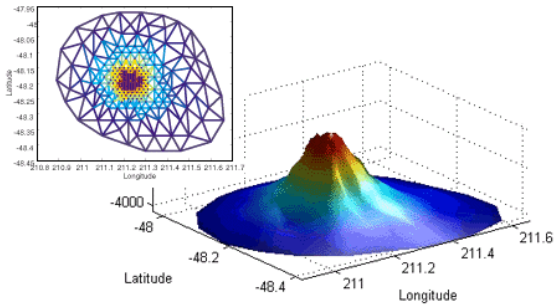
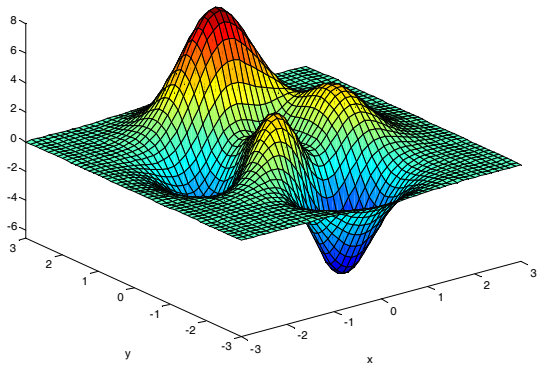
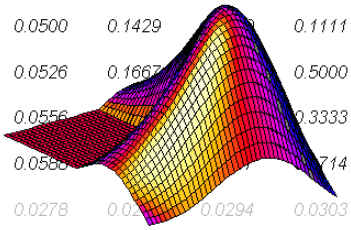
Οδηγός
MATLAB*
για αρχάριους





37	0.0135	0.0133	0.0132	0.0130	0.0128	0.0127
39	0.0233	0.0227	0.0222	0.0217	0.0213	0.0208
41	0.0238	0.0476	0.0455	0.0435	0.0417	0.0400
43	0.0244	0.0500	0.1429		0.1111	0.1000
45	0.0250	0.0526	0.1667		0.5000	0.0909
47	0.0256	0.0556			0.3333	0.0833
49	0.0263	0.0588			0.714	0.0769
52	0.0270	0.0278	0.0	0.0294	0.0303	0.0312
54	0.0156	0.0159	0.0161	0.0164	0.0167	0.0169
00	0.0101	0.0102	0.0103	0.0104	0.0105	0.0106



Χρίστος Ξενοφώντος
Τμήμα Μαθηματικών και Στατιστικής
Πανεπιστήμιο Κύπρου

* MATLAB is a registered trademark of The MathWorks Inc.

ΠΕΡΙΕΧΟΜΕΝΑ

1. *ΕΙΣΑΓΩΓΗ*

Το λογισμικό MATLAB, που παίρνει το όνομά του από τις λέξεις **MAT**rix **LAB**oratory, είναι ένα σύγχρονο ολοκληρωμένο μαθηματικό πακέτο που χρησιμοποιείται εκτενώς στα πανεπιστήμια και στη βιομηχανία. Είναι ένα διαδραστικό (interactive) πρόγραμμα για *αριθμητικούς* υπολογισμούς και για κατασκευή γραφημάτων, αλλά παρέχει επίσης και τη δυνατότητα προγραμματισμού, κάτι που το καθιστά ένα χρησιμότατο εργαλείο για όλους όσους ασχολούνται με τις θετικές επιστήμες (και όχι μόνο). 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 όνομα του, το MATLAB είναι ειδικά σχεδιασμένο για υπολογισμούς με πίνακες, όπως η επίλυση γραμμικών συστημάτων, η εύρεση ιδιοτιμών και ιδιοδιανυσμάτων, η αντιστροφή τετραγωνικού πίνακα 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 κ.α. Η γλώσσα προγραμματισμού του MATLAB δίνει την ευχέρεια στον χρήστη να το επεκτείνει με δικά του προγράμματα. Συχνά θα λέμε 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). Με άλλα λόγια, δεν βρίσκει την ακριβή λύση αλλά μια προσεγγιστική λύση ενός προβλήματος. Αυτή είναι και η βασική του διαφορά από τα συστήματα συμβολικών υπολογισμών όπως το Maple και το Mathematica.

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)!

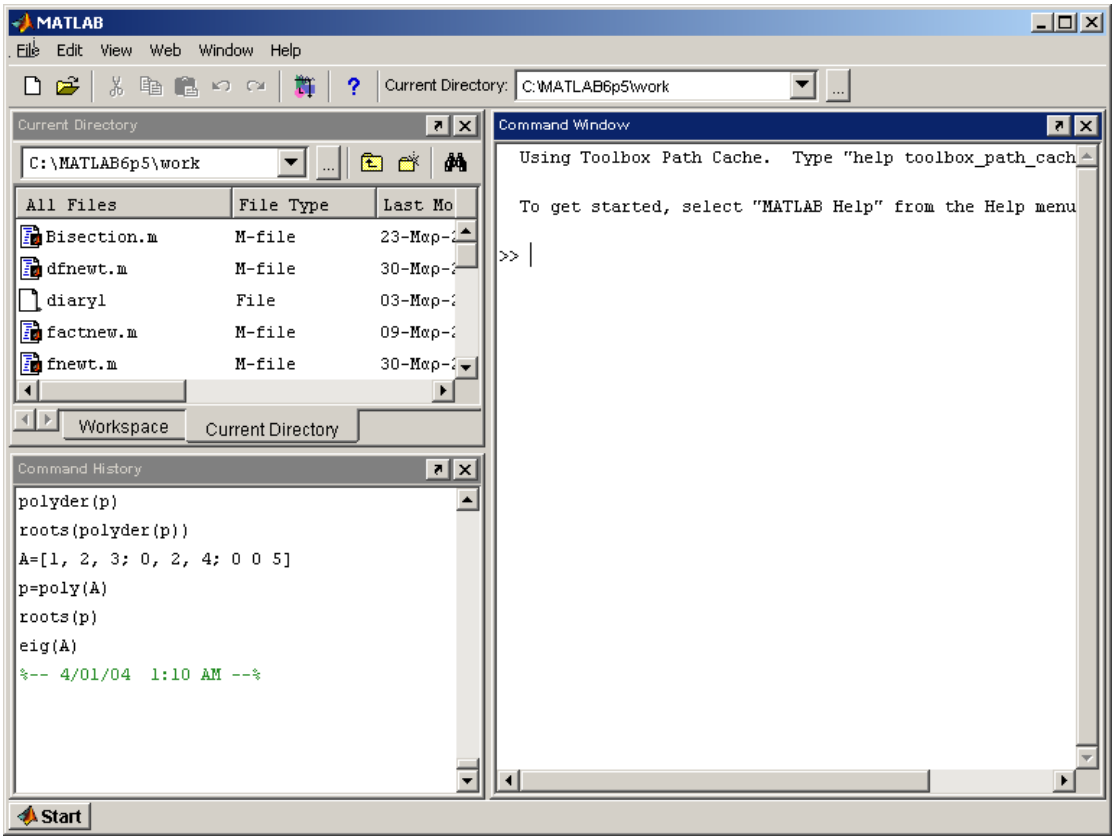
Μια πληθώρα πληροφοριών τόσο για αρχάριους όσο και προχωρημένους είναι διαθέσιμη στην επίσημη ιστοσελίδα της MATLAB:

http://www.mathworks.com

1.1 *Η MATLAB στο Πανεπιστήμιο Κύπρου*

Το λογισμικό πακέτο MATLAB είναι εγκατεστημένο σε διάφορα εργαστήρια του Πανεπιστημίου Κύπρου, π.χ. Β141 στην Πανεπιστημιούπολη. Αφού ξεκινήσουμε τον υπολογιστή, μπορούμε να ξεκινήσουμε το πρόγραμμα με διπλό κλικ πάνω στο εικονίδιο της MATLAB, το οποίο φαίνεται πιο κάτω (για την εκδοχή 6.5):

Μετά από λίγο, αφού το πρόγραμμα φορτώσει, θα εμφανιστεί στην οθόνη μας το παράθυρο έναρξης της MATLAB (MATLAB opening window):



Στο μεγάλο **παράθυρο εντολών** (Command Window) στα δεξιά, εισάγονται οι εντολές της MATLAB μετά την *προτροπή* (prompt) >>**.** Τα αποτελέσματα επίσης τυπώνονται στο παράθυρο αυτό. Μπορούμε να κλείσουμε τα υπόλοιπα παράθυρα, αν θέλουμε, μια και δεν έχουν να κάνουν με τις εντολές (και τα αποτελέσματα που θα πάρουμε), αλλά με την διαχείριση του χώρου εργασίας και των αρχείων (κάτι με το οποίο δεν θα ασχοληθούμε και τόσο σε αυτόν τον οδηγό. Κατά τη διάρκεια μιας εργασίας στη MATLAB μπορεί να εμφανιστούν αυτόματα και άλλα παράθυρα όταν αυτό απαιτείται.

1.2 *Πώς να διαβάσετε αυτόν τον οδηγό*

Οι εντολές της MATLAB θα αναγράφονται ακριβώς μετά από την προτροπή (>>), δηλ. ότι ακολουθεί την προτροπή θα είναι δεδομένα εισόδου (εντολές) που για να επεξεργαστούν θα

πρέπει να ακολουθηθούν με το “enter”. Τα δεδομένα εξόδου, δηλ. η απάντηση που δίνει η MATLAB θα αναγράφονται ακριβώς μετά (εκτός αν καθοριστεί διαφορετικά).

Η MATLAB κάνει διαχωρισμό μεταξύ κεφαλαίων και μικρών γραμμάτων, που σημαίνει ότι, π.χ. το `a + B` είναι διαφορετικό από το `a + b`. Οι διαφορετικές συμβολοσειρές (fonts), όπως αυτές που μόλις είδατε, θα χρησιμοποιούνται για να διαχωρίσουν αυτά που γράφουμε στον υπολογιστή με αυτά που διαβάζετε σε αυτόν τον οδηγό. Παραθέτουμε ένα παράδειγμα:

Η MATLAB μπορεί να δουλέψει όπως ένας υπολογιστής χειρός. Αν ζητήσουμε από τη MATLAB να προσθέσει δύο αριθμούς, τότε παίρνουμε την αναμενόμενη απάντηση.

```
>> 3 + 4

ans =

     7
```

Όπως θα δείτε, η MATLAB κάνει πολύ περισσότερα από το να δουλεύει όπως ένας υπολογιστής χειρός. Για να επωφεληθείτε στο μέγιστο από αυτόν το οδηγό, πρέπει να δοκιμάσετε όλες τις εντολές και να κάνετε όλες τις ασκήσεις στο τέλος κάθε εδαφίου. Εισηγούμαι να διαβάσετε πρώτα το κάθε εδάφιο και μετά να δοκιμάσετε τις εντολές, ίσως περισσότερο από μία φορά.

2. *ΤΑ ΒΑΣΙΚΑ ΤΗΣ MATLAB*

2.1 *Βασικά στοιχεία*

Ας αρχίσουμε με κάτι απλό, όπως για παράδειγμα πως μπορούμε να ορίσουμε ένα διάνυσμα γραμμής που να περιέχει τους αριθμούς 1, 2, 3, 4, 5 και να το καταχωρήσουμε στη μεταβλητή *x*. Γράφουμε

```
>> x = [1 2 3 4 5]
```

και παίρνουμε

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

Παρατηρούμε ότι το σύμβολο " = " χρησιμοποιείται για καταχώρηση δεδομένων σε μια μεταβλητή και οι (τετραγωνικές) παρενθέσεις " [ ] " χρησιμοποιούνται για να εσωκλείσουν τα στοιχεία του διανύσματος, τα οποία διαχωρίζονται με κενό (space). Αυτό ακριβώς θα κάναμε και με το χέρι στο μάθημα της γραμμικής άλγεβρας. Θα μπορούσαμε, αντί για κενό, να χρησιμοποιήσουμε κόμμα (,) μεταξύ των στοιχείων του διανύσματος.

Για να ορίσουμε ένα διάνυσμα στήλης (κάτι που στη MATLAB διαφέρει από το διάνυσμα γραμμής) χρησιμοποιούμε ερωτηματικό (;) αντί για κόμμα (,) μεταξύ των στοιχείων του διανύσματος. Φυσικά, θα μπορούσαμε να ορίσουμε πρώτα ένα διάνυσμα γραμμής και να πάρουμε το ανάστροφο (transpose) του, με την εντολή `'`. Ας δούμε και τους δύο τρόπους μέσω ενός παραδείγματος:

```
>> y = [6;7;8;9;10]

y =
     6
     7
     8
     9
    10

>> y = [6,7,8,9,10]

y =
     6     7     8     9    10

>> y'

ans =
     6
     7
     8
     9
    10
```

Παρατηρούμε ότι η MATLAB τυπώνει το αποτέλεσμα στην οθόνη αμέσως μετά από την εντολή που γράψαμε. Αν για κάποιο λόγο δεν θέλουμε να δούμε την απάντηση τότε γράφουμε ερωτηματικό (;) στο τέλος της εντολής (πριν πατήσουμε το enter).

