ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ & ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ

MATLAB

ΕΙΣΑΓΩΓΗ - ΜΑΘΗΜΑΤΙΚΑ -ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Εργαστηριακός οδηγός

Δρ Ιωάννης Θ. Φαμέλης
Καθηγητής

(2018)

Τι είναι το MATLAB

Το **MATLAB** (MATrix LABoratory) είναι ένα διαδραστικό περιβάλλον για αριθμητικούς υπολογισμούς κυρίως. Μέσω του Symbolic Toolbox μας παρέχει τη δυνατότητα να προβούμε και σε κάποιους συμβολικούς υπολογισμούς. Το 1978 δημιουργήθηκε η πρώτη του έκδοση σε Fortran. Το 1984 η έκδοση MATLAB 1 γράφηκε σε γλώσσα C. Γρήγορα το MATLAB έγινε ένα ιδιαίτερα πολύτιμο και δημοφιλές εργαλείο για επιστήμονες και ερευνητές πολλών επιστημονικών πεδίων. Στην εποχή μας έχουμε φθάσει στην έκδοση MATLAB R2009b (Σεπτέμβριος 2009). Ο πυρήνας του με τις βασικές εσωτερικές του εντολές είναι αναπτυγμένος σε γλώσσα C, ενώ όλες οι υπόλοιπες εντολές του είναι αναπτυγμένες στη γλώσσα του MATLAB.

Τα κύρια χαρακτηριστικά του είναι τα ακόλουθα:

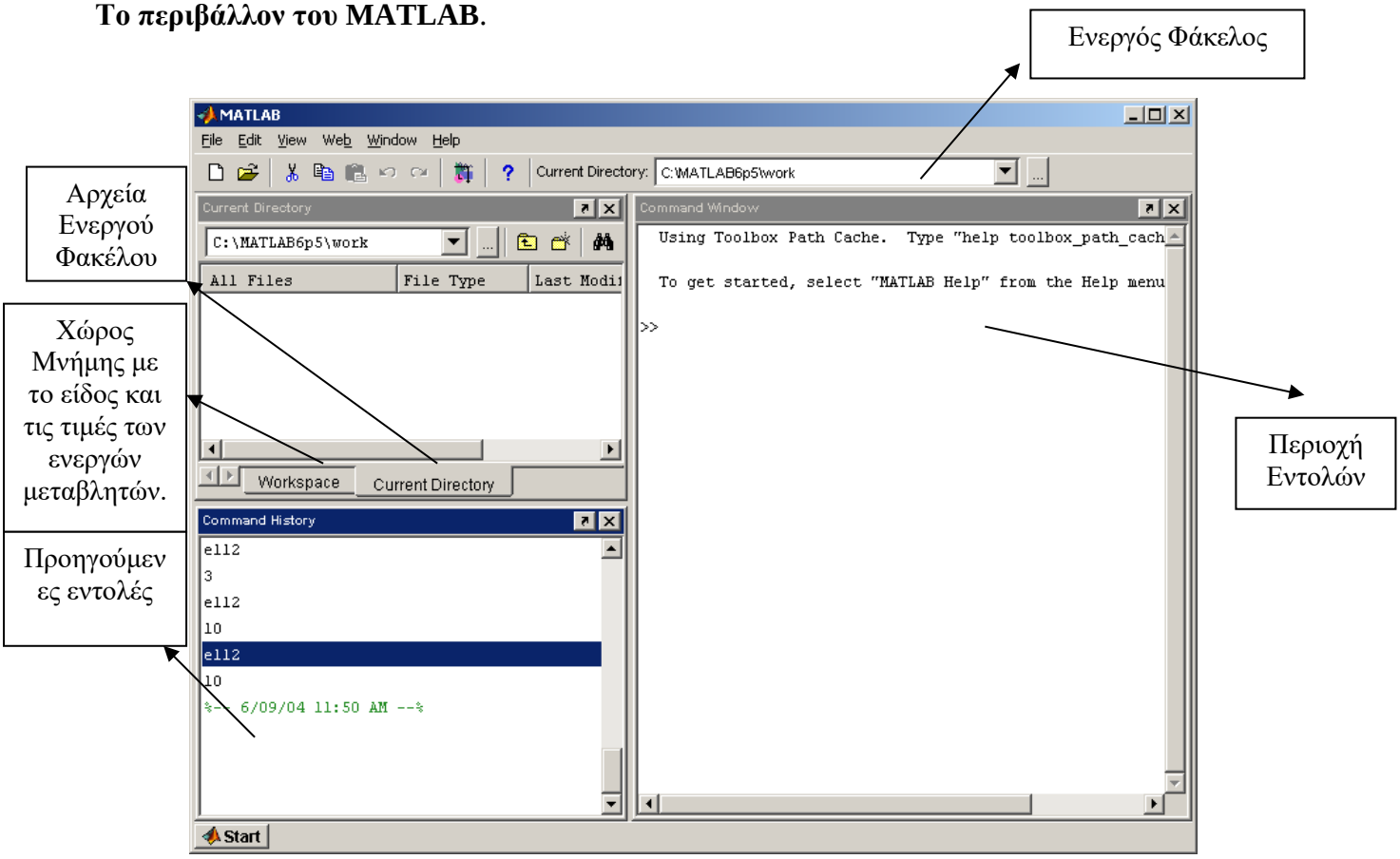
1. Επιτρέπει ταχύ και εύκολο προγραμματισμό σε μία υψηλού επιπέδου γλώσσα.
2. Παρέχει υψηλού επιπέδου εύκολα υλοποιήσιμα γραφικά.
3. Το διαδραστικό του περιβάλλον επιτρέπει τον πειραματισμό πάνω στα δεδομένα και εύκολη εύρεση σφαλμάτων.
4. Είναι μία σύγχρονη γλώσσα προγραμματισμού που υποστηρίζει τον αντικειμενοστραφή προγραμματισμό, χειρίζεται τα δεδομένα με ιδιαίτερη ευκολία (π.χ. χωρίς δηλώσεις). Τα προγράμματα MATLAB μπορούν να μεταφερθούν από ένα σύστημα σε ένα άλλο χωρίς μετατροπή.
5. Υπάρχουν πολλές βιβλιοθήκες (Toolboxes) που δίνουν τη δυνατότητα να χειριστούμε προβλήματα συγκεκριμένου πεδίου π.χ. Fuzzy Logic, Image Processing, Databases, Symbolic Maths, Parallel and Distributed Programming και πολλές άλλες.
6. Υπάρχει άφθονος διαθέσιμος κώδικας στο διαδίκτυο που καλύπτει μεγάλη γκάμα μαθηματικών και επιστημονικών προβλημάτων.

Εκτελώντας το MATLAB μας εμφανίζεται το παράθυρό του μέσα από το οποίο μπορούμε να χειριστούμε όλο το περιβάλλον της εφαρμογής. (Αν το παράθυρό στον υπολογιστή σας δεν είναι αυτό που εμφανίζεται στην εικόνα επιλέξτε τον κατάλογο επιλογών View→ Desktop Layout → Default.) Στο περιβάλλον αυτό αρχικά ορίζουμε τον φάκελο στον οποίο θα εργαστούμε (εξ ορισμού ο C:\MATLAB6p5\work) και θα αποθηκεύσουμε τα προγράμματά σας και τα αρχεία σας.

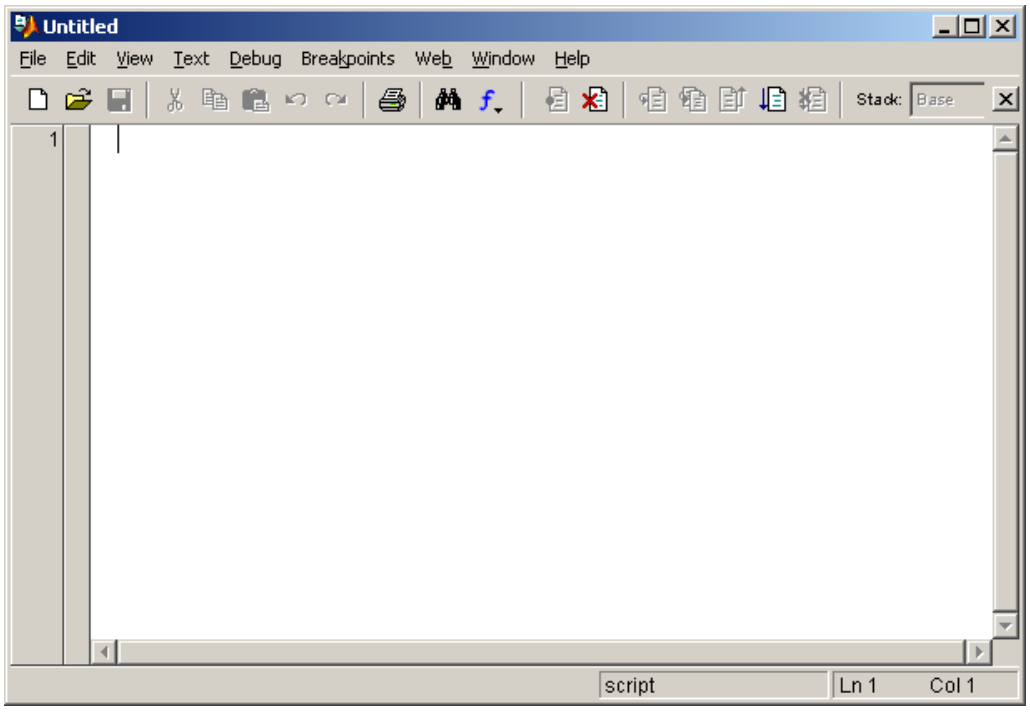
Στα δεξιά υπάρχει το Command Window όπου δίνουμε τις εντολές μας στο MATLAB και λαμβάνουμε τα αποτελέσματα. Οι εντολές δίνονται μετά την προτροπή >> (prompt) και το τέλος τους πιστοποιείται πατώντας Enter. Στα αριστερά στο άνω μέρος υπάρχουν διαθέσιμα το παράθυρο Current Directory, στο οποίο βλέπετε τα περιεχόμενα του ενεργού φακέλου και το παράθυρο Workspace στο οποίο εμφανίζονται οι μεταβλητές που ορίζουμε και χρησιμοποιούμε.

Στο κάτω μέρος στα αριστερά υπάρχει το παράθυρο Command History στο οποίο εμφανίζονται όλες οι εντολές που εκτελούνται στο Command Window από όπου μπορούν να εκτελεσθούν ξανά ή να αντιγραφούν. Τέλος, από τον κατάλογο επιλογών File (New, Open) μπορούμε να ενεργοποιήσουμε τον συντάκτη (editor) του MATLAB στον οποίο συντάσσουμε τα προγράμματά μας.

Το περιβάλλον του MATLAB.



Ο συντάκτης προγραμμάτων:



Στο MATLAB υπάρχει μία γενική αρχή, **όλες οι ποσότητες θεωρούνται πίνακες (ακόμη και οι αριθμοί θεωρούνται 1x1 πίνακες) και αποθηκεύονται κατά στήλες στη μνήμη του υπολογιστή.** Επίσης θα πρέπει να γνωρίζουμε τους ακόλουθους κανόνες:

Μετά την πληκτρολόγηση ενός υπολογισμού πατάμε Enter για να λάβουμε ως απάντηση το αποτέλεσμα.

1. Η χρήση του ; στο τέλος ενός υπολογισμού έχει ως αποτέλεσμα να γίνει ο υπολογισμός αλλά να μην εμφανιστεί το αποτέλεσμα.
2. ( ) και [ ] έχουν διαφορετική σημασία.
3. Στα ονόματα μεταβλητών το MATLAB είναι ευαίσθητο στη χρήση κεφαλαίων και πεζών γραμμάτων. Στα ονόματα δε βάζουμε κενά ή σύμβολα.
4. Πληκτρολόγηση ονόματος μεταβλητής και Enter δίνει ως απάντηση την τιμή της.
5. Με τα βέλη κάνεις scroll σε προηγούμενες εντολές.
6. Η εντολή **help topic** μας δίνει online βοήθεια για το topic. Πιο αναλυτική βοήθεια και παραδείγματα είναι διαθέσιμη από τον κατάλογο επιλογών Help→ MATLAB Help.
7. Υπολογισμός που θέλουμε να συνεχιστεί στην από κάτω γραμμή εισόδου πρέπει στο τέλος της κάθε ενδιάμεσης γραμμής να έχει "...".

ΠΡΟΣΟΧΗ: **Στα παραδείγματα εμφανίζεται η προτροπή εντολών >> που μας εμφανίζει το MATLAB εάν επιθυμείτε να εκτελέσετε κάποια εντολή στο δικό σας υπολογιστή δεν θα πρέπει να πληκτρολογείτε και τους χαρακτήρες >>.**

ΕΡΓΑΣΤΗΡΙΑΚΟΣ ΟΔΗΓΟΣ

Διαδικασία εργαστηρίου: Διαβάζετε το κείμενο που προηγείται των εντολών. Εκτελείτε τις εντολές και στο κενό δεξιά τους γράφετε το αποτέλεσμα και σχόλιά σας για το τι κάνει η εντολή και γιατί έδωσε το συγκεκριμένο αποτέλεσμα. Εάν έχετε οποιαδήποτε απορία απευθύνεστε στον διδάσκοντά σας και το συζητάτε μαζί του.

Εισαγωγικά θέματα

Στο MATLAB εξ ορισμού οι πράξεις μεταξύ αριθμητικών ποσοτήτων γίνονται με διπλή ακρίβεια, δηλαδή με 16 ψηφία σχετική ακρίβεια. Με την χρήση όμως κατάλληλων δηλώσεων μπορούμε να χρησιμοποιήσουμε και ποσότητες απλής ακρίβειας (με 8 ψηφία ακρίβεια). Αυτό όμως είναι κάτι που δεν θα μας απασχολήσει.

Οι πράξεις που εκτελεί το περιβάλλον είναι η πρόσθεση, η αφαίρεση, ο πολλαπλασιασμός, η δεξιά διαίρεση, η αριστερή διαίρεση και η ύψωση σε δύναμη, με τελεστές

+ - * / \ ^

αντίστοιχα. Η προτεραιότητα των πράξεων καθορίζεται με τη χρήση παρενθέσεων και σύμφωνα με τα όσα γνωρίζουμε από τα Μαθηματικά και τις άλλες γλώσσες προγραμματισμού. Με το

=

εκχωρούμε σε μεταβλητές τις μαθηματικές εκφράσεις.

Έστω ότι θέλουμε να υπολογίσουμε την τιμή της παράστασης :

$$2 + \frac{11}{3} + 25^2$$

η οποία έχει αποτέλεσμα τον άρρητο $55\frac{2}{3} = 55.6666\ldots$. Στην προτροπή εντολών (prompt όπως ονομάζεται η οποία εμφανίζεται με τους χαρακτήρες >>) πληκτρολογούμε την ακόλουθη εντολή :

```
>> 2+11/3+2*5^2
```

55.6667

και πατάμε Enter ↵.

Το περιβάλλον του MATLAB εκτελεί τις πράξεις με τη γνωστή προτεραιότητα και υπολογίζει το αποτέλεσμα με 16 ψηφία ακρίβεια. Ωστόσο, το περιβάλλον εμφανίζει το αποτέλεσμα χρησιμοποιώντας τα 5 σημαντικά ψηφία του (με τέσσερα δεκαδικά και μέχρι τρία ακέραια ψηφία στρογγυλοποιώντας το τελευταίο δεκαδικό ψηφίο).

Έστω ότι τώρα θέλουμε να υπολογίσουμε την τιμή της παράστασης :

$$2 + \frac{11}{3} + 25^2 + 100\ (2 + 5^3)$$

η οποία έχει αποτέλεσμα τον άρρητο

$$55\frac{2}{3} + 12700 = 12700 + 55.66666\ldots. = 12755.6666\ldots\ .$$

Σε αυτήν την περίπτωση πληκτρολογούμε:

```
>> 2+11/3+2*5^2+100*(2+5^3)
```

$1.2756e+004 = 1.2756 \times 10^4 \approx 12756$

Εδώ, επειδή δεν μπορεί να εφαρμοστεί ο παραπάνω κανόνας (υπάρχουν παραπάνω από τρία ακέραια ψηφία), το αποτέλεσμα εμφανίζεται σε εκθετική μορφή. **Δεν πρέπει να ξεχνάμε ότι αυτό που βλέπουμε είναι μόνο μία αναπαράσταση του αριθμού με ένα συγκεκριμένο τρόπο και όχι η τιμή που έχει αποθηκεύσει ο υπολογιστής.**

Η **εκθετική μορφή** αναπαράστασης αριθμού έχει τη μορφή:

$$\alpha.\beta e \pm c$$

όπου το α είναι το ακέραιο μέρος, το β το δεκαδικό και $\pm$ το εκθετικό μέρος της αναπαράστασης του αριθμού. Για καθένα από τα α , β και c μπορεί να διατίθενται ένα ένας συγκεκριμένος αριθμός ψηφίων (ανάλογα με το περιβάλλον που εμφανίζεται ο αριθμός).

