάρια στη $2^η$) χρησιμοποιούμε το χαρακτήρα απόδρασης \n:

```
>> sprintf('%6.3f %5.0f \n',W)
```

```
ans =
 3.330  3330
 3.200  3200
 3.080  3080
 3.130  3130
```

Θα παρεμβάλουμε τώρα κείμενο μεταξύ των στηλών:

```
>> sprintf('The weight is %6.3f Kg or %5.0f gr \n',W)
ans =
The weight is  3.330 Kg or  3330 gr
The weight is  3.200 Kg or  3200 gr
The weight is  3.080 Kg or  3080 gr
The weight is  3.130 Kg or  3130 gr
```

Παράδειγμα 7.2.12

Θα τυπώσουμε τρείς αριθμούς με μια εντολή σε διαφορετικές γραμμές:

```
>> x=pi; y=x*x; z=x*y;
>> sprintf('Data 1: %6.3f \npi square:  %7.3f \npi cube: %7.2f',
x, y, z)
ans =
Data 1:  3.142
pi square:    9.870
pi cube:   31.01
```

Μπορούμε να στοιχίσουμε τους αριθμούς προσθέτοντας κενά διαστήματα ως εξής:

```
>> sprintf('Data 1:       %6.3f \npi square:  %7.3f \npi cube:
%7.2f', x, y, z)
ans =
Data 1:       3.142
pi square:    9.870
pi cube:     31.01
```

Παράδειγμα 7.2.13

Στο παράδειγμα που ακολουθεί τυπώνουμε μια απόστροφο καθώς και το σύμβολο %:

```
>> x=13.2;
>> sprintf('Gauss''s theorem gives: %7.3d%%', x)
ans =
Gauss's theorem gives: 1.320e+001%
```

Στην περίπτωση εκτύπωσης ενός διανύσματος ή ενός πίνακα η MATLAB επαναλαμβάνει το format μέχρι να τυπώσει όλα τα στοιχεία. Για παράδειγμα,

```
>> x=1:5
x =
     1     2     3     4     5
>> sprintf('%7.2f \n',x)
ans =
   1.00
   2.00
   3.00
   4.00
   5.00
```

Στην περίπτωση διδιάστατων πινάκων η εκτύπωση γίνεται κατά στήλες από τα αριστερά προς τα δεξιά:

```
>> A=[1 2 3; 4 5 6];
>> sprintf('%7.2f \n',A)
ans =
   1.00
   4.00
   2.00
   5.00
   3.00
   6.00
```

Αν θέλουμε να εκτυπώσουμε τα στοιχεία κατά γραμμές, μπορούμε να τυπώσουμε τον ανάστροφο πίνακα:

```
>> sprintf('%7.2f \n',A')
ans =
   1.00
   2.00
   3.00
   4.00
   5.00
   6.00
```

Γενικά, η εντολή sprintf συμπεριφέρεται όπως και η ομώνυμή της στη γλώσσα ANSI C με κάποιες εξαιρέσεις και επεκτάσεις. Για περισσότερες πληροφορίες δοκιμάστε τη βοήθεια της MATLAB (help sprintf).

7.3 Ανάγνωση από και γράψιμο σε αρχεία

7.3.1 Οι εντολές fopen και fclose

Για να διαβάσουμε δεδομένα από ένα αρχείο ή να γράψουμε δεδομένα σ' αυτό θα πρέπει να το ανοίξουμε με την εντολή **fopen**. Όταν ολοκληρώσουμε την εργασία μας με ένα αρχείο το κλείνουμε με την εντολή **fclose**.

Η εντολή fopen χρησιμοποιείται για το άνοιγμα αρχείων. Ο πιο απλός τρόπος για να ανοίξουμε ένα αρχείο με το όνομα *filename* είναι με την εντολή

fid = fopen(filename)

όπου fid ο **κωδικός αριθμός αρχείου** (file identifier). Υπάρχουν δύο κωδικοί αριθμοί αρχείων που είναι αυτόματα διαθέσιμοι και δεν χρειάζεται να ανοικτούν. Αυτοί είναι οι

- fid=1 (συνήθης εκτύπωση στην οθόνη), και
- fid=2 (συνήθη σφάλματα στην οθόνη)

Αν η fopen δεν μπορεί να ανοίξει ένα αρχείο, τότε επιστρέφει την τιμή -1.

Με την εντολή

fid = fopen(filename, permission)

μπορούμε να ορίσουμε τις επιτρεπόμενες δράσεις (άδειες) που μπορούμε να κάνουμε σ' ένα αρχείο. Για παράδειγμα η

nfile1 = fopen('Data1','r')

ανοίγει το αρχείο με όνομα Data1 για απλή ανάγνωση. Αυτή είναι και η προεπιλογή της MATLAB, δηλ. η πιο πάνω εντολή είναι ισοδύναμη με την

nfile1 = fopen('Data1')

Οι διάφορες άδειες που μπορούμε να χρησιμοποιήσουμε φαίνονται στον παρακάτω πίνακα:

Άδεια (permission)	**Περιγραφή**
'r'	Άνοιγμα αρχείου για ανάγνωση (προεπιλογή)
'w'	Άνοιγμα νέου ή υπάρχοντος αρχείου για γράψιμο με διαγραφή του αρχικού περιεχομένου
'a'	Άνοιγμα νέου ή υπάρχοντος αρχείου για γράψιμο (από το τέλος του αρχείου).
'r+'	Άνοιγμα αρχείου για ανάγνωση και γράψιμο.
'w+'	Άνοιγμα νέου ή υπάρχοντος αρχείου για ανάγνωση και γράψιμο με διαγραφή του αρχικού περιεχομένου
'a+'	Άνοιγμα νέου ή υπάρχοντος αρχείου για ανάγνωση και γράψιμο. Πρόσθεσε νέα δεδομένα στο τέλος του αρχείου.

Για να κλείσουμε ένα αρχείο με κωδικό αριθμό fid χρησιμοποιούμε την εντολή

fclose(fid)

Αν το fid δεν αντιστοιχεί σε ανοικτό αρχείο ή δεν είναι ίσο με 0 (συνήθης είσοδος), 1 (συνήθης εκτύπωση) ή 2 (συνήθη σφάλματα), η fclose επιστρέφει μήνυμα λάθους.

Για να κλείσουμε όλα τα ανοικτά αρχεία (εκτός από τα 0, 1 και 2= χρησιμοποιούμε την εντολή

fclose('all')

Η συνάρτηση feof

Η συνάρτηση **feof** ελέγχει αν έχουμε φτάσει στο τέλος ενός αρχείου. Η

st = feof(fid)

επιστρέφει 1 αν ο **δείκτης τέλους αρχείου** (end-of-file indicator) έχει οριστεί και 0 διαφορετικά. Ο δείκτης τέλους αρχείου ορίζεται όταν μια εντολή ανάγνωσης προσπαθεί να αναγνώσει μετά το τέλος του αρχείου με κωδικό fid.

7.3.2 Η εντολή fprintf

Η εντολή **fprintf** μοιάζει με την sprintf έχοντας ως μόνη διαφορά ότι γράφει τα δεδομένα σε αρχείο ή στην οθόνη και όχι σε αλφαριθμητικό. Με την εντολή

fprintf(fid, format, A, B, ….)

οι πίνακες A, B, … γράφονται στη μορφή που καθορίζεται από το αλφαριθμητικό format στο αρχείο με κωδικό αριθμό fid. Αν παραλειφθεί ο κωδικός ή fid = 1 τότε έχουμε τη συνήθη εκτύπωση στην οθόνη (προεπιλογή). Οι εντολές

fprintf(1, format, X, Y), **fprintf(format, X, Y)** και **sprintf(format, X, Y)**

είναι παρόμοιες. Αν fid = 2, τότε έχουμε εκτύπωση συνήθους σφάλματος. Αν το fid δεν αντιστοιχεί σε ανοικτό αρχείο, ή δεν είναι 0 (συνήθης είσοδος), 1 (συνήθης εκτύπωση) ή 2 (συνήθη σφάλματα), η fclose επιστρέφει μήνυμα λάθους.

Αν γράψουμε

count = fprintf(fid, format, A, B, ….)

τότε το count είναι το πλήθος των bytes που έχουν γραφεί με επιτυχία.

Παράδειγμα 7.3.1

Θα δούμε τις μικροδιαφορές μεταξύ των εντολών fprintf και sprinf. Έστω λοιπόν οι εξής εντολές:

```
>> x=rand
x =
    0.8913
>> fprintf(1,'The new random number is %7.4f',x)
The new random number is  0.8913>>
```

Παρατηρούμε ότι με την εκτέλεση της εντολής η MATLAB δεν αλλάζει γραμμή. Η προτροπή >> εμφανίζεται στο τέλος της εκτυπωθείσας γραμμής. Το ίδιο αποτέλεσμα παίρνουμε αν παραλείψουμε το πρώτο όρισμα (δηλ. το fid):

```
>> fprintf('The new random number is %7.4f',x)
The new random number is  0.8913>>
```

Αν θέλουμε να έχουμε αλλαγή γραμμής, πρέπει να αλλάξουμε το format ως εξής:

```
>> fprintf('The new random number is %7.4f \n',x)
The new random number is  0.8913
```

Ας δούμε τώρα τι παίρνουμε με την εντολή sprintf:

```
>> sprintf('The new random number is %7.4f',x)
ans =
The new random number is  0.8913
```

Παρατηρούμε ότι έχουμε αλλαγή γραμμής αλλά και εκτύπωση του ονόματος ans της πρόχειρης μεταβλητής που δημιουργήσαμε αφού με την εντολή sprintf ορίζουμε στην ουσία μια αλφαριθμητική μεταβλητή. Αυτό μπορούμε να το αποφύγουμε με την εντολή:

```
>> disp(sprintf('The new random number is %7.4f',x))
The new random number is  0.8913
```

Συμπερασματικά, οι εντολές

```
fprintf('The new random number is %7.4f \n',x)
```

και

```
disp(sprintf('The new random number is %7.4f',x))
```

είναι ισοδύναμες. Η πρώτη είναι κάπως πιο απλή και φαίνεται να είναι προτιμότερη.

Παράδειγμα 7.3.2

Θα γράψουμε στο αρχείο x2dat.txt και σε μορφή πίνακα τις τιμές της $f(x) = x^2$ για $x = 0, 0.5, \ldots, 4$:

```
>> x=0:0.5:4;
>> Y=[x; x.^2];
>> fid=fopen('x2dat.txt','w');
>> fprintf(fid,'%5.1f %8.3f \n',Y);
>> fclose(fid);
```

Ας δούμε τι περιέχει το αρχείο x2dat.txt:

```
>> fprintf('%5.1f %8.3f \n',Y);
  0.0    0.000
  0.5    0.250
  1.0    1.000
  1.5    2.250
  2.0    4.000
  2.5    6.250
  3.0    9.000
  3.5   12.250
  4.0   16.000
```

Για να δούμε πόσα bytes έχουν αποθηκευτεί επιτυχώς γράφουμε

```
>> count=fprintf('%5.1f %8.3f \n',Y)
count =
   144
```

το οποίο φυσικά μπορούσαμε επίσης να βρούμε γράφοντας

```
>> whos Y
```

```
  Name      Size            Bytes  Class     Attributes

  Y         2x9               144  double
```

7.3.3 Η εντολή fscanf

Με την εντολή **fscanf** μπορούμε να διαβάσουμε **μορφοποιημένα** (formatted) δεδομένα από ένα αρχείο. Η εντολή

A = fscanf(fid, format, size)

διαβάζει δεδομένα από το αρχείο με κωδικό fid, σύμφωνα με το αλφαριθμητικό format, και τα επιστρέφει στη μεταβλητή εξόδου A.

Οι διαστάσεις του A καθορίζονται στο προαιρετικό όρισμα size, το οποίο περιορίζει τον αριθμό των στοιχείων που θα διαβαστούν από το αρχείο. Αν δεν προσδιοριστεί τότε διαβάζεται ολόκληρο το αρχείο. Πιθανές επιλογές είναι οι:

- **N**: διάβασε το πολύ N στοιχεία σε διάνυσμα στήλης
- **inf**: διάβασε το πολύ μέχρι το τέλος του αρχείου
- **[M, N]**: διάβασε το πολύ M×N στοιχεία γεμίζοντας ένα M×N πίνακα κατά στήλη. Το N μπορεί να είναι inf αλλά όχι το M.

Αν το όρισμα size δεν είναι της μορφής [M, N] τότε το A είναι διάνυσμα στήλης.

Παράδειγμα 7.3.3

Το διάνυσμα A περιέχει τις γωνίες από 0 έως 180^o με βήμα 30^o. Θα τυπώσουμε το διάνυσμα αυτό καθώς και τις αντίστοιχες γωνίες σε ακτίνια στο αρχείο outp1.txt.

```
>> A=[0 30 60 90 120 150 180];
>> fid=fopen('outp1.txt','w');
>> fprintf(fid, '%g degrees = %g radians\n', [A;A*pi/180]);
>> fclose(fid);
```

Το αρχείο outp1.txt έχει ως εξής:

```
0 degrees = 0 radians
30 degrees = 0.523599 radians
60 degrees = 1.0472 radians
90 degrees = 1.5708 radians
120 degrees = 2.0944 radians
150 degrees = 2.61799 radians
180 degrees = 3.14159 radians
```

Για να διαβάσουμε τώρα το αρχείο outp1.txt γράφουμε τις εντολές:

```
>> fid=fopen('outp1.txt','r');
>> X=fscanf(fid, '%g degrees = %g radians\n')
X =
         0
         0
   30.0000
    0.5236
   60.0000
```

```
    1.0472
   90.0000
    1.5708
  120.0000
    2.0944
  150.0000
    2.6180
  180.0000
    3.1416
>> fclose(fid);
```

Τα δεδομένα αποθηκεύονται σε διάνυσμα. Μπορούμε να μετατρέψουμε το διάνυσμα σε ένα 7×2 πίνακα με την εντολή **reshape**:

```
>> X=reshape(X,2,7)'
X =
         0         0
   30.0000    0.5236
   60.0000    1.0472
   90.0000    1.5708
  120.0000    2.0944
  150.0000    2.6180
  180.0000    3.1416
```

Εναλλακτικά, θα μπορούσαμε να γράψουμε απευθείας

```
>> fid=fopen('outp1.txt','r');
>> X=fscanf(fid, '%g degrees = %g radians\n',[2 inf])'
X =
         0         0
   30.0000    0.5236
   60.0000    1.0472
   90.0000    1.5708
  120.0000    2.0944
  150.0000    2.6180
  180.0000    3.1416
```

Παρατηρούμε ότι το τελευταίο όρισμα [2 inf] προσδιορίζει τις διαστάσεις του πίνακα εξόδου, ο οποίος συμπληρώνεται κατά στήλες. Δώσαμε inf για το πλήθος στηλών, για να μπορούμε να έχουμε οποιοδήποτε πλήθος γραμμών στο αρχείο και χρησιμοποιήσαμε τον τόνο για να αναστρέψουμε τον πίνακα.

Παράδειγμα 7.3.4

Το αρχείο data1.txt περιέχει τα πιο κάτω δεδομένα:

```
0.0    0.000
1.0    1.100
2.0    1.182
3.0    2.41
4.0    2.8
```

Θα διαβάσουμε το αρχείο ως εξής:

```
>> fid=fopen('data1.txt');
>> X=fscanf(fid, '%g %g',[2 inf])'
X =
         0         0
    1.0000    1.1000
    2.0000    1.1820
    3.0000    2.4100
    4.0000    2.8000
```

Παράδειγμα 7.3.5

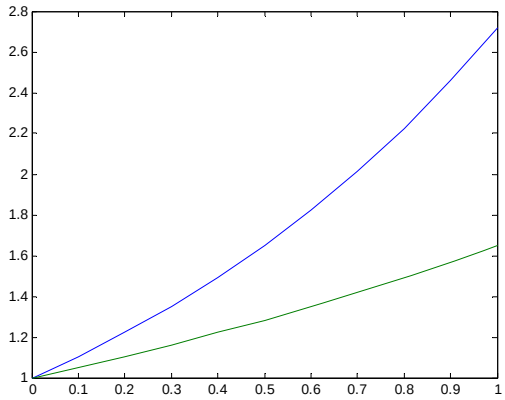
Το αρχείο data2.txt περιέχει στην $1^η$ στήλη τιμές του x και στη $2^η$ και $3^η$ στήλη τις τιμές των $y_1(x)$ και $y_2(x)$:

```
0.0  1.000000  1.000000
0.1  1.105171  1.051271
0.2  1.221403  1.105171
0.3  1.349859  1.161834
0.4  1.491825  1.221403
0.5  1.648721  1.284025
0.6  1.822119  1.349859
0.7  2.013753  1.419068
0.8  2.225541  1.491825
0.9  2.459603  1.568312
1.0  2.718282  1.648721
```

Θα διαβάσουμε το αρχείο και θα σχεδιάσουμε τα γραφήματα των y_1 και y_2, ως εξής:

```
>> fid=fopen('data2.txt');
>> X=fscanf(fid,'%g %g %g',[3 inf])'
X =
         0    1.0000    1.0000
    0.1000    1.1052    1.0513
    0.2000    1.2214    1.1052
    0.3000    1.3499    1.1618
    0.4000    1.4918    1.2214
    0.5000    1.6487    1.2840
    0.6000    1.8221    1.3499
    0.7000    2.0138    1.4191
    0.8000    2.2255    1.4918
    0.9000    2.4596    1.5683
    1.0000    2.7183    1.6487
>> plot(X(:,1), X(:,2), X(:,1), X(:,3))
```

Παίρνουμε έτσι το πιο κάτω γράφημα:



Μια άλλη ισχυρή εντολή για την ανάγνωση κειμένου από αρχείο είναι η **textscan** που χρησιμοποιεί διαφορετική σύνταξη για το format από την fscanf και επιστρέφει την έξοδο σε *πίνακα κελίων* (cell array). Για περισσότερες πληροφορίες χρησιμοποιείστε τη βοήθεια help textscan.

7.3.4 Οι εντολές fgetl και fgets

Η εντολή **fgetl** διαβάζει και επιστρέφει την επόμενη γραμμή από το αρχείο με κωδικό fid. Η δομή της εντολής έχει ως εξής:

tline = fgetl(fid)

όπου η tline είναι το αλφαριθμητικό που περιέχει τη γραμμή που διαβάστηκε.