Επίσης, η MATLAB αυτόματα καταχωρεί την απάντηση που μας δίνει στη μεταβλητή `ans` (όταν εμείς δεν επιλέξουμε κάποιο άλλο όνομα). Πρέπει να σημειωθεί ότι αυτή η μεταβλητή ανακυκλώνεται, έτσι κάθε φορά που ζητάμε μια απάντηση χωρίς να της δώσουμε όνομα, η MATLAB της δίνει το όνομα `ans`.

Είναι καλή ιδέα να ξέρουμε ανά πάσα στιγμή ποιες μεταβλητές έχουν οριστεί και κατέχουν μέρος της μνήμης του χώρου εργασίας μας. Αυτό μπορεί να γίνει με την εντολή

```
>> whos
```

η οποία δίνει ως απάντηση κάτι που μοιάζει με τα πιο κάτω.

```
Name        Size       Elements    Bytes     Density    Complex

ans         5 by 1        5          40        Full        No
x           1 by 5        5          40        Full        No
y           1 by 5        5          40        Full        No

Grand total is 15 elements using 120 bytes
```

Μια άλλη παρόμοια εντολή είναι η `who`, η οποία δίνει μόνο τα ονόματα των ενεργών μεταβλητών.

```
>> who

Your variables are:

ans         x           y
```

Αν θέλουμε να σβήσουμε κάποια μεταβλητή από τη μνήμη, χρησιμοποιούμε την εντολή `clear variable_name`. Αν πούμε απλώς `clear`, τότε σβήνονται όλες οι μεταβλητές – γι'αυτό προσοχή στη χρήση αυτής της εντολής. Ας χρησιμοποιήσουμε την εντολή `clear` για να σβήσουμε την μεταβλητή `ans` :

```
>> clear ans
```

Ελέγχουμε αν πράγματι η μεταβλητή έχει σβηστεί:

```
>> who

Your variables are:

x           y
```

Για να βγούμε από το πρόγραμμα γράφουμε `quit` στην προτροπή. Όταν το κάνουμε αυτό, όλες οι μεταβλητές που έχουμε ορίσει θα εξαφανιστούν. Αν, όμως, γράψουμε `save filename` πριν το `quit` τότε όλες οι μεταβλητές (και οι τιμές τους) γράφονται στο αρχείο

`filename.mat`. Την επόμενη φορά που θα ξεκινήσουμε την MATLAB μπορούμε να καλέσουμε αυτό το αρχείο με την εντολή `load filename`, και έτσι θα έχουμε και πάλι όλες τις (παλιές) μεταβλητές στη διάθεση μας.

Να αναφέρουμε μια τελευταία εντολή, πριν αρχίσουμε να μαθαίνουμε πιο ενδιαφέροντα πράγματα. Είναι η εντολή `help` η οποία μας δίνει βοήθεια για τις άλλες εντολές (δηλ. μας λέει πως δουλεύει μια συγκεκριμένη εντολή). Για παράδειγμα

```
>> help who

 WHO List current variables.
     WHO lists the variables in the current workspace.
     WHOS lists more information about each variable.
     WHO GLOBAL and WHOS GLOBAL list the variables in the
     global workspace.
```

Δοκιμάστε την εντολή `help help` για να δείτε πως δουλεύει η εντολή `help`.
(Στους υπολογιστές που τρέχουν *Windows* μπορούμε, επίσης, να βρούμε βοήθεια μέσω των μενού στο πάνω μέρος του παραθύρου της MATLAB).

2.2 *Διανύσματα και πίνακες*

Έχουμε ήδη δει πώς ορίζουμε διανύσματα και πως τα καταχωρούμε σε μεταβλητές. Πολλές φορές θέλουμε να ορίσουμε διανύσματα (ή και πίνακες) των οποίων τα στοιχεία ισαπέχουν μεταξύ τους. Αυτό μπορεί να γίνει πολύ εύκολα, φτάνει να καθορίσουμε το πρώτο στοιχείο, το βήμα και το τελευταίο στοιχείο. Αν το βήμα είναι 1, τότε δεν χρειάζεται να το συμπεριλάβουμε. Για παράδειγμα, για να ορίσουμε το διάνυσμα u με στοιχεία τους αριθμούς 0, 1, 2, 3, …, 7, 8, γράφουμε

```
>> u = [0:8]
```

και παίρνουμε

```
u =
     0     1     2     3     4     5     6     7     8
```

Βλέπουμε ότι γράψαμε (μέσα σε τετραγωνικές παρενθέσεις) το πρώτο στοιχείο 0, την άνω και κάτω τελεία (:) και το τελευταίο στοιχείο 8. Η MATLAB έκανε τα υπόλοιπα.

Ας ορίσουμε το διάνυσμα v με στοιχεία τους αριθμούς 0, 2, 4, 6, και 8.

```
>> v = [0:2:8]

v =
     0     2     4     6     8
```

Εδώ καθορίσαμε το βήμα 2 μεταξύ του πρώτου και τελευταίου στοιχείου, έτσι πήραμε το ζητούμενο διάνυσμα. Σημειώστε ότι το βήμα μπορεί να είναι και αρνητικό.

Η MATLAB μας επιτρέπει να αποσπάσουμε μέρος κάποιου διανύσματος με τη χρήση της άνω και κάτω τελείας (:). Για παράδειγμα, το διάνυσμα που αποτελείται από τα πρώτα 3 στοιχεία του διανύσματος v, που ορίσαμε πριν, είναι

```
>> v(1:3)

ans =
     0     2     4
```

Παρατηρούμε ότι χρησιμοποιούμε κυκλικές παρενθέσεις, αντί για τετραγωνικές, όταν αναφερόμαστε στα στοιχεία ενός ήδη ορισμένου διανύσματος. Στο συγκεκριμένο παράδειγμα παραλείψαμε το βήμα, έτσι η MATLAB αυτόματα χρησιμοποίησε βήμα ίσο με 1. Η πιο κάτω εντολή δίνει τα 4 πρώτα στοιχεία του v, αλλά με βήμα 2 (δηλ. τα στοιχεία στις θέσεις 1 και 3):

```
>> v(1:2:4)

ans =
     0     4
```

Ο ορισμός ενός πίνακα είναι παρόμοιος με αυτόν των διανυσμάτων. Για να ορίσουμε ένα πίνακα Α, ορίζουμε ένα διάνυσμα στήλης του οποίου τα στοιχεία είναι διανύσματα γραμμής. Συγκεκριμένα, ορίζουμε τον πίνακα γραμμή με γραμμή, ξεχωρίζοντας την κάθε μια με ερωτηματικό (;). Ο ορισμός της κάθε γραμμής γίνεται όπως ένα διάνυσμα, δηλ. βάζουμε κόμμα (,) ή κενό μεταξύ των στοιχείων της κάθε γραμμής. Για παράδειγμα,

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

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

Αντί για ερωτηματικό, μπορούμε να ξεχωρίσουμε τις γραμμές με enter. Δηλαδή, μπορούσαμε να ορίσουμε τον Α σαν

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

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

το οποίο, ίσως μοιάζει πιο πολύ με τον τρόπο που θα ορίζαμε τον Α με το χέρι.

Μπορούμε να αναφερθούμε σε κάποιο στοιχείο του Α χρησιμοποιώντας κυκλικές παρενθέσεις – αυτή τη φορά, έχουμε δύο αριθμούς που χαρακτηρίζουν το στοιχείο, ένα για τη γραμμή και ένα για τη στήλη. Για παράδειγμα, το στοιχείο στη $2^{η}$ γραμμή και $3^{η}$ στήλη του Α είναι

```
>> A(2,3)

ans =
     5
```

(Πρώτα γράφουμε τη γραμμή και μετά τη στήλη. Για αυτό το λόγο, το στοιχείο στη $3^{η}$ γραμμή και $2^{η}$ στήλη είναι ο αριθμός 7.

Δείτε τι δίνει η MATLAB όταν ζητήσουμε το στοιχείο στη $4^{η}$ γραμμή και $1^{η}$ στήλη.

```
>> A(4,1)
???  Index exceeds matrix dimensions.
```

Όπως ήταν αναμενόμενο, πήραμε λάθος, μια και ο A είναι ένας 3×3 πίνακας και δεν έχει $4^{η}$ γραμμή. Τα λάθη που μας δίνει η MATLAB είναι αρκετά πληροφοριακά και σε αυτή τη περίπτωση μας είπε ακριβώς πιο ήταν το πρόβλημα.

Μπορούμε, επίσης, να αναφερθούμε σε υποπίνακες χρησιμοποιώντας τον πιο πάνω τρόπο σε συνδυασμό με την άνω και κάτω τελεία. Ο υποπίνακας του A που αποτελείται από τις γραμμές ένα και δύο, και από τις στήλες δύο και τρία, είναι

```
>> A(1:2,2:3)

ans =
     2     3
     4     5
```

Για να αναφερθούμε σε μια ολόκληρη γραμμή (ή στήλη) απλώς γράφουμε την άνω και κάτω τελεία (χωρίς αρχικό και τελικό αριθμό). Για παράδειγμα, αν θέλουμε να αναφερθούμε στην δεύτερη στήλη του A λέμε

```
>> A(:,2)

ans =
     2
     4
     7
```

όπου η άνω και κάτω τελεία (:) χρησιμοποιήθηκε για να πει στη MATLAB ότι όλες οι γραμμές θα συμπεριληφθούν. Το ίδιο μπορεί να γίνει για να δούμε μια ολόκληρη γραμμή, για παράδειγμα την $3^{η}$:

```
>> A(3,:)

ans =
     6     7     8
```

Ας ορίσουμε τώρα έναν άλλο πίνακα B, και δύο διανύσματα *s* και *t* που θα χρησιμοποιηθούν πιο κάτω.

```
>> B = [-1 3 10; -9 5 25; 0 14 2]
```

```
B =
    -1     3    10
    -9     5    25
     0    14     2

>> s = [-1 8 5]

s =
    -1     8     5

>> t = [7;0;11]

t =
     7
     0
    11
```

Μια από τις σημαντικότερες δυνατότητες της MATLAB είναι η ευκολία με την οποία μπορούμε να χειριστούμε πίνακες και διανύσματα. Για παράδειγμα, για να αφαιρέσουμε 1 από όλα τα στοιχεία του A λέμε

```
>> A-1

ans =
     0     1     2
     2     3     4
     5     6     7
```

Η προσθαφαίρεση δύο συμβατών πινάκων είναι εξίσου εύκολη. Για παράδειγμα, για να προσθέσουμε τους A και B γράφουμε

```
>> A+B

ans =
     0     5    13
    -6     9    30
     6    21    10
```

Το ίδιο ισχύει και για διανύσματα.

```
>> s-t
??? Error using ==> -
Matrix dimensions must agree.
```

Το λάθος που πήραμε ήταν αναμενόμενο, μια και το διάνυσμα *s* είναι 1×3 ενώ το *t* έιναι 3×1. Δεν θα πάρουμε λάθος αν γράψουμε

```
>> s-t'

ans =
    -8     8    -6
```

μια και παίρνοντας το ανάστροφο διάνυσμα του *t* κάνει τα διανύσματα συμβατά.

Πρέπει να είμαστε εξίσου προσεκτικοί όταν πολλαπλασιάζουμε πίνακες ή/και διανύσματα.

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

>> B*t

ans =

   103
   212
    22
```

Μια άλλη σημαντική εντολή είναι αυτή που μας επιτρέπει να λύσουμε γραμμικά συστήματα των οποίων ο πίνακας συντελεστών είναι αντιστρέψιμος[†] . Αν ο M είναι ένας τετραγωνικός, αντιστρέψιμος πίνακας και *b* είναι ένα συμβατό διάνυσμα, τότε

`x = M\b` είναι η λύση του συστήματος M *x* = *b* και
`x = b/M` είναι η λύση του συστήματος *x* M = *b*.

Ας δούμε τις πιο πάνω πράξεις στη περίπτωση που M = B και *b* = *t*.

```
>> x=B\t

x =
    2.4307
    0.6801
    0.7390
```

Το *x* είναι η λύση του συστήματος B *x* = *t* όπως φαίνεται και στην πιο κάτω επαλήθευση.

```
>> B*x

ans =
    7.0000
    0.0000
   11.0000
```

Μια και το *x* αποτελείται από μη-ακέραιους, αξίζει να αναφέρουμε σε αυτό το σημείο, την εντολή `format long`. Η MATLAB τυπώνει στην οθόνη μόνο τέσσερα δεκαδικά ψηφία εκτός αν γράψουμε πρώτα `format long`, το οποίο αναγκάζει την MATLAB να μας δώσει 14 δεκαδικά ψηφία.