Ένας τέτοιος αριθμός διαβάζεται ως

$$\alpha.\beta \times 10^{\pm c}$$

Για παράδειγμα το $2.1423e+02$ είναι η εκθετική αναπαράσταση του $2.1423 \times 10^{+2} = 214.23$, ενώ το $2.1423e-003$ είναι η εκθετική αναπαράσταση του $2.1423 \times 10^{-3} = 0.0021423$.

Η μεταβλητή ans

Όπως πιθανά γνωρίζετε, **μεταβλητή** στα προγραμματιστικά περιβάλλοντα (όπως είναι και το MATLAB) είναι μία θέση στη μνήμη του υπολογιστή η οποία έχει ένα όνομα μπορεί να πάρει κάποιο περιεχόμενο. Όταν αναφερόμαστε στο όνομα της μεταβλητής είναι σαν να αναφερόμαστε το περιεχόμενό της. Την τιμή της μεταβλητής μπορούμε να την αλλάξουμε. Μέχρι τώρα δεν έχουμε μιλήσει καθόλου για μεταβλητές, ωστόσο στο περιβάλλον του MATLAB υπάρχει μία συγκεκριμένη μεταβλητή η οποία λαμβάνει ως τιμή το αποτέλεσμα ενός υπολογισμού τον οποίο δεν έχουμε εκχωρήσει σε κάποια μεταβλητή. Η μεταβλητή αυτή έχει όνομα **ans**. Μπορούμε να χρησιμοποιήσουμε την τιμή της συγκεκριμένης μεταβλητής καλώντας το όνομά της ή χρησιμοποιώντας τη σε πράξεις. Η μεταβλητή με όνομα **ans** διατηρεί πάντα στη μνήμη το αποτέλεσμα της τελευταίας εντολής η οποία δεν έχει εκχωρηθεί σε μία συγκεκριμένη μεταβλητή. Παρατηρήστε στο παράθυρο του χώρου μνήμης (Workspace) έχει εμφανιστεί η μεταβλητή αυτή. Διπλοπατήστε την και δείτε τι γίνεται.

Έστω ότι μετά από όσες εντολές έχουμε εκτελέσει έως τώρα εκτελέσουμε την εντολή:

$1.4756e+004 = 1.4756 \times 10^{4} \approx 14756$

$1.4756e+004 = 1.4756 \times 10^{4} \approx 14756$

```
>> ans+2000
```

Μέχρι πριν την εκτέλεση της συγκεκριμένης εντολής η μεταβλητή με όνομα ans είχε τιμή 12755.6666…. Με τη συγκεκριμένη εντολή προστίθεται στο περιεχόμενό της η τιμή 2000 και εκχωρείται το αποτέλεσμα στην ίδια μεταβλητή.

Εκχώρηση τιμών σε μεταβλητές

Όπως αναφέραμε εκτός από τη δεξιά διαίρεση, ως πράξη υπάρχει και η αριστερή διαίρεση :

$$a \setminus b = a^{-1}b = b / a$$

Για τα δύο είδη διαιρέσεων ισχύει:

$$2 / 3 = 2 \times 3^{-1} = 0.666\ldots \text{ , } 2 \setminus 3 = 2^{-1} \times 3 = 1.5$$

Η χρήση της έχει μεγαλύτερο νόημα στην περίπτωση που εφαρμόζεται μεταξύ πινάκων ωστόσο, ας δούμε την εφαρμογή της με αριθμούς. Στο MATLAB πληκτρολογήστε:

```
>> a=2
```

Με αυτόν τον τρόπο εκχωρήσαμε στη μεταβλητή με όνομα a την τιμή 2. Παρατηρήστε ότι δεν έχουμε κάνει καμία δήλωση τύπου της μεταβλητής αυτής (δεν είναι απαραίτητο άλλωστε). Παρατηρήστε επίσης ότι εάν διπλοπατήσουμε το όνομά της στο Wrokspace ανοίγει ο συντάκτης μεταβλητών, όπου αναφέρεται η μεταβλητή ως ένα πίνακας 1x1 με τιμές διπλής ακρίβειας. Για να κατανοηθούν πλήρως οι διαφορές των δύο τύπων διαίρεσης εκτελέστε τις ακόλουθες εντολές:

```
>> b=3
>> c=a/b
>> d=a\b
```

0.6667

1.5000

Η εντολή format

Με τη χρήση της εντολής **format** μπορούμε να επιλέξουμε τον τρόπο με τον οποίο θα μας παρουσιάζονται οι αριθμητικές ποσότητες. Η προκαθορισμένη κατάσταση είναι να εμφανίζονται οι αριθμοί σε μορφή **short** , δηλαδή εμφανίζει τους αριθμούς με το πολύ 4 δεκαδικά ψηφία και το πολύ 3 ακέραια. Εάν ο αριθμός δεν μπορεί να παρασταθεί έτσι τότε τον εμφανίζει σε εκθετική μορφή με «short» αναπαράσταση του ακέραιου και δεκαδικού μέρους. Άλλη επιλογή είναι η επιλογή **long** όπου για την αναπαράσταση του ακέραιου μέρους μπορούν να εμφανιστούν μέχρι 2 ή 3 ψηφία και για την αναπαράσταση του δεκαδικού μέρους 15 ψηφία. Επίσης η επιλογή **long e** επιβάλει την «long» αναπαράσταση του ακέραιου και δεκαδικού μέρους σε εκθετική μορφή. Τα παραπάνω γίνονται αντιληπτά με τα παρακάτω παραδείγματα.

```
>> c=2+10/3+2*5^2-10*(2+5^3)
```

-1.2147e+003

```
>> d=2/5
```

0.4000

```
>> format long
>> c
```

-1.214666666666667e+003

```
>> d
```

0.400000000000000

```
>> format long e
>> c
```

-1.214666666666667e+003

```
>> d
```

4.000000000000000e-001

```
>> format short e
>> c
-1.2147e+003
>> d
4.0000e-001
>> format short
>> c
-1.2147e+003
>> d
0.4000
```

Ισοδύναμα μπορεί κάποιος να αλλάξει τον τρόπο εμφάνισης των αριθμών από την επιλογή **Preferences** του καταλόγου επιλογών **File**. Υπάρχουν και άλλες επιλογές για την εντολή format αλλά δεν θα μας απασχολήσουν.

Χειρισμός μεταβλητών, περισσότερα για τις αριθμητικές ποσότητες.

Όπως είδαμε, όλες οι μεταβλητές, από τη στιγμή την οποία τους εκχωρούμε μία τιμή αποθηκεύονται σε ένα τμήμα της μνήμης του υπολογιστή το οποίο το MATLAB ονομάζει **Workspace**. Στη συνηθισμένη μορφή του περιβάλλοντος του MATLAB μπορούμε να δούμε στην αριστερή πάνω γωνία τα περιεχόμενα του Workspace σε ένα παράθυρο (δείτε την εικόνα στην πρώτη σελίδα 3). Εκεί εμφανίζονται οι μεταβλητές, ο τύπος τους, και το περιεχόμενό τους. Από τη στιγμή που δε χρειαζόμαστε κάποια μεταβλητή καλό είναι να την διαγράφουμε από το Workspace με την εντολή clear. Πριν ξεκινήσουμε μία νέα εργασία ενδείκνυται να εκτελούμε την εντολή **clear all** ώστε το workspace του MATLAB να είναι απαλλαγμένο από μεταβλητές που αφορούν παλαιότερους υπολογισμούς. Δείτε στο παράθυρο του Workspace την μεταβλητή με όνομα a.

```
>> clear a
>> a
??? Undefined function or variable 'a'
```

Μετά την εκτέλεση της εντολής clear η μεταβλητή δεν υπάρχει πια στο Workspace.

Οι αριθμοί διπλής ακρίβειας στο MATLAB αποθηκεύονται στην μνήμη του υπολογιστή με λέξεις των 64-bit. Οι μη μηδενικές ποσότητες έχουν ένα εύρος από 10^{-308} έως 10^{308} περίπου. Η μονάδα στρογγυλοποίησης είναι $2^{-53} \approx 1.11 \times 10^{-16}$. Η μονάδα στρογγυλοποίησης (roundoff unit) είναι το όριο του **σχετικού** σφάλματος όταν εκτελούνται οι βασικές πράξεις (πρόσθεση, αφαίρεση, πολλαπλασιασμό και διαίρεση) μεταξύ δύο αριθμών κινητής υποδιαστολής και στον υπολογισμό της ρίζας αριθμού. Με απλά λόγια, όλα τα παραπάνω εννοούν ότι όπως έχουμε αναφέρει στο περιβάλλον του MATLAB οι πράξεις γίνονται με 16 σημαντικά ψηφία ακρίβεια.

Οι συναρτήσεις **realmax** και η **realmin** επιστρέφουν τη μεγαλύτερη και τη μικρότερη θετική ποσότητα, αντίστοιχα, που μπορεί να χειριστεί το περιβάλλον.

```
>> realmax
1.7977e+308
>> realmin
2.2251e-308
```

Η **Inf** και η **NaN** αποτελούν δύο «τιμές» που εκφράζουν την άπειρη ποσότητα (δηλαδή ποσότητα η οποία είναι μεγαλύτερη από την μεγαλύτερη ποσότητα που μπορεί να χειριστεί το περιβάλλον) και τη μη οριζόμενη μαθηματική ποσότητα ($\frac{0}{0}, \frac{\infty}{\infty}, 0 \times \infty$) αντίστοιχα.

```
>> 1/0
Inf
>> 0/0
NaN
```

Οποιαδήποτε ποσότητα μεγαλύτερη από τη realmax προκαλεί υπερχείλιση (overflow). Σε μία τέτοια περίπτωση, σε αντίθεση με άλλα προγραμματιστικά περιβάλλοντα, δεν προκαλείται τερματισμός του προγράμματος, απλά η τιμή της μεταβλητής θέτεται ίση με την άπειρη ποσότητα. Παρατηρήστε τα αποτελέσματα των ακόλουθων εντολών.

```
>> 1.1*realmax
Inf
```

```
>>-2*realmax
```
-Inf

Το **eps** αποτελεί τη διαφορά μεταξύ του 1 και του επόμενου αριθμού κινητής υποδιαστολής.

```
>> format long e
>> eps
```
2.220446049250313e-016

```
>> 1+ eps
```
1.000000000000000e+000

Παρατηρήστε ότι η τιμή του για την περίπτωση του αριθμού 1 είναι περίπου δύο φορές τη μονάδα στρογγυλοποίησης. Εάν δηλαδή η το αποτέλεσμα είναι μεταξύ του ένα και μικρότερο του ένα συν τη μονάδα στρογγυλοποίησης ο αριθμός στρογγυλοποιείται στη μονάδα. Σε αντίθετη περίπτωση, εάν ο αριθμός που υπολογίζεται είναι μεταξύ του ένα συν τη μονάδα στρογγυλοποίησης και του ένα συν το eps τότε η στρογγυλοποίηση γίνεται στο 1+eps.

Η συνάρτηση **realmin** επιστρέφει τον μικρότερο θετικό κανονικοποιημένο αριθμό κινητής υποδιαστολής. Οποιοσδήποτε υπολογισμός μικρότερος από την ποσότητα **realmin** είτε θεωρείται μηδέν, εάν έχει αποτέλεσμα μικρότερο του **realmin *eps,** είτε προκαλεί κάποια μικρή προσότητα με πολλά μηδενικά στην εκθετική του αναπαράσταση.

```
>> realmin*eps
```
4.940656458412465e-324

```
>> realmin*eps/2
```
0

Για να κατανοήσει κάποιος πλήρως τους παραπάνω υπολογισμούς πρέπει να εμβαθύνει στους κανόνες αριθμητικών υπολογισμών κινητής υποδιαστολής (IEEE) που εκτελεί ο υπολογιστής.

Βοήθεια στο MATLAB

Εκτελώντας την εντολή **help** ή από τον κατάλογο επιλογών **HELP-> MATLAB Help** μπορείτε να βρείτε βοήθεια σχετικά με την κάθε εντολή του MATLAB.

```
>> help eps
```

Η βοήθεια που παρέχει το περιβάλλον είναι πραγματικά εκτενής και θα καλύψει πλήρως τις ανάγκες του μέσου χρήστη.

Θέμα για συζήτηση στο εργαστήριο:

Οι υπολογισμοί σφαλμάτων, στο περιβάλλον του MATLAB, δεν γίνονται στις απόλυτες ποσότητες. Η απόσταση ενός αριθμού x από τον επόμενο σε μέγεθος αριθμό που κατανοεί και αποθηκεύει το περιβάλλον δεν είναι σταθερή, αλλά υπολογίζονται με **σχετικό** τρόπο. Στις πιο πρόσφατες εκδόσεις του MATLAB **(και όχι στην 6.5 που είναι εγκατεστημένη στα εργαστήριά μας)**, η εντολή **eps(x)** μας επιστρέφει θετική απόσταση μεταξύ του αριθμού x και του επόμενου σε μέγεθος αριθμού. Παρατηρώντας τα παρακάτω αποτελέσματα (τα οποία λαμβάνουμε σε κάποια από τις πιο πρόσφατες εκδόσεις του MATLAB) σημειώστε παρακάτω το πως κατανοείτε την έννοια **απόλυτο σφάλμα** και **σχετικό σφάλμα**. Μπορείτε να το εκφράσετε με ένα τύπο;

```
>> eps(1)
```
2.220446049250313e-016

```
>> eps(1/2)
```
1.110223024625157e-016

```
>> eps(2)
```
4.440892098500626e-016

```
>> eps(100)
```
1.421085471520200e-014

Οι μιγαδικοί αριθμοί στο MATLAB

Ξέρουμε ότι δεν υπάρχει πραγματικός αριθμός που επαληθεύει την εξίσωση $x^2 = -1$. Η ανάγκη επίλυσης τέτοιων εξισώσεων οδηγεί στο σύνολο των μιγαδικών αριθμών. Το σύνολο $\mathbb{C}$ **των μιγαδικών αριθμών** είναι ένα σύνολο που περιέχει το σύνολο $\mathbb{R}$ των πραγματικών αριθμών.

Υπάρχει στο $\mathbb{C}$ ένα στοιχείο i τέτοιο ώστε $i^2 = -1$. Επιπλέον κάθε στοιχείο του $\mathbb{C}$ έχει μοναδική παράσταση της μορφής $a + bi$ όπου $a, b \in \mathbb{R}$. Επομένως, αν $a, a', b, b' \in \mathbb{R}$, έχουμε

$$a + bi = a' + b'i \Leftrightarrow a = a' \text{ και } b = b'.$$

Έστω $z = a + bi \in \mathbb{C}$ όπου $a, b \in R$.

Ο πραγματικός αριθμός a ονομάζεται το **πραγματικό μέρος** του z και συμβολίζεται με $\mathrm{Re}(z)$. Ο πραγματικός αριθμός b ονομάζεται το **φανταστικό μέρος** του z και συμβολίζεται με $\mathrm{Im}(z)$.

Αν για παράδειγμα $z = \frac{1}{2} + 3i$, τότε $\mathrm{Re}(z) = \frac{1}{2}$, $\mathrm{Im}(z) = 3$.

Ένας μιγαδικός αριθμός είναι πραγματικός αν και μόνο αν το φανταστικό μέρος του είναι ίσο με μηδέν:

$$z \in \mathbb{R} \Leftrightarrow \mathrm{Im}(z) = 0.$$

Ξέρουμε ότι ένα τριώνυμο $ax^2 + bx + c$, όπου $a, b, c \in \mathbb{R}$, έχει πραγματικές ρίζες αν και μόνο αν η διακρίνουσά του είναι μη αρνητική, δηλαδή $b^2 - 4ac \geq 0$. Όμως *κάθε* τριώνυμο έχει μιγαδικές ρίζες (ανεξάρτητα από το πρόσημο της διακρίνουσας). Πράγματι, εύκολα επαληθεύεται με πράξεις ότι η εξίσωση $ax^2 + bx + c = 0$, όπου $a \neq 0$, έχει ρίζες τις $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$. Αν $b^2 - 4ac < 0$, τότε οι ρίζες αυτές γράφονται $x = \frac{-b \pm \sqrt{(-1)\left(4ac - b^2\right)}}{2a} = \frac{-b \pm i\sqrt{\left(4ac - b^2\right)}}{2a}$. Για παράδειγμα, οι ρίζες του $x^2 + 2x + 3$ είναι $x = \frac{-2 \pm i\sqrt{8}}{2} = -1 \pm i\sqrt{2}$.

Στο MATLAB οι μεταβλητές i και j είναι ίσες με τη φανταστική μονάδα, εκτός και αν τους αλλάξουμε τιμή εκχωρώντας τους μία άλλη. Αυτό δεν είναι καλό να γίνεται εάν σκοπεύουμε να εργαστούμε με ποσότητες που μπορεί να είναι μιγαδικές.

```
>> i^2
ans =
   -1
>> j^2
ans =
   -1
>> i=2
i =
     2
>> i^2
ans =
     4
>> clear i
>> i
ans =
   0 + 1.0000i
```

Έστω μιγαδικοί $z, w \in \mathbb{C}$ με $z = a + bi$ και $w = c + di$.