Παράδειγμα 7.3.6

Το ακόλουθο function m-file με όνομα readany.m διαβάζει όλες τις γραμμές του αρχείου που δίνει ο χρήστης και τις τυπώνει στην οθόνη:

```
function []=readany()
% READANY
% Reads every line of the file infile
% provided by the user and prints it
% on the screen.
%
infile=input('Enter input file:  ','s')
fid=fopen(infile);
while 1
    tline=fgetl(fid);
    if ~ischar(tline), break, end
    disp(tline)
end
fclose(fid)
% End of READANY
```

Δοκιμάστε για παράδειγμα να διαβάσετε το αρχείο sin.m.

Η εντολή **fgets** είναι παρόμοια με την fgetl αφού επίσης διαβάζει και επιστρέφει την επόμενη γραμμή από το αρχείο με κωδικό fid. Η μόνη διαφορά είναι ότι η fgets επιστρέφει επίσης και τον χαρακτήρα αλλαγής γραμμής. Η δομή της έχει ως εξής:

tline = fgets(fid)

Αν θέλουμε να διαβάσουμε το πολύ nchar χαρακτήρες από την επόμενη γραμμή γράφουμε

tline = fgets(fid, nchar)

7.3.5 Οι εντολές fread και fwrite

Οι εντολές **fread** και **fwrite** χρησιμοποιούνται αντίστοιχα για την ανάγνωση από ή το γράψιμο σε **δυαδικά αρχεία** (binary files). Χρησιμοποιείστε τις σχετικές βοήθειες για περισσότερες πληροφορίες.

7.4 Ασκήσεις

7.1. Γράψτε ένα function m-file με όνομα drawyn.m που θα κάνει τα γραφήματα των συναρτήσεων Bessel δεύτερου είδους και τάξης από 0 έως 6, δηλ. τις Y_0, Y_1 έως Y_5 στο διάστημα [0, *b*] , όπου *b* μεταβλητή εισόδου. Μετά από κάθε γράφημα το πρόγραμμα θα δίνει χρόνο 8 δευτερολέπτων στον χρήστη πριν να δημιουργήσει το επόμενο.

7.2. Γράψτε στο αρχείο με όνομα ask1dat.txt και σε μορφή πίνακα τις τιμές της tan(*x*) για $x = 0, 0.1\pi, \ldots, \pi$.

7.3. Γράψτε στο αρχείο με όνομα ask2dat.txt και σε μορφή πίνακα τις τιμές των tan(*x*) και cot(*x*) για $x = 0, 0.1\pi, \ldots, \pi$.

7.4 Γράψτε στο αρχείο με όνομα x2dat.txt και σε μορφή πίνακα τις τιμές της $f(x) = x^2$ και της $g(x) = 2x$ για $x = 0, 0.5, \ldots, 4$.

7.5 Τροποποιήστε το m-file readany.m του παραδείγματος 7.3.5 έτσι ώστε να διαβάζεται και ο χαρακτήρας αλλαγής γραμμής. Δώστε το όνομα readany1.m στο νέο αρχείο. Τρέξτε τα readany.m και readany1.m και διαβάστε και στις δυο περιπτώσεις το readany.m. Τι παρατηρείτε;

7.6 Γράψτε ένα m-file με όνομα catfiles.m το οποίο θα μπορεί να διαβάζει μέχρι πέντε αρχεία της επιλογής του χρήστη και να τα τυπώνει όλα μαζί σε ένα νέο αρχείο.

8 ΕΠΙΛΥΣΗ ΜΗ ΓΡΑΜΜΙΚΩΝ ΕΞΙΣΩΣΕΩΝ

Στο παρόν κεφάλαιο θα ασχοληθούμε με μεθόδους επίλυσης εξισώσεων την μορφής

$$f(x) = 0.$$

Αναζητούμε μια ακολουθία $\{x_n\}_{n=0}^{\infty}$ προσεγγίσεων της λύσης, έτσι ώστε $\lim_{n\to\infty} x_n = \alpha$, με $f(\alpha) = 0$. Φυσικά, για κάποιο πεπερασμένο n, η τιμή x_n θα είναι μια προσέγγιση του α. Για τις μεθόδους που θα δούμε, θα χρειαστεί να ξέρουμε μια αρχική εκτίμηση του α, ή ένα διάστημα που την περιέχει.

8.1 Η μέθοδος της διχοτόμησης

Το m-file με το όνομα **bisection.m**, που φαίνεται πιο κάτω υλοποιεί τη λεγόμενη **μέθοδο της διχοτόμησης** (bisection method).

Οι μεταβλητές εισόδου είναι οι εξής:

- Η συνάρτηση fName, που αντιπροσωπεύει την $f(x)$ και είναι είτε συνάρτηση βιβλιοθήκης είτε συνάρτηση που έχει οριστεί ανώνυμα ή μέσω κάποιου m-file.
- Τα a και b είναι τα άκρα του αρχικού διαστήματος τέτοια ώστε

$$f(a)\, f(b) < 0$$

- Το delta είναι η ανοχή σφάλματος, έτσι ώστε το τελικό αποτέλεσμα που παίρνουμε να ικανοποιεί $|x_n - \alpha| \le \text{delta}$.

Η μεταβλητή εξόδου είναι η προσέγγιση της λύσης.

Σημειώνουμε επίσης ότι χρησιμοποιούμε τη μεταβλητή eps στο m-file, η οποία είναι η σχετική ακρίβεια της αριθμητικής κινητής υποδιαστολής που χρησιμοποιεί η MATLAB. Χρησιμοποιείται ως προεπιλεγμένη ανοχή (default tolerance). Όπως είδαμε, η τιμή της είναι

```
>> eps
ans =
    2.220446049250313e-016
```

bisection.m

```
function root=bisection(fName,a,b,delta)
%
% Metablhtes eisodou:
%    fname: onoma sunexous sunarthshs mias metablkhths f(x)
%    a, b : orizoun to diasthma [a,b] opou
```

```
%              f(a)*f(b) < 0
%    delta: anoxh (mh arnhjtikos pragmatikos)
%
% Metablhth exodou:
%    root: to meson tou diasthmatos [alpha, beta] me thn
%          idiothta
%             f(alpha)*f(beta) <= 0
%          kai
%               |beta-alpha| <= delta+eps*max(|alpha|, |beta|)
%
   fa = feval(fName,a);
   fb = feval(fName,b);
   while abs(a-b) > delta+eps*max(abs(a), abs(b))
       mid = (a+b)/2;
       fmid = feval(fName,mid);
       if fa*fmid <=0
           % uparxei riza sto [a,mid]
           b = mid;
           fb = fmid;
       else
           % uparxei riza sto [mid,b]
           a = mid;
           fa = fmid;
       end
   end
   root = (a+b)/2;
```

Παράδειγμα 8.1.1

Θεωρούμε την εξίσωση $\cos(x) = 0$ η οποία έχει ρίζα την $x = \pi/2 = 1.5707963267949$. Πιο κάτω φαίνονται τα αποτελέσματα που πήραμε σε διάφορες περιπτώσεις με την Bisection.

```
>> format long
>> root=bisection('cos',0,3,0.001)
root =
   1.57067871093750
>> error=root-pi/2
error =
   -1.176158573965580e-004

>> root=bisection('cos',0,3,0.00001)
root =
   1.57079601287842
>> error=root-pi/2
error =
   -3.139164785892490e-007

>> root=bisection('cos',0,3,0.00000001)
root =
   1.57079632859677
>> error=root-pi/2
error =
    1.801874205398235e-009
```

Παράδειγμα 8.1.2

Θεωρούμε τώρα την εξίσωση

$$f(x) = x^2 - x - 2$$

με (προφανείς) ρίζες τις $x_1 = -1$ και $x_2 = -2$. Επειδή η $f(x)$ δεν αντιστοιχεί σε συνάρτηση βιβλιοθήκης την ορίζουμε σαν ανώνυμη συνάρτηση:

```
>> f = @(x) x.^2 - x - 2;
```

Ακολουθούν αποτελέσματα που πήραμε σε διάφορες περιπτώσεις με την Bisection.

```
>> format long
>> root=bisection(f,0,4,0.000001)
root =
   1.99999952316284

>> error=2-root1
error =
    4.768371582031250e-007

>> root2=bisection(f,-3,1, 0.000001)
root2 =
  -1.000000476837158

>> error=-1-root2
error =
    4.768371582031250e-007
```

8.2. Η μέθοδος Newton

Το m-file με το όνομα **Newton1.m** επιλύει μια βαθμωτή εξίσωση της μορφής

$$f(x) = 0,$$

με τη **μέθοδο Newton**:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0,1,2,\ldots$$

Σημειώνουμε ότι πρέπει να ξέρουμε την παράγωγο της συνάρτησης όπως και μια αρχική εκτίμηση x_0. Επιπλέον, πρέπει να ισχύει $f'(x_k) \neq 0 \ \forall k$.

Οι μεταβλητές εισόδου της Newton1.m είναι:

- Η συνάρτηση fName που είναι είτε συνάρτηση βιβλιοθήκης, είτε function m-file είτε ανώνυμη συνάρτηση που ορίζεται από τον χρήστη. Η συνάρτηση fdName είναι η παράγωγος της συνάρτησης που ορίζεται στην fName, και πάλι πρέπει να οριστεί από τον χρήστη.
- Το x_0 είναι η αρχική εκτίμηση της ζητούμενης λύσης.
- Το delta είναι η ανοχή σφάλματος.
- Το Nmax είναι ο μέγιστος επιτρεπτός αριθμός επαναλήψεων.

Σημειώνεται ότι η Newton1.m τυπώνει σε κάθε επανάληψη τις τιμές των x_k και $f(x_k)$ με την εντολή **sprintf** που είδαμε στο προηγούμενο κεφάλαιο.

Newton1.m

```
function root=Newton1(fName,fdName,x0,delta,Nmax)
%
% Epilush mh grammikhs exiswshs f(x)=0  me th me0odo Newton:
%
%                     f(x_k)
%     x_k+1 = x_k - ---------
%                   f'(x_k)
%
% Metablhtes eisodou:
%    fName : onoma sunexous sunarthshs mias metablhths f(x)
%    fdName: h paragwgos df/dx ths f(x)
%    x0    : arxikh ektimhsh ths rizas
%    delta : anoxh (mh arnhjtikos pragmatikos)
%    Nmax  : megistos ari0mos epanalhyewn
%
% Metablhth exodou:
%    root: h riza pou upologizetai me th me0odo Newton
%
   xk=x0;
   fk=feval(fName,xk);
   dfk=feval(fdName,xk);
   disp('-----------------------------------------------')
   disp('   k              x_k              f(x_k)')
   disp('-----------------------------------------------')
   disp(sprintf(' %3.0f      %14.9f     %14.9f', 0, xk, fk))
   for k=1:Nmax
       xk1=xk-fk/dfk;
       dx=abs(xk1-xk);
       xk=xk1;
       fk=feval(fName,xk);
       dfk=feval(fdName,xk);
       disp(sprintf(' %3.0f      %14.9f     %14.9f', k, xk, fk))
       if dx < delta+eps
          disp('-----------------------------------------------')
          disp('Newton method has converged');
          root=xk;
          return
       end
   end
   disp('No convergence after Nmax iterations');
% Telos tou Newton1.m
```

Παράδειγμα 8.2.1

Για την εξίσωση

$$f(x) = x - \cos(x) = 0$$

ορίζουμε την f και την παράγωγο της σαν ανώνυμες συναρτήσεις:

```
>> fnewt = @(x) x - cos(x);
>> dfnewt = @(x) 1 + sin(x);
```

Καλώντας την Newton1.m παίρνουμε τα πιο κάτω αποτελέσματα:

```
>> Newton1('fnewt','dfnewt',1,0.000001,1000)
-------------------------------------------
   k              x_k              f(x_k)
-------------------------------------------
   0         1.000000000        0.459697694
   1         0.750363868        0.018923074
   2         0.739112891        0.000046456
   3         0.739085133        0.000000000
   4         0.739085133        0.000000000
-------------------------------------------
Newton method has converged
```

8.3 Η συνάρτηση fzero

Η συνάρτηση βιβλιοθήκης **fzero** προσπαθεί να βρει μια ρίζα της εξίσωσης

$$f(x) = 0$$

με αρχική εκτίμηση x_0. Η σύνταξή της είναι

fzero(fun, x0)

Η συνάρτηση fun μπορεί να είναι συνάρτηση βιβλιοθήκης, ανώνυμη συνάρτηση ή function m-file, π.χ.

fzero(@sin, 3)

ή

fzero('sin', 3)

ή ακόμα και

fzero(@(x) sin(3*x), 2)

Παράδειγμα 8.3.1
Για την εξίσωση

$$f(x) = x - \cos(x) = 0$$

ξέρουμε από το προηγούμενο παράδειγμα ότι η λύση της ανήκει στο διάστημα [0, 1], άρα, διαλέγουμε σαν αρχική εκτίμηση το 1/2 και έχουμε:

```
>> f = @(x) x - cos(x);
>> fzero(f,0.5)

ans =
   0.739085133215161
```

Παράδειγμα 8.3.2
Θεωρούμε την εξίσωση $\cos(x) = 0$ η οποία έχει ρίζα την $x = \pi/2 = 1.5707963267949$. Επομένως,

```
>> fzero('cos',1)

ans =
   1.570796326794897
```

8.4 Ασκήσεις

8.1 Θεωρείστε την εξίσωση

$$f(x) = e^{-ax} - \frac{x}{b}$$

όπου *a* και *b* το τρίτο και τέταρτο ψηφίο, αντίστοιχα, του ΑΜ σας. Χρησιμοποιώντας τη MATLAB:

(α) Ορίστε τη συνάρτηση $f(x)$ σαν ανώνυμη συνάρτηση και κατασκευάστε το γράφημα της έτσι ώστε να φαίνεται η μοναδική ρίζα της. Το γράφημα πρέπει να έχει λεζάντα και ετικέτες για τους άξονες.

(β) Επιλύστε την $f(x) = 0$ με τη μέθοδο της διχοτόμησης για διάφορες τιμές της ανοχής delta. Συγκρίνετε στη συνέχεια τα αποτελέσματά σας με το αποτέλεσμα που δίνει η εντολή **fzero**.

8.2 Θεωρούμε την εξίσωση

$$f(x) = \frac{4}{3} e^{2-x/2} \left(1 + \frac{\log x}{x}\right) = 0$$

Χρησιμοποιώντας τη MATLAB:

(α) Ορίστε τη συνάρτηση $f(x)$ και την παράγωγό της σαν ανώνυμες συναρτήσεις, και κατασκευάστε το γράφημα της $f(x)$ έτσι ώστε να φαίνεται η μοναδική ρίζα της. Το γράφημα πρέπει να έχει λεζάντα και ετικέτες για τους άξονες.

(β) Επιλύστε την $f(x) = 0$ με τη μέθοδο Newton για διάφορες τιμές της ανοχής delta και της αρχικής τιμής x_0. Για την τελευταία δοκιμάστε τιμές στα διαστήματα (0, 0.8), (0.8, 1.2) και (1.2, ∞). Συγκρίνετε στη συνέχεια τα αποτελέσματά σας με το αποτέλεσμα που δίνει η εντολή **fzero**.

8.3 Πριν 200 χρόνια ο Gauss ανάπτυξε μία μέθοδο για τον υπολογισμό των τροχιών ουρανίων σωμάτων, η οποία βασιζόταν στη γωνιακή θέση του σώματος που εκτελούσε την τροχιά. Ο Laplace ανέπτυξε παρόμοια μέθοδο με την οποία, όπως και με αυτή του Gauss, παίρνουμε ένα πολυώνυμο $8^{ου}$ βαθμού του οποίου η θετική ρίζα μας δίνει το ζητούμενο. Για παράδειγμα, αν εφαρμόσουμε τη μέθοδο του Gauss σε ένα διαστημόπλοιο που περιστρέφεται γύρω από τη Γη με ελλειπτική τροχιά που έχει μεγάλο άξονα 2.9 επί την ακτίνα της Γης, και εκκεντρότητα 0.3, τότε παίρνουμε την εξής εξίσωση:

$$r^8 - 0.9945225r^6 - 3.370698r^3 - 365.7847 = 0\,.$$

Η θετική ρίζα της πιο πάνω εξίσωσης χρησιμοποιείται για τον υπολογισμό των στοιχείων που καθορίζουν την τροχιά του διαστημοπλοίου.

Να κάνετε τη γραφική παράσταση του πιο πάνω πολυωνύμου για να βρείτε μια καλή αρχική εκτίμηση για την θετική του ρίζα. Στη συνέχεια, να βρείτε την θετική ρίζα του πολυωνύμου με ακρίβεια 10^{-6}, χρησιμοποιώντας όποια μέθοδο θέλετε.

9 ΕΠΙΛΥΣΗ ΓΡΑΜΜΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

9.1 Γενικά περί γραμμικών συστημάτων

Έχουμε ήδη δει ότι στη MATLAB για να λύσουμε ένα γραμμικό σύστημα $Ax = b$, με $A \in \mathbb{R}^{n \times n}$ και $b \in \mathbb{R}^n$ δεδομένα και με τον A αντιστρέψιμο, χρησιμοποιούμε την **αριστερή διαίρεση** '\'. Για παράδειγμα,

```
>> A=[1 2 3;4 5 6;7 8 0]
A =
     1     2     3
     4     5     6
     7     8     0
>> b=[1;2;3]
b =
     1
     2
     3
>> x=A\b
x =
   -0.3333
    0.6667
         0
```

Μια άλλη εντολή βιβλιοθήκης που μας δίνει τόσο τη λύση όσο και τον ανηγμένο κλιμακωτό του πίνακα είναι η **rref**. Γενικά, η

rref(A)

μας δίνει την **ανηγμένη κλιμακωτή (κατά γραμμές) μορφή** του πίνακα A (*row reduced echelon form*) ο οποίος προκύπτει με εφαρμογή πεπερασμένου πλήθους στοιχειωδών μετασχηματισμών γραμμών πάνω στον A με τη μέθοδο απαλοιφής Gauss-Jordan.

Αν χρησιμοποιήσουμε τη συνάρτηση rref για τον επαυξημένο πίνακα $[A \mid b]$ τότε παίρνουμε στη γενική περίπτωση την ανηγμένη κλιμακωτή μορφή $[R \mid s]$. Στην περίπτωση που είναι $R = I$ τότε στα δεξιά έχουμε τη μοναδική λύση του συστήματος, $x = s$. Άρα, για να λύσουμε το γραμμικό σύστημα του προηγούμενου παραδείγματος, γράφουμε

```
>> rref([A b])
ans =
```

```
    1.0000         0         0   -0.3333
         0    1.0000         0    0.6667
         0         0    1.0000         0
```

που μας δίνει τη λύση ως την τελευταία στήλη του αποτελέσματος. Αυτή η εντολή είναι ιδιαίτερα χρήσιμη όταν ο πίνακας του συστήματος δεν είναι αντιστρέψιμος (άρα παίρνουμε άπειρες ή 0 λύσεις) ή όταν το σύστημα δεν είναι τετραγωνικό.