```
>> format long
>> x

x =
   2.43071593533487
   0.68013856812933
   0.73903002309469
```

[†] Ανακαλούμε ότι ένας πίνακας $M \in \mathbb{R}^{n\times n}$ καλείται *αντιστρέψιμος αν* $Mx = 0 \quad \Rightarrow \quad x = 0 \quad \forall\, x \in \mathbb{R}^n$.

Πολλές φορές θέλουμε να πραγματοποιήσουμε μια πράξη σε όλα τα στοιχεία ενός διανύσματος (ή πίνακα). Η MATLAB μας το επιτρέπει, εφόσον της το ζητήσουμε. Για παράδειγμα, ας υποθέσουμε ότι θέλουμε να πολλαπλασιάσουμε όλα τα στοιχεία του διανύσματος *s* με τον εαυτό τους, δηλ. θέλουμε να υπολογίσουμε το διάνυσμα $s^2 = [s(1)*s(1), s(2)*s(2), s(3)*s(3)]$.

Η εντολή `s*s` δεν θα δουλέψει λόγω έλλειψης συμβατότητας. Πρέπει, λοιπόν, να πούμε στη MATLAB να κάνει τον πολλαπλασιασμό κατά-στοιχεία. Αυτό επιτυγχάνεται με τα σύμβολα "`.*`". Για την ακρίβεια, κάθε φορά που βάζουμε τελεία (.) πριν από κάποια πράξη, η MATLAB κάνει τους υπολογισμούς κατά-στοιχεία.

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

>> s.*s

ans =
     1    64    25
```

Θα μπορούσαμε να υψώσουμε το *s* στη $2^η$ δύναμη, παίρνοντας το ίδιο αποτέλεσμα. (Η τελεία είναι και σε αυτή τη περίπτωση απαραίτητη.)

```
>> s.^2

ans =
     1    64    25
```

Η πιο κάτω πίνακας δίνει τα σύμβολα για τις πιο συνηθισμένες πράξεις που μπορούμε να κάνουμε στη MATLAB.

+	πρόσθεση
-	αφαίρεση
*	πολλαπλασιασμός
^	δύναμη
'	ανάστροφος
\	αριστερή διαίρεση
/	δεξιά διαίρεση

Υπενθυμίζουμε ότι ο πολλαπλασιασμός, η ύψωση σε δύναμη και η διαίρεση μπορούν να γίνουν κατά-στοιχεία αφού βάλουμε τελεία (.) πριν από το κάθε σύμβολο.

Ασκήσεις

Ορίστε τα πιο κάτω και χρησιμοποιήστε τα για να απαντήσετε τις ερωτήσεις.

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

1. Υπολογίστε τα εξής (όπου ορίζονται)

(a) $A \cdot b$ **(b)** $a + 4$ **(c)** $b \cdot a$ **(d)** $a \cdot b^{\mathrm{T}}$ **(e)** $A \cdot a^{\mathrm{T}}$

2. Εξηγήστε τις διαφορές στις απαντήσεις που δίνει η MATLAB όταν γράψουμε `A*A, A^2` και `A.^2`.

3. Ποια εντολή δίνει τον υποπίνακα που αποτελείται από τη $2^{η}$ μέχρι τη $3^{η}$ γραμμή του A?

4. Λύστε το σύστημα $A\,x = b$ ως προς x. Επαληθεύσετε την απάντησή σας.

2.3 *Συναρτήσεις βιβλιοθήκης*

Υπάρχουν πάρα πολλές συναρτήσεις βιβλιοθήκης (δηλ. εντολές) στη MATLAB. Εδώ θα δούμε μερικές από αυτές, χωρίζοντας τις σε κατηγορίες.

Βαθμωτές συναρτήσεις

Οι εντολές σε αυτή τη κατηγορία χρησιμοποιούνται για βαθμωτές ποσότητες και όταν χρησιμοποιηθούν σε διανύσματα (ή πίνακες) λειτουργούν κατά-στοιχεία. Οι πιο βασικές δίδονται στον πιο κάτω πίνακα.

`sin`	Ημίτονο
`cos`	Συνημίτονο
`tan`	Εφαπτομένη
`asin`	Ημίτονο τόξου
`acos`	Συνημίτονο τόξου
`atan`	Εφαπτομένη τόξου
`exp`	Εκθετική συνάρτηση
`log`	Λογάριθμος βάσης *e*
`abs`	Απόλυτη τιμή
`sqrt`	Τετραγωνική ρίζα
`rem`	Υπόλοιπο
`round`	Στρογγυλοποίηση προς τον πλησιέστερο ακέραιο
`floor`	Στρογγυλοποίηση προς τα κάτω
`ceil`	Στρογγυλοποίηση προς τα πάνω

Εισηγούμαστε να ζητήσετε βοήθεια (help) μέσω της MATLAB για την κάθε συνάρτηση βιβλιοθήκης του πιο πάνω πίνακα. Εμείς θα δούμε ορισμένες από αυτές σε παραδείγματα.

Οι τριγωνομετρικές συναρτήσεις παίρνουν σαν δεδομένο εισόδου τιμές σε ακτίνια (radians). Η MATLAB αναγνωρίζει τον αριθμό π = 3.1415… , ο οποίος γράφεται σαν `pi`. Άρα,

```
>> sin(pi/2)

ans =
     1

>> cos(pi/2)

ans =
    6.1230e-017
```

Το ημίτονο του π/2 είναι πράγματι 1 αλλά αναμέναμε να δούμε ότι το συνημίτονο του π/2 είναι 0. Μην ξεχνάτε ότι η MATLAB επεξεργάζεται τα δεδομένα με *αριθμητική πεπερασμένης ακρίβειας* και έτσι η απάντηση που πήραμε είναι *σχεδόν* 0 (`6.1230e-017` $= 6.1230\times10^{-17} \approx 0$). Αυτό δεν επηρεάζει δυσμενώς τους υπολογισμούς μας φτάνει να το γνωρίζουμε και να το θυμόμαστε πάντα.

Οι συναρτήσεις `exp` και `log` είναι αυτονόητες, έτσι ας δούμε ένα παράδειγμα με την συνάρτηση `rem` η οποία δίνει το υπόλοιπο μιας διαίρεσης. Επομένως, το υπόλοιπο του 12 δια 4 είναι 0

```
>> rem(12,4)

ans =
     0
```

και το υπόλοιπο του 12 δια 5 είναι 2

```
>> rem(12,5)

ans =
     2
```

Οι συναρτήσεις `floor`, `ceil` και `round` δίνουν τα εξής:

```
>> floor(1.4)

ans =
    1

>> ceil(1.4)

ans =
    2

>> round(1.4)

ans =
    1
```

Υπενθυμίζουμε ότι οι πιο πάνω εντολές λειτουργούν κατά-στοιχεία όταν χρησιμοποιηθούν σε διανύσματα (ή πίνακες). Για παράδειγμα, αν x = [0, 0.1, 0.2, . . ., 0.9, 1], τότε $y = \exp(x)$

θα δώσει ένα άλλα διάνυσμα y , του ιδίου μεγέθους όπως το x, και του οποίου τα στοιχεία δίδονται από $y = [e^0, e^{0.1}, e^{0.2}, \ldots, e^1]$.

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

x =

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

>> y = exp(x)

y =

Columns 1 through 7

1.0000  1.1052  1.2214  1.3499  1.4918  1.6487  1.8221

Columns 8 through 11

2.0138  2.2255  2.4596  2.7183
```

Αυτό θα μας είναι χρήσιμο όταν κατασκευάσουμε γραφήματα (βλ. την ενότητα 2.4).

Παρατηρείστε επίσης ότι η MATLAB έδωσε το αποτέλεσμα τυπώνοντας ένα πίνακα 1×11 (ή ένα διάνυσμα γραμμής με 11 στοιχεία). Μια και δεν χώρεσαν τα αποτελέσματα σε μια γραμμή στην οθόνη, η MATLAB τύπωσε και τους αριθμούς των στηλών.

Συναρτήσεις για διανύσματα

Κάποιες συναρτήσεις βιβλιοθήκης της MATLAB λειτουργούν πάνω σε διανύσματα, δίδοντας ως απάντηση μια βαθμωτή ποσότητα. Ορισμένες από αυτές φαίνονται στον πιο κάτω πίνακα.

`max`	Μέγιστο στοιχείο
`min`	Ελάχιστο στοιχείο
`length`	Μήκος διανύσματος
`sort`	Αύξουσα κατάταξη
`sum`	Άθροισμα στοιχείων διανύσματος
`prod`	Γινόμενο στοιχείων διανύσματος
`median`	Διάμεση τιμή στοιχείων διανύσματος
`mean`	Μέση τιμή στοιχείων διανύσματος
`std`	Τυπική απόκλιση στοιχείων διανύσματος

Και πάλι, εισηγούμαι να ζητήσετε βοήθεια (`help`) για όλες τις πιο πάνω συναρτήσεις. Ας δούμε ορισμένες από αυτές μέσω παραδειγμάτων.

Ορίζουμε το διάνυσμα *z* ως

```
>> z = [0.9347,0.3835,0.5194,0.8310]

z =
    0.9347    0.3835    0.5194    0.8310
```

Τότε

```
>> max(z)

ans =
    0.9347

>> min(z)

ans =
    0.3835

>> sort(z)

ans =
    0.3835    0.5194    0.8310    0.9347

>> sum(z)

ans =
    2.6686

>> mean(z)

ans =
    0.6671
```

Οι συναρτήσεις βιβλιοθήκης για διανύσματα μπορούν να χρησιμοποιηθούν και σε πίνακες. Σε αυτή την περίπτωση λειτουργούν κατά-στήλες, και η απάντηση δίδεται ως ένα διάνυσμα γραμμής. Ας δούμε ένα παράδειγμα.

Έστω ο πίνακας

```
>> M = [
0.7012,0.2625,0.3282
0.9103,0.0475,0.6326
0.7622,0.7361,0.7564];
```

Η εντολή `max(M)` θα δώσει ως απάντηση ένα διάνυσμα του οποίου τα στοιχεία αντιστοιχούν στο μέγιστο στοιχείο της κάθε στήλης του M:

```
>> max(M)
```

```
ans =
    0.9103    0.7361    0.7564
```

Αν τώρα πούμε `max(ans)` τότε παίρνουμε το μέγιστο στοιχείο του πίνακα:

```
>> max(ans)

ans =
    0.9103
```

Τα δύο αυτά βήματα μπορούν να συνδυαστούν:

```
>> max(max(M))

ans =
    0.9103
```

Ο συνδυασμός διαφόρων συναρτήσεων/εντολών της MATLAB μπορεί να φανεί πολύ χρήσιμος όταν γράφουμε πολύπλοκα προγράμματα. Θα πούμε περισσότερα για αυτό στην ενότητα 3.

Συναρτήσεις βιβλιοθήκης για πίνακες

Αυτές οι συναρτήσεις είναι που δίνουν την πιο πολλή ευελιξία στη MATLAB, και μπορούν να χωριστούν σε δυο υποκατηγορίες. Η πρώτη υποκατηγορία είναι οι συναρτήσεις που βοηθούν στην κατασκευή (και διαχείριση) πινάκων, και ορισμένες τέτοιες συναρτήσεις φαίνονται πιο κάτω.

`eye`	Ταυτοτικός πίνακας
`zeros`	Μηδενικός πίνακας
`ones`	Μοναδιαίος πίνακας
`diag`	Εξαγωγή της διαγωνίου ενός πίνακα ή κατασκευή διαγώνιου πίνακα
`triu`	Το άνω τριγωνικό μέρος ενός πίνακα
`tril`	Το κάτω τριγωνικό μέρος ενός πίνακα
`rand`	(ψευδο)τυχαίος πίνακας

Μην ξεχάσετε να ζητήσετε βοήθεια για τις πιο πάνω εντολές.

Για να κατασκευάσουμε τον 4×4 ταυτοτικό πίνακα (δηλ. ένα 4×4 πίνακα με 1 στη κύρια διαγώνιο και 0 σε όλα τα άλλα στοιχεία), λέμε

```
>> eye(4,4)

ans =
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

Οι αριθμοί στην παρένθεση προσδιορίζουν το αριθμό γραμμών και στηλών, αντίστοιχα. Αν ο πίνακας είναι τετραγωνικός, τότε μπορούμε να δώσουμε μόνο ένα αριθμό για το μέγεθος, π.χ. `eye(4)`. Το ίδιο ισχύει και για τις επόμενες δύο συναρτήσεις βιβλιοθήκης.