- **Πρόσθεση - Αφαίρεση** Ορίζουμε το άθροισμα $z \pm w$ ως εξής

$$z \pm w = (a \pm c) + (b \pm d)i.$$

- **Πολλαπλασιασμός** Ορίζουμε το γινόμενο zw ως εξής

$$zw = (ac - bd) + (ad + bc)i$$

- ο **αντίστροφος** του $a+bi$ και συμβολίζεται με $(a+bi)^{-1}$ ή $\frac{1}{a+bi}$ και θα είναι ο μιγαδικός αριθμός $\frac{a}{a^2+b^2}+\frac{-b}{a^2+b^2}i$. Ισχύει $(a+bi)(a+bi)^{-1}=1$

- Η **διαίρεση** ορίζεται ως πολλαπλασιασμός με τον αντίστροφο $z/w = z\cdot w^{-1}$

Στο MATLAB μπορούμε να ορίσουμε μιγαδικούς αριθμούς και να κάνουμε τις βασικές πράξεις με αυτούς. Η φανταστική μονάδα θα πρέπει είτε να πολλαπλασιάζεται με τον τελεστή * είτε να πληκτρολογείται κολλητά στο φανταστικό μέρος. Η συνάρτηση **sqrt( )** υπολογίζει την τετραγωνική ρίζα του ορίσματός της.

```
>> clear all
>> z=sqrt(2)-5i
>> w=1+3*i
>> z+w
>> z*w
>> z/w
>> z*w^-1
```

1.4142 - 5.0000i

1.0000 + 3.0000i

2.4142 - 2.0000i

16.4142 - 0.7574i

-1.3586 - 0.9243i

-1.3586 - 0.9243i

Έστω $z=a+bi\in\mathbb{C}$ όπου $a,b\in\mathbb{R}$.

Ο πραγματικός αριθμός a ονομάζεται το **πραγματικό μέρος**, ενώ ο πραγματικός αριθμός b ονομάζεται το **φανταστικό μέρος** του z. Μέτρο Μιγαδικού Αριθμού $z=a+bi$ είναι ο αριθμός $\sqrt{a^2+b^2}$ και συμβολίζεται με $|z|$. Ο **συζυγής** του μιγαδικού αριθμού $z=a+bi$ είναι ο $a-bi$ και συμβολίζεται με $\bar{z}$ και ισχύει η σχέση $z\bar{z}=a^2+b^2=|z|^2$.

Με τις συναρτήσεις **abs( )**, **conj( )**, **real( )**, **imag( )** υπολογίζουμε το μέτρο, τον συζυγή, το πραγματικό και το μιγαδικό μέρος ενός μιγαδικού αριθμού αντίστοιχα. Οπότε χρησιμοποιώντας τους παραπάνω τύπους και ορισμούς οι τρεις ακόλουθες εντολές θα πρέπει να δώσουν το ίδιο αποτέλεσμα.

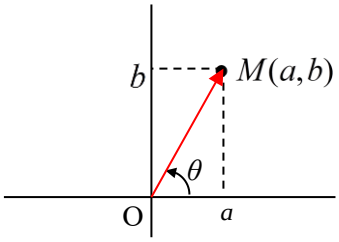
```
>> abs(z)^2
>> real(z)^2+imag(z)^2
>> z*conj(z)
```

27

27

27

Ας θεωρήσουμε την εικόνα ενός μη μηδενικού μιγαδικού αριθμού $z=a+bi$ στο επίπεδο. Έστω θ η γωνία (μετρημένη σε ακτίνια) που σχηματίζεται αν κινηθούμε με φορά αντίθετη της κίνησης των δεικτών ρολογιού από τον ημιάξονα Ox στο OM όπως φαίνεται στο σχήμα[1].



Τότε έχουμε $a=|OM|\cos\theta$ και $b=|OM|\sin\theta$. Συνεπώς ισχύει $z=a+bi=|OM|(\cos\theta+i\sin\theta)$.

[1] Εννοούμε ότι $0\le\theta<2\pi$.

Η παράσταση $|OM|(\cos\theta + i\sin\theta)$ ονομάζεται η **τριγωνομετρική μορφή** του $a+bi$. Παρατηρούμε ότι το $|OM|$ είναι το μέτρο του $a+bi$. Η δε γωνία θ $(0 \le \theta < 2\pi)$ ονομάζεται το **(πρωτεύον) όρισμα** του $a+bi$. Συνεπώς η τριγωνομετρική μορφή είναι

$$\rho(\cos\theta + i\sin\theta)$$

όπου $\rho = \sqrt{a^2+b^2}$ και θ είναι το όρισμα του $a+bi$.

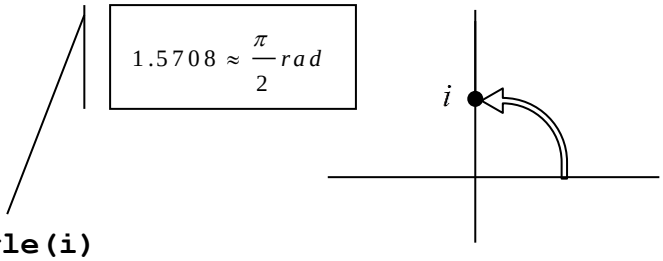
Το όρισμα του $a+bi$ προσδιορίζεται από τις σχέσεις

$$a = \rho\cos\theta,\ b = \rho\sin\theta,\ 0 \le \theta < 2\pi\ .$$

Διαιρώντας τους δύο αυτούς τύπους συμπεραίνουμε ότι

$$\tan\theta = b/a \Rightarrow \theta = \arctan(b/a),\quad 0 \le \theta < 2\pi\ .$$

Με τη συνάρτηση **angle( )** υπολογίζουμε το όρισμα (σε ακτίνια) ενός μιγαδικού αριθμού. Το αποτέλεσμα μπορεί να είναι και αρνητικός αριθμός μιας και η συγκεκριμένη συνάρτηση επιστρέφει το όρισμα σε τιμές [-π,π]. Δηλαδή, για μιγαδικούς με όρισμα μεγαλύτερο του π επιστρέφει αρνητικές γωνίες. Για παράδειγμα για την φανταστική μονάδα το όρισμα είναι:




```
>> angle(i)
```

Η τριγωνομετρική συνάρτηση **atan()** επιστρέφει το τόξο εφαπτομένης γωνίας σε ακτίνια. Σύμφωνα με όσα έχουμε πει οι δύο ακόλουθοι υπολογισμοί θα πρέπει να δώσουν το ίδιο αποτέλεσμα.

```
>> angle(z)
```
-1.2952

```
>> atan(imag(z)/real(z))
```
-1.2952

Έστω ότι $z = \rho_1(\cos(\theta_1) + i\sin(\theta_1))$ και $w = \rho_2(\cos(\theta_2) + i\sin(\theta_2))$ τότε για το γινόμενο των δύο μιγαδικών αριθμών τότε ισχύει $zw = \rho_1\rho_2(\cos(\theta_1+\theta_2) + i\sin(\theta_1+\theta_2))$. Οπότε, θεωρώντας τους μιγαδικούς του παραδείγματός μας τότε οι ακόλουθες ποσότητες αναμένουμε να ισούνται με μηδέν.

```
>> angle(z*w)-angle(z)-angle(w)
```
0

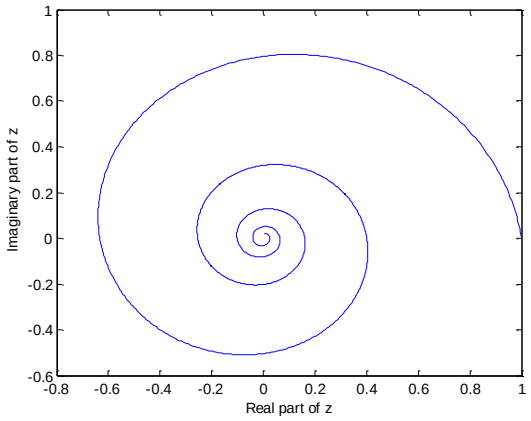
```
>> abs(z*w)-abs(z)*abs(w)
```
-3.5527e-015

Παρατηρείστε ότι, λόγω του ότι το MATLAB εκτελεί τις πράξεις με ακρίβεια δεκαέξι ψηφίων, δεν επιστρέφει το αναμενόμενο μηδενικό αποτέλεσμα αλλά μία πολύ μικρή ποσότητα της τάξης του 10^{-15}.

Ένα πρώτο γράφημα

Όταν ζητήσουμε το γράφημα μίας μιγαδικής ποσότητας με την εντολή **plot( )** το MATLAB εμφανίζει ένα σημείο με τετμημένη (στον άξονα των x) το πραγματικό μέρος του μιγαδικού και τεταγμένη (στον άξονα των y) το μιγαδικό μέρος του.

Με τις παρακάτω εντολές υπολογίζουμε τις δυνάμεις $z^0, z^{0.01}, z^{0.02}, \ldots, z^{20}$ ενός μιγαδικού αριθμού z και τις τοποθετούμε σε ένα διάνυσμα και κάνουμε το γράφημά των στοιχείων του διανύσματος στο επίπεδο xy ενώνοντας τα διαδοχικά σημεία με ευθύγραμμα τμήματα. Για το ακριβώς είναι διάνυσμα, το πώς σχηματίζουμε τις παραπάνω δυνάμεις με τη χρήση της εντολής n=0:0.01:20 και τη χρήση της πράξης .^ θα αναφερθούμε λίγο παρακάτω. Ωστόσο τα αποτελέσματά τους φαίνονται δίπλα στις εντολές. Οι **xlabel,ylabel** βάζουν τίτλους στους άξονες του γραφήματος.




```
>> clear all
>> z=0.2+0.8i;
>> n=0:0.01:20;
>> w=z.^n;
>> plot(w),xlabel('Real part of z'),ylabel('Imaginary part of z')
```

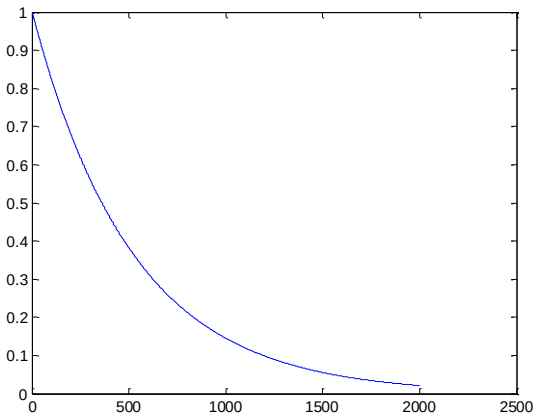
$$n = [0, 0.01, 0.02, ..., 19.99, 20]$$

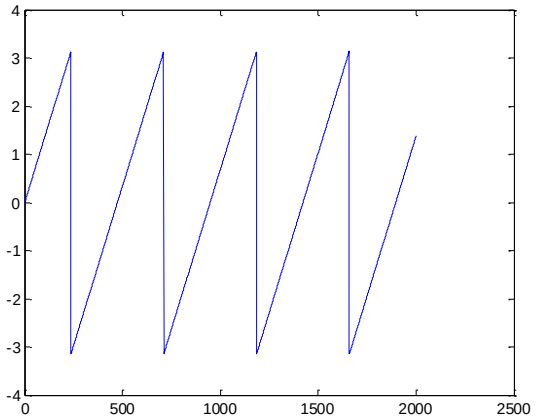
$$w = [z^0, z^{0.01}, z^{0.02}, ..., z^{19.99}, z^{20}]$$

Εάν θέλουμε να εξηγήσουμε το όμορφο σχήμα που εμφανίζεται θα πρέπει να παρατηρήσουμε ότι ο αριθμός z έχει μέτρο μικρότερο της μονάδας. Οι δυνάμεις του συγκεκριμένου αριθμού είναι μιγαδικοί με μέτρο τις αντίστοιχες δυνάμεις του μέτρου τους z και ορίσματα πολλαπλάσια του ορίσματος του z. Η πρώτη δύναμη είναι το z^0 το οποίο είναι το 1. Όλες οι υπόλοιπες δυνάμεις έχουν μέτρο δύναμη του μέτρου του z οπότε είναι αριθμοί με μέτρο μικρότερο του 1. Τα ορίσματα των δυνάμεων είναι γωνίες από 0 έως το όρισμα του z για τις πρώτες 101 δυνάμεις $[0, 0.01, 0.02, ..., 0.99, 1]$ και γωνίες πολλαπλάσιες του ορίσματος του z για τις υπόλοιπες δυνάμεις $[1.01, 1.02, ..., 19.99, 20]$.

Μπορείτε να διαπιστώσετε τα παραπάνω βλέποντας τις τιμές της μεταβλητής w στο Workspace ή με το να κάνουμε γράφημα τις τιμές των γραφημάτων των μέτρων και των ορισμάτων των δυνάμεων του z.

```
>> figure
>> plot(abs(w))
>> figure
>> plot(angle(w))
```





Με την εντολή **figure** δημιουργείται ένα νέο παράθυρο γραφικών στο οποίο εμφανίζεται το αποτέλεσμα της επόμενης εντολής plot. Έτσι κάθε γράφημα εμφανίζεται σε διαφορετικό παράθυρο από το προηγούμενο.

Άσκηση για εξάσκηση:

Καθαρίστε τη μνήμη από τις όποιες τιμές μεταβλητών υπάρχουν ενεργές. Αλλάξτε την εμφάνιση των αποτελεσμάτων σε εκθετική μορφή με αναπαράσταση του δεκαδικού μέρους με πολλά ψηφία. Ορίστε στο MATLAB τον μιγαδικό αριθμό $z = 2 + 5i$, και στη συνέχεια υπολογίστε στο MATLAB την ποσότητα

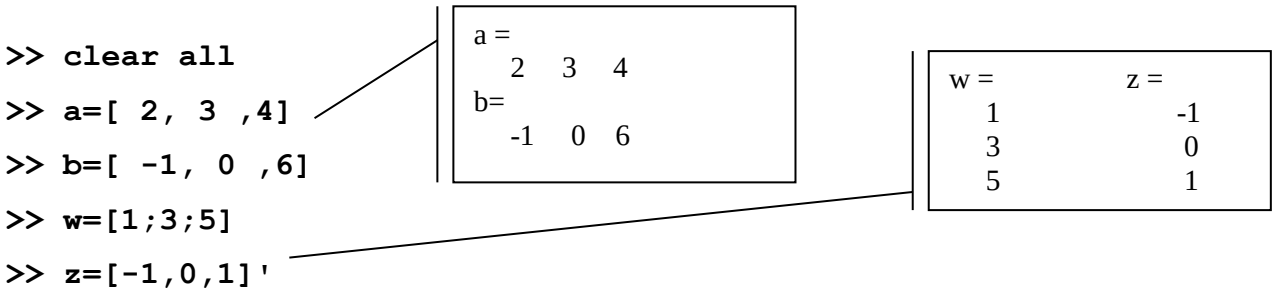
$$\left(\frac{\sin(20\pi + 2)}{3} + \frac{2^{Re(\mathrm{z})}}{\mathrm{Im(z)}}\cos(1)\right)^{50}$$

Πως διαβάζετε τον αριθμητικό αποτέλεσμα που λάβατε ως απάντηση;

Παρατήρηση: Στην παραπάνω έκφραση οι συμβολισμοί Re(z) και Im(z) αφορούν το πραγματικό και μιγαδικό μέρος του μιγαδικού αριθμού z αντίστοιχα. Δεν είναι εντολές ή συναρτήσεις του MATLAB αλλά έκφραση της μαθηματικής αναπαράστασης των ποσοτήτων αυτών. Πιο πάνω έχετε συναντήσει τις αντίστοιχες συναρτήσεις του MATLAB, Επίσης θα χρειαστείτε τη συνάρτηση **sin( )** η οποία επιστρέφει το ημίτονο γωνίας εκφρασμένης σε ακτίνια, και την **cos( )** η οποία ανάλογα επιστρέφει το συνημίτονο. Τέλος η μαθηματική σταθερά π στο περιβάλλον του MATLAB προσεγγίζεται από τη σταθερά pi.

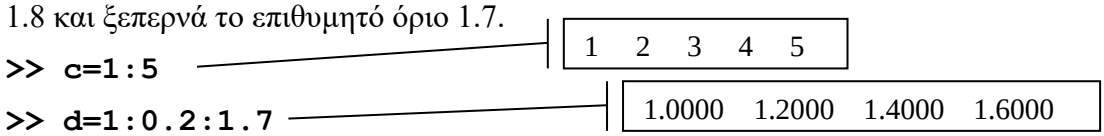
Ορισμός διανυσμάτων και βασικές διανυσματικές (στοιχείο προς στοιχειό) πράξεις

Ένα διάνυσμα στο MATLAB είναι μία δομή στην οποία μπορούμε να συγκεντρώσουμε πολλές τιμές και να τις εκχωρήσουμε σε μία μεταβλητή. Υπάρχουν τα διανύσματα γραμμή τα οποία ορίζονται παραθέτοντας τα στοιχεία τους μέσα σε αγκύλες χωρισμένα με κόμμα ή με κενό. Επίσης υπάρχουν και τα διανύσματα στήλη τα οποία ορίζονται παρόμοια, χωρίζοντας όμως τα στοιχεία τους με ; ή εισάγοντας τα στοιχεία της κάθε γραμμής και πατώντας Enter.