Παράδειγμα 9.1.1

Θεωρούμε το εξής γραμμικό σύστημα:

$$x_1 + x_2 = 2$$
$$2x_1 + 2x_2 = 3$$

Ορίζουμε

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

και γράφουμε

```
>> rref([A b])
ans =
     1     1     0
     0     0     1
```

που σημαίνει ότι μετά από την απαλοιφή παίρνουμε $x_1 + x_2 = 0$ και $0 \cdot x_1 + 0 \cdot x_2 = 1$, δηλαδή το σύστημα είναι ασυμβίβαστο και δεν υπάρχει λύση.

Παράδειγμα 9.1.2

Θεωρούμε το εξής γραμμικό σύστημα:

$$x_1 - 2x_2 + x_3 = 1$$
$$2x_1 - x_2 + x_3 = 2$$

Ορίζουμε

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

και γράφουμε

```
>> format rat
>> rref([A b])
ans =
```

```
1          0          1/3          1
0          1          -1/3         0
```

Άρα

$$x_1 + \frac{1}{3}x_3 = 1 \quad \text{και} \quad x_1 - \frac{1}{3}x_2 = 0\,.$$

Εδώ έχουμε άπειρο πλήθος λύσεων. Θέτοντας $x_3 = s$, βρίσκουμε

$$x_1 = 1 - s/3, \qquad x_2 = s/3, \qquad x_3 = s\,, \qquad s \in \mathbf{R}\,.$$

9.2 Απαλοιφή Gauss

Στη γενική περίπτωση, ο στόχος της απαλοιφής Gauss είναι η μετατροπή ενός γραμμικού συστήματος σε ένα ισοδύναμο τριγωνικό σύστημα το οποίο επιλύεται εύκολα αφού δεν απαιτεί την εφαρμογή οποιουδήποτε μετασχηματισμού στον πίνακα των συντελεστών. Ως εκ τούτου, θα επικεντρωθούμε πρώτα σε τριγωνικά συστήματα.

Κάτω τριγωνικά συστήματα και εμπρός αντικατάσταση

Θεωρούμε σαν πρώτο παράδειγμα το 3×3 κάτω τριγωνικό σύστημα:

$$\begin{bmatrix} a_{11} & 0 & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Οι άγνωστοι μπορούν να υπολογιστούν ως εξής:

$$\begin{aligned} x_1 &= b_1 / a_{11} \\ x_2 &= (b_2 - a_{21}x_1) / a_{22} \\ x_3 &= (b_3 - a_{31}x_1 - a_{32}x_2) / a_{33} \end{aligned}$$

Ο αλγόριθμος που χρησιμοποιήσαμε είναι γνωστός ως **εμπρός αντικατάσταση** (forward substitution). Είναι φανερό ότι η διαδικασία εφαρμόζεται μόνο όταν ο A είναι μη ιδιάζων, δηλ. όταν η $\det A = a_{11}\,a_{22}\,a_{33}$ είναι μη μηδενική.

Στη γενική περίπτωση γραμμικού συστήματος της μορφής

$$Lx = b$$

όπου ο L είναι **κάτω τριγωνικός**, η i-οστή εξίσωση

$$\ell_{i1}x_1 + \cdots + \ell_{ii}x_i = b_i$$

επιλύεται ως προς τον άγνωστο x_i ως εξής:

$$x_i = \left(b_i - \sum_{j=1}^{i-1} \ell_{ij}x_j \right) / \ell_{ii}$$

Ο υπολογισμός του x_i απαιτεί περίπου $2i$ πράξεις, οπότε όλη η διαδικασία απαιτεί περίπου

$$2(1+2+\cdots+n) \approx n^2$$

πράξεις.

Παράδειγμα 9.2.1

Το κάτωθι function m-file με το όνομα LTriSol.m επιλύει ένα κάτω τριγωνικό γραμμικό σύστημα με εμπρός αντικατάσταση.

```
function x = LTriSol(L,b)
%
% Input:
%   L   n-by-n nonsingular lower triangular matrix
%   b   n-by-1 vector
%
% Output:
%   x   Solution of the system Lx=b
%
n = length(b);
x = zeros(n,1);
for j=1:n-1
    x(j) = b(j)/L(j,j);
    b(j+1:n) = b(j+1:n) - x(j)*L(j+1:n,j);
end
x(n) = b(n)/L(n,n);

% End of LTriSol.m
```

Για το σύστημα

$$\begin{bmatrix} 2 & 0 & 0 \\ 1 & 5 & 0 \\ 7 & 9 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 6 \\ 2 \\ 5 \end{bmatrix}$$

πήραμε τα πιο κάτω αποτελέσματα:

```
>> A=[ 2 0 0;1 5 0; 7 9 8 ];
>> b = [ 6; 2; 5];
>> x=LTriSol(A,b)
x =
    3.0000
   -0.2000
   -1.7750
```

Άνω τριγωνικά συστήματα και πίσω αντικατάσταση

Η επίλυση ενός άνω τριγωνικού συστήματος είναι εντελώς ανάλογη. Η μόνη διαφορά έγκειται στο ότι ο υπολογισμός των αγνώστων γίνεται προς τα πίσω. Έτσι για να επιλύσουμε το σύστημα

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

εργαζόμαστε από κάτω προς τα πάνω:

$$\begin{aligned} x_3 &= b_3 / u_{33} \\ x_2 &= (b_2 - u_{23}x_3) / u_{22} \\ x_1 &= (b_3 - u_{12}x_2 - u_{13}x_3) / u_{11} \end{aligned}$$

Η διαδικασία αυτή είναι γνωστή ως **πίσω αντικατάσταση** (back substitution).

Παράδειγμα 9.2.2

Το κάτωθι function m-file με το όνομα UTriSol.m επιλύει ένα άνω τριγωνικό γραμμικό σύστημα με πίσω αντικατάσταση.

```
function x = UTriSol(U,b)
%
% Input:
%   U   n-by-n nonsingular upper triangular matrix
%   b   n-by-1 vector
%
% Output:
%   x   Solution of the system Ux=b
%
n = length(b);
x = zeros(n,1);
for j=n:-1:2
    x(j) = b(j)/U(j,j);
    b(1:j-1) = b(1:j-1) - x(j)*U(1:j-1,j);
end
x(1) = b(1)/U(1,1);

%End of UTriSol.m
```

Για το σύστημα

$$\begin{bmatrix} 6 & 7 & 8 \\ 0 & 3 & 4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 9 \\ 5 \\ 2 \end{bmatrix}$$

πήραμε τα πιο κάτω αποτελέσματα:

```
>> U=[ 6 7 8
       0 3 4
       0 0 1];

>> b=[9; 5; 2];

>> x=UTriSol(U,b)
x =
     0
    -1
     2
```

9.2.1 Παραγοντοποίηση *LU*

Θεωρούμε το γραμμικό σύστημα

$$A\,x = b.$$

Με την **παραγοντοποίηση *LU*** (LU factorization) βρίσκουμε ένα κάτω τριγωνικό πίνακα L και ένα άνω τριγωνικό πίνακα U έτσι ώστε

$$P\,A = L\,U.$$

Ο L έχει στη διαγώνιο μονάδες και στις άλλες μη μηδενικές θέσεις τους πολλαπλασιαστές της απαλοιφής Gauss (που χρησιμοποιήθηκαν για την απαλοιφή κάθε στοιχείου). Ο U είναι ο άνω τριγωνικός πίνακας που προκύπτει με την απαλοιφή Gauss (δηλ. ο κλιμακωτός πίνακας). Τέλος, ο πίνακας P είναι ο πίνακας μεταθέσεων που αντιστοιχεί στις εναλλαγές γραμμών που έγιναν κατά τη διαδικασία της απαλοιφής. Είναι φανερό ότι στην περίπτωση που δεν γίνονται εναλλαγές γραμμών ισχύει $P = I$ και έτσι

$$A = L\,U.$$

Με την εύρεση των L, U και P η επίλυση του συστήματος $Ax = b$ ανάγεται στην επίλυση των κάτωθι (αντίστοιχα κάτω και άνω) τριγωνικών συστημάτων:

$$L\,y = P\,b$$

$$U\,x = y$$

Η συνάρτηση lu της MATLAB

Στη MATLAB η παραγοντοποίηση LU πραγματοποιείται απλά με τη συνάρτηση βιβλιοθήκης **lu**. Η μορφή της είναι:

[L, U, P] = lu(A)

όπου η σημασία των μεταβλητών εισόδου και εξόδου είναι προφανής.

Παράδειγμα 9.2.3

Θεωρούμε τον πίνακα

$$A = \begin{bmatrix} 4 & 2 & 0 \\ 2 & 3 & 1 \\ 0 & 1 & 5/2 \end{bmatrix}$$

ο οποίος παραγοντοποιείται **χωρίς εναλλαγές γραμμών** ως εξής:

$$A = LU = \begin{bmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ 0 & 1/2 & 1 \end{bmatrix} \begin{bmatrix} 4 & 2 & 0 \\ 0 & 2 & 1 \\ 0 & 0 & 2 \end{bmatrix}$$

Η πιο πάνω παραγοντοποίηση επαληθεύεται εύκολα στη MATLAB:

```
A=[ 4 2 0
    2 3 1
    0 1 5/2 ];
```

```
>> [L,U,P]=lu(A)
L =
    1.0000         0         0
    0.5000    1.0000         0
         0    0.5000    1.0000

U =
     4     2     0
     0     2     1
     0     0     2

P =
     1     0     0
     0     1     0
     0     0     1
```

Παρατηρούμε ότι $P = I$ αφού δεν έχουμε εναλλαγές γραμμών. Αν γράψουμε

D = lu(A)

τότε ο D περιέχει τους L και U σε συμπυκνωμένη μορφή:

```
>> D=lu(A)
D =
    4.0000    2.0000         0
    0.5000    2.0000    1.0000
         0    0.5000    2.0000
```

Αυτή είναι ακριβώς η μορφή που παίρνουμε όταν κάνουμε την παραγοντοποίηση LU με το χέρι (γράφουμε τους αντίστοιχους πολλαπλασιαστές Gauss στις θέσεις των μηδενικών που δημιουργούμε). Μπορείτε να βρείτε περισσότερες πληροφορίες με την εντολή **help lu**.

Παράδειγμα 9.2.4

Θεωρούμε τον πίνακα

$$A = \begin{bmatrix} 0 & 1 & 2 \\ 2 & -2 & 1 \\ 5 & 3 & 1 \end{bmatrix}$$

Κατά την παραγοντοποίηση LU απαιτείται τουλάχιστον μια εναλλαγή γραμμών αφού $a_{11} = 0$. Πιο κάτω φαίνονται τα αποτελέσματα που έδωσε η MATLAB.

```
>> A=[ 0 1 2
       2 -2 1
       5 3 1];
>> [L,U,P] = lu (A)
L =
    1.0000         0         0
    0.4000    1.0000         0
         0   -0.3125    1.0000

U =
    5.0000    3.0000    1.0000
         0   -3.2000    0.6000
         0         0    2.1875
```

```
P =
     0     0     1
     0     1     0
     1     0     0
```

9.2.2 Η μέθοδος Gauss

Παράδειγμα 9.2.5 gaussel.m

Το πιο κάτω m-file υλοποιεί την απαλοιφή Gauss (με πίσω αντικατάσταση) για τη εξεύρεση της λύσης ενός γραμμικού συστήματος $A\,x = b$.

```
function [x] = gaussel(A,b)
% [x] = gaussel(A,b)
%
% Luvei to grammiko susthma Ax = b me apaloifh Gauss kai
% pisw avtikatastash.
%
% Dedomeva eisodou : O pivakas A kai to diavusma b
%
% Dedomeva eksodou : To diavusma (lush) x

N = max(size(A));

% Apaloifh Gauss

for j=2:N
    for i=j:N
        m = A(i,j-1)/A(j-1,j-1);
        A(i,:) = A(i,:) - A(j-1,:)*m;
        b(i) = b(i) - m*b(j-1);
    end
end

% Pisw avtikatastash
x = zeros(N,1);
x(N) = b(N)/A(N,N);
for j=N-1:-1:1
    x(j) = (b(j)-A(j,j+1:N)*x(j+1:N))/A(j,j);
end

% Telos tou gaussel.m
```

Ας τρέξουμε το πιο πάνω m-file για το σύστημα

$$\begin{aligned} 4x_1 + 3x_2 + 2x_3 + 3x_4 &= 1 \\ x_1 + 2x_2 + 3x_3 + 6x_4 &= 0 \\ 4x_1 + 2x_2 + 2x_3 + x_4 &= 2 \\ 9x_1 + 9x_2 + x_3 - 2x_4 &= -5 \end{aligned}$$

```
>> A = [4 3 2 3;1 2 3 6;4 2 2 1;9 9 1 -2]
A =
     4     3     2     3
     1     2     3     6
     4     2     2     1
     9     9     1    -2

>> b=[1;0;2;-5]
```

```
b =
     1
     0
     2
    -5
```

Μια και το m-file δεν ελέγχει αν ο πίνακας είναι αντιστρέψιμος, θα το κάνουμε εμείς:

```
>> det(A)
ans =
   -94
```

Καλούμε τώρα τη gaussel.m και παίρνουμε

```
>> gaussel(A,b)
ans =
   1.297872340425532
  -1.765957446808510
  -0.021276595744681
   0.382978723404255
```

Ελέγχουμε την απάντηση μας με την ενσωματωμένη (και καταλληλότερη) εντολή επίλυσης γραμμικών συστημάτων:

```
>> A\b
ans =
   1.297872340425532
  -1.765957446808511
  -0.021276595744681
   0.382978723404255
```

9.3 Παραγοντοποίηση Cholesky

Γνωρίζουμε από τη θεωρία ότι κάθε **συμμετρικός θετικά ορισμένος πίνακας** A **παραγοντοποιείται κατά Cholesky** ως εξής:

$$A = LL^T$$

όπου L κάτω τριγωνικός πίνακας με θετικά διαγώνια στοιχεία. Έτσι η επίλυση του γραμμικού συστήματος

$$Ax = b$$

όπου A συμμετρικός θετικά ορισμένος πίνακας ανάγεται στην επίλυση του κάτω τριγωνικού συστήματος

$$Ly = b$$

με *εμπρός αντικατάσταση* και του άνω τριγωνικού συστήματος

$$L^T x = y$$

με *πίσω αντικατάσταση*.

Η συνάρτηση chol της MATLAB

Η συνάρτηση **chol** βρίσκει την παραγοντοποίηση Cholesky ενός συμμετρικού θετικά ορισμένου πίνακα. Μπορούμε να την καλέσουμε με δύο τρόπους:

i) **R = chol(A)**

Εφόσον ο A είναι συμμετρικός θετικά ορισμένος ο R είναι ο άνω τριγωνικός πίνακας της παραγοντοποίησης Cholesky, δηλ.

$$R = L^T$$

και έτσι

$$A = R^T R$$

Σημειώνουμε ότι η MATLAB διαβάζει μόνο το άνω τριγωνικό μέρος του A και θεωρεί ότι ο πίνακας είναι συμμετρικός (ή ερμιτιανός στην περίπτωση μιγαδικού πίνακα).

Αν ο πίνακας A δεν είναι θετικά ορισμένος (για την ακρίβεια αν το άνω τριγωνικό του μέρος δεν αντιστοιχεί σε συμμετρικό θετικά ορισμένο πίνακα) η MATLAB επιστρέφει το ακόλουθο μήνυμα σφάλματος:

??? Error using ==> chol
Matrix must be positive definite.

ii) **[R, p]= chol(A)**

Αν το άνω τριγωνικό μέρος του A αντιστοιχεί σε συμμετρικό θετικά ορισμένο πίνακα, η MATLAB επιστρέφει τον R και **p = 0**. Διαφορετικά ο R είναι κενός και **p = 1** (ή γενικά θετικός ακέραιος). Με τη μορφή αυτή η MATLAB δεν επιστρέφει μήνυμα σφάλματος.