Η εντολή `zeros` κατασκευάζει ένα πίνακα με μηδενικά, και η εντολή `ones` κατασκευάζει ένα πίνακα με τον αριθμό 1 σε όλες τις θέσεις.

```
>> zeros(2,3)

ans =
     0     0     0
     0     0     0

>> ones(2)

ans =
     1     1
     1     1
```

Μπορούμε να κατασκευάσουμε ένα (ψευδο)τυχαίο πίνακα με την εντολή `rand`. (Τα στοιχεία θα είναι ομοιόμορφα κατανεμημένα μεταξύ του 0 και 1.)

```
>> C = rand(5,4)

C =
    0.2190    0.3835    0.5297    0.4175
    0.0470    0.5194    0.6711    0.6868
    0.6789    0.8310    0.0077    0.5890
    0.6793    0.0346    0.3834    0.9304
    0.9347    0.0535    0.0668    0.8462
```

Οι εντολές `triu` και `tril`, αποσπούν το άνω και κάτω τριγωνικό μέρος, αντίστοιχα, ενός πίνακα. Για παράδειγμα.

```
>> triu(C)

ans =
    0.2190    0.3835    0.5297    0.4175
         0    0.5194    0.6711    0.6868
         0         0    0.0077    0.5890
         0         0         0    0.9304
         0         0         0         0

>> tril(C)

ans =
    0.2190         0         0         0
    0.0470    0.5194         0         0
    0.6789    0.8310    0.0077         0
    0.6793    0.0346    0.3834    0.9304
    0.9347    0.0535    0.0668    0.8462
```

Παρατηρούμε ότι η MATLAB “γέμισε τις άδειες θέσεις” με μηδενικά.

Όπως αναφέραμε και προηγουμένως η εντολή `diag` έχει δύο χρήσεις. Η πρώτη είναι για να αποσπάσουμε την κύρια διαγώνιο ενός πίνακα. Έστω

```
>> D = [
0.9092 0.5045 0.9866
0.0606 0.5163 0.4940
0.9047,0.3190,0.2661];
```

Τότε,

```
>> diag(D)

ans =
    0.9092
    0.5163
    0.2661
```

δίνει ένα διάνυσμα στήλης του οποίου τα στοιχεία είναι τα ίδια με αυτά στην κύρια διαγώνιο του D.

Η δεύτερη χρήση της εντολής `diag` είναι για την κατασκευή διαγώνιων πινάκων. Για παράδειγμα,

```
>> diag([0.9092;0.5163;0.2661])

ans =
    0.9092         0         0
         0    0.5163         0
         0         0    0.2661
```

δηλ., πήραμε ένα διαγώνιο πίνακα του οποίου τα στοιχεία είναι τα ίδια με αυτά που δώσαμε σαν δεδομένο εισόδου στην εντολή `diag`. Δοκιμάστε την εντολή `diag(diag(D))` για να δείτε τι δίνει.

Η δεύτερη υποκατηγορία συναρτήσεων βιβλιοθήκης για πίνακες περιλαμβάνει εντολές που είτε δίνουν πληροφορίες για κάποιο πίνακα, ή επεξεργάζονται τον πίνακα με κάποιο τρόπο. Ο πιο κάτω πίνακας περιέχει ορισμένες από αυτές τις εντολές.

`size`	Μέγεθος ενός πίνακα
`det`	Ορίζουσα ενός πίνακα
`inv`	Αντίστροφος ενός πίνακα
`rank`	Βαθμός ενός πίνακα
`rref`	Κλιμακωτή μορφή ενός πίνακα (reduced row echelon form)
`eig`	Ιδιοτιμές (και ιδιοδιανύσματα) ενός πίνακα
`poly`	Χαρακτηριστικό πολυώνυμο ενός πίνακα
`norm`	Νόρμα ενός πίνακα (1-νορμα, 2-νόρμα, ∞ -νόρμα)
`cond`	Δείκτης κατάστασης ενός πίνακα (στη 2-νόρμα)
`lu`	Παραγοντοποίηση LU
`qr`	Παραγοντοποίηση QR
`chol`	Παραγοντοποίηση Cholesky
`svd`	Ανάλυση ιδιαζουσών τιμών (singular value decomposition)

Ως συνήθως, ζητήστε βοήθεια για τις πιο πάνω εντολές. Για να δούμε πως χρησιμοποιούνται κάποιες από αυτές, ορίζουμε τον εξής πίνακα.

```
>> A = [9,7,0;0,8,6;7,1,-6]

A =
     9     7     0
     0     8     6
     7     1    -6
```

Έχουμε

```
>> size(A)

ans =
     3     3

>> det(A)

ans =
  -192
```

Μια και η ορίζουσα του A δεν είναι 0, μπορούμε να πούμε

```
>> inv(A)

ans =
    0.2812   -0.2187   -0.2187
   -0.2187    0.2812    0.2812
    0.2917   -0.2083   -0.3750
```

Ας ελέξουμε ότι πράγματι $AA^{-1} = I$ and $A^{-1}A = I$.

```
>> A*inv(A)

ans =
    1.0000    0.0000    0.0000
    0.0000    1.0000    0.0000
    0.0000    0.0000    1.0000

>> inv(A)*A

ans =
    1.0000    0.0000         0
    0.0000    1.0000         0
    0.0000         0    1.0000
```

Όπως ήδη αναφέραμε, η MATLAB χρησιμοποιεί αριθμητική πεπερασμένης ακρίβειας, και έτσι η απάντηση που πήραμε περιέχει 0 και 0.0000. Αυτό συμβαίνει λόγω του ότι ο αλγόριθμος που υπολογίζει τον αντίστροφο δεν κάνει τις πράξεις ακριβώς και υποπίπτει σε σφάλματα στρογγυλοποίησης. Από πρακτική άποψη, το 0 και το 0.0000 είναι τα ίδια.

Οι ιδιοτιμές και τα ιδιοδιανύσματα του A (δηλ. οι αριθμοί λ και τα διανύσματα x τα οποία ικανοποιούν $Ax = \lambda x$) υπολογίζονται με την εντολή `eig`.

```
>> eig(A)

ans =
   12.6462
    3.1594
   -4.8055
```

δίνει ένα διάνυσμα στήλης με τις ιδιοτιμές λ, και

```
>> [X,D]=eig(A)

X =
   -0.8351   -0.6821    0.2103
   -0.4350    0.5691   -0.4148
   -0.3368   -0.4592    0.8853

D =
   12.6462         0         0
         0    3.1594         0
         0         0   -4.8055
```

δίνει ένα διαγώνιο πίνακα D με τις ιδιοτιμές στη κύρια διαγώνιο, και ένα πίνακα X του οποίου οι στήλες είναι τα ιδιοδιανύσματα που αντιστοιχούν στην κάθε ιδιοτιμή.

Ασκήσεις

Χρησιμοποιείστε τον πίνακα A και τα διανύσματα a και b από τις ασκήσεις της προηγούμενης ενότητας για να απαντήσετε τα πιο κάτω.

1. Με συνδυασμό εντολών, βρείτε το μέγιστο και ελάχιστο στοιχείο του πίνακα A.

2. Κατασκευάστε ένα (ψευδο)τυχαίο 2×2 πίνακα του οποίου τα στοιχεία να είναι θετικοί ακέραιοι.

3. Ταξινομήστε τα στοιχεία του b.

4. **(a)** Βρείτε τις ιδιοτιμές και τα ιδιοδιανύσματα του πίνακα $B = A^{-1}$. Αποθηκεύσετε τις ιδιοτιμές σε ένα διάνυσμα στήλης με την ονομασία *lambda*.
(b) Με I τον 4×4 ταυτοτικό πίνακα, υπολογίστε την ορίζουσα του πίνακα $B - lambda_j\ I$, για $j = 1, 2, 3, 4$. (Σημείωση: $lambda_1$ είναι η πρώτη ιδιοτιμή, $lambda_2$ η δεύτερη, κ.ο.κ.)

2.4 *Γραφήματα*

Κλείνουμε την συζήτηση των βασικών στοιχείων της MATLAB με λίγα λόγια για την κατασκευή γραφημάτων. Γράφοντας `help plot` θα δείτε τις δυνατότητες της εντολής `plot`, η οποία είναι η βασική εντολή για την κατασκευή δυδιάστατων γραφημάτων.

Βασικά, αν *x* και *y* είναι δύο διανύσματα ίδιου μήκους, τότε η εντολή `plot(x,y)` δίνει τη γραφική παράσταση του *y* ως προς *x*.

Για παράδειγμα, για να κατασκευάσουμε τη γραφική παράσταση της συνάρτησης $y = \cos(x)$ στο διάστημα $[-\pi, \pi]$, ορίζουμε το διάνυσμα *x* με τιμές ισαπέχοντα σημεία από $-\pi$ μέχρι π, με βήμα, ας πούμε, 0.01.

```
>> x=-pi:0.01:pi;
```

Βάλαμε ερωτηματικό (;) στο τέλος της εντολής για να μην τυπωθούν οι αριθμοί στην οθόνη. Σημειώστε ότι όσο πιο μικρό είναι το βήμα τόσο πιο "ομαλή" θα είναι η καμπύλη του γραφήματος. Ισοδύναμα, θα μπορούσαμε να ορίσουμε το *x* μέσω της εντολής `linspace` η οποία αντί για βήμα παίρνει τον αριθμό των σημείων για δεδομένο εισόδου, μετά το διάστημα:

```
>> x=linspace(-pi,pi,101);
```

(Εδώ ορίσαμε το *x* σαν ένα διάνυσμα με 101 ισαπέχοντα σημεία στο διάστημα $[-\pi, \pi]$.)

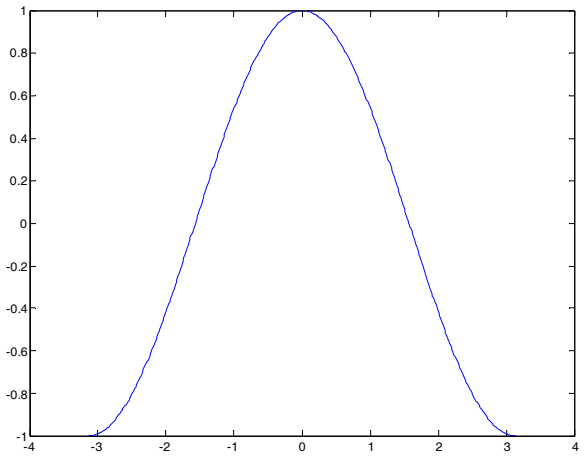
Μετά, ορίζουμε το *y* ως

```
>> y=cos(x);
```

(βάζοντας ερωτηματικό και πάλι στο τέλος για να μην τυπωθεί το αποτέλεσμα στην οθόνη) και κατασκευάζουμε το γράφημα:

```
>> plot(x,y)
```

Σε αυτό το σημείο θα ανοίξει ένα καινούργιο παράθυρο το οποίο θα περιέχει το γράφημα (σαν αυτό που φαίνεται πιο κάτω).



Μπορούμε, αν θέλουμε, να βάλουμε ετικέτες στους άξονες με τις εντολές `xlabel` και `ylabel`, όπως επίσης και τίτλο με την εντολή `title`, μετά που θα ανοίξει το παράθυρο με το γράφημα.

```
>> xlabel('x')
>> ylabel('y=cos(x)')
>> title('Graph of cosine from - pi to pi')
```

Το κείμενο που θέλουμε να εμφανιστεί στο γράφημα γράφεται μέσα σε τόνους (`' '`).

Υπάρχουν διάφορα είδη γραμμών και χρωμάτων που μπορούμε να χρησιμοποιήσουμε για γραφήματα, όπως φαίνεται πιο κάτω.

y	yellow	.	point
m	magenta	o	circle
c	cyan	x	x-mark
r	red	+	plus
g	green	-	solid
b	blue	*	star
w	white	:	dotted
k	black	-.	dashdot
		--	dashed

Άρα, για να πάρουμε το ίδιο γράφημα, αλλά με χρώμα πράσινο, λέμε

```
>> plot(x,y,'g')
```

όπου το χρώμα εμφανίζεται στο τέλος μέσα σε τόνους. Η εντολή

```
>> plot(x,y,'--')
```

δίνει το ίδιο γράφημα αλλά αντί για συνεχή γραμμή, η καμπύλη σχεδιάζεται με διακεκομμένη. Επιτρέπεται και ο συνδυασμός των δύο, έτσι για να πάρουμε το γράφημα με μπλε γραμμή η οποία αποτελείται από τελείες λέμε

```
>> plot(x,y,'b:')
```

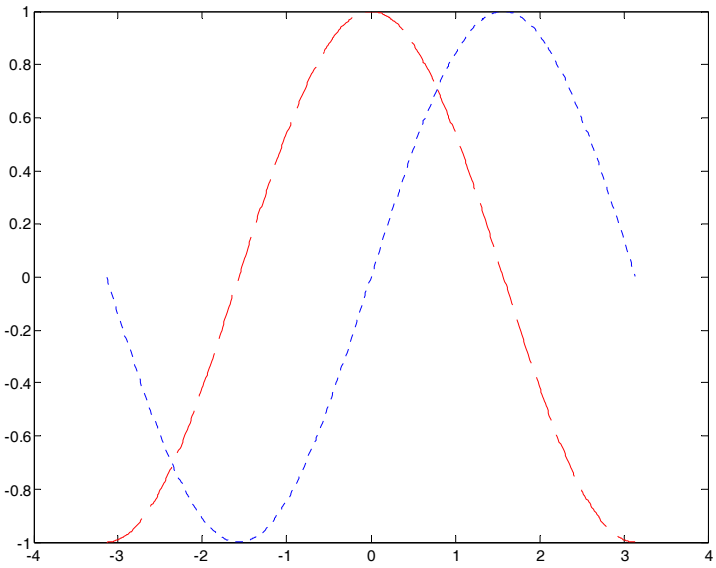
Μπορούμε να κάνουμε πολλαπλά γραφήματα στους ίδιους άξονες με ευκολία. Αν ορίσουμε ένα καινούργιο διάνυσμα

```
>> z = sin(x);
```