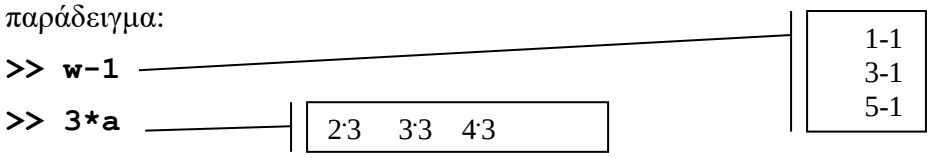


Η **απόστροφος (') επιστρέφει τον ανάστροφο του διανύσματος μετατρέποντας διανύσματα γραμμή σε διανύσματα στήλες και το αντίστροφο**. Για τα παραπάνω διανύσματα, τα a,b είναι 1x3, δηλαδή έχουν μία γραμμή (line) και τρεις στήλες (columns), και τα w,z είναι 3x1, δηλαδή έχουν 3 γραμμές και μία στήλη.

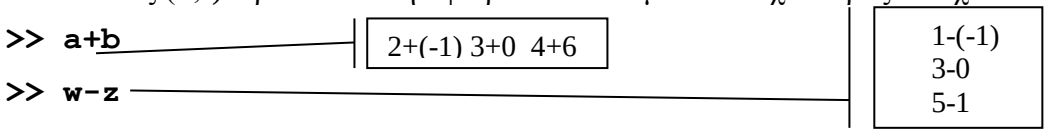
Ένας άλλος τρόπος για να ορίσουμε διανύσματα γραμμή γίνεται ορίζοντας τον πρώτο στοιχείο του (π.χ. α), το βήμα μεταβολής των τιμών του διανύσματος (π.χ. κ) και τον αριθμό που μπορεί να είναι μικρότερο ή ίσο το τελευταίο στοιχείο του (π.χ. β). Χωρίζουμε τους αριθμούς αυτούς με άνω κάτω τελεία. Δηλαδή, όταν πληκτρολογούμε **α:κ:β** το MATLAB ορίζει ένα διάνυσμα με πρώτο στοιχείο το **α**, επόμενα στοιχεία τα **α+κ,α+2κ,…** και τελευταίο στοιχείο το **α+νκ** όπου **α+(ν+1)κ>β**. Στην περίπτωση που δεν έχουμε ορίσει βήμα μεταβολής αυτό παίρνει εξ ορισμού την τιμή 1. Για την ακόλουθη μεταβλητή c, η πρώτη τιμή του διανύσματος είναι 1, το βήμα μεταβολής 1 και η τελική τιμή 5. Για τη μεταβλητή d, η πρώτη τιμή είναι 1, το βήμα μεταβολής 0.2 και η τελική τιμή το 1.6 διότι ο επόμενος αριθμός που υπολογίζεται με βάση το βήμα μεταβολής είναι 1.8 και ξεπερνά το επιθυμητό όριο 1.7.



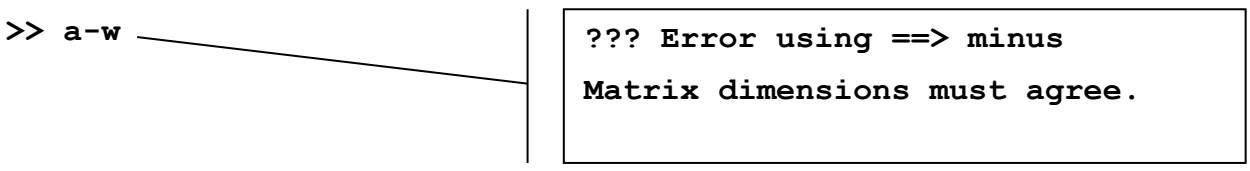
Όταν σε ένα διάνυσμα προσθέτουμε ή αφαιρούμε έναν αριθμό **προστίθεται ή αφαιρείται σε κάθε στοιχείο** του διανύσματος ο αριθμός αυτός. Όταν πολλαπλασιάζουμε ή διαιρούμε ένα διάνυσμα επί αριθμό **πολλαπλασιάζεται κάθε στοιχείο** του διανύσματος επί τον αριθμό. Για παράδειγμα:



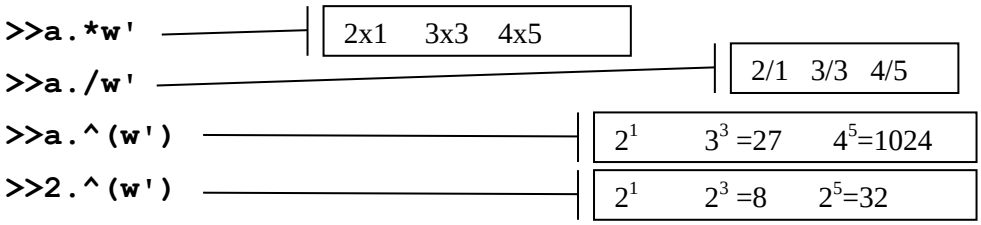
Οι τελεστές (+,-) προστεθούν ή αφαιρούν διανύσματα στοιχείο προς στοιχείο.



Οι πράξεις αυτές γίνονται όταν τα διανύσματα έχουν ακριβώς την ίδια διάσταση.



Οι **διανυσματικοί τελεστές (vectorized) (.* , ./ , .^)** μας δίνουν τη δυνατότητα να εφαρμόσουμε την αντίστοιχη πράξη (*,/,^) μεταξύ διανυσμάτων στοιχείο προς στοιχείο. **Οι πράξεις αυτές πρέπει να γίνονται μεταξύ διανυσμάτων ιδίου μεγέθους (σε γραμμές και στήλες).** Για την περίπτωση του w η εφαρμογή της πράξης της αναστροφής (') επιστρέφει ένα διάνυσμα με μία γραμμή και τρεις στήλες, οπότε τα a,w έχουν την ίδια διάσταση και μπορεί να εφαρμοστεί η πράξη στοιχείο προς στοιχείο. Η ύψωση σε δύναμη έχει την μεγαλύτερη προτεραιότητα από όλες τις πράξεις (και την αναστροφή) οπότε είναι απαραίτητη η παρένθεση ώστε να εφαρμοστεί πρώτα η αναστροφή και μετά η ύψωση σε δύναμη.



Παρατήρηση:
Τώρα μπορούμε να κατανοήσουμε τις εντολές που χρησιμοποιήσαμε στο παράδειγμα με το γράφημα του μιγαδικού.
Η εντολή n=0:0.01:20; δημιούργησε το διάνυσμα γραμμή με όνομα **n** και στοιχεία [0,0.01,0.02,…,19.98,19.99,20]. Η εντολή w=z.^n δημιουργεί ένα διάνυσμα όπου έχει ως στοιχεία $z^0, z^{0.01}, z^{0.02}, \ldots, z^{20}$. Εδώ έχουμε μία εξαίρεση στον κανόνα που αναφέραμε παραπάνω. Μόνο για την ύψωση αριθμού σε δύναμη σε ένα διάνυσμα, με την πράξη .^ το αποτέλεσμα είναι ένας διάνυσμα με στοιχεία τον αριθμό υψωμένο σε κάθε στοιχείο του διανύσματος.

Μαθηματικές συναρτήσεις και απλές γραφικές παραστάσεις

Οι κυριότερες μαθηματικές συναρτήσεις παρατίθενται σε πίνακα παρακάτω. Οι πιο βασικές, εκτός την **sqrt( )** που αναφέραμε παραπάνω, είναι η συνάρτηση **exp( )** υψώνει στην εκθετική δύναμη, η **sin( )** που επιστρέφει το ημίτονο, η **cos( )** που επιστρέφει το συνημίτονο, η **log( )** υπολογίζει τον νεπέριο λογάριθμο και η **log10( )** τον δεκαδικό λογάριθμο. Για τις τριγωνομετρικές συναρτήσεις οι γωνίες θεωρούνται σε ακτίνια. Τα ορίσματα μπορεί να είναι πραγματικοί ή μιγαδικοί (όπου ορίζονται) αριθμοί, διανύσματα ή πίνακες. **Γενικά, όταν τα ορίσματα είναι διανύσματα ή πίνακες τότε η συνάρτηση (με μικρές εξαιρέσεις) εφαρμόζεται σε κάθε στοιχείο του διανύσματος ή του πίνακα αντίστοιχα.** Το **ερωτηματικό στο τέλος μίας εντολής** έχει το αποτέλεσμα να εκτελείται η εντολή **δίχως να επιστρέφει κάποια έξοδο στην οθόνη**. Στο MATLAB η σταθερά pi περιέχει προσέγγιση της μαθηματικής ποσότητας π. Στο επόμενο παράδειγμα παρατηρούμε ότι αφού το όριο

$$\lim_{x \to 0} \frac{\sin(\pi x)}{x} = \pi$$

τα στοιχεία του διανύσματος y προσεγγίζουν τον αριθμό π, όσο τα στοιχεία του διανύσματος x τείνουν στο 0.

```
>> clear all
>> format long
>> x=[0.1;0.01;0.001;0.0001];
>> y=sin(pi*x)./x
>> y-pi
```

Για να κατανοήσουμε τις παραπάνω πράξεις δείτε ακολούθως τι υπολογίζει το MATLAB.

$$x = \begin{bmatrix} 0.1 \\ 0.01 \\ 0.001 \\ 0.0001 \end{bmatrix},\ pi * x = \begin{bmatrix} \pi \times 0.1 \\ \pi \times 0.01 \\ \pi \times 0.001 \\ \pi \times 0.0001 \end{bmatrix},\ \sin(pi * x) = \begin{bmatrix} \sin(\pi \times 0.1) \\ \sin(\pi \times 0.01) \\ \sin(\pi \times 0.001) \\ \sin(\pi \times 0.0001) \end{bmatrix},$$

$$y = \sin(pi * x)./x = \begin{bmatrix} \sin(\pi \times 0.1)/0.1 \\ \sin(\pi \times 0.01)/0.01 \\ \sin(\pi \times 0.001)/0.001 \\ \sin(\pi \times 0.0001)/0.0001 \end{bmatrix},\ ans = \sin(pi * x)./x - \pi = \begin{bmatrix} \sin(\pi \times 0.1)/0.1 - \pi \\ \sin(\pi \times 0.01)/0.01 - \pi \\ \sin(\pi \times 0.001)/0.001 - \pi \\ \sin(\pi \times 0.0001)/0.0001 - \pi \end{bmatrix}$$

Γράφημα συνάρτησης τρόπος Α.

Υπάρχουν διάφοροι τρόποι για να ορίσουμε μία σύνθετη μαθηματική συνάρτηση και να κάνουμε το γράφημά της. Στην πιο απλή μορφή μπορούμε να **ορίσουμε ένα διάνυσμα που να περιέχει μία πυκνή διαμέριση ισαπέχοντων σημείων του διαστήματος (τις τετμημένες) στο οποίο θέλουμε να σχεδιάσουμε τη συνάρτηση.** Μία **διαμέριση** ενός διαστήματος [α,β] είναι γενικά ένα σύνολο διατεταγμένων αριθμών που περιέχονται στο διάστημα. Για να ορίσουμε τη διαμέριση ενός διαστήματος ορίζουμε ένα διάνυσμα της μορφής **α:κ:β,** σύμφωνα με τα όσα είπαμε παραπάνω. Για παράδειγμα εάν θέλουμε να ορίσουμε στο διάστημα $[-2\pi, 2\pi]$ μία διαμέριση με **σταθερό πλάτος (δηλαδή απόσταση μεταξύ δύο διαδοχικών αριθμών)** 0.1 πληκτρολογούμε **-2*pi:0.1:2*pi**. Αφού την ορίσουμε μπορούμε να εφαρμόσουμε σε κάθε σημείο της διαμέρισης τον τύπο της συνάρτησης ώστε να **υπολογίσουμε ένα διάνυσμα που να περιέχει τις τιμές (εικόνες, τεταγμένες των σημείων) της συνάρτησης για αυτά τα σημεία.**

Στον ορισμό της μαθηματικής έκφρασης της συνάρτησης μπορούμε να χρησιμοποιούμε τις μαθηματικές συναρτήσεις του MATLAB και θα **πρέπει να χρησιμοποιούμε τους διανυσματικούς τελεστές** (.* ./ .^) που αναφέραμε παραπάνω. Με τη χρήση της **plot( )** μπορούμε να σχεδιάσουμε τα ζεύγη των τιμών της συνάρτησης. Μέσα στην παρένθεση χωρίζουμε τα διανύσματα με στοιχεία τα σημεία της διαμέρισης και των εικόνων χωρισμένα με κόμμα. Δηλαδή ζητάμε με τη χρήση της plot να κάνει το γράφημα όλων των ζευγών $(x, f(x))$ για κάθε σημείο της διαμέρισης που έχουμε επιλέξει. Το πρώτο διάνυσμα περιέχει τις τετμημένες των σημείων και το δεύτερο τις τεταγμένες.

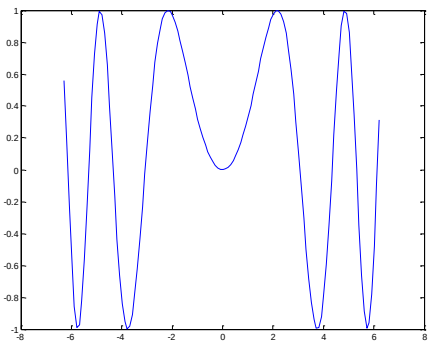
Το MATLAB αντιστοιχεί το κάθε σημείο του πρώτου διανύσματος με το κάθε σημείο του δεύτερου διανύσματος που βρίσκεται στην ίδια θέση και κάνει το γράφημα του σημείου σε άξονα xy, θεωρώντας ως τετμημένες των σημείων του γραφήματος τα στοιχεία του πρώτου διανύσματος και ως τεταγμένες τους τα στοιχεία του δεύτερου διανύσματος.

Προσοχή: Στην συγκεκριμένη και πιο απλή μορφή εκτέλεσης της εντολής plot, τα διαδοχικά σημεία ενώνονται με ευθύγραμμα τμήματα. Εάν η διαμέριση δεν είναι αρκετά πυκνή (δηλαδή τα σημεία έχουν μεγάλη απόσταση μεταξύ τους) τότε το αποτέλεσμα είναι γραφήματα που δεν είναι καλές αναπαραστάσεις της γραφικής παράστασης της συνάρτησης που μας ενδιαφέρει.

Όπως αναφέραμε το **ερωτηματικό στο τέλος** της εντολής δεν επιτρέπει την εμφάνιση του αποτελέσματος της εντολής, το οποίο μπορεί να είναι ενοχλητικό όταν πρόκειται για μεγάλα διανύσματα.

Για παράδειγμα έστω ότι θέλουμε να κάνουμε το γράφημα της $f(x) = \sin\left(x^2/3\right)$ στο διάστημα $[-2\pi, 2\pi]$ πληκτρολογούμε τις ακόλουθες εντολές:

```
>> clear all
>> x=-2*pi:0.1:2*pi;
>> f=sin(x.^2/3);
>> plot(x,f)
```



Για να κατανοήσουμε τις παραπάνω πράξεις δείτε ακολούθως τι υπολογίζει το MATLAB.

$$x = \begin{bmatrix} -2\pi \\ -2\pi + 0.1 \\ \mathrm{M} \\ 2\pi - 0.1 \\ 2\pi \end{bmatrix}, x.\wedge 2 = \begin{bmatrix} (-2\pi)^2 \\ (-2\pi + 0.1)^2 \\ \mathrm{M} \\ (2\pi - 0.1)^2 \\ (2\pi)^2 \end{bmatrix}, x.\wedge 2/3 = \begin{bmatrix} (-2\pi)^2/3 \\ (-2\pi + 0.1)^2/3 \\ \mathrm{M} \\ (2\pi - 0.1)^2/3 \\ (2\pi)^2/3 \end{bmatrix}, f = \sin(x.\wedge 2/3) = \begin{bmatrix} \sin\left[(-2\pi)^2/3\right] \\ \sin\left[(-2\pi + 0.1)^2/3\right] \\ \mathrm{M} \\ \sin\left[(2\pi - 0.1)^2/3\right] \\ \sin\left[(2\pi)^2/3\right] \end{bmatrix}$$

Γράφημα συνάρτησης τρόπος Β.

Εναλλακτικά, μπορούμε να ορίσουμε τη συνάρτηση με τη χρήση της **inline( )**. Η συνάρτηση θα πρέπει να δοθεί με μορφή κειμένου (μέσα σε εισαγωγικά).