Είναι φανερό ότι με τους πιο πάνω συμβολισμούς το γραμμικό σύστημα

$$Ax = b$$

μπορεί να αντικατασταθεί από το

$$R^T Rx = b$$

το οποίο επιλύεται εύκολα στη MATLAB με την εντολή

```
x = R\ (R'\b)
```

Παράδειγμα 9.3.1

Θεωρούμε τον συμμετρικό θετικά ορισμένο πίνακα

$$A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

και τους πίνακες

$$B=\begin{bmatrix}2 & -1 & 0\\ 0 & 2 & -1\\ 0 & 0 & 2\end{bmatrix},\qquad C=\begin{bmatrix}-2 & -1 & 0\\ -1 & 2 & -1\\ 0 & -1 & 2\end{bmatrix}$$

Παρατηρούμε ότι ο B είναι άνω τριγωνικός πίνακας που προκύπτει από τον A αν μηδενίσουμε τα στοιχεία του δεύτερου που βρίσκονται κάτω από τη διαγώνιο. Είναι φανερό ότι ο B δεν είναι συμμετρικός. Σύμφωνα με αυτά που αναφέραμε πιο πάνω η συνάρτηση **chol** δεν κάνει διάκριση μεταξύ των A και B, αφού τα άνω τριγωνικά τους μέρη ταυτίζονται. Τέλος ο πίνακας C είναι συμμετρικός αλλά όχι θετικά ορισμένος. Ακολουθούν τα αποτελέσματα που δίνει η MATLAB για τους τρεις πίνακες.

```
>> A = [2,-1,0;-1,2,-1;0,-1,2]
A =
     2    -1     0
    -1     2    -1
     0    -1     2

>> chol(A)

ans =
    1.4142   -0.7071         0
         0    1.2247   -0.8165
         0         0    1.1547
>> [R,p]=chol(A)

R =
    1.4142   -0.7071         0
         0    1.2247   -0.8165
         0         0    1.1547

p =
     0

>> B = [2,-1,0;0,2,-1;0,0,2]

B =
     2    -1     0
     0     2    -1
     0     0     2

>> chol(B)
ans =
    1.4142   -0.7071         0
         0    1.2247   -0.8165
         0         0    1.1547

>> [R,p]=chol(B)

R =
    1.4142   -0.7071         0
         0    1.2247   -0.8165
         0         0    1.1547

p =
     0

>> C = [-2,-1,0;-1,2,-1;0,-1,2]

C =
    -2    -1     0
    -1     2    -1
     0    -1     2
```

```
>> chol(C)
??? Error using ==> chol
Matrix must be positive definite.
>> [R,p]=chol(C)
R =
     []
p =
     1
```

Παράδειγμα 9.3.2

Οι **πίνακες Pascal** αποτελούν ένα ενδιαφέρον παράδειγμα συμμετρικών θετικά ορισμένων πινάκων. Τα στοιχεία ενός πίνακα Pascal είναι οι διωνυμικοί συντελεστές οι οποίοι εμφανίζονται στο περίφημο **τρίγωνο του Pascal**. Για παράδειγμα ο 4×4 πίνακας Pascal είναι ο

$$P = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 3 & 6 & 10 \\ 1 & 4 & 10 & 20 \end{bmatrix}$$

Παρατηρούμε ότι στην πρώτη γραμμή (άρα και στην πρώτη στήλη) έχουμε μονάδες και ότι κάθε άλλο στοιχείο είναι το άθροισμα των στοιχείων που βρίσκονται αριστερά και πάνω από αυτό. Αποδεικνύεται ότι ο πίνακας L της παραγοντοποίησης Cholesky έχει επίσης ως στοιχεία τους διωνυμικούς συντελεστές. Έτσι για τον P βρίσκουμε

$$L^T = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Η MATLAB κατασκευάζει τον $n \times n$ πίνακα Pascal με τη συνάρτηση **pascal**. Έτσι μπορούμε να επαληθεύσουμε εύκολα το πιο πάνω αποτέλεσμα:

```
>> A=pascal(4)
A =
     1     1     1     1
     1     2     3     4
     1     3     6    10
     1     4    10    20
>> [R,p]=chol(A)
R =
     1     1     1     1
     0     1     2     3
     0     0     1     3
     0     0     0     1
p =
     0
```

9.4 Νόρμες διανυσμάτων και πινάκων

9.4.1 Νόρμες διανυσμάτων

Η p-νόρμα ($1 \le p < \infty$) διανύσματος $x \in \mathbb{R}^n$ ορίζεται ως εξής:

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

Στην πράξη χρησιμοποιούνται συνήθως οι 1-, η 2- και η ∞-νόρμα για τις οποίες ισχύουν:

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|, \qquad \|x\|_2 = \left(\sum_{i=1}^{n} |x_i|^2 \right)^{1/2}, \qquad \|x\|_\infty = \max_{1 \le i \le n} |x_i|$$

Η συνάρτηση norm της MATLAB

Στη MATLAB η p-νόρμα ενός **διανύσματος** x υπολογίζεται με τη συνάρτηση

norm(x,p)

Όλες οι τιμές $1 \le p < \infty$ είναι αποδεκτές. Όπως θα δούμε στη συνέχεια **δεν ισχύει το ίδιο και για τις νόρμες πίνακα.** Επιπλέον υπάρχουν οι εξής επιλογές:

- **norm(x, inf)**: υπολογίζει την $\|x\|_\infty$.
- **norm(x,-inf)**: υπολογίζει την $\|x\|_{-\infty}$ η οποία ορίζεται ως εξής:
$$\|x\|_{-\infty} = \min_{1 \le i \le n} |x_i|$$
- **norm(x, 'fro')**: υπολογίζει τη **διανυσματική νόρμα Frobenius** η οποία ορίζεται από την
$$\|x\|_F = \sqrt{tr(x^T x)}$$

Αν δεν προσδιορίσουμε το p, η MATLAB υπολογίζει τη 2-νόρμα. Έτσι οι **norm(x, 2)** και **norm(x)** είναι ισοδύναμες.

Παράδειγμα 9.4.1

Θα υπολογίσουμε την 1-, 2-, 4- και την ∞-νόρμα του x = [1, 2, 3, 4] με τη MATLAB.

```
>> x=[1 2 3 4];
>> norm(x,1)
ans =
    10
>> norm(x)
ans =
    5.4772
>> norm(x,2)
ans =
    5.4772
```

```
>> norm(x,4)
ans =
    4.3376
>> norm(x,inf)
ans =
     4
```

Παράδειγμα 9.4.2

Θα υπολογίσουμε την 1-, 2- και την ∞-νόρμα του διανύσματος x = [0.2, 0.4, 0.6, 0.8] με τη MATLAB.

```
>> x=(1:4)/5;
>> norm1=norm(x,1)
norm1 =
     2
>> norm2=norm(x)
norm2 =
    1.0954
>> norminf=norm(x,inf)
norminf =
    0.8000
```

9.4.2 Νόρμες πινάκων

Η p-νόρμα του πίνακα $A \in \mathbb{R}^{n\times n}$ ορίζεται ως εξής:

$$\|A\|_p = \sup_{x\neq 0} \frac{\|Ax\|_p}{\|x\|_p}.$$

Στην πράξη χρησιμοποιούνται συνήθως οι 1-, η 2- και η ∞-νόρμα για τις οποίες ισχύουν:

$$\|A\|_1 = \max_{1\le j\le n} \sum_{i=1}^{n} |a_{ij}|, \qquad \|A\|_2 = \sqrt{r_\sigma\left(A^T A\right)}, \qquad \|A\|_\infty = \max_{1\le i\le n} \sum_{j=1}^{n} |a_{ij}|$$

όπου r_σ η **φασματική ακτίνα** (δηλ. η μεγαλύτερη σε απόλυτη τιμή ιδιοτιμή ενός πίνακα).

Στη MATLAB η p-νόρμα ενός πίνακα υπολογίζεται με τη συνάρτηση **norm** που είδαμε πιο πάνω:

norm(A, p)

Σε αντίθεση με τις διανυσματικές νόρμες, **το *p* μπορεί να πάρει μόνο τις τιμές 1, 2, inf και 'fro'.** Η τελευταία επιλογή μας δίνει τη **νόρμα Frobenius** (για πίνακα):

$$\|A\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} |a_{ij}|^2}.$$

Όπως και στις διανυσματικές νόρμες οι **norm(A, 2)** και **norm(A)** είναι ισοδύναμες.

Παρατήρηση: Για κάθε p-νόρμα του $n \times n$ ταυτοτικού πίνακα ισχύει $\|I\|_p = 1$ ενώ για τη νόρμα Frobenius έχουμε $\|I\|_F = \sqrt{n}$.

Η MATLAB διαθέτει και άλλες συναρτήσεις για την εκτίμηση της νόρμας πίνακα (ειδικά της 2-νόρμας ο υπολογισμός της οποίας είναι χρονοβόρος):

- **normest(A):** Μας δίνει μια **εκτίμηση** της 2-νόρμας του A. Προορίζεται κυρίως για αραιούς πίνακες αλλά δίνει καλά αποτελέσματα και για μεγάλους πυκνούς πίνακες.
- **normest1(A):** Μας δίνει μια **εκτίμηση** της 1-νόρμας του A.

Παράδειγμα 9.4.3

Θα υπολογίσουμε με τη MATLAB την 1-, 2- και την ∞-νόρμα του πίνακα:

$$A = \begin{bmatrix} 2 & -1 & 4 \\ -2 & 3 & -1 \\ 6 & -1 & 4 \end{bmatrix}$$

Ακολουθούν τα αποτελέσματα:

```
>> A= [ 2 -1 4;-2 3  -1;6 -1 4]
A =
     2    -1     4
    -2     3    -1
     6    -1     4
>> norm(A,1)
ans =
    10
>> norm(A,inf)
ans =
    11
>> norm(A',inf)
ans =
    10
>> format long
>> norm(A,2)
ans =
    8.814112199862889
```

Ας δούμε τι δίνει η εντολή **normest(A):**

```
>> normest(A)
ans =
   8.814112195524853
```

Η εκτίμηση που παίρνουμε είναι πράγματι πολύ καλή.

9.5 Δείκτης κατάστασης αντιστρέψιμου πίνακα

Ο **δείκτης κατάστασης** (condition number) ενός αντιστρέψιμου πίνακα A ορίζεται ως εξής:

$$\kappa_p(A) = \|A\|_p \left\|A^{-1}\right\|_p$$

Αποδεικνύεται ότι

$$\kappa_p(A) \geq 1$$

αν η χρησιμοποιούμενη νόρμα πίνακα είναι **φυσική**. Αν $\kappa_p(A) >> 1$ λέμε ότι ο A είναι **κακής κατάστασης** (ill or badly conditioned). Διαφορετικά λέμε ότι ο A είναι **καλής κατάστασης** (well conditioned).

Η συνάρτηση cond της MATLAB

Στη MATLAB δείκτης κατάστασης πίνακα A υπολογίζεται με τη συνάρτηση

cond(A, p)

η οποία είναι ισοδύναμη με την εντολή

norm(A, p)* norm(inv(A), p)

Έτσι το *p* μπορεί να πάρει μόνο τις επιτρεπόμενες τιμές για νόρμα πίνακα στη MATLAB: 1, 2, inf και 'fro'. Αν δεν προσδιορίσουμε το p, η MATLAB θεωρεί ότι p = 2 (προεπιλεγμένη τιμή). Έτσι οι **cond(A, 2)** και **cond(A)** μας δίνουν το ίδιο αποτέλεσμα. Για τον κ_∞ γράφουμε **cond(A, inf)**.

Η **cond(A, 2)** υπολογίζεται πιο δύσκολα από τις **cond(A, 1)** και **cond(A, inf)** (αφού περιλαμβάνει πολλαπλασιασμό πινάκων και την εύρεση των ιδιοτιμών του A^TA). Χρησιμοποιείται συνήθως για μικρούς πίνακες ή όταν η $\|\cdot\|_2$ είναι απαραίτητη.

Η MATLAB διαθέτει και άλλες συναρτήσεις για την εκτίμηση του δείκτη κατάστασης:

- **condest(A):** Μας δίνει μια **εκτίμηση** (για την ακρίβεια ένα κάτω φράγμα) του $\kappa_1(A)$. Χρησιμοποιεί την **lu(A)** και είναι κατάλληλη για μεγάλους αραιούς πίνακες.
- **rcond(A):** Μας δίνει μια **εκτίμηση** του $1/\kappa_1(A)$. Χρησιμοποιεί την **lu(A)** και ένα παλαιό αλγόριθμο που αναπτύχθηκε στα προγράμματα LINPACK και LAPACK. Έχει καθαρά ιστορική αξία.

Παράδειγμα 9.5.1

Οι **πίνακες Hilbert** είναι συμμετρικοί πίνακες με γενικό στοιχείο της μορφής

$$h_{ij} = \frac{1}{i+j-1}$$

Έτσι,

$$H_4 = \begin{bmatrix} \frac{1}{1} & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \end{bmatrix}$$

Η MATLAB βρίσκει τον $n \times n$ πίνακα Hilbert με τη συνάρτηση **hilb(n)**. Για παράδειγμα,

```
>> H4=hilb(4)
H4 =
    1.0000    0.5000    0.3333    0.2500
    0.5000    0.3333    0.2500    0.2000
    0.3333    0.2500    0.2000    0.1667
    0.2500    0.2000    0.1667    0.1429
```

Οι πίνακες Hilbert είναι γνωστοί για την κακή τους κατάσταση. Θα το επαληθεύσουμε υπολογίζοντας όλους τους πιθανούς δείκτες κατάστασης του H4.

```
>> cond(H4,1)
ans =
  2.8375e+004

>> cond(H4)
ans =
  1.5514e+004

>> cond(H4,inf)
ans =
  2.8375e+004

>> cond(H4,'fro')
ans =
  1.5614e+004
```

Παράδειγμα 9.5.2

Η λύση του γραμμικού συστήματος

$$\begin{bmatrix} .780 & .563 \\ .913 & .659 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} .217 \\ .254 \end{bmatrix}$$

είναι $x_1 = 1$ και $x_2 = -1$. Πράγματι με τη MATLAB παίρνουμε:

```
>> A=[.780 .563; .913 .659]

A =
    0.7800    0.5630
    0.9130    0.6590

>> b=[.217; .254]

b =
    0.2170
    0.2540

>> x=A\b
```

```
x =
    1.0000
   -1.0000
```

Αν διαταράξουμε πολύ λίγο τον πίνακα των συντελεστών, και λύσουμε το σύστημα

$$\begin{bmatrix} .780 & .563001 \\ .913 & .659 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} .217 \\ .254 \end{bmatrix}$$

τότε η λύση που παίρνουμε είναι $x_1 = 8.574712\ldots$ και $x_2 = -1.49425\ldots$.
Πράγματι,

```
>> format long

>> A(1,2)=.563001
A =
   0.78000000000000   0.56300100000000
   0.91300000000000   0.65900000000000

>> x=A\b

x =
   8.57471264411556
 -11.49425287416920
```

Ο πίνακας A είναι προφανώς κακής κατάστασης. Θα το επαληθεύσουμε βρίσκοντας τον δείκτη κατάστασης $\kappa_1(A)$.

```
>> format short

>> A=[.780, .563;.913 .659]

A =
    0.7800    0.5630
    0.9130    0.6590

>> cond(A,1)

ans =
  2.6614e+006
```

9.6 Επαναληπτικές μέθοδοι

9.6.1 Η μέθοδος Jacobi

Έστω ότι θέλουμε να *προσεγγίσουμε* τη λύση του γραμμικού συστήματος $Ax = b$, χρησιμοποιώντας μια επαναληπτική μέθοδο της μορφής

$$x^{(k+1)} = Mx^{(k)} + d \ , k = 0,1,...$$

για κάποιο πίνακα M και κάποιο διάνυσμα d, αφού έχουμε μια αρχική εκτίμηση $x^{(0)}$ για την λύση. Για αυτές της μεθόδους θα υποθέσουμε ότι τα διαγώνια στοιχεία a_{ii} του A είναι μη μηδενικά

Μια τέτοια μέθοδος είναι η μέθοδος ***Jacobi***, όπου

$$M = D^{-1}B \ \text{ και } \ d = D^{-1}b \ , \text{ με } \ D = diag(A), B = D - A \, .$$

Το πιο κάτω m-file υλοποιεί την πιο πάνω μέθοδο για τον υπολογισμό της λύσης του γραμμικού συστήματος $Ax = b$ με ακρίβεια `tol` (την οποία αποφασίζει ο χρήστης), ξεκινώντας από μια αρχική προσέγγιση $x^{(0)}$ και κάνοντας μέχρι και `maxit` επαναλήψεις.

```
function [x] = jacobi(A,b,tol,maxit,x0)
% function [x] = jacobi(A,b,tol,maxit,x0)
%
% This function takes as input an n-by-n matrix A, an n-vector b
% and solves the linear system Ax=b using Jacobi's iteration
% with tol being the desired accuracy, maxit being the maximum
% number of allowed iterations and x0 being the initial guess
% for the solution.
%
D = diag(diag(A));
B = D-A;
D_inv = diag(1./diag(A));
D_inv_B = D_inv*B;
D_inv_b = D_inv*b;
k = 1;
xold = x0;
while k < maxit
   x=D_inv_B*xold+D_inv_b;
   if norm(x-xold,inf) < tol
      disp('');
      disp('Iteration converged! Number of iterations:')
      iter = k
      disp('');
      disp('Error between successive iterates is:');
      error = norm(x-xold,inf)
      break
   end
   xold = x;
   k = k+1;
end
%End of jacobi.m
```

Παράδειγμα 9.6.1

Ας υποθέσουμε ότι ορίζουμε τα εξής δεδομένα στη MATLAB:

```
>> A=[1 1 -1;-1 3 0; 1 0 -2]
A =
     1     1    -1
    -1     3     0
     1     0    -2
>> b=[0;2;-3]
b =
     0
     2
    -3
>> x0=[1;1;1]
x0 =
     1
     1
     1
>> tol=1e-06
tol =
  1.0000e-006
>> maxit=100
maxit =
   100
```

Τότε, τρέχουμε το m-file και παίρνουμε

```
>> format long
>> x=jacobi(A,b,tol,maxit,x0)
Iteration converged! Number of iterations:
iter =
    18
Error between successive iterates is:
error =
    5.953741807340762e-007
x =
   1.00000000000000
   0.99999980154194
   1.99999970231291
```

Η ακριβής λύση είναι

```
>> A\b
ans =
     1
     1
     2
```

και παρατηρούμε ότι

```
>> abs(x-ans)
ans =
```

```
1.0e-006 *
                  0
 0.198458059985640
 0.297687090089482
```

άρα η μέθοδος προσέγγισε τη λύση με την απαιτούμενη ακρίβεια

9.6.2 Η μέθοδος Gauss-Seidel

Μια άλλη επαναληπτική μέθοδος είναι η ***Gauss-Seidel*** για την οποία έχουμε

$$x^{(k+1)} = Mx^{(k)} + d \ , k = 0,1,...$$

με $M = \left(D - L\right)^{-1} U$, $\vec{d} = \left(D - L\right)^{-1} \vec{b}$, όπου $D = diag(A)$, L κάτω τριγωνικός με μηδενική διαγώνιο και U άνω τριγωνικός με μηδενική διαγώνιο, έτσι ώστε

$$A = D - L - U.$$

Μπορούμε να γράψουμε την πιο πάνω μέθοδο χρησιμοποιώντας συνιστώσες εξής:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right].$$

Η υλοποίηση της μεθόδου ***Gauss-Seidel*** αφήνεται σαν άσκηση.

9.7 Ασκήσεις

9.1 Έστω το γραμμικό σύστημα:

$$\begin{bmatrix} 4 & 0 & 1 & 1 \\ 3 & 1 & 3 & 1 \\ 0 & 1 & 2 & 0 \\ 3 & 2 & 4 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

Αφού ελέγξετε ότι ο A είναι αντιστρέψιμος, επιλύστε το σύστημα με τους εξής τρόπους:
(α) Χρησιμοποιώντας την ενσωματωμένη εντολή "\" της MATLAB.

(β) Βρείτε τον αντίστροφο A^{-1} του A και υπολογίστε στη συνέχεια το x:

(γ) Βρείτε την παραγοντοποίηση LU του A και στη συνέχεια επιλύστε τα τριγωνικά συστήματα με τα m-files LTriSol.m και UTriSol.m που γράψαμε.

9.2. Δίνεται ο συμμετρικός θετικά ορισμένος πίνακας

$$A = \begin{bmatrix} 3 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 3 \end{bmatrix}$$

Να βρεθεί η παραγοντοποίηση Cholesky του A
(α) με τη MATLAB

(β) με το χέρι.

9.3 Βρείτε με το χέρι την παραγοντοποίηση Cholesky του 3×3 πίνακα Pascal.

9.4 Βρείτε με τη MATLAB την παραγοντοποίηση Cholesky του 6×6 πίνακα Pascal.