Τότε η εντολή

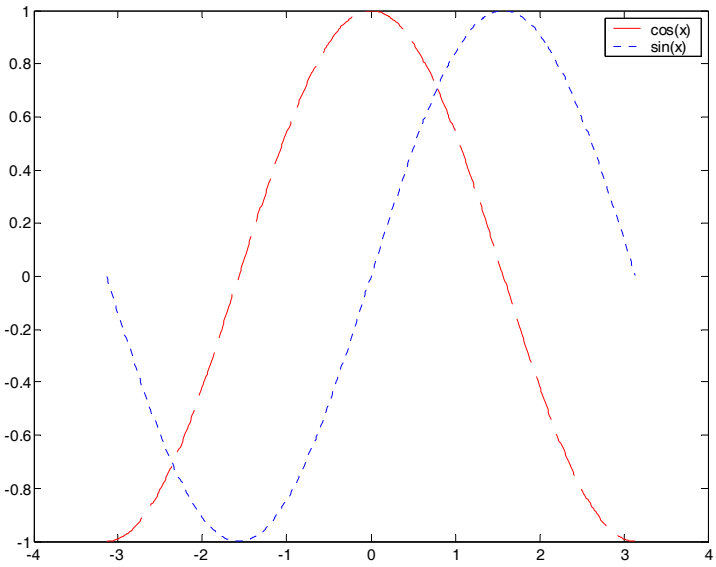
```
>> plot(x,y,'r--',x,z,'b:')
```

δίνει το γράφημα που φαίνεται πιο κάτω, δηλ. την γραφική παράσταση της συνάρτησης $y = \cos(x)$ με κόκκινο διακεκομμένη γραμμή και της συνάρτησης $z = \sin(x)$ με μπλε γραμμή από τελείες.



Όταν έχουμε πολλαπλά γραφήματα στους ίδιους άξονες, είναι καλή ιδέα να βάζουμε λεζάντα που να επεξηγεί ποια καμπύλη αντιστοιχεί σε ποια συνάρτηση. Η εντολή `legend` κάνει αυτή τη δουλεία.

```
>> legend('cos(x)','sin(x)')
```



Η σειρά με την οποία γράφουμε το κείμενο της λεζάντας πρέπει να είναι η ίδια με αυτή που ακολουθήσαμε στο γράφημα.

Αν θέλουμε να τυπώσουμε το γράφημα το κάνουμε χρησιμοποιώντας το μενού εντολών που βρίσκεται στο παράθυρο του γραφήματος. Μπορούμε ακόμα να μεταφέρουμε την εικόνα του γραφήματος (μέσω των μενού του παραθύρου του γραφήματος) στο αγαπημένο μας επεξεργαστή κειμένων, π.χ. Microsoft Word.

Η εντολή `grid` τοποθετεί πλέγμα σε ένα υφιστάμενο γράφημα. Άλλες εντολές για γραφήματα δίδονται στον πιο κάτω πίνακα.

`subplot`	Πολλαπλά γραφήματα στο ίδιο παράθυρο
`loglog`	Λογαριθμικό γράφημα
`semilogx`	Ημιλογαριθμικό γράφημα (άξονας των *x*)
`semilogy`	Ημιλογαριθμικό γράφημα (άξονας των *y*)
`surf`	Γράφημα επιφάνειας
`surfl`	Γράφημα επιφάνειας με φωτισμό
`mesh`	Γράφημα επιφάνειας με πλέγμα

Ζητήστε βοήθεια για τις πιο πάνω εντολές για να δείτε πως χρησιμοποιούνται.

Δίνουμε τώρα ένα σύντομο παράδειγμα για γραφήματα επιφανειών (στις τρεις διαστάσεις). Ορίζουμε τις τιμές που θα πάρουν τα *x* και *y*:

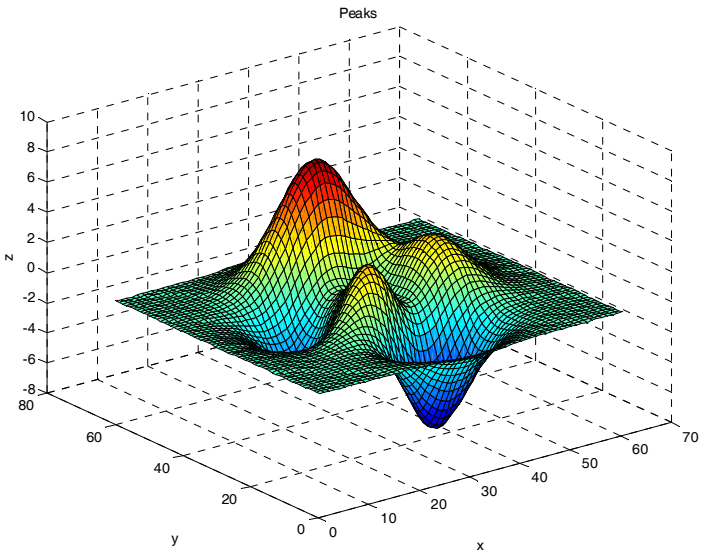
```
>> [x,y] = meshgrid(-3:.1:3,-3:.1:3);
```

Ορίζουμε την συνάρτηση $z = f(x, y)$:

```
>> z = 3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) ...
   - 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) ...
   - 1/3*exp(-(x+1).^2 - y.^2);
```

Κάνουμε το γράφημα, βάζοντας ετικέτες στους άξονες και τίτλο:

```
>> surf(z)
>> xlabel('x')
>> ylabel('y')
>> zlabel('z')
>> title('Peaks')
```



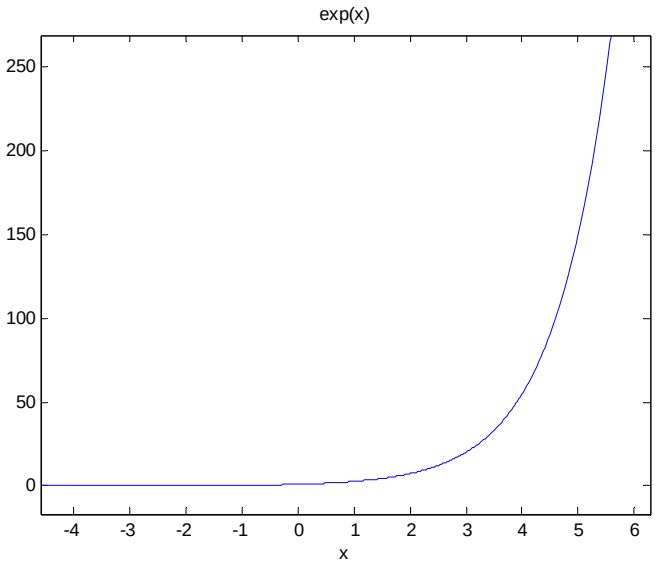
Γράψτε `help meshgrid`, `help surf` και `help peaks` για περισσότερες πληροφορίες για το πιο πάνω γράφημα.

Οι εντολές 'ez'

Για γρήγορα γραφήματα μαθηματικών συναρτήσεων (στις 2- και 3-διαστάσεις), οι πιο κατάλληλες εντολές είναι οι εντολές 'ez'. Για παράδειγμα, η εντολή

```
>> ezplot('exp(x)')
```

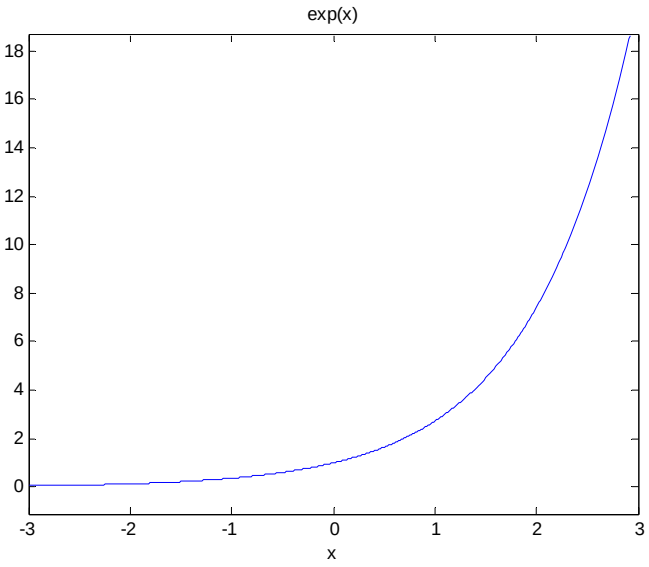
δίνει το γράφημα της συνάρτησης e^x σε προεπιλεγμένο (από τη MATLAB) διάστημα τιμών για το x.



Η εντολή

```
>> ezplot('exp(x)', -3, 3 )
```

δίνει το ίδιο γράφημα, αλλά αυτή τη φορά για τιμές του x στο διάστημα [–3, 3].



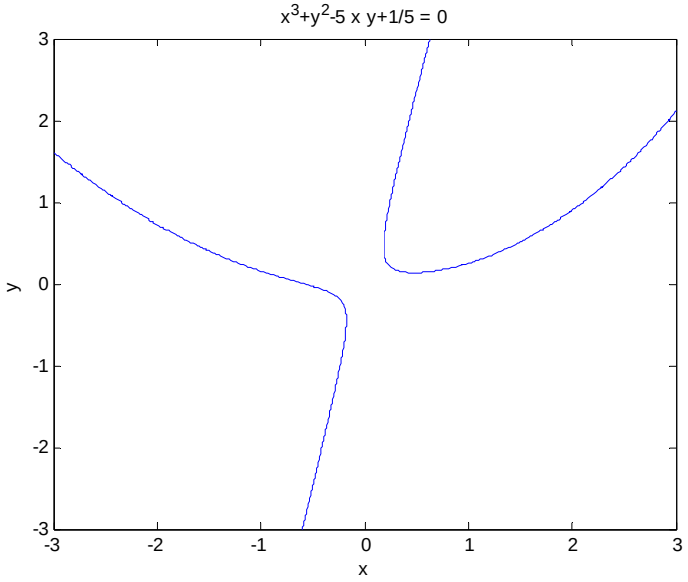
Για να κατασκευάσουμε το γράφημα της πεπλεγμένης συνάρτησης

$$x^3 + y^2 - 5xy + \frac{1}{5} = 0$$

για τιμές του x στο διάστημα [–3, 3], γράφουμε

```
>> ezplot('x^3+y^2-5*x*y+1/5',[-3,3])
```

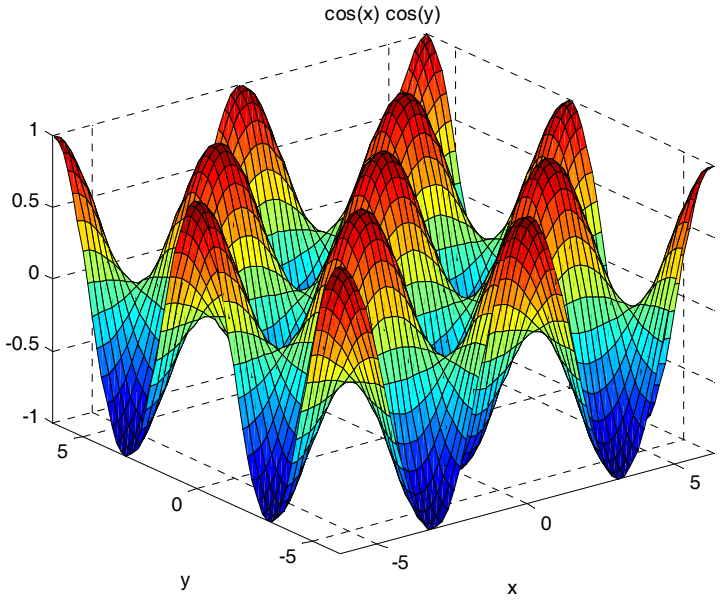
και παίρνουμε το γράφημα που ακολουθεί.



Οι εντολές `ezsurf, esurfl, ezmesh` , κ.λ.π. δουλεύουν με παρόμοιο τρόπο. Για παράδειγμα,

```
>> ezsurf('cos(x)*cos(y)')
```

δίνει



Ασκήσεις

1. Κάντε τις γραφικές παραστάσεις των συναρτήσεων $f(x) = x^2$, $g(x) = x^3$ για $x = -1, \ldots, 1$ στους ίδιους άξονες. Βάλτε ετικέτες , λεζάντα, και τυπώστε το γράφημα στον εκτυπωτή.

2. Να κάνετε την γραφική παράσταση της $z(x, y) = \dfrac{xy^2}{x^2 + y^2}$ για $x \in [-1, 3], y \in [1, 4]$.