Για **παράδειγμα** έστω ότι θέλουμε να κάνουμε το γράφημα της $g(x) = \dfrac{\cos\left(e^{x^2}\right)}{3}$ με τη μεθοδολογία της inline στο διάστημα $[-2,2]$. Αρχικά θα πρέπει να ορίσουμε τη συνάρτηση με τον ακόλουθο τρόπο:

```
>> clear all
>> g=inline(vectorize('cos(exp(x^2))/3'))
```

Δώστε **προσοχή** το πως υλοποιείται η e^{x^2} με την εντολή exp(x^2). Οποιαδήποτε άλλη έκφραση π.χ. **`e^(x^2)` θα ήταν λανθασμένη**. Το **`e`** για το περιβάλλον του MATLAB είναι απλά το όνομα μίας μεταβλητής και όχι η μαθηματική ποσότητα $e = 2.718\ldots$. Εάν δεν της έχουμε δώσει κάποια τιμή ο υπολογισμός δεν μπορεί να γίνει. Όπως θα δούμε και παρακάτω η μαθηματική ποσότητα e υπολογίζεται με την μαθηματική συνάρτηση υπολογισμού εκθετικών δυνάμεων **`exp(1)`.**

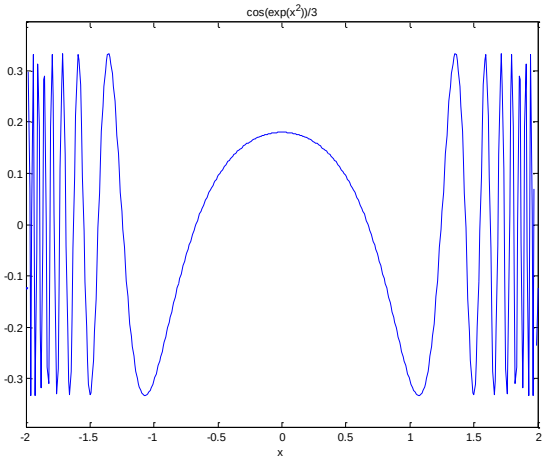
Η συνάρτηση **vectorize( )** μπορεί να μετατρέψει την μαθηματική έκφραση της συνάρτησης σε διανυσματική μορφή, ώστε να μπορεί να εφαρμοστεί σε διανύσματα. Εάν δεν επιθυμούμε να εφαρμοστεί η συνάρτηση σε διανύσματα αυτή δεν είναι απαραίτητο να τη χρησιμοποιήσουμε. Από τη στιγμή που ορίσαμε τη συνάρτηση με τη χρήση της **inline,** το περιβάλλον τη θεωρεί ως ένα μαθηματικό αντικείμενο το οποίο μπορεί να εφαρμοστεί σε αριθμούς ή και σε διανύσματα στην περίπτωση που έχει χρησιμοποιηθεί η **vectorize.** Για παράδειγμα:

```
>> g(1)
```
→ -0.3039

```
>> g([-1,0,1])
```
→ -0.3039 0.1801 -0.3039

Η **ezplot( )** με παραμέτρους το όνομα και το διάστημα σχεδίασης εμφανίζει το γράφημα μίας συνάρτησης η οποία έχει οριστεί με την **inline**. Η διαδικασία αυτή είναι αυτοματοποιημένη και το περιβάλλον αποφασίζει πόσο πυκνή διαμέριση σημείων θα χρησιμοποιήσει στο γράφημα. Για να εμφανιστεί το γράφημα πληκτρολογούμε τις ακόλουθες εντολές:

```
>> ezplot(g,[-2,2])
```



Γράφημα συνάρτησης τρόπος Γ.
Εναλλακτικά, μπορούμε να ορίσουμε τη συνάρτηση με τη χρήση function handles. Για παράδειγμα οι εντολές

```
>> clear all
>> x=-2*pi:0.1:2*pi;
>> f=@(x) cos(exp(x.^2))/3;
>> plot(x,f(x))
```

Θα επιστρέψουν το ίδιο γράφημα. Επειδή εφαρμόζουμε την συνάρτηση σε διάνυσμα θα πρέπει να οριστεί σε διανυσματική μορφή. Φυσικά μπορεί να κληθεί για μοναδική τιμή και να επιστρέψει την τιμή της π.χ.

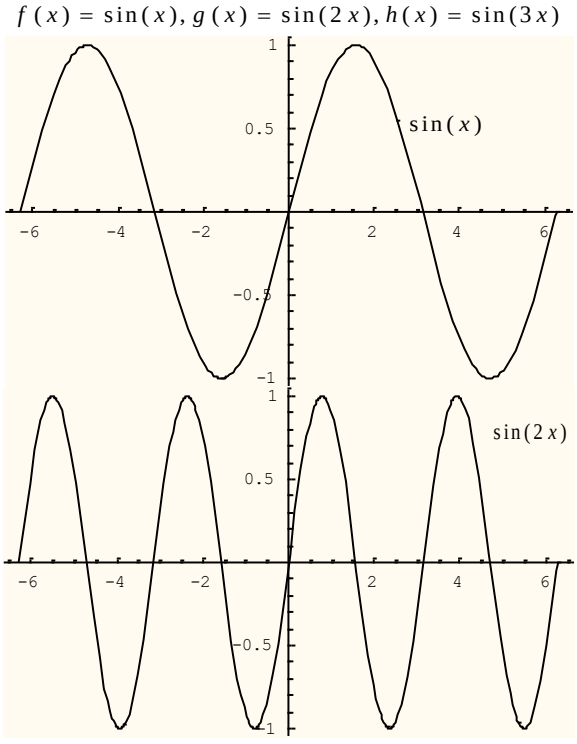
```
>> f(3)
```

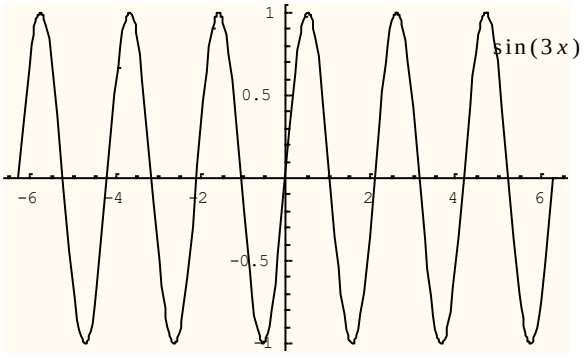
Εφαρμοσμένο παράδειγμα: Περιοδικές συναρτήσεις:

Μία συνάρτηση *f(x)* είναι **περιοδική** με **περίοδο** *T* όταν ισχύει *f(x+T)=f(x).* Η ελάχιστη δυνατή περίοδος λέγεται και θεμελιώδης περίοδος. Εμείς όταν λέμε περίοδο θα αναφερόμαστε σε αυτήν.

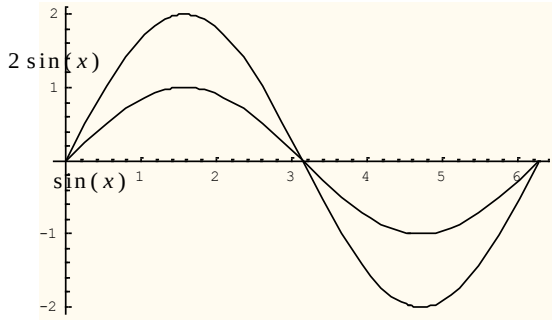
Μία συνηθισμένη περιοδική συνάρτηση είναι η *y=A sin(ωx)* όπου το *ω* ονομάζεται (γωνιακή ή κυκλική) **συχνότητα** και το Α είναι το **πλάτος**.

Στο παρακάτω σχήμα παρατηρούμε ότι για την συνάρτηση *sin(ωx)* όσο το *ω* μεγαλώνει τόσο μικραίνει η περίοδος της συνάρτησης η οποία ισούται με *T=2π/ω*.

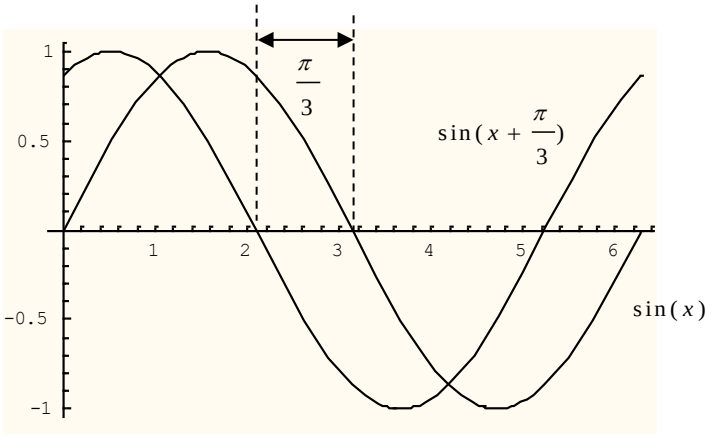




Επίσης για την συνάρτηση *y=A sin(x)* η οποία έχει πεδίο τιμών [*A,-A*] όσο το *A* (θετικό) μεγαλώνει τόσο το πλάτος της ταλάντωσης μεγαλώνει.



Η γραφική παράσταση της συνάρτησης $\sin(x+\theta)$ είναι η γραφική παράσταση της $\sin(x)$ μετατοπισμένη αριστερά κατά θ εάν το θ είναι θετικό, ενώ εάν το θ είναι αρνητικό μετατοπίζεται δεξιά κατά θ. Το θ ονομάζεται **φάση**.



Ανάλογη είναι και η συμπεριφορά της συνάρτησης συνημίτονο.

Ο **Jean Baptiste Fourier (1768-1830)** απόδειξε ότι κάθε περιοδική *y=f(χ)* συνάρτηση μπορεί να γραφεί ως ένα άπειρο άθροισμα ημιτονοειδών συναρτήσεων της μορφής:

$$f(x) = A_0 + A_1 \sin(\omega x + \phi_1) + A_2 \sin(2\omega x + \phi_2) + ... + A_n \sin(n\omega x + \phi_n) + ...$$

$$= A_0 + \sum_{n=1}^{\infty} A_n \sin(n\omega x + \phi_n)$$

και τελικά να προσεγγιστεί από ένα πεπερασμένο

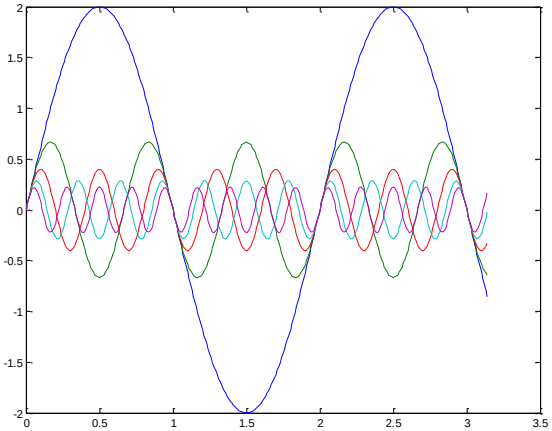
$$f(x) \approx A_0 + A_1 \sin(\omega x + \phi_1) + A_2 \sin(2\omega x + \phi_2) + ... + A_n \sin(n\omega x + \phi_n)$$

Αυτή είναι η βάση της θεωρίας των σειρών Fourier. Ο όρος $A_1 \sin(\omega x + \phi_1)$ ονομάζεται πρώτη αρμονική, ο $A_2 \sin(2\omega x + \phi_2)$ δεύτερη αρμονική κ.λ.π.

Παράδειγμα: Έστω $y = 2\sin(\pi x) + \frac{2}{3}\sin(3\pi x) + \frac{2}{5}\sin(5\pi x) + \frac{2}{7}\sin(7\pi x) + \frac{2}{9}\sin(9\pi x)$

Για να δούμε τις πέντε αρμονικές ξεχωριστά στο MATLAB εκτελούμε τις εντολές

```
>> clear all
>> x=0:0.01:pi;
>> y1=2*sin(pi*x);
>> y2=2/3*sin(3*pi*x);
>> y3=2/5*sin(5*pi*x);
>> y4=2/7*sin(7*pi*x);
>> y5=2/9*sin(9*pi*x);
>> plot(x,y1,x,y2,x,y3,x,y4,x,y5)
```



Με τη συγκεκριμένη εκτέλεση της plot εμφανίζουμε στο ίδιο γράφημα τη γραφική παράσταση πέντε ζευγών διανυσμάτων (των πέντε αρμονικών). Για κάθε ζευγάρι διανυσμάτων, το πρώτο διάνυσμα (εδώ και στα τέσσερα ζεύγη το x) περιέχει τις τετμημένες των σημείων του γραφήματος και το δεύτερο (για την πρώτη αρμονική το y1, για τη δεύτερη το y2, … , για την πέμπτη το y5) περιέχει τις τεταγμένες.

Αν θεωρήσουμε τους δύο πρώτους όρους του αθροίσματος και κάνουμε το αντίστοιχο γράφημα:

```
>> plot(x,y1+y2)
```

Αν θεωρήσουμε τους τρεις πρώτους όρους του αθροίσματος και κάνουμε το αντίστοιχο γράφημα:

```
>> plot(x,y1+y2+y3)
```

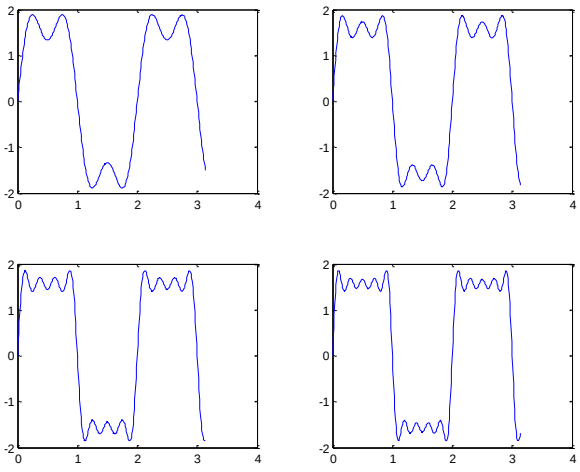
Αν θεωρήσουμε τους τέσσερεις πρώτους όρους του αθροίσματος και κάνουμε το αντίστοιχο γράφημα:

```
>> plot(x,y1+y2+y3+y4)
```

Τέλος, αν θεωρήσουμε και τους πέντε όρους του αθροίσματος και κάνουμε το αντίστοιχο γράφημα:

```
>> plot(x,y1+y2+y3+y4+y5)
```

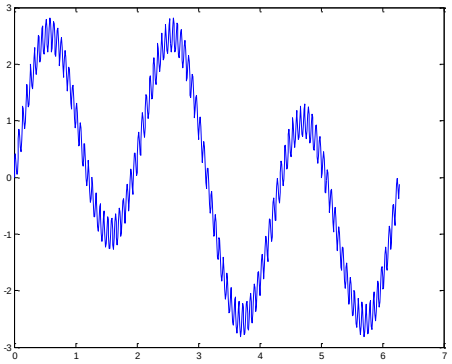
Αν παρατηρήσουμε τα τέσσερα γραφήματα:



Είναι φανερό ότι προσθέτοντας και άλλους παρόμοιους όρους το γράφημα μοιάζει όλο και περισσότερο με μία συγκεκριμένη περιοδική συνάρτηση τετραγωνικού παλμού. Προσθέτοντας και άλλες αρμονικές

τέτοια αθροίσματα μπορούν να χρησιμοποιηθούν για να προσεγγίσουν με ικανοποιητική ακρίβεια τέτοιου είδους συναρτήσεις.

Επίσης, όταν $y = \sin(x) + 2\sin(3x) + 0.3\sin(100x)$



Πληκτρολογώντας τις εντολές

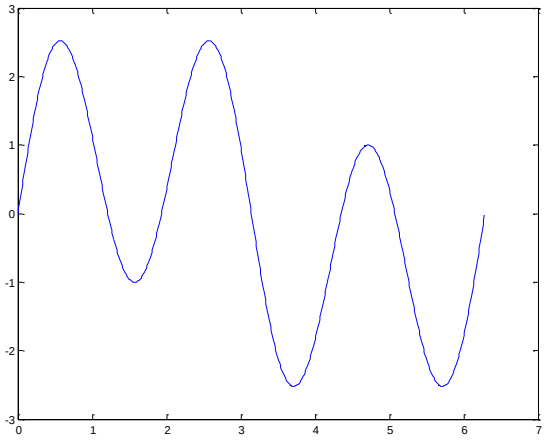
```
>> clear all
>> x=0:0.01:2*pi;
>> y1=sin(x)+2*sin(3*x)+0.3*sin(100*x);
>> plot(x,y1)
```

βλέπουμε το γράφημα της παραπάνω συνάρτησης στο διάστημα [0,2π].

Αποκόπτοντας το τελευταίο όρο, δηλαδή όταν θεωρήσουμε ότι $y = \sin(x) + 2\sin(3x)$, όταν πληκτρολογήσουμε

```
>> y2=sin(x)+2*sin(3*x);
>> plot(x,y2)
```

παρατηρούμε ότι η αποκοπή αυτού του όρου μοιάζει να λειτούργησε ως φίλτρο το οποίο καθάρισε τη συνάρτηση από το «θόρυβο».