9.5 Χρησιμοποιώντας τις κατάλληλες εντολές της MATLAB, υπολογίστε την 1-, 2-, τη νόρμα Frobenius και την ∞-νόρμα του x = [1, -2, 5, 0, 6].

9.6 Χρησιμοποιώντας τις κατάλληλες εντολές της MATLAB υπολογίστε την 1-, 2-, τη νόρμα Frobenius και την ∞-νόρμα του x = [0.2, 0.25, 0.3, …., 1.3].

9.7 Γράψτε ένα function m-file, που να καλείται mynorm.m**,** το οποίο να παίρνει σαν δεδομένα εισόδου ένα διάνυσμα x και ένα αριθμό p και υπολογίζει τις διανυσματικές νόρμες, όπως ακριβώς και η norm(x, p), χρησιμοποιώντας τον ορισμό της κάθε νόρμας. Το πρόγραμμα πρέπει να γράφει κατάλληλο μήνυμα σφάλματος όταν η τιμή του p δεν είναι αποδεκτή.

9.8 Υπολογίστε την 1-, 2-, τη νόρμα Frobenius και την ∞-νόρμα του πίνακα

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ -2 & 0 & 3 & 1 \\ 5 & -1 & -1 & 4 \\ 10 & 4 & 2 & 2 \end{bmatrix}$$

9.9 Υπολογίστε την 1- και την 2-νόρμα του 1000×1000 πίνακα Pascal.

9.10 Υπολογίστε τους δείκτες κατάστασης κ_1, κ_2, κ_∞ και κ_F του 10×10 πίνακα Hilbert.

9.11 Οι πίνακες Hilbert και Pascal είναι γνωστοί για την κακή τους κατάσταση η οποία χειροτερεύει καθώς η διάσταση n των πινάκων αυτών αυξάνει. Υπολογίστε τους 2-δείκτες κατάστασης για n = 3, 6, 9, 12, 15 και γράψτε τα αποτελέσματά σας σε πίνακα της μορφής

n	$\kappa_2(H_n)$	$\kappa_2(P_n)$
3		
6		
9		
12		
15		

Στη συνέχεια επιλύστε (με τη MATLAB) το σύστημα

$$P_3\, x = b$$

για $b = [1, 6, 10]^T$ και $b = [1.0001, 6, 10]^T$.

9.12 Είναι οι πίνακες

$$A = \begin{bmatrix} 1 & -1 & 2 \\ 1 & 0 & 1 \\ 2 & -1 & 3.0001 \end{bmatrix} \quad \text{και} \quad B = \begin{bmatrix} 1 & 2 & -3 & 1 \\ 2 & 4 & 1 & 4 \\ -1 & 1 & 1 & 0 \\ -2 & 2 & -5 & -1 \end{bmatrix}$$

καλής ή κακής κατάστασης;

9.13 Γράψτε ένα m-file με όνομα GaussSeidel.m της πιο κάτω μορφής:

```
[x, p] = GaussSeidel( A, b, x0, Nmax, eps)
```

όπου οι μεταβλητές εισόδου και εξόδου συμβολίζουν τα εξής:

A: πίνακας των συντελεστών
b: πίνακας των σταθερών
x0: αρχική εκτίμηση της λύσης
Nmax: μέγιστος επιτρεπτός αριθμός επαναλήψεων
eps: ανοχή (ε)

x: προσεγγιστική λύση
p: παίρνει την τιμή 1 σε περίπτωση σύγκλισης και την τιμή 0 διαφορετικά

Το πρόγραμμα τυπώνει την προσέγγιση $x^{(m)}$ της λύσης σε κάθε επανάληψη και τερματίζει όταν

$$\left\| \boldsymbol{x}^{(m+1)} - \boldsymbol{x}^{(m)} \right\|_\infty \le \varepsilon$$

9.14 Επιλύστε με το m-file GaussSeidel.m το σύστημα με

$$A=\begin{bmatrix}10 & -11 & 2 & 0\\ -1 & 11 & -1 & 3\\ 2 & -1 & 10 & -1\\ 0 & 3 & -1 & 8\end{bmatrix},\quad b=\begin{bmatrix}6\\ 25\\ -11\\ 15\end{bmatrix},\quad \varepsilon=10^{-10}$$

9.15 Επιλύστε με το m-file GaussSeidel.m το σύστημα με A τον πιο πάνω πίνακα και b το διάνυσμα των τεσσάρων τελευταίων ψηφίων της ταυτότητάς σας.

9.16 Με τη βοήθεια της MATLAB βρείτε πόσες επαναλήψεις της μεθόδου απαιτούνται για την επίλυση οποιουδήποτε συστήματος της μορφής $Ax = b$, όπου A ο πιο πάνω πίνακας, για να ισχύει

$$\left\|\boldsymbol{x}^{(m)}-\boldsymbol{x}\right\|_\infty \le 10^{-8}$$

αν

$$\left\|\boldsymbol{x}^{(0)}-\boldsymbol{x}\right\|_\infty = 10.$$

10 ΑΡΙΘΜΗΤΙΚΗ ΟΛΟΚΛΗΡΩΣΗ

10.1 Αθροίσματα Riemann

Στο κεφάλαιο αυτό θα ασχοληθούμε με αριθμητικές μεθόδους υπολογισμού του ορισμένου ολοκληρώματος

$$\int_a^b f(x)\,dx$$

όπου τα a, b είναι γνωστά και η συνάρτηση $f(x)$ είναι συνεχής και επίσης γνωστή. Γνωρίζουμε ότι αν διαμερίσουμε το διάστημα $[a, b]$ σε n υποδιαστήματα ίσου μήκους $\Delta x = (b-a)/n$ και κομβικά σημεία τα

$$x_0 = a, \quad x_1 = a + \Delta x, \quad x_2 = a + 2\Delta x, \quad \ldots, \quad x_i = a + i\Delta x, \quad \ldots, \quad x_n = a + n\Delta x = b$$

τότε έχουμε

$$\int_a^b f(x)dx = \lim_{n\to\infty} \sum_{i=0}^{n-1} f(x_i)\Delta x = \lim_{n\to\infty} \sum_{i=1}^{n} f(x_i)\Delta x\,.$$

Τα πιο πάνω αθροίσματα καλούνται **αθροίσματα Riemann** (αριστερό και δεξιό, αντίστοιχα). Αν τώρα, χρησιμοποιήσουμε ένα πεπερασμένο αριθμό κομβικών σημείων (δηλ. δεν πάρουμε το όριο όταν το n τείνει στο άπειρο) τότε έχουμε μια προσέγγιση για το ολοκλήρωμα.

Τα πιο κάτω m-files υπολογίζουν το αριστερό και δεξιό άθροισμα Riemann, αντίστοιχα, παίρνοντας σαν δεδομένα εισόδου τη συνάρτηση συνάρτηση f, τα άκρα a, b του διαστήματος $[a, b]$, και ένα ακέραιο αριθμό n, και δίνουν σαν δεδομένο εξόδου την προσέγγιση του ολοκληρώματος της $f(x)$ στο διάστημα $[a, b]$.

```
function [I] = leftsum(f,a,b,n)
%
% function [I] = leftsum(f,a,b,n)
%
% Ypologizei to Aristero A0roisma Reimann gia to oloklhrwma
% ths f(x) sto diasthma [a,b] me n upodiasthmata.
%

dx = (b-a)/n;
x=linspace(a,b,n+1);
I=0;
for i=1:n
    I = I + feval(f,x(i))*dx;
end

% Telos tou leftsum.m
```

```
function [I] = rightsum(f,a,b,n)
%
% function [I] = rightsum(f,a,b,n)
%
% Ypologizei to Deksio A0roisma Reimann gia to oloklhrwma
% ths f(x) sto diasthma [a,b] me n upodiasthmata.
%

dx = (b-a)/n;
x=linspace(a,b,n+1);
I=0;
for i=2:n+1
    I = I + feval(f,x(i))*dx;
end

% Telos tou rightsum.m
```

Θα χρησιμοποιήσουμε τα πιο πάνω m-files για να προσεγγίσουμε το ολοκλήρωμα

$$\int_0^1 \frac{1}{25+x^2}\,dx\,,$$

το οποίου η ακριβής τιμή είναι $\frac{1}{5}\arctan\left(\frac{1}{5}\right) \approx 0.03947911197$, χρησιμοποιώντας $n = 500$ κομβικά σημεία.

```
>> f = @(x) 1./(25+x.^2);

>> format long

>> leftsum(f,0,1,500)

ans =
   0.039480649445321

>> rightsum(f,0,1,500)

ans =
   0.039477572522244
```

Παράδειγμα 10.1.1

Θεωρούμε τώρα το ολοκλήρωμα $\int_0^2 x^2 e^{-x} dx = 2 - 10e^{-2} \left(\approx 0.64664716763387\right)$. Θα χρησιμοποιήσουμε τα πιο πάνω m-files για διάφορες τιμές του n για να διαπιστώσουμε ότι όταν το n αυξάνεται, η προσέγγιση που παίρνουμε τείνει στην ακριβή τιμή του ολοκληρώματος.

Έχουμε

```
>> f = @(x) x.^2.*exp(-x);

>> n=500:500:10000;

>> for i=1:length(n)
    Ileft(i) = leftsum(f,0,2,n(i));
    Iright(i) = rightsum(f,0,2,n(i));
  end
```

```
    >> Ileft'
ans =
   0.645564485365751
   0.646105826500787
   0.646286273545214
   0.646376497067392
   0.646430631180690
   0.646466720589555
   0.646492498738745
   0.646511832350637
   0.646526869604328
   0.646538899407283
   0.646548741973337
   0.646556944111715
   0.646563884382652
   0.646569833186310
   0.646574988816147
   0.646579499992256
   0.646583480441763
   0.646587018619101
   0.646590184356723
   0.646593033520580

>> Iright'
ans =
   0.647729849897536
   0.647188508766680
   0.647008061722476
   0.646917838200338
   0.646863704087047
   0.646827614678186
   0.646801836529000
   0.646782502917111
   0.646767465663416
   0.646755435860461
   0.646745593294408
   0.646737391156030
   0.646730450885098
   0.646724502081437
   0.646719346451599
   0.646714835275493
   0.646710854825986
   0.646707316648645
   0.646704150911028
   0.646701301747169
```

Πράγματι, οι προσεγγίσεις τείνουν προς την ακριβή τιμή, αλλά αυτό γίνεται με αρκετά αργό ρυθμό. Στη συνέχεια θα δούμε ένα άλλο τρόπο προσέγγισης ολοκληρωμάτων ο οποίος *συγκλίνει* ταχύτερα.

Παράδειγμα 10.1.2

Ένας πιο εύκολος τρόπος να υπολογίσουμε τα αθροίσματα Riemann είναι με την εντολή **diff** που βρίσκει τις διαφορές των στοιχείων ενός διανύσματος. Για παράδειγμα

```
    >> u=[1 3 4 4.5 8];
```

```
>> diff(u)
ans =
    2.0000    1.0000    0.5000    3.5000
```

Μπορούμε λοιπόν να τροποποιήσουμε το leftsum.m ως εξής:

```
function [I] = leftsum1(f,a,b,n)
%
% function [I] = leftsum1(f,a,b,n)
%
% Ypologizei to Aristero A0roisma Reimann gia to oloklhrwma
% ths f(x) sto diasthma [a,b] me n upodiasthmata.
%

x=linspace(a,b,n+1);
Y=feval(f,x);
Yleft=Y(1:n);
I=sum(diff(x).*Yleft);

% Telos tou leftsum1.m
```

Παράδειγμα 10.1.3

Το m-file leftsum2.m υπολογίζει το αριστερό άθροισμα Riemann και επιπλέον σχεδιάζει γραφικά το εμβαδόν που υπολογίζεται με τη χρήση της συνάρτησης **fill**:

```
function [I] = leftsum2(f,a,b,n)
%
% function [I] = leftsum2(f,a,b,n)
%
% Ypologizei to Aristero A0roisma Reimann gia to oloklhrwma
% ths f(x) sto diasthma [a,b] me n upodiasthmata.
% ISxedia;zei episis to embadon poy ypologizei.
%

x=linspace(a,b,n+1);
Y=feval(f,x);
Yleft=Y(1:n);
I=sum(diff(x).*Yleft);
% Draw area
xnew=linspace(a,b,1001);
ynew=feval(f,xnew);
plot(xnew,ynew); xlabel('x'); ylabel('y'); hold on
for i=1:n
    xx(1)=x(i); yy(1)=0;
    xx(2)=x(i+1); yy(2)=0;
    xx(3)=x(i+1); yy(3)=feval(f,xx(1));
    xx(4)=x(i); yy(4)=feval(f,xx(1));
    fill(xx,yy,'m')
end
plot(xnew,ynew); plot(x,Y,'ro')
hold off

% Telos tou leftsum2.m
```

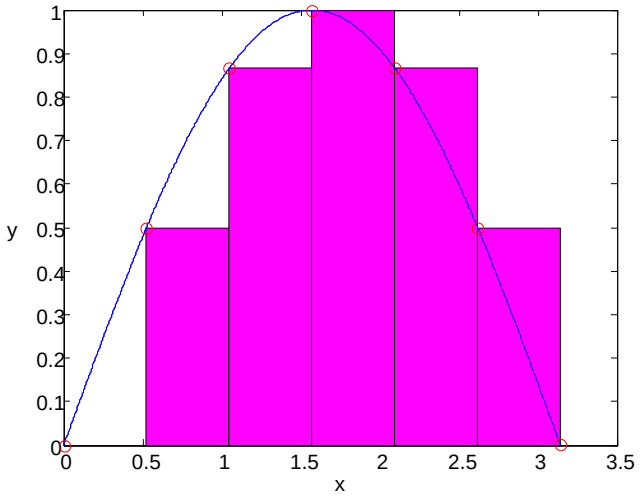
Θεωρούμε τώρα το ολοκλήρωμα

$$\int_0^{\pi} \sin x \, dx$$

Για $n = 6$ βρίσκουμε

```
>> leftsum2(@sin,0,pi,6)
ans =
   1.954097233313707
```

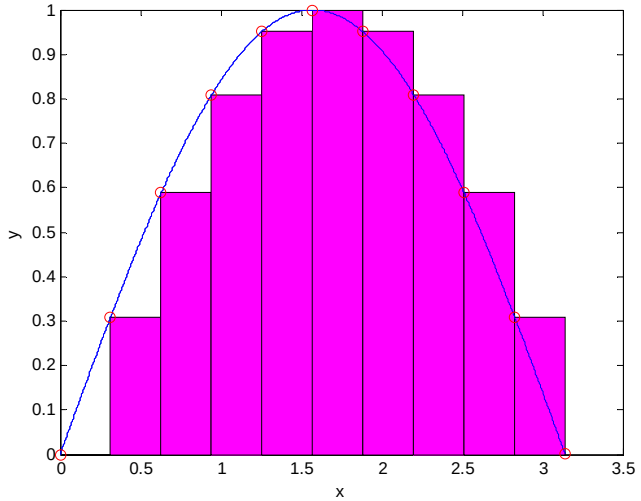
και παίρνουμε το γράφημα:



Για $n = 10$ βρίσκουμε

```
>> leftsum2(@sin,0,pi,10)
ans =
   1.983523537509454
```

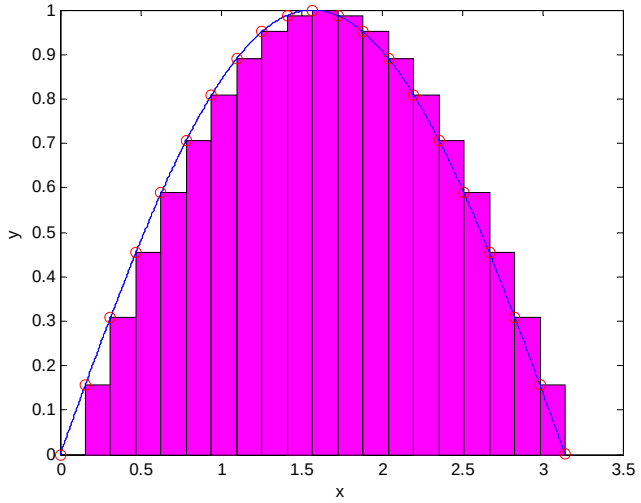
και παίρνουμε το γράφημα:



Για $n = 20$ βρίσκουμε

```
>> leftsum2(@sin,0,pi,20)
ans =
   1.995885972708715
```

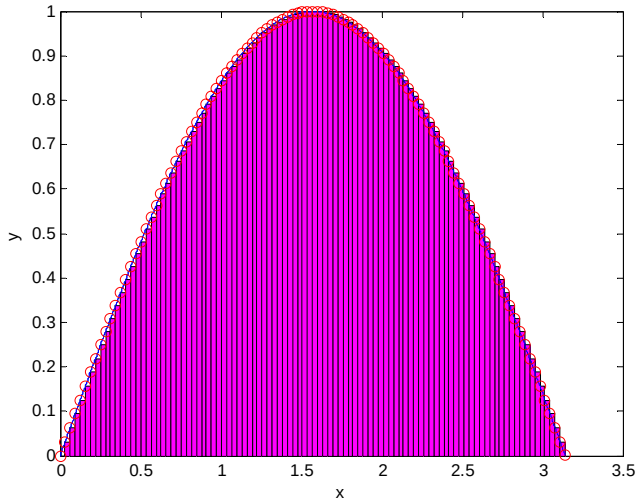
και παίρνουμε το γράφημα:



Τέλος για $n = 100$ βρίσκουμε

```
>> leftsum2(@sin,0,pi,100)
ans =
   1.999835503887444
```

και παίρνουμε το γράφημα:



Βλέπουμε πως με την αύξηση του n προσεγγίζεται το εμβαδόν που θέλουμε να υπολογίσουμε.