3. Σχεδιάστε την καμπύλη που ορίζεται από την πεπλεγμένη συνάρτηση $y^2 - x^2 - 1 = 0$ με $-3 < x < 2$ και $-2 < y < 3$.

3. *ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΣΤΗ MATLAB*

3.1 *M-files: αρχεία script αρχεία συναρτήσεων*

Για να εκμεταλλευτούμε πλήρως τις ικανότητες της MATLAB πρέπει να δούμε πως γράφουμε τις δικές μας εντολές (και προγράμματα). Αυτό επιτυγχάνεται μέσω αρχείων που καλούνται "m-files" λόγω του ότι πρέπει να έχουν το επίθεμα (extension) "`.m`" (στο όνομά τους), π.χ. `compute.m`.

Υπάρχουν δύο είδη m-files: τα αρχεία script (*script files*) και τα αρχεία συναρτήσεων (*function files*). Τα αρχεία script περιέχουν απλώς μια σειρά από εντολές που επεξεργάζονται από τη MATLAB όταν καλέσουμε το αρχείο με το όνομά του. Με άλλα λόγια, δεν υπάρχουν δεδομένα εισόδου ή εξόδου σε αυτά τα αρχεία, και η χρήση τους περιορίζεται στις περιπτώσεις που θα θέλαμε να εκτελέσουμε πολλές εντολές (τη μια μετά την άλλη) χωρίς να τις γράφουμε μία-μία. Η άλλη χρησιμότητα αυτών των αρχείων είναι όταν θέλουμε να εισαγάγουμε πολλές πληροφορίες στη MATLAB (π.χ. σαν στοιχεία ενός πίνακα).

Τα αρχεία συναρτήσεων, τα οποία πολλές φορές καλούνται απλώς m-files, έχουν δεδομένα εισόδου (και εξόδου) και λειτουργούν σαν μια καινούργια εντολή της MATLAB ή ακόμα και σαν ένα υπολογιστικό πρόγραμμα.

Για να γράψουμε ένα m-file επικαλούμαστε τον συντάκτη (editor) που περιέχει η MATLAB με την εντολή `edit` η οποία ανοίγει ένα καινούργιο παράθυρο όπου γράφουμε το m-file. Η πρώτη γραμμή πρέπει πάντα να περιέχει τα εξής: Τη λέξη κλειδί `function`, τα (πιθανά) δεδομένα εξόδου (σε τετράγωνες παρενθέσεις), το όνομα του m-file, και τα (πιθανά) δεδομένα εισόδου (σε στρογγυλές παρενθέσεις), όπως φαίνεται πιο κάτω:

```
function [a] = log3(x)
```

Η μεταβλητή `a` είναι το δεδομένο εξόδου, το `log3` είναι το όνομα του m-file (δηλ. το αρχείο καλείται `log3.m`) και το `x` είναι το δεδομένο εισόδου. Κάτω από αυτή την επικεφαλίδα μπορούμε να γράψουμε σχόλια (που να επεξηγούν τι κάνει το m-file) και τις εντολές που θα επεξεργαστούν τα δεδομένα εισόδου έτσι ώστε να δώσουν το δεδομένο εξόδου που θα θέλαμε. Ας δούμε ένα απλό παράδειγμα:

```
function [a] = log3(x)

%  [a] = log3(x) - Calculates the base 3 logarithm of x.

    a = log(abs(x))./log(3);

% End of function
```

Όποιο κείμενο ακολουθεί το σύμβολο "`%`" δεν επεξεργάζεται από την MATLAB και λειτουργεί σαν σχόλια. Το πιο πάνω m-file έχει μόνο μία γραμμή, αυτή που υπολογίζει το δεδομένο εξόδου `a`, συναρτήσει του δεδομένου εισόδου `x` – εδώ απλώς υπολογίζεται ο λογάριθμος του `x` στη βάση 3. Για να τρέξουμε το πιο πάνω m-file γράφουμε,

```
>> log3(5)
```

```
ans =
    1.4650
```

Να αναφέρουμε ο υπολογισμός γίνεται βάσει του τύπου

$$\log_3(x) = \ln(|x|) / \ln(3)$$

και τη διαίρεση την κάναμε κατά-στοιχεία σε περίπτωση που το δεδομένο εισόδου είναι διάνυσμα (ή πίνακας). Επίσης, χρησιμοποιούμε απόλυτη τιμή μια και ο λογάριθμος ενός αρνητικού αριθμού δεν ορίζεται.

Παρατηρείστε τι παίρνουμε αν γράψουμε `help log3`:

```
>> help log3
[a] = log3(x) - Calculates the base 3 logarithm of x.
```

Πήραμε ως "βοήθεια" τα σχόλια που εμείς γράψαμε μέσα στο m-file. Άρα, η χρήση τέτοιων σχολίων είναι πάρα πολύ σημαντική.

Τέλος να πούμε ότι τα m-files μπορούν να καλέσουν άλλα m-files, ακόμα και τους ίδιους τους εαυτούς τους με αναδρομικό τρόπο.

Άσκηση

Γράψετε ένα *script* m-file που να καλείται `rand_int.m` και που να δίνει ένα τυχαίο ακέραιο όταν επικαλεστεί.

3.2 *Βρόχοι*

Θα δούμε τώρα πως μπορούμε να επαναλάβουμε εντολές στη MATLAB, μέσω βρόχων (loops). Αυτό θα μας χρησιμεύσει στην κατασκευή πιο πολύπλοκων προγραμμάτων. Υπάρχουν δύο είδη βρόχων, ο βρόχος "for" και ο βρόχος "while". Αρχίζουν με τη λέξη κλειδί `for` (ή `while`) και τελειώνουν με τη λέξη κλειδί `end`.

Ο βρόχος "for" μας επιτρέπει να επαναλάβουμε κάποιες εντολές σε ένα συγκεκριμένο αριθμό περιπτώσεων. Για παράδειγμα,

```
>> for j=1:4
     j+2
   end

j =
     3

j =
     4

j =
     5

j =
```

```
     6
```

Ζητήσαμε από την MATLAB την μετάβαση από τους αριθμούς 1 μέχρι 4, και κάθε φορά να τυπώσει (στην οθόνη) τον τρέχοντα αριθμό συν 2. Η βωβή μεταβλητή `j` παίζει μόνο το ρόλο του δείκτη.

Ας δούμε ένα άλλο παράδειγμα. Ορίζουμε το διάνυσμα $x = [1, 2, \ldots, 10]$ και υπολογίζουμε το $x^2 = [1^2, 2^2, \ldots, 10^2]$, το οποίο καταχωρούμε στη μεταβλητή `x2`. Θα βάλουμε ερωτηματικό στο τέλος της εντολής για να μη δούμε τα ενδιάμεσα αποτελέσματα.

```
>> x = 1:10

x =
     1     2     3     4     5     6     7     8     9    10

>> for i=1:10
     x2(i) = x(i)^2;
   end

>> x2

x2 =
     1     4     9    16    25    36    49    64    81   100
```

Αν και οι βρόχοι “for” είναι πολύ χρήσιμοι, μερικές φορές θα μπορούσαμε να πετύχουμε το ίδιο αποτέλεσμα με πιο εύκολο τρόπο. Στο πιο πάνω παράδειγμα θα ήταν καλύτερα αν λέγαμε

```
>> x2 = x.^2

x2 =
     1     4     9    16    25    36    49    64    81   100
```

Οι βρόχοι “for” μπορούν επίσης να εγκιβωτιστούν. Το πιο κάτω παράδειγμα δείχνει πως μπορούμε να υψώσουμε τα στοιχεία ενός πίνακα στη δεύτερη δύναμη.

```
>> 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
```

Και πάλι θα μπορούσαμε να επιτύχουμε τον σκοπό μας με τον πιο εύκολο τρόπο `A2 = A.^2`. Για ένα πιο ρεαλιστικό παράδειγμα, παραθέτουμε το πιο κάτω m-file, `gaussel.m`, που υλοποιεί την απαλοιφή Gaussian (μαζί με ανάδρομη αντικατάσταση) για την επίλυση του γραμμικού συστήματος A $x = b$.

```
function [x] = gaussel(A,b)

% [x] = gaussel(A,b)
%
%   This subroutine will perform Gaussian elimination
%   and back substitution to solve the system Ax = b.
%   INPUT :  A - matrix for the left hand side.
%            b - vector for the right hand side
%
%   OUTPUT : x - the solution vector.

N = max(size(A));

% Perform Gaussian Elimination

 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

% Perform back substitution

 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

% End of function
```

Για να δούμε πως λειτουργεί το πιο πάνω m-file, ορίζουμε τον πίνακα A και το διάνυσμα b:

```
>> 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
```

Ελέγχουμε αν ο πίνακας Α είναι αντιστρέψιμος (αν, δηλ., η ορίζουσά του δεν είναι 0):

```
>> det(A)


ans =

   -94
```

Η λύση του συστήματος Α $x = b$ παρέχεται από

```
>> x = gaussel(A,b)

x =
    1.2979
   -1.7660
   -0.0213
    0.3830
```

Φυσικά, θα μπορούσαμε επίσης να πάρουμε τη λύση του συστήματος με εντολή/συνάρτηση βιβλιοθήκης, ως εξής: `x = A\b`. Δοκιμάστε το!

Το δεύτερο είδος βρόχου είναι ο βρόχος "while". Χρησιμοποιείται όταν θέλουμε να επαναλάβουμε κάποιες εντολές μέχρι που να ισχύει μία συνθήκη (αντί για προκαθορισμένο αριθμό περιπτώσεων). Για παράδειγμα, δοθέντος του αριθμού n, το m-file που ακολουθεί (`exple.m`) θα δώσει τον πιο μικρό θετικό ακέραιο a έτσι ώστε $2^a \geq n$.

```
function [a] = exple(n)

% [a] = exple(n)
%

a = 0;
while 2^a < n
  a = a + 1;
end

% End of function
```

```
>> a = exple(4)

a =
     2
```

Ο βρόχος συνέχιζε να τρέχει εφόσον ίσχυε η σχέση

```
2^a < n
```

και σταμάτησε όταν η σχέση έπαψε να ισχύει. Τέτοιες σχέσεις χρησιμοποιούνται και στις εντολές "if" που θα δούμε σε λίγο. Η MATLAB γνωρίζει τους εξής σχεσιακούς τελεστές.

<	Μικρότερο
>	Μεγαλύτερο
<=	Μικρότερο ή ίσο
>=	Μεγαλύτερο ή ίσο
==	Ίσο
~=	Άνισο

Η διαφορά μεταξύ του "=" και του "= =" έγκειται στο ότι το πρώτο χρησιμοποιείται για καταχώρηση τιμών σε μεταβλητές ενώ το δεύτερο για έλεγχο σχέσεων. Οι διάφορες σχέσεις στη MATLAB μπορούν να ενωθούν με τους πιο κάτω λογικούς τελεστές.

&	Λογικό και
|	Λογικό ή
~	Λογικό όχι

Άσκηση

Ο $n \times n$ πίνακας Hilbert H, έχει σαν στοιχεία $H_{i,j} = 1/(i + j - 1)$, $i,j = 1, 2, \ldots, n$. Χρησιμοποιώντας ένα διπλό (εγκιβωτισμένο) βρόχο "for" κατασκευάστε τον 5×5 πίνακα Hilbert και ελέγξετε την απάντησή σας με την εντολή βιβλιοθήκης της MATLAB `hilb` που κάνει την ίδια δουλειά.

3.3 *Εντολή if*

Μερικές φορές το πρόγραμμά μας πρέπει να "πάρει κάποια απόφαση" πριν προχωρήσει και η εντολή "if" κάνει αυτή τη δουλειά. Η γενική δομή της έχει ως εξής:

```
if relation
  statement(s)
elseif relation     % if applicable
  statement(s)      % if applicable
else                % if applicable
  statement(s)      % if applicable
end
```

Οι λογικοί τελεστές (`&, |, ~`) μπορούν να χρησιμοποιηθούν για πιο πολύπλοκες σχέσεις.