Τα αναπτύγματα (περιοδικών κυρίως) συναρτήσεων σε τριγωνομετρικά αθροίσματα βρίσκουν πολλές εφαρμογές. Για παράδειγμα, μπορούν να χρησιμοποιηθούν για το φιλτράρισμα θορύβου στην ανάλυση σημάτων. Επίσης στις τηλεπικοινωνίες βρίσκουν εφαρμογή στη μεταφορά του συνεχούς σήματος της φωνής μέσω δορυφόρου από ένα σημείο του πλανήτη σε ένα άλλο. Η μεταφορά του συνεχούς σήματος της φωνής, αφού ψηφιοποιηθεί, και η αποστολή του bit προς bit απαιτεί την επεξεργασία και μεταφορά μεγάλου όγκου δεδομένων. Εάν το σήμα αναπτυχθεί σε ένα τριγωνομετρικό άθροισμα, αρκεί να μεταφερθούν μόνο οι συντελεστές (φάσεις, συχνότητες και πλάτη) και στον προορισμό να εφαρμοστεί ο κατάλληλος τύπος ώστε να αναπαραχθεί το σήμα. Μία τέτοια διαδικασία είναι πολύ πιο οικονομική.

Περισσότερες Μαθηματικές Συναρτήσεις στο MATLAB

Οι κυριότερες μαθηματικές συναρτήσεις παρατίθενται στον πίνακα που ακολουθεί. Όπως έχουμε ήδη αναφέρει, για τις τριγωνομετρικές συναρτήσεις οι γωνίες θεωρούνται σε ακτίνια. Τα ορίσματα μπορεί να είναι πραγματικοί ή μιγαδικοί (όπου ορίζονται) αριθμοί, διανύσματα ή πίνακες (για τα διανύσματα και τους πίνακες θα μιλήσουμε αναλυτικότερα σε επόμενα εργαστήρια). Η εφαρμογή των συναρτήσεων σε πίνακες ή διανύσματα έχουν ως αποτέλεσμα (με μικρές εξαιρέσεις) το να εφαρμόζεται η συνάρτηση σε κάθε στοιχείο του πίνακα.

Συνάρτηση	**Περιγραφή**
abs(x)	απόλυτη τιμή
sqrt(x)	τετραγωνική ρίζα
sin(x)	ημίτονο
cos(x)	συνημίτονο
tan(x)	εφαπτομένη
asin(x)	αντίστροφη τριγωνομετρική συνάρτηση ημιτόνου
acos(x)	αντίστροφη τριγωνομετρική συνάρτηση συνημίτονου
atan(x)	αντίστροφη τριγωνομετρική συνάρτηση εφαπτομένης
sinh(x)	υπερβολικό ημίτονο
cosh(x)	υπερβολικό συνημίτονο
tanh(x)	υπερβολική εφαπτομένη
asinh(x)	αντίστροφη τριγ. συνάρτηση υπερβολικού ημιτόνου
acosh(x)	αντίστροφη τριγ. συνάρτηση υπερβολικού
atanh(x)	αντίστροφη τριγ. συνάρτηση υπερβολικής
sign(x)	+1 αν x>0, -1 αν x<0
exp(x)	εκθετική συνάρτηση e^x
log(x)	λογάριθμος με βάση το e, δηλαδή $ln(x)$
log10(x)	λογάριθμος με βάση το 10, δηλαδή $log_{10}(x)$ ή $\log(x)$
ceil(x)	στρογγυλοποιεί στο κοντινότερο μεγαλύτερο ακέραιο
floor(x)	στρογγυλοποιεί στο κοντινότερο μικρότερο ακέραιο
round(x)	στρογγυλοποιεί στο κοντινότερο ακέραιο
fix(x)	στρογγυλοποιεί στο κοντινότερο, προς το 0, ακέραιο
factorial(x)	παραγοντικό $n! = 1 \cdot 2 \cdot \ldots \cdot (n-1) \cdot n$ και $0! = 1$

Με τις ακόλουθες εντολές υλοποιούμε απλά παραδείγματα εφαρμογής διάφορων μαθηματικών συναρτήσεων σε αριθμούς.

```
>> abs(-2)
>> sqrt(-2)
```

Στην τελευταία εντολή το αποτέλεσμα της $\sqrt{-2}$ είναι ο μιγαδικός $0 + i\sqrt{2}$.

```
>> sin(2*pi)
```

Παρατηρήστε ότι ενώ το παραπάνω αποτέλεσμα θα πρέπει να ήταν μηδέν το MATLAB το υπολογίζει με μία ακρίβεια 16 ψηφίων. Ενώ η επόμενη έκφραση έχει τιμή $\sqrt{3} \approx 1.7321$.

```
>> cos(4*pi)*tan(pi/3)
```

Για τα παρακάτω θυμίζουμε τους ορισμούς των αντίστροφων τριγωνομετρικών συναρτήσεων (τόξο ημιτόνου, συνημιτόνου κ.λ.π.). Ξέρουμε ότι ισχύει:

$$\sin(\pi/2) = 1 \Rightarrow \arcsin(1) = sin^{-1}(1) = \pi/2 \approx 1.5708$$

και ότι

$$\cos(0) = 1 \Rightarrow \arccos(1) = cos^{-1}(1) = 0 .$$

```
>> asin(1)
>> acos(1)
```

Θυμίζουμε τις ιδιότητες των λογαρίθμων $e^{ln(x)} = x$, $ln(e) = 1$, $10^{log_{10}(x)} = x$, $\log(x^p) = p \cdot log(x)$ και τις εφαρμόζουμε στις παρακάτω εντολές.

```
>> exp(log(3))
```
→ 3

```
>> log10(10^5)
```
→ 5

Επίσης ορίζονται και οι συναρτήσεις υπερβολικού ημιτόνου και συνημιτόνου ως

$$\sinh(x) = \frac{e^x - e^{-x}}{2} \quad \text{και} \quad \cosh(x) = \frac{e^x + e^{-x}}{2}$$

οπότε $\cosh(0) = \frac{e^0 + e^{-0}}{2} = 1$.

```
>> cosh(0)
```
→ 1

Για τις παρακάτω δείτε τον ορισμό τους στον παραπάνω πίνακα, και κατανοείστε τα αποτελέσματα.

```
>> ceil(2.1)
```
→ 3

```
>> floor(-2.1)
```
→ -3

```
>> fix(-2.6)
```
→ -2

```
>> factorial(3)
```
→ 6

```
>> factorial(0)
```
→ 1

Άσκηση για εξάσκηση:

Καθαρίστε τη μνήμη από τις όποιες τιμές μεταβλητών υπάρχουν ενεργές. Σε ένα διάνυσμα με όνομα x δημιουργείστε μία διαμέριση του διαστήματος $[-\pi, \pi]$ με πλάτος 0.01. Ορίστε στη συνέχεια διανύσματα τα οποία να περιέχουν τις τιμές των συναρτήσεων

$$f1 = \frac{\sin(\sqrt{2}x + \frac{\pi}{3})}{x + 10} \quad \text{και} \quad f2 = \cos(5x + \frac{\pi}{10})$$

για τις τιμές του x. Τα αποτελέσματα των παραπάνω εντολών δεν πρέπει να εμφανίζονται. Δημιουργήστε τα ακόλουθα γραφήματα το ένα μετά το άλλο σε διαφορετικά παράθυρα:

A) Τη γραφική παράσταση της $f1(x)$

B) Τη γραφική παράσταση της $f2(x)$

C) Τη γραφική παράσταση της $f1(x) + f2(x)$

D) Τη γραφική παράσταση των $f1(x), f2(x)$ σε ένα γράφημα.

Για να το καταφέρετε αυτό θα πρέπει να εκτελείτε πριν από κάθε εντολή του γραφήματος την εντολή **`>>figure`**

Γράψτε τις κατάλληλες εντολές και κάντε τα σκαριφήματα των γραφημάτων που ζητούνται στο ακόλουθο κενό:

Διανύσματα και πίνακες

Έχουμε ήδη αναφέρει ότι στο MATLAB όλες οι ποσότητες θεωρούνται πίνακες (ακόμη και οι αριθμοί θεωρούνται 1x1 πίνακες) και αποθηκεύονται κατά στήλες στη μνήμη του υπολογιστή. Αυτό επηρεάζει ακόμη και τις τεχνικές προγραμματισμού στο συγκεκριμένο περιβάλλον. Για αυτό το λόγο θα πρέπει να ασχοληθούμε πιο αναλυτικά με τα διανύσματα και τους πίνακες. Στα προηγούμενα μαθήματα έχουμε γνωρίσει το πως ορίζουμε διανύσματα γραμμή και διανύσματα στήλη. Αυτό μπορεί να γίνει είτε παραθέτοντας όλα τα στοιχεία του διανύσματος είτε δίνοντας το πρώτο στοιχείο ενός διανύσματος, το βήμα μεταβολής και στην συνέχεια τη μεγαλύτερη τιμή που μπορεί να έχει το τελευταίο στοιχείο. Με αυτήν την τεχνική ορίζουμε τα ακόλουθα διανύσματα γραμμή.

$$c = [1\ ,2\ ,3\ ,4\ ,5\ ], d = [1\ ,1.2,1.4,1.6]$$

```
>> c=1:5
```

```
>> d=1:0.2:1.7
```

Μπορούμε επίσης να ορίσουμε διανύσματα γραμμή συνθέτοντας τα από άλλα διανύσματα γραμμή. Για παράδειγμα η εκτέλεση της εντολής :

```
>> e=[c d]
```

μας δημιουργεί το ακόλουθο διάνυσμα:

$e = [1,2,3,4,5,1,1.2,1.4,1.6]$.

Έχουμε πει ότι τα διανύσματα στήλη ορίζονται παρόμοια χωρίζοντας τα στοιχεία τους με ; ή εισάγοντας τα στοιχεία της κάθε γραμμής και πατώντας Enter ή ως ανάστροφα διανυσμάτων γραμμή με τη χρήση του τελεστή αναστροφής '. Θυμηθείτε ο τελεστής αναστροφής επιστρέφει το ανάστροφο του διανύσματος μετατρέποντας διανύσματα γραμμή σε διανύσματα στήλες και το αντίστροφο. Όσο αφορά τον ορισμό διανυσμάτων στήλη ως σύνθεση άλλων διανυσμάτων στήλη αυτός μπορεί να γίνει με παρόμοιο (με την περίπτωση διανυσμάτων γραμμής) τρόπο. Έτσι πληκτρολογώντας:

```
>> f=[c' ; d']
```

δημιουργούμε το διάνυσμα στήλη $f = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 1 \\ 1.2 \\ 1.4 \\ 1.6 \end{bmatrix}$.

Μπορούμε να αναφερθούμε στο στοιχείο ενός διανύσματος χρησιμοποιώντας τον δείκτη της θέσης του π.χ. για το δεύτερο στοιχείο του διανύσματος e πληκτρολογούμε:

```
>> e(2)
```

αλλά και να το θέσουμε ίσο με μία νέα τιμή, αλλάζοντας έτσι τις τιμές στοιχείων του διανύσματος. Δηλαδή με την εκτέλεση της εντολής

```
>> e(2)=10
```

το διάνυσμα θα γίνει τώρα $e = [1,10,3,4,5,1,1.2,1.4,1.6]$.

Επίσης, μπορούμε να ορίσουμε κενά διανύσματα (δηλαδή διανύσματα με κανένα στοιχείο) με τον ακόλουθο τρόπο:

```
>> g=[ ]
```

Μπορούμε να προσθέσουμε στοιχεία σε κάποιο διάνυσμα γραμμή και με τον ακόλουθο τρόπο:

```
>> g=[g,3]
>> g=[g,2]
```

g = [3]

g = [3 2]

Εάν αντί για κόμμα χρησιμοποιούσαμε ; θα προσθέταμε στοιχεία δημιουργώντας ένα διάνυσμα στήλη.

```
>> g=[ ]
>> g=[g;3]
>> g=[g;2]
```

g = [3
2]

Εάν προσπαθήσουμε να αναφερθούμε σε στοιχείο διανύσματος **πέρα από τα όρια του**, τότε επιστρέφεται κατάλληλο μήνυμα λάθους (που διαφέρει ως προς την έκφρασή του από έκδοση σε έκδοση του MATLAB), για παράδειγμα:

```
>> g(5)
```

??? Attempted to access g(5); index out of bounds because numel(g)=2.

Ωστόσο, εάν εκχωρήσουμε τιμή σε θέση πέρα από τα όρια του διανύσματος τότε το διάνυσμα μεγαλώνει και τα ενδιάμεσα στοιχεία είναι μηδενικά. Το συγκεκριμένο διάνυσμα έχει εννέα ψηφία, εάν εκτελέσουμε την εντολή

```
>> e(12)=-2
```

με την οποία ορίζουμε τιμή του δωδέκατου στοιχείου του διανύσματος το διάνυσμα θα γίνει:

$$e = [1,10,3,4,5,1,1.2,1.4,1.6,0,0,-2].$$

Παρατηρούμε ότι οι διαστάσεις των διανυσμάτων (αλλά και των πινάκων, όπως θα δούμε παρακάτω) δεν είναι προκαθορισμένες αλλά μπορούν αν μεταβληθούν με δυναμικό τρόπο.

Η size χρησιμοποιείται για να μας επιστραφεί η διάσταση ενός διανύσματος ως πίνακας ενώ η length μας επιστρέφει τη μεγαλύτερη διάσταση (γραμμών ή στηλών). Στο παράδειγμα που ακολουθεί τα διανύσματα έχουν τρία στοιχεία το καθένα ενώ ως διάνυσμα το v είναι 3x1 ενώ το w είναι 1x3.

```
>> clear all
>> v=[1:2:5]'
>> w=[1;3;5]'
>> length(v)
>> size(v)
>> length(w)
>> size(w)
```

3

3 1

3

1 3

Οι πίνακες δύο διαστάσεων αποτελούν τη φυσική γενίκευση των διανυσμάτων στα οποία καμία από τις δύο διαστάσεις δεν είναι μονάδα. Όπως και στη γραμμική άλγεβρα ένας πίνακας με διάσταση m, n είναι μία διάταξη $n \times m$ πραγματικών (ή μιγαδικών αριθμών) σε m γραμμές και n στήλες.
Οι πίνακες στο MATLAB ορίζονται εισάγοντας τα στοιχεία τους διαχωρίζοντας τις γραμμές είτε πατώντας Enter είτε βάζοντας ερωτηματικό (;). Σε κάθε γραμμή παραθέτουμε τα στοιχεία χωρισμένα κενά ή κόμματα ή ορίζοντας την με το γνωστό τρόπο **α:κ:β** όπου η γραμμή έχει πρώτο στοιχείο το **α**, επόμενα στοιχεία τα **α+κ,α+2κ,…** και τελευταίο στοιχείο το **α+νκ** όπου **α+(ν+1)κ>β**. Οι διάφοροι τρόποι εφαρμόζονται στα ακόλουθα παραδείγματα. **Καλό είναι να χρησιμοποιούμε το κόμμα ανάμεσα στα στοιχεία κάθε γραμμής και το ερωτηματικό ως διαχωριστικό των γραμμών αντί για κενό και Enter αντίστοιχα.**

Πατήστε Enter και συνεχίζετε στην κάτω γραμμή

```
>> clear all
>> a=[5 7 9
     1 -3 -7]
>> b=[1 2 5 ; 9 0 5]
>> c=[0,1;3,-2;4,2]
>> d=[1:3,2:3;1:5;1:2:10]
```

```
a =
     5     7     9
     1    -3    -7
```

```
b =
     1     2     5
     9     0     5
```

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

```
d =
     1     2     3     2     3
     1     2     3     4     5
     1     3     5     7     9
```

Προσοχή όμως, οι γραμμές που ορίζουμε στους πίνακες θα πρέπει να έχουν όλες τον ίδιο αριθμό στηλών διαφορετικά, όπως βλέπουμε στο ακόλουθο παράδειγμα όπου η πρώτη γραμμή έχει δύο στοιχεία και η δεύτερη τρία, λαμβάνουμε μήνυμα λάθους.

```
>> e=[1,2;5,-6,0]
```

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

Όπως συμβαίνει και στα διανύσματα, εάν εκχωρήσουμε τιμή σε θέση πέρα από τα όρια ενός πίνακα τότε εάν ο πίνακας έχει ορισθεί αυτός μεγαλώνει και τα ενδιάμεσα στοιχεία του είναι μηδενικά. Στην περίπτωση που δεν έχει ορισθεί ο πίνακας τότε δημιουργείται άμεσα στη μνήμη και πάλι τα ενδιάμεσα στοιχεία του είναι μηδενικά.

```
>> e(3,3)=2
```

```
e =     0     0     0
        0     0     0
        0     0     2
```

Και για τους πίνακες, η **size** χρησιμοποιείται για να μας επιστραφεί η διάσταση του διανύσματος ως πίνακας και η **length** μας επιστρέφει τη μεγαλύτερη διάσταση (γραμμών ή στηλών).

```
>> size(d)
>> length(d)
```

Μπορούμε επίσης να εκχωρήσουμε τον αριθμό των στηλών και των γραμμών σε μεταβλητές με την εκτέλεση της εντολής **size** όπως στο ακόλουθο παράδειγμα:

```
>> [k,l]=size(d)
```

Ο τελεστής αναστροφής εφαρμόζεται φυσικά και στους πίνακες και επιστρέφει τον ανάστροφο ενός πίνακα. Θυμίζουμε ότι, στη Γραμμική Άλγεβρα, ο **ανάστροφος** $\mathbf{A}^T$ (transpose) ενός πίνακα προκύπτει εάν στον πίνακα αλλάξουμε τις γραμμές σε στήλες και τις στήλες σε γραμμές.

```
>> b'
```