10.2 Η μέθοδος του τραπεζίου

Στη **μέθοδο του τραπεζίου**, η προσέγγιση ολοκληρώματος επιτυγχάνεται ως εξής:

$$\int_a^b f(x)dx \approx \frac{\Delta x}{2}\left\{f(x_0)+2\left[f(x_1)+f(x_2)+\ldots+f(x_{n-1})\right]+f(x_n)\right\}$$

όπου, $\Delta x=\left(b-a\right)/n$ και

$$x_0=a,\quad x_1=a+\Delta x,\quad x_2=a+2\Delta x,\quad \ldots,\quad x_i=a+i\Delta x,\quad \ldots,\quad x_n=a+n\Delta x=b\,.$$

Το πιο κάτω m-file υλοποιεί τη μέθοδο του τραπεζίου.

```
function [I] = trapezoid(f,a,b,n)

% function [I] = trapezoid(f,a,b,n)
%
% This function uses the (composite) Trapezoidal Rule to
% approximate the int(f(x),x=a..b) with n subintervals.
% The function f(x) is defined externally (as an anonymous
% function or using, e.g. the "inline" command.
%
if a==b
    I=0;
    return
end
dx = (b-a)/n;
x=linspace(a,b,n+1);
I=0;
for i=2:n
    I = I + 2*feval(f,x(i))*dx;
end
I = 0.5*(dx*feval(f,a)+dx*feval(f,b)+I);
%End of trapezoid.m
```

Θα τρέξουμε το πιο πάνω m-file για να προσεγγίσουμε το ολοκλήρωμα

$$\int_0^1 \frac{1}{25+x^2}\,dx\,,$$

όπως κάναμε και στην προηγούμενη παράγραφο, χρησιμοποιώντας 500 κομβικά σημεία.

```
>> f = @(x) 1./(25+x.^2);
>> trapezoid(f,0,1,500)

ans =
   0.039479110983783
```

Συγκρίνοντας την τιμή που πήραμε με την ακριβή τιμή (0.03947911197) και με αυτές που έδωσαν τα leftsum.m (0.039480649445321) και rightsum.m (0.039477572522244), βλέπουμε ότι το trapezoid.m δίνει καλύτερο αποτέλεσμα.

Παράδειγμα 10.2.1.

Θεωρούμε το ολοκλήρωμα

$$\int_0^2 x^2 e^{-x} dx = 2 - 10e^{-2} \left(\approx 0.64664716763387 \right).$$

Θα χρησιμοποιήσουμε το m-file trapezoid.m για διάφορες τιμές του *n* για να διαπιστώσουμε ότι όταν το *n* αυξάνεται, η προσέγγιση που παίρνουμε τείνει στην ακριβή τιμή του ολοκληρώματος.

Έχουμε

```
>> f = @(x) x.^2.*exp(-x);

>> n=10:10:200;

>> for i=1:length(n)
     Itrap(i) = trapezoid(f,0,2,n(i));
   end

>> Itrap'

ans =

   0.646633273909148
   0.646646297350553
   0.646646995655494
   0.646647113211022
   0.646647145340791
   0.646647156882577
   0.646647161830465
   0.646647164231974
   0.646647165510063
   0.646647166240432
   0.646647166682129
   0.646647166961876
   0.646647167145984
   0.646647167271144
   0.646647167358621
   0.646647167421247
   0.646647167467032
   0.646647167501131
   0.646647167526947
   0.646647167546781
```

Πράγματι, οι προσεγγίσεις τείνουν προς την ακριβή τιμή, και με αρκετά γρήγορο ρυθμό (αφού οι τιμές του *n* που πήραμε δεν ήταν και τόσο μεγάλες). Για *n* παρόμοιο με αυτό που πήραμε για τα αθροίσματα Riemann, παίρνουμε

```
>> trapezoid(f,0,2,500)

ans =
   0.646647167631643

>> trapezoid(f,0,2,1000)

ans =
   0.646647167633734

>> trapezoid(f,0,2,10000)

ans =
   0.646647167633874
```

Υπάρχουν και άλλες παρόμοιες μέθοδοι, τις οποίες δεν θα δούμε εδώ. (Μια από αυτές όμως θα δοθεί σαν άσκηση.)

10.2.1 Η εντολή trapz

Στην περίπτωση που αντί για την συνάρτηση $f(x)$ μας έχει δοθούν διακριτές τιμές της $f(x_1), f(x_2), ..., f(x_N)$ που αντιστοιχούν στα σημεία $x_1, x_2, ..., x_N$, τότε μπορούμε να βρούμε το ολοκλήρωμα της $f(x)$ στο διάστημα $[x_1, x_N]$ με την εντολή

trapz(x, y)

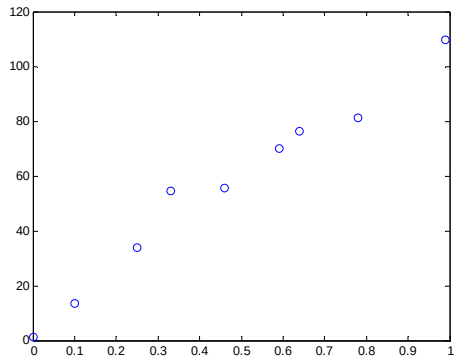
η οποία παίρνει σαν δεδομένα εισόδου τα διανύσματα $x = [x_1, x_2, ..., x_N]$ και $y = [f(x_1), f(x_2), ..., f(x_N)]$. Βασικά, χρησιμοποιείται η μέθοδος του τραπεζίου (όπως φαίνεται και από το όνομα trapz) χωρίς καν να γνωρίζουμε την συνάρτηση (αφού μόνο οι τιμές της χρειάζονται στο υπολογισμό του ολοκληρώματος με τον κανόνα του τραπεζίου).

Έστω για παράδειγμα τα δεδομένα:

```
>> x = [0 0.1 0.25 0.33 0.46 0.59 0.64 0.78 0.99];
>> y = [1.14 13.45 33.99 54.68 55.5 70.0 76.3 81.19 109.7];
```

Η γραφική παράσταση των πιο πάνω δεδομένων είναι

```
>> plot(x,y,'o')
```



Τότε, το ολοκλήρωμα της συνάρτησης που περνά από τα πιο πάνω σημεία είναι

```
>> trapz(x,y)
ans =
  57.878749999999997
```

Παράδειγμα 10.2.2

Είναι εύκολο να δούμε ότι με τον κανόνα του τραπεζίου υπολογίζουμε το πιο κάτω άθροισμα εμβαδών ορθογωνίων:

$$A = \sum_{i=1}^{n-1} (x_{i+1} - x_i)(y_{i+1} + y_i)/2$$

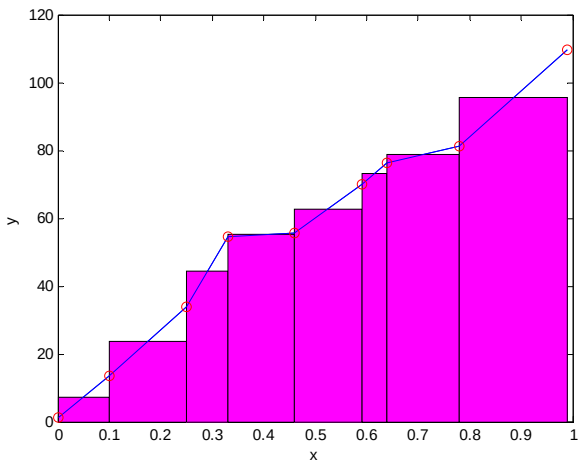
Το πιο κάτω m-file υλοποιεί τον κανόνα του τραπεζίου και ταυτόχρονα σχεδιάζει το εμβαδόν που υπολογίζεται:

```
function [I]=trapznew(x,y)
%
% TRAPZNEW
% Ypologizei kai sxediazei to embadon katw apo ta shmeia (x,y)
% sumfvna me ton kanona toy trapeziou
%
nx=length(x);
ny=length(y);
if nx~=ny
    disp('x and y must be of the same length!')
    return
else
    n=nx;
end
avg_y=y(1:n-1)+diff(y)/2;
I=sum(diff(x).*avg_y);
% Draw area
plot(x,y); xlabel('x'); ylabel('y'); hold on
for i=1:n-1
    xx(1)=x(i); yy(1)=0;
    xx(2)=x(i+1); yy(2)=0;
    xx(3)=x(i+1); yy(3)=avg_y(i);
    xx(4)=x(i); yy(4)=avg_y(i);
    fill(xx,yy,'m')
end
plot(x,y); plot(x,y,'ro')
hold off

% Telos tou trapznew.m
```

Για τα δεδομένα που είδαμε πιο πάνω παίρνουμε:

```
>> x = [0 0.1 0.25 0.33 0.46 0.59 0.64 0.78 0.99];
>> y = [1.14 13.45 33.99 54.68 55.5 70.0 76.3 81.19 109.7];
>> trapznew(x,y)
ans =
  57.878749999999997
```

και το γράφημα



Παράδειγμα 10.2.3

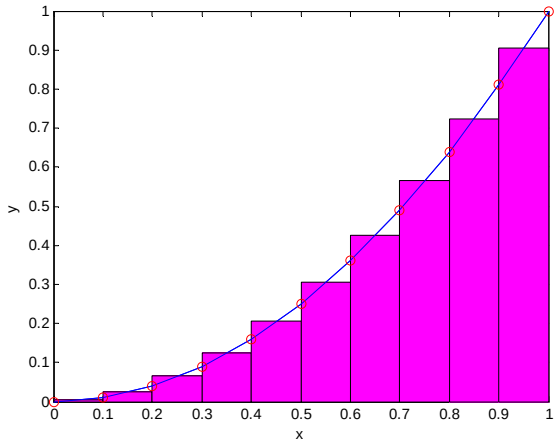
Θα χρησιμοποιήσουμε το trapznew για να υπολογίσουμε προσεγγιστικά το ολοκλήρωμα

$$\int_0^1 x^2 dx = \frac{1}{3}$$

Για $n = 11$ παίρνουμε:

```
>> x=0:0.1:1;
>> y=x.^2;
>> trapznew(x,y)
ans =
   0.33500000000000
```

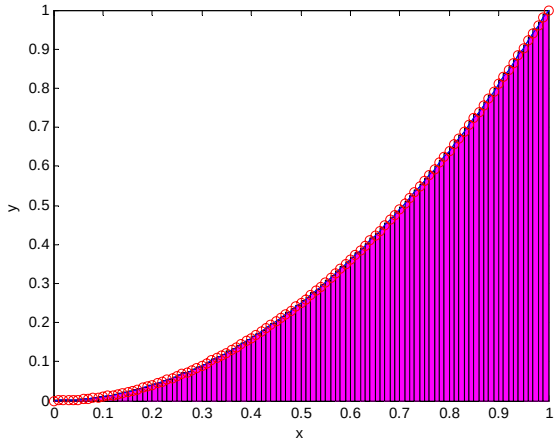
και το γράφημα



Για $n = 101$ παίρνουμε:

```
>> x=0:0.01:1;
>> y=x.^2;
>> trapznew(x,y)
ans =
   0.33335000000000
```

και το γράφημα



10.3 Οι εντολές quad και quadl

Η MATLAB διαθέτει δύο συναρτήσεις/εντολές βιβλιοθήκης για τον υπολογισμό του ολοκληρώματος

$$\int_a^b f(x)dx\,,$$

οι οποίες καλούνται **quad** και **quadl** (από τη λέξη *quadrature* που σημαίνει αριθμητική ολοκλήρωση) και έχουν την εξής δομή:

quad(fun, a, b, tol)

quadl(fun, a, b, tol)

Τα δεδομένα εισόδου είναι η συνάρτηση fun που αντιστοιχεί στην *f*(*x*) (η οποία ορίζεται σαν ανώνυμη συνάρτηση ή μέσω ενός m-file), τα άκρα *a*, *b* του διαστήματος ολοκλήρωσης, και η επιθυμητή ανοχή tol. Σημειώνουμε ότι η **quad** χρησιμοποιεί την λεγόμενη **προσαρμοστική μέθοδο του Simpson**, ενώ η **quadl** χρησιμοποιεί την **προσαρμοστική μέθοδο Lobatto**. (Γράψτε help quad ή help quadl για περισσότερες πληροφορίες.)

Ας δούμε τι τιμές μας δίνουν αυτές οι συναρτήσεις για τα ολοκληρώματα των προηγούμενων παραδειγμάτων:

```
>> f = @(x) 1./(25+x.^2);
>> quad(f,0,1,1e-06)
ans =
   0.039479111966923
>> quad(f,0,1,1e-016)
ans =
   0.039479111969976
>> quadl(f,0,1,1e-06)
ans =
   0.039479111969976
>> quadl(f,0,1,1e-016)
ans =
   0.039479111969976
```

Παρατηρούμε ότι, για αυτό το παράδειγμα, η quadl δίνει καλύτερα αποτελέσματα ακόμα και όταν η ζητούμενη ακρίβεια είναι χαμηλή.

Για το άλλο παράδειγμα, έχουμε

```
>> f = @(x) x.^2.*exp(-x);
>> quad(f,0,2,1e-06)
ans =
   0.646647196668170
>> quad(f,0,2,1e-016)
ans =
   0.646647167633873
```

```
>> quadl(f,0,2,1e-06)
ans =
   0.646647172284436
>> quadl(f,0,2,1e-016)
ans =
   0.646647167633873
```

Παράδειγμα 10.3.1

Θα υπολογίσουμε το ολοκλήρωμα $\int_0^1 e^{x^2} dx$ χρησιμοποιώντας τις quad και quadl για διαφορετικές ανοχές. Με τη quad βρίσκουμε

```
>> for i = 1:16
    inte = quad('exp(x.^2)',0,1,10^(-i));
    disp(sprintf('For i= %3.0f the integral is %15.13g',i,inte))
   end
For i=   1 the integral is  1.462652811157
For i=   2 the integral is  1.462652811157
For i=   3 the integral is  1.462652811157
For i=   4 the integral is  1.462652811157
For i=   5 the integral is  1.462651884484
For i=   6 the integral is  1.462651771591
For i=   7 the integral is  1.462651746962
For i=   8 the integral is  1.462651745972
For i=   9 the integral is  1.462651745914
For i=  10 the integral is  1.462651745907
For i=  11 the integral is  1.462651745907
For i=  12 the integral is  1.462651745907
For i=  13 the integral is  1.462651745907
For i=  14 the integral is  1.462651745907
For i=  15 the integral is  1.462651745907
For i=  16 the integral is  1.462651745907
```

Με τη quadl βρίσκουμε

```
>> for i = 1:16
    inte = quadl('exp(x.^2)',0,1,10^(-i));
    disp(sprintf('For i= %3.0f the integral is %15.13g',i,inte))
   end
For i=   1 the integral is  1.462651763478
For i=   2 the integral is  1.462651763478
For i=   3 the integral is  1.462651763478
For i=   4 the integral is  1.462651763478
For i=   5 the integral is  1.462651763478
For i=   6 the integral is  1.462651763478
For i=   7 the integral is  1.462651763478
For i=   8 the integral is  1.462651745907
For i=   9 the integral is  1.462651745907
For i=  10 the integral is  1.462651745907
For i=  11 the integral is  1.462651745907
For i=  12 the integral is  1.462651745907
For i=  13 the integral is  1.462651745907
For i=  14 the integral is  1.462651745907
For i=  15 the integral is  1.462651745907
For i=  16 the integral is  1.462651745907
```

Παρατηρούμε ξανά ότι η quadl είναι πιο ακριβής.

10.4 Ασκήσεις

10.1 Ο λεγόμενος (σύνθετος) κανόνας του *Simpson*, για την προσέγγιση ενός ολοκληρώματος, είναι

$$\int_a^b f(x)dx \approx \frac{h}{3}\left[f(x_0)+4f(x_1)+2f(x_2)+4f(x_3)+2f(x_4)+\ldots+2f(x_{n-2})+4f(x_{n-1})+f(x_n)\right]$$

όπου $h=(b-a)/n$, $x_i=a+ih, i=0,\ldots,n$ με το $n>0$ **άρτιο**. Να γράψετε ένα m-file, που να καλείται csimpson.m, το οποίο να παίρνει σαν δεδομένα εισόδου την συνάρτηση *f*, τα άκρα *a* και *b* του διαστήματος [*a*, *b*], και το *n*, και να δίνει σαν δεδομένο εξόδου την προσέγγιση του ολοκληρώματος.

Χρησιμοποιείστε το m-files σας για να υπολογίσετε τα πιο κάτω ολοκληρώματα με $n = 50$.

(α) $\int_0^1 \frac{1}{25+x^2}dx = \frac{1}{5}\arctan\left(\frac{1}{5}\right)\left(\approx 0.03947911197\right)$

(β) $\int_0^2 x^2e^{-x}dx = 2-10e^{-2}\left(\approx 0.64664716763387\right)$

(γ) $\int_0^1 \cos\left(\frac{\pi x^2}{2}\right)dx (\approx 0.779893400376823)$

10.2 Με τη βοήθεια σταθερού άνεμου, ένα χαρταετός πετά με δυτική κατεύθυνση. Το ύψος του χαρταετού από την (οριζόντια) αρχική θέση $x = 0$ μέχρι την τελική θέση $x = 80$, δίνεται από συνάρτηση $f(x)=150-\frac{1}{40}(x-50)^2$. Να βρείτε την απόσταση που ταξίδεψε ο χαρταετός. (ΥΠΟΔΕΙΞΗ: Για να απαντήσετε σε αυτή την ερώτηση πρέπει να υπολογίσετε το μήκος τόξου της καμπύλης $y = f(x)$ από $x = 0$ μέχρι $x = 80$.)

10.3 Χρησιμοποιώντας τις εντολές quad και quadl, να υπολογίσετε τα πιο κάτω ολοκληρώματα με ακρίβεια 10^{-6}.

(α) $\int_0^1 \cos(\pi x)\sin(\pi x/2)dx = -\frac{2}{3\pi}\left(\approx -0.2122065907\right)$

(β) $\int_{-1}^1 \sqrt{1+x^2/4}dx = \frac{1}{2}\sqrt{5}+2\ln(2)-2\ln(\sqrt{5}-1)\left(\approx 2.080457639\right)$

(γ) $\int_0^1 e^{-x^2}dx\left(\approx .7468241330\right)$

10.4 Δημιουργείστε ένα function m-file με όνομα rightsum2.m που να υπολογίζει το δεξιό άθροισμα Riemann της συνάρτησης *f* στο διάστημα [*a*, *b*] για *n* ίσου μήκους υποδιαστήματα και να σχεδιάζει το υπολογιζόμενο εμβαδόν.

Υπόδειξη: Τροποποιείστε το leftsum2.m που δίνεται στο Παράδειγμα 10.1.3 και χρησιμοποιείστε τη συνάρτηση diff.

10.5 Χρησιμοποιώντας τις εντολές quad και quadl, υπολογίστε το ολοκλήρωμα

$$\int_0^2 \frac{1}{x^3 - 2x - 5} dx$$

με τους δύο τρόπους που δίνει η βοήθεια help quad.