Ας δούμε πως μπορούμε να χρησιμοποιήσουμε την εντολή "if" για να ορίσουμε την κατά-τμήματα συνάρτηση

$$F = \begin{cases} x^2 & if \quad -1 < x < 0.5 \\ 0.25 & if \quad 0.5 \le x < 1 \end{cases}$$

Πρώτα ορίζουμε το διάνυσμα x με τιμές στο διάστημα $[-1, 1]$:

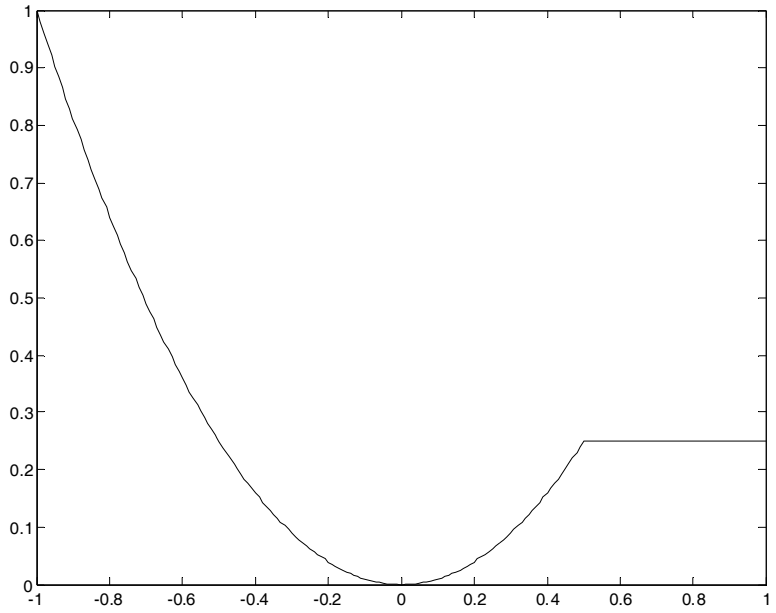
```
>> x=-1:0.01:1;
```

Μετά, χρησιμοποιούμε ένα βρόχο "for" και για κάθε στοιχείο του *x* ελέγχουμε αν ισχύει η σχέση $x_i < 0.5$. Αν ναι, τότε η *F* παίρνει την τιμή x_i^2. Διαφορετικά, η *F* παίρνει την τιμή 0.25. Το αποτέλεσμα είναι ένα διάνυσμα *F*, ίδιου μήκους με το *x*, του οποίου τα στοιχεία δίνουν τις τιμές της (κατά-τμήματα) συνάρτησης, όπως ορίζεται πιο πάνω.

```
» for i=1:length(x)
   if x(i) < 0.5
     F(i) = x(i)^2;
   else
     F(i) = 0.25;
   end
  end
```

Μπορούμε να δούμε τη γραφική παράσταση της *F* ως εξής:

```
>> plot(x,F,'-k')
```



Στο επόμενο παράδειγμα θέλουμε να γράψουμε ένα m-file που να καλείται `chk_inv.m`, το οποίο να παίρνει σαν δεδομένο εισόδου ένα τετραγωνικό πίνακα A και να δίνει σαν δεδομένο εξόδου τον αντίστροφο του A, αν ο A είναι αντιστρέψιμος και πράγματι τετραγωνικός. Αν ο A δεν είναι αντιστρέψιμος, τότε το m-file θα τυπώνει το κατάλληλο μήνυμα στην οθόνη με την εντολή `disp('Matrix is singular.')` και θα τερματίζει. Αν, πάλι, ο πίνακας δεν είναι τετραγωνικός, τότε το m-file πάλι θα τυπώνει το κατάλληλο μήνυμα στην οθόνη με την εντολή `disp('Matrix is not square.')`.

```
function [Ainv] = chk_inv(A)

% [Ainv] = chk_inv(A)
% Calculate the inverse of a matrix A if it exists.
```

```
[m,n] = size(A); % compute the size of the matrix A
if m~=n  % check if A is square
  disp('Matrix is not square.');
  A = [];
  return  % quit the function
elseif det(A)==0  % check if A is singular
  disp('Matrix is singular.');
  A = [];
  return  % quit the function
else
  Ainv = inv(A);  % compute the inverse
end

% End of function
```

Να ένα ενδεικτικό παράδειγμα χρήσης του πιο πάνω προγράμματος.

```
>> A=rand(3,3)

A =
    0.0475    0.6326    0.3653
    0.7361    0.7564    0.2470
    0.3282    0.9910    0.9826

>> chk_inv(A)

ans =
   -2.4101    1.2551    0.5806
    3.1053    0.3544   -1.2437
   -2.3270   -0.7767    2.0783
```

Δοκιμάστε και άλλους πίνακες για να δείτε τι παίρνετε.

Ας πούμε δυο λόγια για τις εντολές `disp` και `return` που είδαμε στο πιο πάνω πρόγραμμα. Η εντολή `return` είναι αυτονόητη και αναγκάζει το πρόγραμμα να τερματίσει. Η εντολή `disp` τυπώνει το κείμενο που βρίσκεται στην παρένθεση (μεταξύ τόνων) στην οθόνη. Για περισσότερες πληροφορίες ζητάτε βοήθεια για τις εντολές αυτές.

Το τελευταίο μας παράδειγμα δείχνει τη χρήση ενός αναδρομικού m-file, δηλ. που καλεί τον εαυτό του. Το m-file `fact.m` που ακολουθεί υπολογίζει το παραγοντικό ενός αριθμού n, που ορίζεται ως $n = 1 \cdot 2 \cdot 3 \cdot 4 \ldots (n-1) \cdot n$, χρησιμοποιώντας τον τύπο $n! = n\,(n-1)$. Να σημειώσουμε ότι $0! = 1$ και $1! = 1$.

```
function [N] = fact(n)

% [N] = fact(n)
```

```
% Calculate n factorial

if (n = = 1) | (n = = 0)
  N = 1;
else
  N = n*fact(n-1);
end

% End of function
```

Ασκήσεις

1. Τροποποιήστε το m-file `log3.m` της ενότητας 3.1, αφαιρώντας την απόλυτη τιμή από το λογάριθμο. Το καινούργιο m-file θα πρέπει να ελέγχει αν το δεδομένο εισόδου είναι θετικό, αρνητικό ή μηδέν. Στην πρώτη περίπτωση θα πρέπει να δίνει το αποτέλεσμα, ενώ στις άλλες δύο περιπτώσεις να τυπώνει το κατάλληλο μήνυμα στην οθόνη και να τερματίζει.

2. Γράψτε ένα m-file που να καλείται `div5.m` το οποίο να παίρνει σαν δεδομένο εισόδου κάποιο αριθμό και να ελέγχει αν είναι διαιρετός από το 5, τυπώνοντας το κατάλληλο μήνυμα στην οθόνη. (Το m-file αυτό δεν θα έχει δεδομένο εξόδου, άρα στις τετραγωνικές παρενθέσεις της επικεφαλίδας δεν γράφουμε τίποτα, δηλ. απλώς λέμε [ ].)

4. *ΕΠΙΠΡΟΣΘΕΤΑ ΘΕΜΑΤΑ*

4.1 *Πολυώνυμα στη MATLAB*

Η MATLAB έχει πολύ καλές ικανότητες στον χειρισμό πολυωνύμων, τα οποία αντιπροσωπεύονται ως διανύσματα (με τους συντελεστές του πολυωνύμου). Για παράδειγμα, το πολυώνυμο

$$p(x) = x^2 - 3x + 5$$

αντιπροσωπεύεται από το διάνυσμα `p = [1, -3, 5]` , ενώ το

$$q(x) = x^4 + 7x^2 - x$$

από το διάνυσμα `q = [1, 0, 7, -1, 0]` .

Γενικά, η MATLAB μπορεί να συσχετίσει ένα διάνυσμα μήκους $n + 1$ με ένα πολυώνυμο βαθμού n. Άρα, αν κάποιες από τις δυνάμεις του x "λείπουν" από το πολυώνυμο (δηλ. κάποιοι από τους συντελεστές του πολυωνύμου είναι 0) τότε πρέπει να θυμηθούμε να τους συμπεριλάβουμε στο διάνυσμα, όπως ακριβώς έγινε στο πιο πάνω παράδειγμα.

Για να βρούμε τη τιμή ενός πολυωνύμου (που έχει οριστεί στην MATLAB ως διάνυσμα) χρησιμοποιούμε την εντολή `polyval`. Για παράδειγμα, για να βρούμε την τιμή του πιο πάνω πολυωνύμου q όταν το $x = -1$, λέμε

```
>> polyval(q,-1)

ans =
     7
```

Η εντολή `roots` δίνει τις ρίζες δοθέντος πολυωνύμου:

```
>> roots(q)

ans =
         0
   0.0712 + 2.6486i
   0.0712 - 2.6486i
  -0.1424
```

Όπως βλέπετε, η MATLAB δουλεύει και με μιγαδικούς αριθμούς, όπου `i=sqrt(-1)`. (Δύο από τις 4 ρίζες του πολυωνύμου είναι μιγαδικές.)

Αν θέλουμε να πολλαπλασιάσουμε δύο πολυώνυμα, κάνουμε συνέλιξη (*convolution*) των διανυσμάτων με τους συντελεστές των πολυωνύμων. Για παράδειγμα, αν $s(x) = x + 2$ και $t(x) = x^2 + 4x + 8$ τότε

$$z(x) = s(x)\, t(x) = x^3 + 6x^2 + 16x + 16.$$

Στη MATLAB, γράφουμε

```
>> s = [1 2];
>> t = [1 4 8];
```

```
>> z = conv(s,t)

z =
     1     6    16    16
```

Η διαίρεση δύο πολυωνύμων είναι παρόμοια και επιτυγχάνεται με την εντολή `deconv` , η οποία μπορεί να δώσει και το υπόλοιπο της διαίρεσης. Ας διαιρέσουμε το *z* με το *t* για να δούμε αν θα πάρουμε το *s*:

```
>> [s,r] = deconv(z,t)

s =
     1     2

r =
     0     0     0     0
```

Πράγματι, πήραμε το *s* μαζί με το διάνυσμα/υπόλοιπο *r*, το οποίο σε αυτή τη περίπτωση είναι 0, όπως και θα έπρεπε.

Η MATLAB μπορεί να δώσει και την παράγωγο ενός πολυωνύμου με την εντολή `polyder` η οποία παίρνει ως δεδομένο εισόδου το διάνυσμα με τους συντελεστές του πολυωνύμου και δίνει ως δεδομένο εξόδου το διάνυσμα με τους συντελεστές της παραγώγου του πολυωνύμου. Για παράδειγμα, αν $p(x) = x^2 - 3x + 5$, όπως και πριν, τότε

```
>> polyder(p)

ans =
     2    -3
```

Τι νομίζετε ότι θα δώσει ο συνδυασμός εντολών `polyval(polyder(p),1);` Τι θα λέγατε για τον συνδιασμό `roots(polyder(p));`

Ασκήσεις

1. Γράψετε ένα m-file, που να καλείται `polyadd.m,` το οποίο προσθέτει δύο πολυώνυμα (που δεν έχουν κατ'ανάγκη τον ίδιο βαθμό). Τα δεδομένα εισόδου θα πρέπει να είναι τα δύο διανύσματα που αντιστοιχούν στα πολυώνυμα που θέλουμε να προσθέσουμε, και το δεδομένο εξόδου θα πρέπει να είναι το διάνυσμα που αντιστοιχεί στο άθροισμά τους.

2. Να βρείτε το τοπικά μέγιστα (local maxima) και ελάχιστα (local minima), αν υπάρχουν, της συνάρτησης $f(x) = x^3 - x^2 - 3x$. Να κάνετε το γράφημα της συνάρτησης, μαζί με τα μέγιστα/ελάχιστα χρησιμοποιώντας 'o' για κάθε μέγιστο και '*' για κάθε ελάχιστο.

4.2 *Μαθηματικές συναρτήσεις*

Υπάρχουν δύο βασικοί τρόποι με τους οποίους μπορούμε να ορίσουμε μία μαθηματική συνάρτηση στη MATLAB, η οποία δεν είναι ήδη ορισμένη ως συνάρτηση βιβλιοθήκης (όπως, π.χ. $\sin(x)$, $\cos(x)$, e^x, κ.λ.π.)

Ο πρώτος τρόπος είναι μέσω ενός m-file, όπως έγινε με την κατά-τμήματα ορισμένη συνάρτηση F στη σελίδα 34. Για να επαναλάβουμε τη διαδικασία, ας ορίσουμε την συνάρτηση $f(x) = e^x - x^2$ μέσω ενός m-file που θα ονομάσουμε `eff.m`, με το εξής περιεχόμενο:

```
function [F] = eff(x)

% [F] = eff(x)

F = exp(x) - x.^2;

% End of function
```

Με αυτό τον τρόπο μπορούμε να πάρουμε τιμές της συνάρτησης καλώντας την μέσω της MATLAB.

```
>> eff(0)

ans =
     1

>> eff(-1/2)

ans =
    0.3565
```

Ο άλλος τρόπος με τον οποίο μπορούμε να ορίσουμε μια μαθηματική συνάρτηση στη MATLAB είναι μέσω μίας ανώνυμης συνάρτησης, όπως λέγεται, κάτι που είναι δυνατό στις εκδοχές της MATLAB 7.0 και μετέπειτα. Για να ορίσουμε την πιο πάνω συνάρτηση γράφουμε

```
>> f=@(x) exp(x) - x.^2;
```

και παίρνουμε τιμές με τον ίδιο τρόπο.