Μπορούμε να κάνουμε αναφορά σε στοιχεία πινάκων αλλά και γραμμές, στήλες ή τμήματα τους. Με τη χρήση της άνω και κάτω τελείας : αναφερόμαστε σε ολόκληρες γραμμές ή στήλες. Για παράδειγμα η πρώτη εντολή μας επιστρέφει την πρώτη γραμμή του πίνακα a ενώ η δεύτερη τη δεύτερη στήλη. Η τρίτη εντολή επιστρέφει το τμήμα του πίνακα που ορίζεται από την δεύτερη ως την τρίτη γραμμή και από την δεύτερη έως την τέταρτη στήλη του πίνακα d.

```
>> a(1,:)
>> a(:,2)
```

```
a =
     5     7     9
     1    -3    -7
```

```
a =
     5     7     9
     1    -3    -7
```

```
>> d(2:3,2:4)
```

```
d =
     1     2     3     2     3
     1     2     3     4     5
     1     3     5     7     9
```

Έχουμε ορίσει στη θεωρία ότι ο μηδενικός πίνακας έχει όλα τα στοιχεία του μηδέν και ο μοναδιαίος πίνακας είναι ένας τετραγωνικός διαγώνιος πίνακας ο οποίος έχει μηδέν παντού εκτός από τη διαγώνιο στην οποία έχει μονάδες. Δηλαδή ο μοναδιαίος πίνακας έχει τη μορφή:

$$\begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}.$$

Στο MATLAB, οι **eye**( ), **zeros**( ) και **ones**( ) δημιουργούν μοναδιαίους, μηδενικούς και πίνακες με στοιχεία μονάδες αντίστοιχα. Οι μηδενικοί και οι μοναδιαίοι πίνακες είναι τετραγωνικοί πίνακες για αυτό η **eye**( ) και η **zeros**( ) στη κλίση τους έχουν ως όρισμα τον αριθμό των γραμμών (που είναι ο ίδιος με των στηλών). Στην **ones**( ), όπου δημιουργούνται σε γενικές γραμμές μη τετραγωνικοί πίνακες, χρειάζεται να βάλουμε ως όρισμα την πλήρη διάσταση. Για παράδειγμα ο πίνακας c παρακάτω είναι 3x2.

```
>> clear all
>> a=eye(3)
>> b=zeros(2)
>> c=ones(3,2)
```

Η εντολή **rand**( ) επιστρέφει πίνακα με (ψεύδο) τυχαία στοιχεία με τιμές από το 0 έως το 1. Ως όρισμα η συνάρτηση αυτή δέχεται τη διάσταση του πίνακα.

```
>>d =rand(2,3)
```

Τέλος, μπορούμε να συνθέσουμε πίνακες από άλλους πίνακες ή διανύσματα. Αυτό μπορεί να γίνει με το να δηλώσουμε ως στοιχεία πίνακα κατάλληλους πίνακες ή διανύσματα. Σε μία τέτοια περίπτωση λέμε ότι θεωρούμε τον πίνακα ως **block** πίνακα.

```
>> e=[b, d;c,a]
```

Στο παραπάνω παράδειγμα ορίζουμε ένα πίνακα χρησιμοποιώντας 4 block (πίνακες).

e =

	b		**d**	
	0	0	0.9501	0.6068	0.8913
	0	0	0.2311	0.4860	0.7621
c	1.0000	1.0000	1.0000	0	0
	1.0000	1.0000	0	1.0000	0
	1.0000	1.0000	0	0	1.0000

a

Όταν εκτελέσετε την εντολή είναι πιθανό τα στοιχεία του πίνακα d να είναι διαφορετικά από αυτά που φαίνονται παραπάνω. Αυτό συμβαίνει επειδή η τα στοιχεία αυτού του πίνακα είναι (ψεύδο) τυχαίοι αριθμοί και κάθε φορά που εκτελείται η εντολή rand( ) επιστρέφει άλλους αριθμούς.

Επίσης είπαμε στη θεωρία ότι ως p διαγώνιο ενός πίνακα A ορίζουμε τα στοιχεία του (εφόσον αυτά υπάρχουν) που ικανοποιούν $A_{i,i+p}$, ενώ ως διαγώνιο –p αυτά που ικανοποιούν $A_{i,i-p.}$ Για παράδειγμα στον ακόλουθο πίνακα:

$$\begin{bmatrix} 1 & -1 & 0 & 2 \\ 3 & -4 & 1 & 19 \\ 2 & 0 & -25 & -1 \\ 9 & -2 & 3 & 1 \end{bmatrix}$$

Η κύρια διαγώνιος είναι:

$$\begin{bmatrix} 1 & & & \\ & -4 & & \\ & & -25 & \\ & & & 1 \end{bmatrix}$$

Η 2-διαγώνιος είναι:

$$\begin{bmatrix} & 0 & \\ & & 19 \\ & & \\ & & \end{bmatrix}$$

Και η -1 διαγώνιος

$$\begin{bmatrix} & & \\ 3 & & \\ & 0 & \\ & & 3 \end{bmatrix}$$

Μπορούμε επίσης να δημιουργήσουμε πίνακα ορίζοντας τις τιμές μίας από τις διαγώνιους του (κύριας ή άλλης) με τη συνάρτηση **diag( )**. Ως πρώτο όρισμα η συνάρτηση δέχεται τα στοιχεία της διαγωνίου σε ένα διάνυσμα και ως δεύτερο τον αριθμό που χαρακτηρίζει τη διαγώνιο. Για την κύρια διαγώνιο δεν βάζουμε όρισμα. Για παράδειγμα η εντολή

```
>> diag([2,3,4])
```

δημιουργεί τον πίνακα

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

ενώ η εντολή:

```
>> diag([2,3,4],1)
```

τον πίνακα:

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

Ως παράδειγμα θα ορίσουμε ως άθροισμα πινάκων με κατάλληλες διαγώνιους τον ακόλουθο πίνακα:

$$c = \begin{bmatrix} 3 & 0 & 1 & 0 & 0 \\ 6 & 10 & 0 & 1 & 0 \\ 0 & -2 & 5 & 0 & -1 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & \sqrt{2} & -1 \end{bmatrix}$$

Θεωρώντας τον πάνω πίνακα ως άθροισμα

$$c = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & -1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sqrt{2} & 0 \end{bmatrix}$$

εκτελούμε:

```
>> c=diag([3,10,5,3,-1])+diag([1,1,-1],2)+diag([6,-2,0,sqrt(2)],-1)
```

Την πράξη του αθροίσματος πινάκων δεν την έχουμε ορίσει ακόμη. Θα την συναντήσουμε παρακάτω. Απλά αναφέρουμε ότι για να προσθέσουμε πίνακες αυτοί θα πρέπει να έχουν την ίδια διάσταση και το αποτέλεσμα είναι το ίδιο με την πρόσθεση πινάκων που γνωρίζουμε από την Γραμμική Άλγεβρα.

Άλλες βασικές συναρτήσεις διανυσμάτων και πινάκων

Έχοντας ως κύριο εργαλείο τα διανύσματα και τους πίνακες το MATLAB μας παρέχει μία μεγάλη γκάμα εντολών που μπορούν να εφαρμοστούν σε πίνακες. Για τα διανύσματα η **sum( )** επιστρέφει το άθροισμα και η **prod( )** το γινόμενο των στοιχείων ενός διανύσματος. Η εντολή **mean()** επιστρέφει τη μέση τιμή των στοιχείων του διανύσματος. Εδώ εκχωρούμε το άθροισμα και το γινόμενο των στοιχείων του διανύσματος x ως δύο στοιχεία ενός διανύσματος γραμμή.

```
>>clear all
>>x=[ 2, 3, -4, 5, 6];
>>s=[sum(x) , prod(x)]
>> mean(x)
```

Το άθροισμα 12

Το γινόμενο -720

Η **sort( )** επιστρέφει ένα διάνυσμα με ταξινομημένα τα στοιχεία διανύσματος κατά αύξουσα διάταξη. Εάν θέλουμε να ταξινομήσουμε κατά φθίνουσα σειρά τότε ταξινομούμε το αντίθετο του διανύσματος και αλλάζουμε πάλι τα πρόσημα.

```
>> sort(x)
>> -sort(-x)
```

Η **max( )** και η **min( )** επιστρέφουν το μεγαλύτερο και μικρότερο στοιχείο ενός διανύσματος αντίστοιχα.

```
>> max(x)
```

Μπορούμε να εκχωρήσουμε τόσο την τιμή του μεγαλύτερου (ή του μικρότερου) στοιχείου ενός διανύσματος όσο και τη θέση του στο διάνυσμα σε μεταβλητές με την εκτέλεση της αντίστοιχης εντολής όπως στο ακόλουθο παράδειγμα. Στην πρώτη μεταβλητή επιστρέφει την τιμή και στη δεύτερη τη θέση.

```
>> [m,k]=min(x)
```

Το μικρότερο στοιχείο m = -4
Η θέση του k = 3

Σε περίπτωση που υπάρχουν περισσότερα από ένα στοιχεία με τη μεγαλύτερη ή τη μικρότερη τιμή το MATLAB επιστρέφει το πρώτο που βρίσκει.

Τέλος, η εντολή **diff( )** επιστρέφει ένα διάνυσμα με στοιχεία τις διαφορές των διαδοχικών στοιχείων του διανύσματος (επόμενο - προηγούμενο). Δηλαδή για το x που έχουμε ορίσει παραπάνω θα επιστρέψει το διάνυσμα [3-2, -4-3, 5-(-4), 6-5].

```
>> diff(x)
```

```
ans =  1   -7    9    1
```

Οι **max( ), min( ), sum( ), prod( ), mean( )** μπορούν να εφαρμοστούν και σε πίνακα. Σε αυτήν την περίπτωση επιστρέφουν ένα διάνυσμα του οποίου κάθε στοιχείο είναι το αποτέλεσμα της εφαρμογής τους σε καθεμία από τις στήλες του πίνακα:

$$B = \begin{bmatrix} 1 & 2 & 3 \\ 4 & -2 & 0 \\ 10 & 2 & 1 \end{bmatrix}$$

```
>> clear all
>> b=[1,2,3;4,-2,0;10,2,1]
>> min(b)
```

Η ακόλουθη εντολή επιστρέφει δύο διανύσματα όπου το πρώτο περιέχει τιμή του μεγαλύτερου στοιχείου κάθε στήλης και το δεύτερο την αντίστοιχη θέση του στη στήλη.

```
>> [m,k]=max(b)
>> sum(b)
>> prod(b)
>> mean(b)
```

Οπότε, για παράδειγμα, για να βρούμε το μέγιστο στοιχείο πίνακα πρέπει να εφαρμόσουμε τη συνάρτηση **max( )** δύο φορές, δηλαδή **max(max( ))**. Το ίδιο ισχύει και για το ελάχιστο αλλά και για το άθροισμα (ή το γινόμενο) όλων των στοιχείων πίνακα.

```
>>max(max(b))
```

Η **sort( )**, όταν την καλούμε με όρισμα πίνακα, επιστρέφει πίνακα με στήλες τις ταξινομημένες στήλες του αρχικού πίνακα.

```
>> sort(b)
```

Η **find( )** μας βρίσκει τις θέσεις των στοιχείων ενός πίνακα που ικανοποιούν μία λογική συνθήκη, εδώ είναι μικρότερα ή ίσα του μηδέν. Περισσότερα για το πώς ορίζουμε λογικές συνθήκες στο MATLAB θα δούμε όταν μιλήσουμε για τις δομές επιλογής.

Συγκεκριμένη εντολή μπορεί να εφαρμοστεί με δύο τρόπους. Κατά τον πρώτο τρόπο η εκτέλεση της εντολής

```
>> [k,l]=find(b<=0)
```

επιστρέφει δύο διανύσματα όπου το πρώτο περιέχει τον αριθμό της γραμμής και το δεύτερο τον αριθμό της στήλης κάθε στοιχείου που ικανοποιεί τη συνθήκη.

Ο δεύτερος τρόπος εκτέλεσης της εντολής εμπλέκει τον τρόπο που το περιβάλλον καταχωρεί στη μνήμη του υπολογιστή τα διανύσματα. Για παράδειγμα η εκτέλεση της

```
>> n=find(b<=0)
```

επιστρέφει τη θέση που έχει το στοιχείο του πίνακα που ικανοποιεί τη συνθήκη όταν θεωρήσουμε τον πίνακα ως διάνυσμα με στοιχεία τις στήλες του πίνακα. **Έτσι άλλωστε φυλάσσει το MATLAB τους πίνακες στην μνήμη του υπολογιστή. Όλοι οι πίνακες αποθηκεύονται στη μνήμη ως διανύσματα με την κάθε στήλη να ακολουθεί την προηγούμενή της.**

Ο πίνακας του παραδείγματός μας είναι ο

$$b = \begin{bmatrix} 1 & 2 & 3 \\ 4 & -2 & 0 \\ 10 & 2 & 1 \end{bmatrix}.$$

Στο MATLAB ο b αποθηκεύεται στη μνήμη του Η/Υ ως [1,4,10,2,-2,2,3,0,1]. Τα στοιχεία που ικανοποιούν τη συνθήκη είναι το -2 στη θέση (2,2) και το 0 στη θέση (2,3). Η εντολή όμως επιστρέφει στο διάνυσμα n τις θέσεις με τις οποίες έχει αποθηκεύσει τον πίνακα στη μνήμη. Με βάση την αποθήκευση του πίνακα στη μνήμη οι θέσεις των στοιχείων είναι η $5^{η}$ και η $8^{η}$. Μπορούμε να προσπελάσουμε τα στοιχεία ενός πίνακα είτε με τον κλασσικό τρόπο, αναφέροντας αριθμό γραμμής και στήλης, είτε με τον τρόπο που αναφέραμε στην παράγραφο αυτή.

```
>> b(n)
```

Άσκηση

Δημιουργείστε έναν **block** πίνακα ο οποίος συντίθεται από τα ακόλουθα blocks (πίνακες): Στην πρώτη γραμμή των blocks θα έχει, έναν μοναδιαίο πίνακα 5x5, έναν τριδιαγώνιο πίνακα 5x5 με στοιχεία της 1 διαγωνίου το -2, στοιχεία της κύριας διαγωνίου το 1 και στοιχεία της -1 διαγωνίου τους αριθμούς από το ένα έως το 5. Στην δεύτερη γραμμή των blocks θα έχει, έναν πίνακα 3x10 με στοιχεία τυχαίους αριθμούς. Στη συνέχεια βρείτε το άθροισμα των στοιχείων κάθε στήλης και το άθροισμα των στοιχείων κάθε γραμμής του πίνακα. Βρείτε το μέγιστο των μέσων όρων των στοιχείων κάθε στήλης. Βρείτε τις θέσεις στον πίνακα των στοιχείων που είναι μεγαλύτερα από το 0.4 και στη συνέχεια, χρησιμοποιώντας το πώς αποθηκεύει τα στοιχεία του πίνακα στην μνήμη το MATLAB, εμφανίστε τα στοιχεία αυτά.

Πράξεις πινάκων και διανυσμάτων

Ως πράξεις πινάκων ορίζονται οι γνωστές, από την γραμμική άλγεβρα, πρόσθεση, αφαίρεση πολλαπλασιασμός και ύψωση σε δύναμη φυσικού αριθμό, εφόσον η πράξη είναι δυνατό να γίνει.

Για την **πρόσθεση και την αφαίρεση πινάκων** ισχύει

$$C = A \pm B \Leftrightarrow c_{ij} = a_{ij} \pm b_{ij} \quad \forall \ 1 \le i \le m, \ 1 \le j \le n$$

Ορίζεται το **γινόμενο (πραγματικού ή μιγαδικού) αριθμού επί πίνακα** ως ένας πίνακας που έχει ως στοιχεία το γινόμενο του αριθμού επί το στοιχείο του πίνακα σε κάθε θέση.

$$kA = \left[k a_{ij} \right]$$

Επίσης για να ορίζεται το γινόμενο δύο πινάκων **A και B** πρέπει **ο αριθμός των στηλών του πίνακα A θα πρέπει να είναι ίσος με τον αριθμό των γραμμών του πίνακα B.** Όταν πολλαπλασιάζουμε πίνακα $n \times m$ με έναν πίνακα $m \times k$ τότε το αποτέλεσμα είναι ένα πίνακας $n \times k$.

Η πιο απλή περίπτωση είναι να πολλαπλασιάσουμε ένα διάνυσμα γραμμή με n στοιχεία, το οποίο μπορεί να θεωρηθεί ως ένας πίνακας με μέγεθος με $1 \times n$, με ένα διάνυσμα στήλη με n στοιχεία, το οποίο μπορεί να θεωρηθεί ως ένας πίνακας με μέγεθος με $n \times 1$. Ο πολλαπλασιασμός μπορεί να γίνει διότι ο αριθμός των στηλών του διανύσματος γραμμή (n) είναι ίσος με τον αριθμό των γραμμών του διανύσματος στήλη (n) και σύμφωνα με όσα είπαμε παραπάνω το αποτέλεσμα θα είναι ένας πίνακας 1×1 .