11 ΣΥΝΗΘΕΙΣ ΔΙΑΦΟΡΙΚΕΣ ΕΞΙΣΩΣΕΙΣ

11.1 Γενικά περί συνήθων διαφορικών εξισώσεων

Μια συνήθης διαφορική εξίσωση (ΣΔΕ) $1^{ης}$ τάξης έχει τη μορφή

$$\frac{dy}{dt} = f(t, y(t))$$

όπου $f(t, y)$ γνωστή και $y(t)$ άγνωστη συνάρτηση. Η πιο πάνω εξίσωση καλείται *διαφορική* γιατί περιέχει την παράγωγο μιας συνάρτησης. Καλείται *συνήθης* γιατί η παράγωγος είναι συνήθης (και όχι για *μερική*), και τέλος είναι $1^{ης}$ τάξης γιατί περιλαμβάνει μόνο την $1^{η}$ παράγωγο της άγνωστης συνάρτησης. Αν και μπορούμε να μιλήσουμε για ΣΔΕ $2^{ης}$, $3^{ης}$, … τάξης, θα περιοριστούμε μόνο σε αυτές που είναι $1^{ης}$ τάξης γιατί οι υπόλοιπες μπορούν να εκφραστούν σαν ένα *σύστημα* ΣΔΕ $1^{ης}$ τάξης. Άρα είναι αρκετό να ξέρουμε πώς να λύνουμε συστήματα ΣΔΕ $1^{ης}$ τάξης στη MATLAB.

Σημειώνουμε ότι στη ειδική περίπτωση που η συνάρτηση f εξαρτάται μόνο από το t, τότε έχουμε

$$\frac{dy}{dt} = f(t) \quad \Rightarrow \quad y = \int f(t)dt + C$$

όπου C μια αυθαίρετη σταθερά, και έτσι παίρνουμε τη λύση αναλυτικά. (Για την ακρίβεια, παίρνουμε άπειρες λύσεις, μια για κάθε σταθερά.)

Αν εκτός από την διαφορική εξίσωση μας δοθεί και μια *αρχική συνθήκη*, π.χ. $y(t_0) = y_0$, με τα t_0, y_0 δοσμένα, τότε η λύση της Σ.Δ.Ε. είναι μοναδική, με την προϋπόθεση ότι η συνάρτηση $f(t, y)$ ικανοποιεί κάποιες συνθήκες ομαλότητας. Σε αυτή την περίπτωση έχουμε ένα *πρόβλημα αρχικών τιμών* (ΠΑΤ):

$$\begin{cases} y'(t) = f(t, y(t)) \\ y(t_0) = y_0 \end{cases}$$

Στο παρόν κεφάλαιο θα ασχοληθούμε με μεθόδους (αριθμητικής) επίλυσης ΠΑΤ χρησιμοποιώντας την MATLAB.

Παράδειγμα 11.1.1 Η μέθοδος του Euler

Η πιο παλιά μέθοδος αριθμητικής επίλυσης του ΠΑΤ

$$\begin{cases} y'(t) = f(t, y(t)) \\ y(t_0) = y_0 \end{cases}$$

είναι η λεγόμενη μέθοδος του Euler, με την οποία βρίσκουμε (διακριτές) προσεγγίσεις $y_i \approx y(t_i)$, $i = 0,1,2,...$ για κάποια t_i, $i = 0,1,2,...$ που μας ενδιαφέρουν. Αν επιλέξουμε κάποιο $h > 0$ (το λεγόμενο *βήμα*) και ορίσουμε $t_{i+1} = t_i + h$, $i = 0,1,2,...,N$ για κάποιο N, τότε μια και τα f, t_0 και y_0 είναι γνωστά, μπορούμε να βρούμε τα $y_i \approx y(t_i)$, $i = 0,1,2,...,T$ μέσω της λεγόμενης *επανάληψης του Euler*:

$$y_{i+1} = y_i + hf(t_i, y_i), \; i = 0,1,2,...,N$$

Το πιο κάτω m-file υλοποιεί την πιο πάνω διαδικασία, παίρνοντας σαν δεδομένα εισόδου την συνάρτηση f, τις αρχικές τιμές y_0, t_0, το T έτσι ώστε $t \in [t_0, T]$ και το βήμα h, και δίνει σαν δεδομένα εξόδου τα διανύσματα $t = [t_0, t_1, ..., t_N]$, $y = [y_0, y_1, ..., y_N]$ έτσι ώστε $y_i \approx y(t_i)$, $i = 0,1,2,...,N$.

```
function [y,t] = euler(fun,y0,t0,T,h)
%  [y,t] = euler(fun,y0,t0,T,h) -
%
%  This function computes the solution to the IVP
%          y'(t) = fun(y,t), y(t0)= y0 ,
%  for a given function "fun(t,y)" using Euler's method.
%
%  The function can be defined via the m-file fun.m,
%  or as an anonymous function (or even using the "inline"
%  command).
%
%  y0 is the initial value, T is the maximum value for t, h
%  is the stepsize and t0=initial value for t.
%
%  The output is a vector containing the approximate
%  solution y_euler.
%

   y(1) = y0;
   t(1) = t0;

   for i=1:ceil((T-t0)/h)
     y(i+1) = y(i) + h*feval(fun,t(i),y(i));
     t(i+1) = t(i) + h;
   end;
   t=t';
   y=y';

% End of m-file euler.m
```

Ας χρησιμοποιήσουμε το πιο πάνω m-file για το εξής ΠΑΤ:

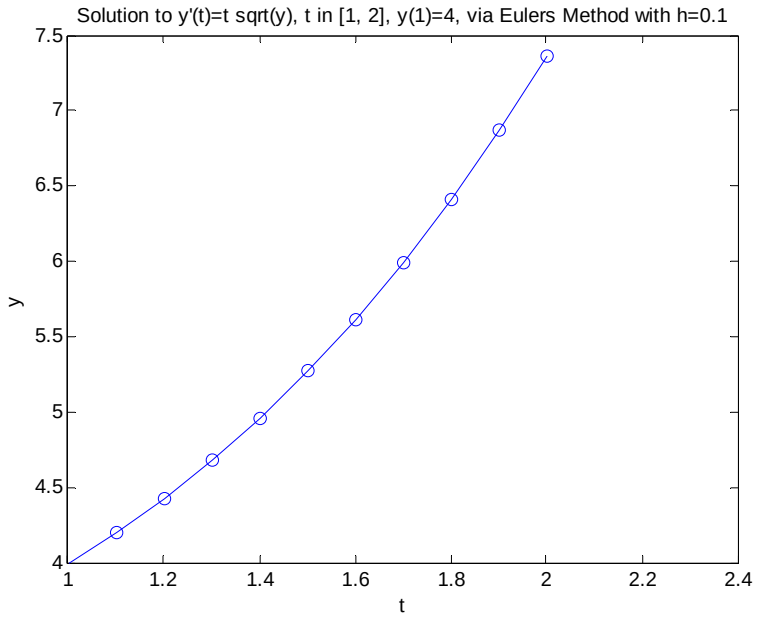
$$y'(t) = t\sqrt{y}, \; y(1) = 4$$

όπου το t ανήκει στο διάστημα [1, 2], και ας επιλέξουμε αρχικά $h = 0.1$.

```
>> f = @(t,y) t*sqrt(y);
>> [y,t]=euler(f,4,1,2,0.1)
y =
    4.0000
    4.2000
    4.4254
    4.6779
    4.9590
    5.2708
    5.6152
    5.9943
    6.4105
    6.8663
    7.3642
t =
    1.0000
    1.1000
    1.2000
    1.3000
    1.4000
    1.5000
    1.6000
    1.7000
    1.8000
    1.9000
    2.0000
```

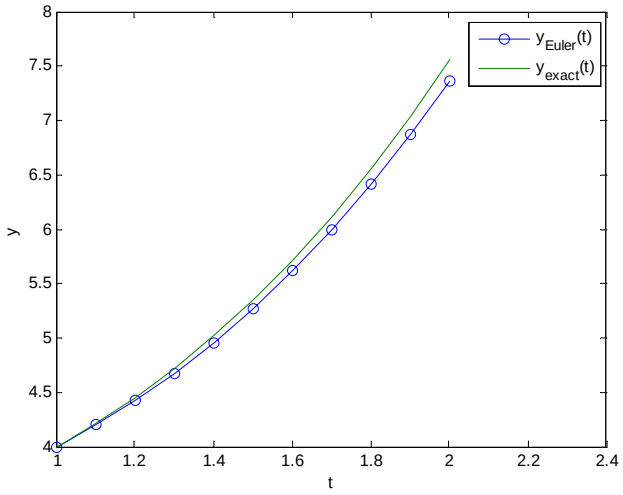
Παίρνουμε τη γραφική παράσταση της λύσης που πήραμε, ως

```
>> plot(t,y,'o-')
>> xlabel('t')
>> ylabel('y')
>> title('Solution to y''(t)=t sqrt(y), t in [1, 2], y(1)=4, via
Eulers Method with h=0.1')
```



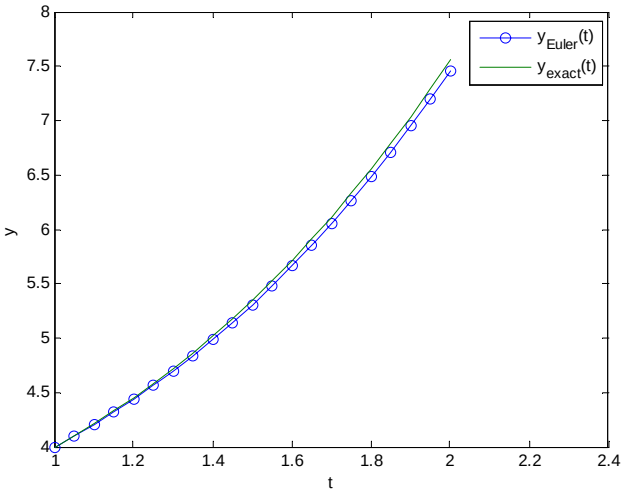
Η ακριβής λύση του πιο πάνω Π.Α.Τ. είναι $y_{ex}(t)=\frac{1}{16}\left(t^2+7\right)^2$, και το πιο κάτω γράφημα μα δείχνει ότι η λύση που πήραμε με την μέθοδο του Euler δεν είναι παρά μια προσέγγιση.

```
>> yex=@(t) (1/16)*(t.^2+7).^2;
>> plot(t,y,'o-',t,yex(t))
>> xlabel('t')
>> ylabel('y')
>> legend('y_{Euler}(t)','y_{exact}(t)')
```



Αν χρησιμοποιήσουμε πιο μικρό βήμα h, τότε η προσέγγιση θα είναι καλύτερη. Αυτό φαίνεται πιο κάτω, όπου πήραμε $h = 0.05$.

```
>> [y,t]=euler(f,4,1,2,0.05);
>> plot(t,y,'o-',t,yex(t))
>> xlabel('t')
>> ylabel('y')
>> legend('y_{Euler}(t)','y_{exact}(t)')
```



11.2 Η εντολή ode 45

Η MATLAB διαθέτει αρκετές συναρτήσεις/εντολές βιβλιοθήκης για την επίλυση Π.Α.Τ., όπως οι **ode23, ode45, ode113, ode15s, ode23s, ode23t, ode23tb**. (Γράψτε **doc ode23** για περισσότερες πληροφορίες.) Εμείς θα ασχοληθούμε μόνο με την **ode45**, η οποία λύνει το Π.Α.Τ.

$$\begin{cases} y'(t) = f(t, y(t)) \\ y(t_0) = y_0 \end{cases}$$

και έχει την εξής δομή:

[t_out,y_out] = ode45(odefun, t_span, y0)

Τα δεδομένα εισόδου και εξόδου έχουν ως εξής:

- odefun: η συνάρτηση $f(t, y)$ (σαν m-file ή ανώνυμη συνάρτηση)
- t_span: το διάνυσμα $[t_0, T]$ όπου ανήκει το t – αυτό πρέπει να δοθεί με αγκύλες
- y0: η αρχική τιμή $y_0 (= y(t_0))$
- t_out: το διάνυσμα με τα σημεία $t_0, t_1, t_2 ...$
- y_out: το διάνυσμα με τις προσεγγιστικές τιμές $y_0, y_1, y_2 ...$ της λύσης

Όπως βλέπετε, τα δεδομένα εισόδου και εξόδου είναι παρόμοια με αυτά που είχαμε στο m-file που γράψαμε για την μέθοδο του Euler. Η διαφορά έγκειται στο ότι για το m-file που γράψαμε, δίναμε και το βήμα h, ενώ στην ode45 δεν το δίνουμε. Η MATLAB επιλέγει της το βήμα με τέτοιο τρόπο ώστε η λύση που παίρνουμε να έχει (απόλυτη) ακρίβεια 10^{-6}. Αυτό σημαίνει ότι το μέγεθος των διανυσμάτων t και y δεν εξαρτάται από εμάς αλλά από τη MATLAB και ότι το διάνυσμα t μπορεί να μην είναι ομοιόμορφα κατανεμημένο. Μπορούμε, αν θέλουμε, να αυξομειώσουμε την ακρίβεια στην εντολή ode45 με το να δώσουμε περισσότερα δεδομένα εξόδου. Για να δείτε πως γράψτε help ode45 στη MATLAB.

Παράδειγμα 11.2.1

Θα βρούμε μια προσέγγιση για τη λύση του Π.Α.Τ.

$$\begin{cases} y'(t) = y(2 - y) \quad , \quad 0 \le t \le 1 \\ y(0) = 3 \end{cases}$$

χρησιμοποιώντας την εντολή ode45.

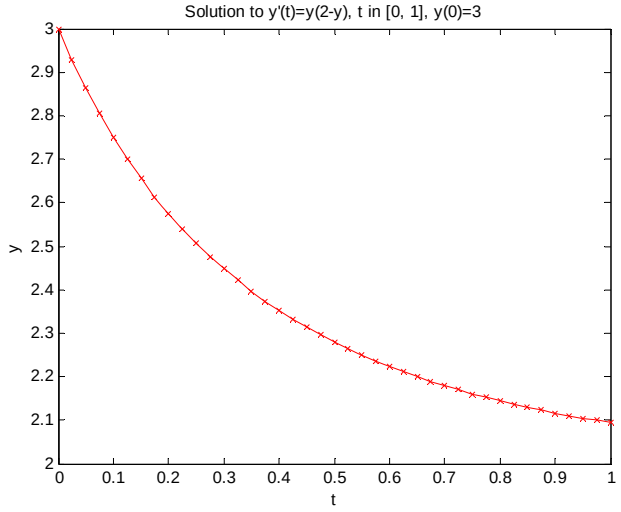
```
>> f = @(t,y) y.*(2-y);
>> [t,y]=ode45(f,[0,1],3);
```

Βάλαμε ‘ ; ’ στο τέλος της εντολής για να μην δούμε τις τιμές που παίρνουμε μια και τα διανύσματα μπορεί να είναι αρκετά μεγάλα:

```
>> length(y)

ans =
    41
```

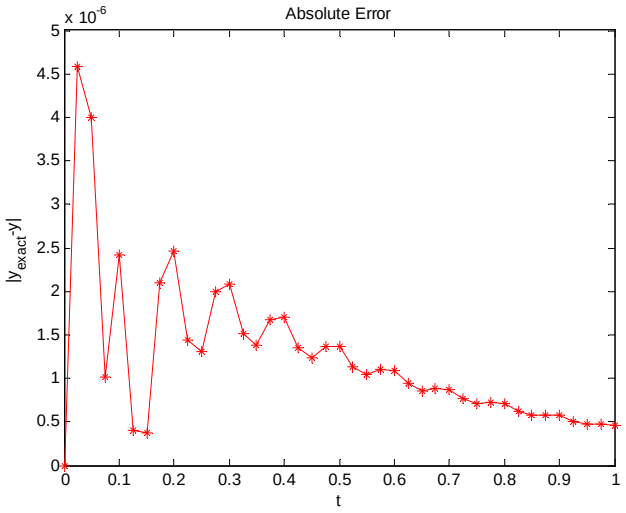
Η γραφική παράσταση της λύσης που πήραμε δίνεται πιο κάτω:

```
>> plot(t,y,'-rx')
>> xlabel('t')
>> ylabel('y')
>> title('Solution to y''(t)=y(2-y), t in [0, 1], y(0)=3')
```



Η ακριβής λύση του πιο πάνω Π.ΑΤ είναι $y_{ex}(t) = \dfrac{6}{3-e^{-2t}}$, και πιο κάτω δείχνουμε τη γραφική παράσταση του σφάλματος $\left|y(t) - y_{ex}(t)\right|$:

```
>> yex=@(t)  6./(3-exp(-2*t));
>> plot(t,abs(y-yex(t)),'*r')
>> plot(t,abs(y-yex(t)),'*-r')
>> xlabel('t')
>> ylabel('|y_{exact}-y|')
>> title('Absolute Error')
```



Πράγματι, έχουμε ακρίβεια της τάξης του 10^{-6}.

11.3 Συστήματα ΣΔΕ

Εκτός από Π.Α.Τ., η εντολή ode45 μπορεί να χρησιμοποιηθεί και για συστήματα Σ.Δ.Ε., όπως για παράδειγμα το πιο κάτω 3×3 *σύστημα αρχικών τιμών* (Σ.ΑΤ):

$$\begin{cases} x_1'(t) = f_1(t, x_1, x_2, x_3) \\ x_2'(t) = f_2(t, x_1, x_2, x_3) \\ x_3'(t) = f_3(t, x_1, x_2, x_3) \\ x_1(t_0) = a_1, x_2(t_0) = a_2, x_3(t_0) = a_3 \end{cases}$$

όπου οι συναρτήσεις $f_i(t, x_1, x_2, x_3), i = 1,2,3$, το σημείο t_0 και οι τιμές a_i, i = 1, 2, 3 είναι δεδομένα, και θέλουμε να προσδιορίσουμε τις συναρτήσεις $x_1(t), x_2(t), x_3(t)$. Αν γράψουμε το πιο πάνω σύστημα σε διανυσματική μορφή, τότε έχουμε

$$\begin{cases} \dfrac{d}{dt}\left(\vec{x}(t)\right) = [F]\,\vec{x}(t) \\ \vec{x}(t_0) = [a_1, a_2, a_3] \end{cases}$$

που μοιάζει με το ΠΑΤ που μελετήσαμε προηγουμένως. Ας δούμε ένα παράδειγμα:

$$\begin{cases} x_1'(t) = -\dfrac{8}{3}x_1(t) + x_2(t)x_3(t) \\ x_2'(t) = -10x_2(t) + 10x_3(t) \\ x_3'(t) = -x_2(t)x_1(t) + 28x_2(t) - x_3(t) \\ x_1(0) = 20, x_2(0) = 5, x_3(0) = -5 \end{cases}$$

το οποίο γράφουμε σε διανυσματική μορφή ως εξής:

$$\begin{bmatrix} x_1'(t) \\ x_2'(t) \\ x_3'(t) \end{bmatrix} = \begin{bmatrix} -\frac{8}{3} & 0 & x_2(t) \\ 0 & -10 & 10 \\ -x_2(t) & 28 & -1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} \Leftrightarrow \frac{d}{dt}\left(\vec{x}(t)\right) = \underbrace{\begin{bmatrix} -\frac{8}{3} & 0 & x_2(t) \\ 0 & -10 & 10 \\ -x_2(t) & 28 & -1 \end{bmatrix}}_{[F]} \vec{x}(t)$$

και $\vec{x}(0) = [20, 5, -5]$.