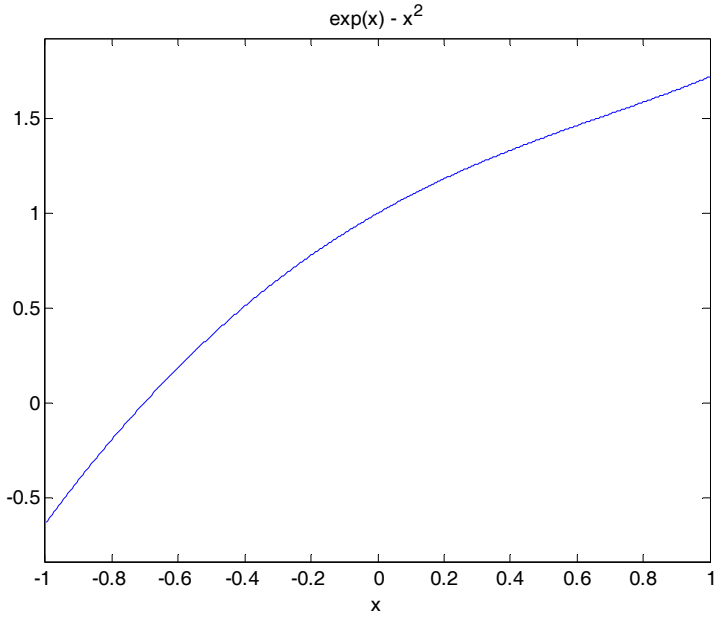
```
>> f(-2)

ans =
   -3.8647

>> f(0.1)

ans =
    1.0952
```

Το σύμβολο `@` και η μεταβλητή `x` που το ακολουθεί σε παρενθέσεις, χρησιμοποιούνται για να υποδείξουν στη MATLAB ότι ορίζουμε μία συνάρτηση η οποία έχει ως μεταβλητή (δηλ. δεδομένο εισόδου) ότι ακολουθεί το `@` μέσα σε παρενθέσεις. Έτσι για να ορίσουμε, για παράδειγμα, τη συνάρτηση δύο μεταβλητών $g(x,y)=\sqrt{x^2+y^2}$, γράφουμε

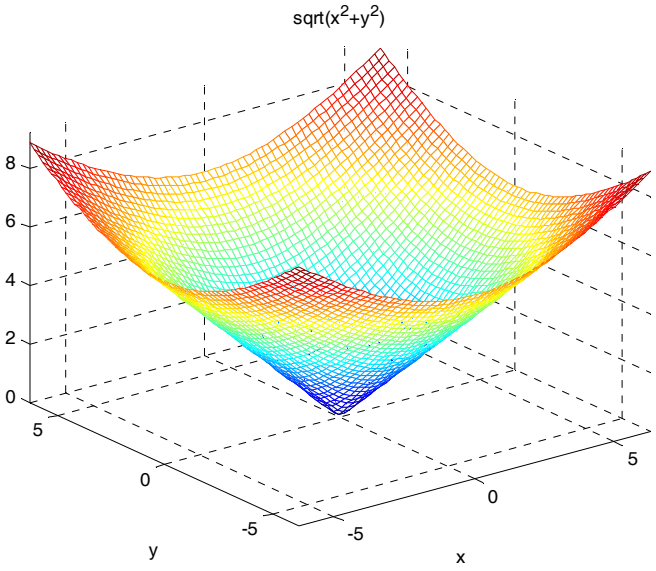
```
>> g=@(x,y) sqrt(x.^2+y.^2)
```

Αφού ορίσουμε μια τέτοια συνάρτηση, μπορούμε στη συνέχεια να πάρουμε τις τιμές (όπως ήδη είδαμε), να κάνουμε τη γραφική της παράσταση (όπως φαίνεται πιο κάτω) και να τη χρησιμοποιήσουμε ως δεδομένο εισόδου σε άλλες εντολές (όπως θα δούμε στην επόμενη ενότητα).

```
>> ezplot(f,[-1,1])
```




```
>> ezmesh(g)
```



4.3 *Αριθμητικές μέθοδοι*

Σε αυτή την τελευταία ενότητα θα ασχοληθούμε με εντολές που έχουν σχέση με την αριθμητική επίλυση (δηλ. προσέγγιση) διαφόρων προβλημάτων.

Έχουμε ήδη δει πώς να βρίσκουμε τις ρίζες κάποιου πολυωνύμου στη MATLAB. Τι γίνεται, όμως αν θέλουμε να βρούμε τις ρίζες μιας γενικής συνάρτησης; Στην προηγούμενη ενότητα είδαμε πώς να ορίσουμε μαθηματικές συναρτήσεις. Επομένως αφού ορίσουμε την συνάρτηση της οποίας η ρίζα είναι το ζητούμενο, χρησιμοποιούμε την εντολή `fzero`, η οποία βρίσκει μια προσέγγιση της ρίζας δεδομένης μίας αρχικής εκτίμησης. Για παράδειγμα, ας βρούμε την ρίζα της συνάρτησης $f(x) = e^x - x^2$ που ορίσαμε πιο πάνω. Η γραφική παράσταση της $f(x)$ δείχνει ότι υπάρχει μια ρίζα κοντά στο –0.8. Άρα, θα χρησιμοποιήσουμε το –0.8 ως την αρχική εκτίμηση και θα πούμε

```
>> fzero(f,-0.8)

ans =
   -0.7035
```

Για να πάρουμε περισσότερα δεκαδικά ψηφία στην απάντηση γράφουμε

```
>> format long
```

και έχουμε

```
>> ans

ans =
  -0.703467422498392
```

Ας δούμε πόσο ακριβής είναι η απάντηση που πείραμε:

```
>> f(ans)

ans =
     0
```

Αυτό δεν σημαίνει ότι έχουμε την ακριβή απάντηση, αλλά ότι η απάντηση έχει 16 σωστά δεκαδικά ψηφία. Αν πούμε, για παράδειγμα,

```
>> f(-0.7035)

ans =
   -6.195673228776011e-005
```

βλέπουμε ότι το `-0.7035` έχει 5 σωστά δεκαδικά ψηφία.

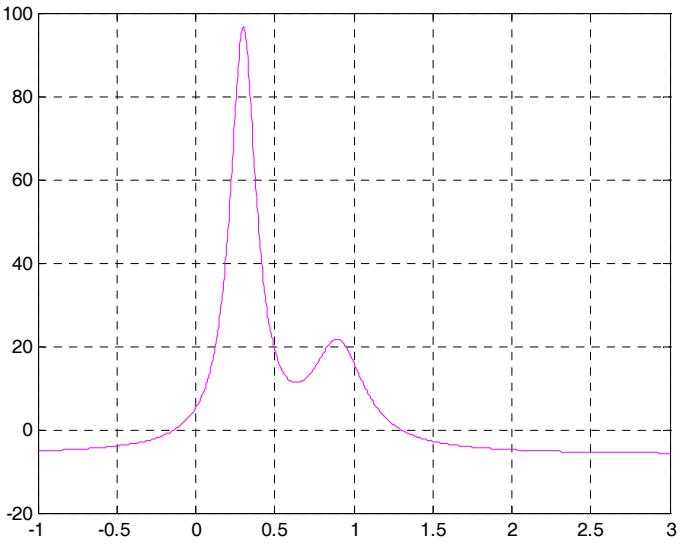
Στη περίπτωση που έχουμε πέραν της μίας ρίζα, καλούμε την πιο πάνω εντολή περισσότερες φορές, δίνοντας κατάλληλες εκτιμήσεις ως δεδομένο εισόδου κάθε φορά.

Μια άλλη χρήσιμη εντολή για αριθμητικούς υπολογισμούς είναι η `fminbnd`, η οποία βρίσκει το ελάχιστο δοθείσας συνάρτησης (μίας μεταβλητής) σε ένα διάστημα, και ενεργεί με παρόμοιο τρόπο όπως η εντολή `fzero`. Για παράδειγμα, ορίζουμε τη συνάρτηση

$$g(x) = \frac{1}{(x-0.3)^2 + 0.01} + \frac{1}{(x-0.9)^2 + 0.04} - 6$$

ως ανώνυμη συνάρτηση και κατασκευάζουμε το γράφημα της στο διάστημα [–1, 3].

```
>> g =@(x)  1./((x-0.3).^2+0.01) + 1./((x-0.9).^2+0.04) - 6;
>> x=linspace(-1,3,1001);
>> plot(x,g(x),'m')
>> grid
```



Βλέπουμε ότι έχουμε σχετικά ελάχιστο μεταξύ του $x = 0.5$ και $x = 1$. Άρα, γράφουμε

```
>> fminbnd(g,0.5,1)

ans =

   0.637010674590588
```

Και έχουμε τη συντεταγμένη του x για το ελάχιστο. Για να βρούμε την συντεταγμένη του y λέμε

```
>> g(ans)

ans =

  11.252754126564305
```

Στη περίπτωση που έχουμε περισσότερα από ένα ελάχιστα, καλούμε την πιο πάνω εντολή περισσότερες από μία φορά, παρέχοντας τα κατάλληλα διαστήματα ως δεδομένα εισόδου.

Πώς νομίζετε ότι μπορούμε να βρούμε το μέγιστο μιας συνάρτησης;

Η τελευταία εντολή που θα δούμε αφορά την αριθμητική ολοκλήρωση. Η εντολή `quad` βρίσκει μια προσέγγιση για το ορισμένο ολοκλήρωμα μιας συνάρτησης. Παίρνει ως

δεδομένα εισόδου την συνάρτηση (που έχει οριστεί, π.χ. σαν ανώνυμη), τα άκρα του διαστήματος ολοκλήρωσης και την απαιτούμενη ακρίβεια. Το τελευταίο είναι προαιρετικό – αν δεν ζητήσουμε κάποια απαιτούμενη ακρίβεια, τότε η MATLAB προεπιλέγει το 10^{-6}, δηλ. δίνει απάντηση με 6 σωστά δεκαδικά ψηφία.

Ας βρούμε το ολοκλήρωμα της $f(x) = e^x - x^2$ από $x = 0$ μέχρι 1. Αφού ήδη ορίσαμε την $f(x)$ απλώς γράφουμε

```
>> quad(f,0,1)

ans =
   1.384948498868295
```

(Η ακριβής απάντηση είναι $e - 4/3$, δηλ. περίπου 1.384948495125712.)

Τι γίνεται με το ολοκλήρωμα του $sin(x)$ από 0 μέχρι π; Ξέρουμε ότι η απάντηση είναι 2, αλλά ας δούμε τι δίνει η εντολή `quad`. Η συνάρτηση *sin* είναι συνάρτηση βιβλιοθήκης, άρα την δίνουμε ως δεδομένο εισόδου μέσα σε τόνους (`' '`). (Θα μπορούσαμε να την δώσουμε επίσης και ως `@sin`.)

```
>> quad('sin',0,pi)

ans =

   1.999999996398431
```

Η απάντηση που πήραμε είναι κοντά στο 2 και έχει 8 σωστά δεκαδικά ψηφία, όπως φαίνεται και από τον υπολογισμό

```
>> 2-ans

ans =
    3.601569042999131e-009
```

Για να πάρουμε καλύτερη απάντηση θα πρέπει να καθορίσουμε την απαιτούμενη ακρίβεια (π.χ. 10^{-16} που μπορεί να γραφτεί ως `1e-16`) στην εντολή `quad`, ως εξής:

```
>> quad('sin',0,pi,1e-16)

ans =

   2.000000000000000
```

Αναφέρουμε ότι η εντολή `quad` χρησιμοποιεί την αναπροσαρμοστική μέθοδο Simpson για την αριθμητική ολοκλήρωση. Μια άλλη, παρόμοια, εντολή είναι η `quadl`, που χρησιμοποιεί την μέθοδο Gauss-Legendre. Ζητήστε βοήθεια για τις δύο αυτές εντολές εάν θα θέλατε περισσότερες πληροφορίες.

5. *ΕΠΙΛΟΓΟΣ*

Ελπίζω αυτός ο οδηγός να σας έχει δώσει ένα καλό υπόβαθρο, έτσι ώστε να μπορέσετε να κάνετε απλούς υπολογισμούς και γραφήματα, όπως επίσης και για να συνεχίσετε τη μελέτη της MATLAB. Υπάρχει πληθώρα εντολών που δεν είδαμε σε αυτόν τον οδηγό και που μπορεί να σας ενδιαφέρουν. Εισηγούμαι να ασχοληθείτε με το πακέτο έτσι ώστε να αποκτήσετε την απαιτούμενη πείρα και άνεση χρήσης. Ένα καλό βοήθημα που περιέχει περισσότερες πληροφορίες, εντολές και παραδείγματα βρίσκετε στο σύνδεσμο

http://www.ucy.ac.cy/~georgios/courses/mas191/MATLABbook.pdf

σε μορφή PDF. Μπορείτε, αν θέλετε, να αγοράσετε το πιο πάνω βοήθημα από το βιβλιοπωλείο Καντζελάρης.

Αναφέρω επίσης και την ιστοσελίδα της εταιρείας που έχει δημιουργήσει τη MATLAB

www.mathworks.com

η οποία περιέχει, μεταξύ άλλων, πολλά βοηθήματα (στα αγγλικά) και αρχεία που έγραψαν άλλοι χρήστες της MATLAB.