Τώρα, για να χρησιμοποιήσουμε την εντολή ode45, θα πρέπει να ορίσουμε τον πίνακα [F], που ορίζει τις συναρτήσεις του δεξιού μέλους του συστήματος, σε ένα m-file (μια και δεν μπορούμε να τον ορίσουμε σαν μια ανώνυμη συνάρτηση). Για το παράδειγμα μας τον ορίζουμε στο m-file odefun.m, που φαίνεται πιο κάτω:

```
function [xprime] = odefun(t,x)
%  [xprime] = odefun(t,x) -
%  This function corresponds to the RHS of the system of ODEs, in
%  which x=[x(1), x(3), x(3)] represents the (vector) of unknown
%  functions.
   xprime = [-8/3,0,x(2);0,-10,10;-x(2),28,-1]*x;
% End of m-file odefun.m
```

Η εντολή ode45 χρησιμοποιείται όπως και πριν

[t, x] = ode45(@odefun, t_span, a)

αλλά, τώρα, τα δεδομένα εισόδου και εξόδου έχουν ως εξής:

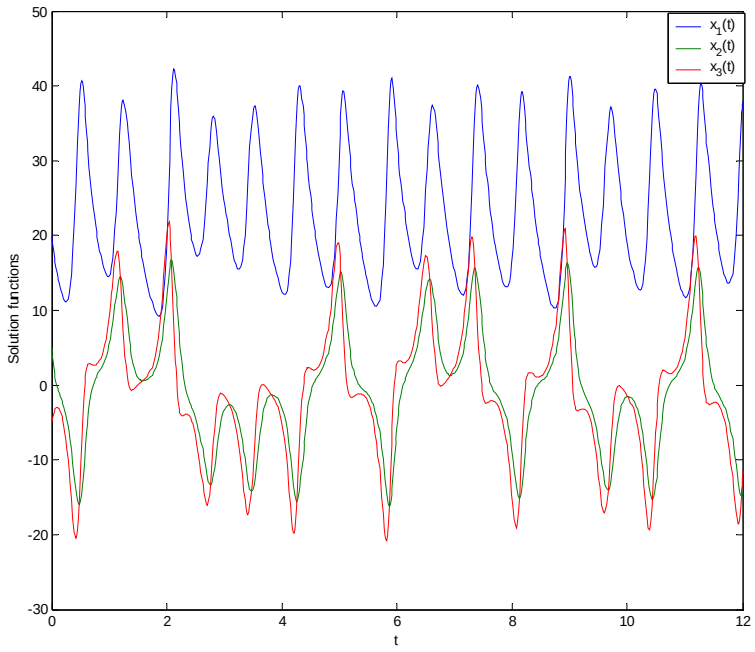
- odefun: το όνομα του m-file που ορίζει τον πίνακα $[F]$. Παρατηρούμε ότι μια και δίνουμε ένα m-file σαν δεδομένο εισόδου, χρειάζεται να βάλουμε το '@' πριν από το όνομα του m-file όταν καλούμε την ode45.
- t_span: το διάνυσμα $[t_0, T]$ όπου ανήκει το t – αυτό πρέπει να δοθεί με αγκύλες
- a: το διάνυσμα $[a_1, a_2, a_3]$ με τις αρχικές τιμές
- t: το διάνυσμα με τα σημεία $t_0, t_1, t_2 ...$
- x: ένας πίνακας του οποίου η κάθε στήλη αντιστοιχεί στις προσεγγιστικές τιμές των λύσεων $[x_1, x_2, x_3]$

Για το παράδειγμα μας έχουμε $t_0 = 0$, $[a_1, a_2, a_3] = [20, 5, -5]$, και ας υποθέσουμε ότι το t ανήκει στο διάστημα [0, 12]. Γράφουμε

```
>> [t,x] = ode45(@odefun,[0,12],[20,5,-5]);
```

με ';' στο τέλος για να μην τυπωθούν οι απαντήσεις στην οθόνη. Για να δούμε τις γραφικές παραστάσεις όλων των λύσεων (στους ίδιους άξονες) γράφουμε:

```
>> plot(t,x)
>> xlabel('t')
>> ylabel('Solution functions')
>> legend('x_1(t)','x_2(t)','x_3(t)')
```



Για να πάρουμε τη γραφική παράσταση μιας από τις λύσεις, π.χ. της $x_2(t)$, γράφουμε

```
>> plot(t,x(:,2))
```

Παράδειγμα 11.3.1

Θεωρούμε το εξής Σ.ΑΤ:

$$\left.\begin{aligned} x_1'(t) &= 2x_1(t) - 2x_1(t)x_2(t) \\ x_2'(t) &= x_1(t)x_2(t) - x_2(t) \end{aligned}\right\} \Leftrightarrow \begin{bmatrix} x_1'(t) \\ x_2'(t) \end{bmatrix} = \begin{bmatrix} 2 & -2x_1(t) \\ x_2(t) & -1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$

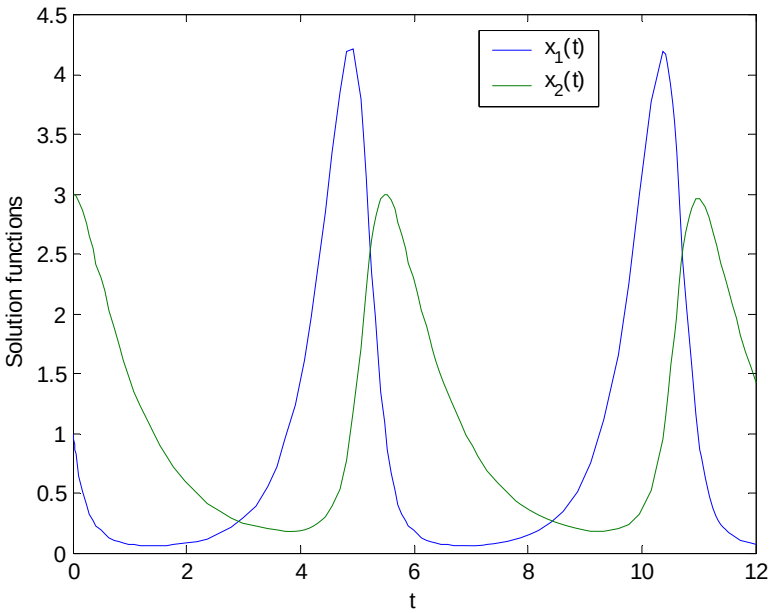
με αρχικές συνθήκες $x_1(0) = 1, x_2(0) = 3$, και $t \in [0, 20]$. Το m-file που ορίζει τον πίνακα του δεξιού μέλους έχει ως εξής:

```
function [xprime] = odefun(t,x)
%  [xprime] = odefun(t,x) -
%
%  This function corresponds to the RHS of the system of ODEs,
%  in which x=[x(1),x(2)] represents the (vector) of unknown
%  functions.
%
%  The function will be passed as input into the ODE solver.
%
xprime = [2, -2*x(1); x(2), -1]*x;
% End of m-file odefun.m
```

Γράφουμε

```
>> [t,x] = ode45(@odefun,[0,12],[1,3]);
>> plot(t,x)
>> legend('x_1(t)','x_2(t)')
>> xlabel('t')
>> ylabel('Solution functions')
```

και παίρνουμε το γράφημα:



11.4 Ασκήσεις

11.1 Θεωρούμε το εξής ΠΑΤ.

$$\begin{cases} y'(t) = \cos(t+y)\,, t \in [0,3] \\ y(0) = 0 \end{cases}$$

του οποίου η ακριβής λύση είναι $y(t) = -t + 2\arctan(t)$. Χρησιμοποιείστε το m-file `euler.m` που γράψαμε, με βήμα $h = 0.1$, για να βρείτε μια προσέγγιση της λύσης. Να κάνετε τη γραφική παράσταση της λύσης που πήρατε μαζί με την ακριβή λύση στους ίδιους άξονες οι οποίοι πρέπει να έχουν ετικέτες, τίτλο και λεζάντα. Επαναλάβετε με βήμα $h = 0.05$ και 0.01.

11.2 Για το ΠΑΤ της προηγούμενης άσκησης, χρησιμοποιείστε την εντολή βιβλιοθήκης ode45 για να βρείτε μια προσέγγιση της λύσης. Να κάνετε τη γραφική παράσταση της λύσης που πήρατε μαζί με την ακριβή λύση στους ίδιους άξονες οι οποίοι πρέπει να έχουν ετικέτες, τίτλο και λεζάντα.

11.3 Θεωρούμε το ΠΑΤ $y'(x) = x - y$, $y(0) = 0$, $x \in [0,1]$, του οποίου η ακριβής λύση είναι $y(x) = e^{-x} + x - 1$. Ορίστε το δεξιό μέλος της ΣΔΕ σαν μια ανώνυμη συνάρτηση *f*, όπως επίσης και το διάνυσμα με τα εξής βήματα

```
>> h=[0.2,0.1,0.05,0.01,0.005];
```

και τρέξτε το m-file `euler.m` που γράψαμε, μέσω του βρόχου

```
>> for i=1:length(h)
     [y,x] = euler(f,0,0,1,h(i));
     yex=exp(-x)+x-1;
     E(i) = max(abs(yex-y));
   end
```

για να υπολογίσετε την προσέγγιση της λύσης (`y`), την ακριβή λύση (`yex`) και το μέγιστο σφάλμα (`E`) μεταξύ της προσέγγισης και της ακριβής λύσης στα σημεία που δίνονται από το διάνυσμα `x`. Στη συνέχεια, κάντε τη γραφική παράσταση του σφάλματος έναντι του βήματος, σε λογαριθμική κλίμακα:

```
>> loglog(h,E,'o-')
```

Τι είναι η κλίση της ευθείας που παίρνετε;

Σημείωση: Η πιο πάνω άσκηση μας δίνει την απάντηση στο ερώτημα 'Πόσο γρήγορα τείνει το σφάλμα στο μηδέν όταν το βήμα τείνει στο μηδέν;' Η εξήγηση έχει ως εξής: Εύκολα μπορούμε να παρατηρήσουμε ότι όταν το $h \to 0$, έχουμε $\max_{0\le x\le 1} |Error| \to 0$. Έστω ότι $E(h) := \max_{0\le x\le 1} |Error| \sim h^p$, για κάποιο $p > 0$ το οποίο καθορίζει την ταχύτητα σύγκλισης της μεθόδου, δηλ. μας λέει πόσο γρήγορα τείνει το σφάλμα στο μηδέν όταν το βήμα τείνει στο μηδέν. Τότε, έχουμε $E(h) \sim h^p \;\Rightarrow\; E(h) \approx Ch^p$ για κάποια σταθερά C (όταν το h είναι αρκετά μικρό). Παίρνοντας τον λογάριθμο, βρίσκουμε

$$\ln(E(h)) \approx \ln(Ch^p) \;\Rightarrow\; \ln(E(h)) \approx \ln(C) + p\ln(h)$$

Αν θέσουμε $Y = \ln(E(h))$, $X = \ln(h)$ και $B = \ln(C)$, τότε έχουμε την σχέση

$$Y = pX + B$$

άρα η γραφική παράσταση του X έναντι στο Y θα είναι ευθεία της οποίας η κλίση θα είναι το p.

11.4 Χρησιμοποιήστε την εντολή ode45 για να λύσετε τα πιο κάτω ΣΑΤ:

(α)
$$\frac{dx_1}{dt} = 3x_1 - 4x_2 \; ; \; x_1(0) = 1$$
$$\frac{dx_2}{dt} = 2x_1 - 3x_2 \; ; \; x_2(0) = 1$$

(β)
$$\frac{dx_1}{dt} = (-0.1)x_1x_2 \quad ; \quad x_1(0) = 10$$
$$\frac{dx_2}{dt} = -x_1 \quad ; \qquad x_2(0) = 15$$

(γ)
$$\frac{dx}{dt} = xz \; ; \; x(0) = 0$$
$$\frac{dy}{dt} = -xz \; ; \; y(0) = 1$$
$$\frac{dz}{dt} = -xy/2 \; ; \; z(0) = 1$$

Βιβλιογραφία

1. *The MathWorks*, http://www.mathworks.com

2. L.W. Fausett, *Applied Numerical Analysis using MATLAB*, Pearson-Prentice Hall, New Jersey, 2008.

3. D.J. Ingham and N.J. Ingham, *MATLAB Guide* (2nd Ed.), SIAM, 2005.

4. C.B. Moler, *Numerical Computing with MATLAB*, SIAM, 2004.

5. H. Moore, *MATLAB for Engineers*, Pearson-Prentice Hall, New Jersey, 2007.

6. Γ.Δ. Ακρίβης και Β.Α. Δουγαλής, *Εισαγωγή στην Αριθμητική Ανάλυση*, Πανεπιστημιακές Εκδόσεις Κρήτης, Ηράκλειο, 2004.

Ευρετήριο

A
abs 12
acos 12
all 111, 113, 114
and 111-113
ans 8, 28
any 111, 113, 114
area 162
asin 12
atan 12
axis 141, 142, 173

B
bar 162-164
bar3 162-164
bar3h 162, 163
barh 162, 163
beep 127
besseli 204
besselj 204-206
besselk 204
bessely 204
blkdiag 56
break 121

C
case 127-130
cat 57
ceil 12, 14, 274
chol 241-244
clc 15, 16
clear 15, 17
clock 20, 21, 119
colorbar 170, 171
colormap 162, 169
comet 148
comet3 175
company 45
cond 67, 248-250
condest 248
continue 121
contour 171
contourf 171

conv 187, 189
cos 12, 138
cross 64

D
date 20, 21
deconv 187, 189
demo 20
det 54, 67, 68, 82, 87
diag 67, 69, 251
diary 15, 19
diff 259, 260, 266
dir 20
disp 22, 82, 207, 208
doc 20, 107, 277
dot 64

E
edit 81
eig 67, 70, 190
elfun 12
else 122-125
elseif 122-125
end 48, 50, 114-122,
eps 28
eq 107
etime 119
exit 3, 15
exp 12, 143, 144, 151, 153
expm 71
eye 42, 56, 59
ezcontour 169, 173
ezcontourf 169, 170, 173
ezplot 143-147, 159
ezsurf 167, 173, 179
ezsurfc 173

F
factorial 121, 126
false 11, 103
fclose 218, 219, 224
feof 219
feval 96, 97, 228, 230, 257, 258, 263, 274
fgetl 224
fgets 224
figure 141

find 109, 110
fix 12, 14, 21
floor 12, 14
fopen 218, 221-224
for 103, 114-119
format 5, 22, 23
format compact 5
format long 6, 13, 24
format loose 23
format short 24
fplot 93, 143, 148, 149, 156, 159, 206
fread 224
frpintf 207, 219, 220
fscanf 221-223
function 81, 84, 85, 90, 228, 230, 236, 237
funm 72
fwrite 224
fzero 206, 231

G
gallery 45
ge 107
ginput 205, 206
global 81
graf3d 169
grid 140, 141, 153, 157, 158, 166
gt 107

H
hadamard 45
hankel 45
help 1, 15, 16
help elfun 12
help elmat 44
help format 25
help matfun 67
help specfun 12
hist 166
hlib 42, 44, 59, 249
hold 141, 154, 169

I
if 103, 122-125
imag 142
inf 28
inline 94, 95

input 22, 26, 27, 83, 123, 203
inv 67, 68, 87, 125
invhilb 42, 44
ischar 108
isempty 108
isequal 107, 108
isfinite 108
isinf 108
isinteger 108
iskeyword 108
isletter 108
islogical 108
isnan 108
isreal 108
isscalar 108
issorted 108
isvarname 108
isvector 108

L
le 107
legend 139-141, 153, 168
length 63, 67, 185, 236, 237, 266
linspace 91, 124, 138, 139, 154, 188, 192, 257, 258
load 15, 19
log 12
log2 16
log10 12
logical 11
loglog 156, 157
logm 71
logspace 139
ls 20
lt 107
lu 238, 239

M
magic 42, 43
max 63, 67, 68
mean 63, 64, 89
median 63
menu 130, 131
mesh 168, 172
meshgrid 167, 169, 171, 179
min 63, 67, 68
mod 12, 13
more on 44

more off 44

N
NaN 28
ne 107
norm 63, 67, 245-247, 251
normest 247
normest1 247
not 111-113
num2str 207, 208

O
ode113 277
ode15s 277
ode23 277
ode23s 277
ode23t 277
ode23tb 277
ode45 277-281
ones 42, 57
or 111-113
otherwise 127-130

P
pascal 42, 43, 244
pause 203, 204
pi 13, 28
pi 165
pi3 165
plot 64, 91, 124, 137-142, 150-157, 159, 180, 193, 194
plot3 174
polar 160, 161
poly 67, 190
polyder 187, 190
polyfit 191-199
polyval 187, 188, 192, 193, 197, 198
prod 63

Q
quad 268, 269
quadl 268, 269
quit 3, 15

R
rand 42, 43, 88, 110
randn 42, 43
rank 67, 68, 87
rcond 248
real 142
relop 111
rem 12, 13
repmat 56
reshape 222
return 121, 125, 126, 263
roots 187-190
rose 166
rosser 45
round 12, 14
rref 67, 70, 233, 234

S
save 15, 19
semilogx 156, 158
semilogy 156, 157
sin 12, 59, 149, 153
size 67, 68, 214
sort 63
specfun 12
sprintf 207-220, 230
sqrt 12, 60
sqrtm 71
std 63
subplot 158, 159, 161, 163, 164, 172, 173, 193, 194
sum 63, 133, 134
surf 167-169, 171, 172, 179
surfc 172
surfl 168
switch 103, 127-130

T
tan 12
text 141
textscan 223
tic 20
title 139-141, 157, 158, 168
toc 20
toeplitz 45
trace 67, 68
trapz 265
tril 67

triu 67
true 11, 103

U

V
vander 45
ver 4
version 3

W
what 20
while 103, 120, 121
who 15, 16, 18
whos 15, 16, 17, 18, 36, 58
wilkinson 45

X
xlabel 139-141, 154, 157, 158, 168
xor 111

Y
ylabel 139-141, 154, 157, 158, 168

Z
zeros 42, 55, 236, 237
zlabel 168