Η πράξη που γίνεται σε μία τέτοια περίπτωση είναι η ακόλουθη:

$$\begin{bmatrix} a_1 & a_2 & \dots & a_{n-1} & a_n \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \text{M} \\ b_{n-1} \\ b_n \end{bmatrix} = \begin{bmatrix} a_1 b_1 + a_2 b_2 + \dots + a_{n-1} b_{n-1} + a_n b_n \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} a_i b_i \end{bmatrix}$$

Για παράδειμα:

```
> >clear all;v=[1,2,3,4],w=[4;5;6;7]
>> v*w
```

ans =1x4+2x5+3x6+4x7=60, **1x4 * 4x1 επιστρέφει πίνακα 1x1**

Επίσης, όταν πολλαπλασιάσουμε ένα διάνυσμα στήλη με n στοιχεία, το οποίο μπορεί να θεωρηθεί ως ένας πίνακας με μέγεθος με $n \times 1$, με ένα διάνυσμα γραμμή με n στοιχεία, το οποίο μπορεί να θεωρηθεί ως ένας πίνακας με μέγεθος με $1 \times n$, ο πολλαπλασιασμός μπορεί να γίνει διότι ο αριθμός των στηλών του διανύσματος γραμμή (1) είναι ίσος με τον αριθμό των γραμμών του διανύσματος στήλη (1) . Σύμφωνα με όσα είπαμε παραπάνω το αποτέλεσμα θα είναι ένας πίνακας $n \times n$.

Η πράξη που γίνεται σε μία τέτοια περίπτωση είναι η ακόλουθη:

$$\begin{bmatrix} a_1 \\ a_2 \\ \text{M} \\ a_{n-1} \\ a_n \end{bmatrix} \begin{bmatrix} b_1 & b_2 & \dots & b_{n-1} & b_n \end{bmatrix} = \begin{bmatrix} a_1 b_1 & a_1 b_2 & \text{L} & a_1 b_{n-1} & a_1 b_n \\ a_2 b_1 & a_2 b_2 & \text{L} & a_2 b_{n-1} & a_2 b_n \\ \text{M} & \text{M} & \text{M} & \text{M} & \text{M} \\ a_{n-1} b_1 & a_{n-1} b_2 & \text{L} & a_{n-1} b_{n-1} & a_{n-1} b_n \\ a_n b_1 & a_n b_2 & \text{L} & a_n b_{n-1} & a_n b_n \end{bmatrix}$$

Για τα διανύσματα του παραπάνω παραδείγματος έχουμε:

```
>> w*v
```

```
[4       [1 2 3 4]   Θα εκτελεστούν                ή όταν γίνουν οι πράξεις
 5  επί              4x1  4x2  4x3  4x4                4   8   12  16
 6                   5 x1 5x2  5x3  5x4                5   10  15  20
 7]                  6x1  6x2  6x3  6x4                6   12  18  24
                     7x1  7x2  7x3  7x4                7   14  21  28
```

Στη γενική περίπτωση πολλαπλασιασμού πίνακα $n \times m$ με έναν πίνακα $m \times k$, το (i, j)- στοιχείο του γινομένου πινάκων προκύπτει από το γινόμενο της i -γραμμής του πίνακα Α επί της j -στήλης του πίνακα Β, όταν το θεωρούμε ως γινόμενο διανύσματος γραμμή επί διάνυσμα στήλη. Δηλαδή, στη γενική περίπτωση το αποτέλεσμα θα είναι ένας πίνακας $n \times k$:

$$C = AB \Rightarrow [c_{ij}] = \left[\sum_{p=1}^{n} a_{ip} b_{pj} \right]$$

Του οποίου το (i, j)- στοιχείο θα είναι το ακόλουθο:

$$c_{ij} = \begin{bmatrix} a_{i1} & a_{i2} & \dots & a_{i,n-1} & a_{i,n} \end{bmatrix} \begin{bmatrix} b_{1j} \\ b_{2j} \\ \text{M} \\ b_{n-1,j} \\ b_{n,j} \end{bmatrix} = a_{i1}b_{1j} + a_{i2}b_{2j} + \dots + a_{i,n-1}b_{n-1,j} + a_{i,n}b_{n,j} = \sum_{k=1}^{n} a_{ik}b_{kj}$$

Στο ακόλουθο παράδειγμα θα ορίσουμε τους πίνακες

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

και από τις $A + A^T$, AB, BA, $B - 2AB$, AA^T, CB, BC, $A + B$ παραστάσεις θα υπολογίσουμε όσες έχουν νόημα.

$$A + A^T = \begin{bmatrix} 1 & -1 \\ 0 & 2 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ -1 & 2 \end{bmatrix} = \begin{bmatrix} 2 & -1 \\ -1 & 4 \end{bmatrix}$$

$$AB = \begin{bmatrix} 1 & -1 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 1 & 3 & -2 \\ 0 & 0 & 3 \end{bmatrix} = \begin{bmatrix} 1\times 1 + (-1)\times 0 & 1\times 3 + (-1)\times 0 & -2\times 1 + 3\times(-1) \\ 0\times 1 + 2\times 0 & 3\times 0 + 2\times 0 & -2\times 0 + 2\times 3 \end{bmatrix} = \begin{bmatrix} 1 & 3 & -5 \\ 0 & 0 & 6 \end{bmatrix}$$

$$B - 2AB = \begin{bmatrix} 1 & 3 & -2 \\ 0 & 0 & 3 \end{bmatrix} - 2\begin{bmatrix} 1 & -1 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 1 & 3 & -2 \\ 0 & 0 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 3 & -2 \\ 0 & 0 & 3 \end{bmatrix} - \begin{bmatrix} 2 & 6 & -10 \\ 0 & 0 & 12 \end{bmatrix} = \begin{bmatrix} -1 & -3 & 8 \\ 0 & 0 & -9 \end{bmatrix}$$

$$AA^T = \begin{bmatrix} 1 & -1 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & 2 \end{bmatrix} = \begin{bmatrix} 1\times 1 + (-1)\times(-1) & 1\times 0 + (-1)\times 2 \\ 1\times 0 + (-1)\times 2 & 0\times 0 + 2\times 2 \end{bmatrix} = \begin{bmatrix} 2 & -2 \\ -2 & 4 \end{bmatrix}$$

$$BC = \begin{bmatrix} 1 & 3 & -2 \\ 0 & 0 & 3 \end{bmatrix} \begin{bmatrix} -1 \\ 2 \\ 4 \end{bmatrix} = \begin{bmatrix} 1\times(-1) + 3\times 2 - 2\times 4 \\ -1\times 0 + 2\times 0 + 3\times 4 \end{bmatrix} = \begin{bmatrix} -3 \\ 12 \end{bmatrix}$$

Οι υπόλοιπες παραστάσεις δεν έχουν νόημα. Για παράδειγμα, το πλήθος των στηλών του B δεν είναι ίσο με το πλήθος των γραμμών του A και επομένως δεν ορίζεται το γινόμενο BA. Για τον ίδιο λόγο δεν ορίζεται το γινόμενο CB . Το άθροισμα $A+B$ δεν ορίζεται γιατί οι πίνακες A, B είναι διαφορετικού μεγέθους. Όπου το η πράξη δεν ορίζεται από το MATLAB επιστρέφεται μήνυμα λάθους όπου αναφέρεται

ότι «…**matrix dimensions must agree**», δηλαδή ότι οι διαστάσεις των πινάκων πρέπει να συμφωνούν, ή κάτι ανάλογο.

```
>>clear all; a=[1,-1;0,2];b=[1,3,-2;0,0,3];c=[-1;2;4];d=[1,-1;3,2];
>> a+a'
>> a*b
>> b*a
>> b-a*b
>> a*a'
>> c*b
>> b*c
>> a+b
```

Επίσης, ορίζεται η **κ-δύναμη τετραγωνικού πίνακα** ως $A^k = \underbrace{A \cdots A}_{k \text{ φορές}}$ και $A^0 = I$.

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

```
>> a^0
>> a^3
```

Για τους διαγώνιους πίνακες, δηλαδή πίνακες με μη μηδενικά στοιχεία μόνο στην κύρια διαγώνιο, ισχύει ότι η κ-δύναμη ενός διαγώνιου πίνακα είναι πίνακας με διαγώνια στοιχεία τις κ- δυνάμεις του αρχικού πίνακα.

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & -1 \end{bmatrix}^3 = \begin{bmatrix} 2^3 & 0 & 0 \\ 0 & 3^3 & 0 \\ 0 & 0 & (-1)^3 \end{bmatrix} = \begin{bmatrix} 8 & 0 & 0 \\ 0 & 27 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

```
>> diag([2,3,-1])^3
```

Τέλος, όπως και στα διανύσματα, και για τους πίνακες ορίζονται οι στοιχείο προς στοιχείο πράξεις πολλαπλασιασμού .*, διαίρεσης **./** και ύψωσης σε δύναμη **.^**. Ισχύουν δε ακριβώς όσα είχαμε αναφέρει παραπάνω, ότι δηλαδή η ύψωση πίνακα με τον τελεστή **.^** σε αριθμό υψώνει κάθε στοιχείο του πίνακα στον αριθμό και ότι οι παραπάνω πράξεις ορίζονται για πίνακες που έχουν την ίδια διάσταση. Για παράδειγμα οι εντολές

```
>> a.^3
>> 3.^ a
>> a.*d
>> a./d
>> a.^d
```

Εκτελούν τις ακόλουθες πράξεις:

$$a.\wedge 3 = \begin{bmatrix} 1^3 & (-1)^3 \\ 0 & 2^3 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 0 & 8 \end{bmatrix},$$

$$3.\hat{}\,a = \begin{bmatrix} 3^1 & 3^{(-1)} \\ 3^0 & 3^2 \end{bmatrix} = \begin{bmatrix} 3 & \frac{1}{3} \\ 1 & 9 \end{bmatrix},$$

$$a.*d = \begin{bmatrix} 1 & -1 \\ 0 & 2 \end{bmatrix}.* \begin{bmatrix} 1 & -1 \\ 3 & 2 \end{bmatrix} = \begin{bmatrix} 1\times 1 & (-1)\times(-1) \\ 0\times 3 & 2\times 2 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 4 \end{bmatrix},$$

$$a./d = \begin{bmatrix} 1 & -1 \\ 0 & 2 \end{bmatrix}./ \begin{bmatrix} 1 & -1 \\ 3 & 2 \end{bmatrix} = \begin{bmatrix} 1/1 & -1/(-1) \\ 0/3 & 2/2 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix},$$

$$a.\hat{}\,d = \begin{bmatrix} 1 & -1 \\ 0 & 2 \end{bmatrix}.\hat{} \begin{bmatrix} 1 & -1 \\ 3 & 2 \end{bmatrix} = \begin{bmatrix} 1^1 & (-1)^{-1} \\ 0^3 & 2^2 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 0 & 4 \end{bmatrix}$$

Ως πράξεις μεταξύ διανυσμάτων ορίζονται οι γνωστές πράξεις της πρόσθεσης και της αφαίρεσης, εφόσον τα διανύσματα είναι ιδίου τύπου.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \pm \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} x \pm x' \\ y \pm y' \\ z \pm z' \end{bmatrix}$$

Για παράδειγμα εάν ορίσουμε $a = [1,2,3], b = [1,4,7], c = [1,2,\cdots,9,10]$ με τις εντολές

```
>> clear all; a=1:3;b=1:3:7;c=1:10;
```

το άθροισμα και η διαφορά διανυσμάτων υπολογίζεται από τις εντολές:

```
>> a+b
```

```
>> a-b
```

Το ερωτηματικό στο τέλος μίας εντολής δηλώνει το τέλος της και αποτρέπει την εμφάνιση του αποτελέσματος της. Το χρησιμοποιούμε όταν θέλουμε να βάλουμε πολλές εντολές σε μία γραμμή και όταν δεν θέλουμε να εμφανίζονται μακροσκελή αποτελέσματα στο περιβάλλον εργασίας του MATLAB.

Φυσικά όλες οι παραπάνω πράξεις ορίζονται είτε για διανύσματα γραμμή είτε για στήλη οποιασδήποτε διάστασης.

Οι πράξεις εκτελούνται φυσικά όταν ορίζονται, κάτι που δε συμβαίνει στην παρακάτω εντολή.

```
>> a+c
```

Επίσης, όπως στη Γραμμική Άλγεβρα, ορίζονται και ο πολλαπλασιασμός και διαίρεση διανύσματος με αριθμό.

$$a\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} ax \\ ay \\ az \end{bmatrix}, \begin{bmatrix} x \\ y \\ z \end{bmatrix}/a = \begin{bmatrix} x/a \\ y/a \\ z/a \end{bmatrix}$$

```
>> 3*a
```

```
>> a/3
```

Κατά την πρόσθεση (ή αφαίρεση) αριθμού σε διάνυσμα ή πίνακα ο αριθμός προστίθεται (ή αφαιρείται αντίστοιχα) σε κάθε στοιχείο του διανύσματος. Δηλαδή ισχύει:

$$k \pm \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix} = \begin{bmatrix} a \pm k & d \pm k & g \pm k \\ b \pm k & e \pm k & h \pm k \\ c \pm k & f \pm k & i \pm k \end{bmatrix}, k \pm \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} k \pm a \\ k \pm b \\ k \pm c \end{bmatrix}$$

Για παράδειγμα:

```
>> a-1
>> b+2
```

Προσοχή: Αυτή η πράξη φυσικά δεν ορίζεται στη γραμμική άλγεβρα.

Τέλος, όπως έχουμε δει, για διανύσματα ίδιου μεγέθους, ορίζονται και οι **στοιχείο προς στοιχείο** πράξεις πολλαπλασιασμού .*, διαίρεσης **./** και ύψωσης σε δύναμη **.^**. Η ύψωση διανύσματος με τον τελεστή **.^** σε αριθμό υψώνει κάθε στοιχείο του διανύσματος στον αριθμό.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} .* \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} x*a \\ y*b \\ z*c \end{bmatrix}, \quad \begin{bmatrix} x \\ y \\ z \end{bmatrix} .\wedge \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} x \wedge a \\ y \wedge b \\ z \wedge c \end{bmatrix}, \quad \begin{bmatrix} x \\ y \\ z \end{bmatrix} .\wedge n = \begin{bmatrix} x.\wedge n \\ y.\wedge n \\ z.\wedge n \end{bmatrix}, \quad n.\wedge \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} n.\wedge a \\ n.\wedge b \\ n.\wedge c \end{bmatrix}$$

Ας τις θυμηθούμε ξανά:

```
> >clear all;v=[1,2,3,4],w=[4;5;6;7]
>> w'.*v
>> v./w'
>> v.^2
>> v'.^w
>> v.^(w')
```

Παρατήρηση: Όταν **εντολές γράφονται σε μία γραμμή και χωρίζονται με κόμμα**, τότε εμφανίζουν τα αποτελέσματά τους. Όταν χωρίζονται με ερωτηματικό, τότε δεν εμφανίζουν τα αποτελέσματά τους.

Φυσικά όλες οι παραπάνω πράξεις ορίζονται είτε για διανύσματα γραμμή είτε για διανύσματα στήλη οποιασδήποτε διάστασης, αρκεί τα διανύσματα να έχουν την ίδια διάσταση. Τέλος, ορίζεται και η δύναμη αριθμού υψωμένου σε διάνυσμα (στοιχείο προς στοιχείο).

```
>>2.^w
```

Στο παράδειγμά μας, η εφαρμογή της πράξης της αναστροφής (**'**) δημιουργεί διανύσματα που έχουν την ίδια διάσταση και μπορεί να εφαρμοστεί η πράξη στοιχείο προς στοιχείο. Η ύψωση σε δύναμη έχει την μεγαλύτερη προτεραιότητα από όλες τις πράξεις (και την αναστροφή) οπότε είναι απαραίτητη η παρένθεση ώστε να εφαρμοστεί πρώτα η αναστροφή και μετά η ύψωση σε δύναμη.

Αντίστροφος και Ορίζουσα πίνακα

Ένας τετραγωνικός πίνακας $\mathbf{A}$, δηλαδή πίνακας με ίδιο αριθμό γραμμών και στηλών είναι **αντιστρέψιμος** εάν υπάρχει πίνακας, τον οποίο συμβολίζουμε με $\mathbf{A}^{-1}$, για τον οποίο ισχύει $\mathbf{A} \cdot \mathbf{A}^{-1} = \mathbf{A}^{-1} \cdot \mathbf{A} = I$. Όπου I ο μοναδιαίος. Ο πίνακας $\mathbf{A}^{-1}$ ονομάζεται ο **αντίστροφος** (inverse) του $\mathbf{A}$. Ένας πίνακας για τον οποίο δεν υπάρχει αντίστροφος λέγεται **ιδιάζοντας** (singular) .

Εάν $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ τότε ορίζουμε την **ορίζουσα του πίνακα** ως τον αριθμό

$$\det(A) = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

Συμβολίζουμε την ορίζουσα του πίνακα και ως $|A|$.

Εάν $A = \begin{bmatrix} -2 & 3 \\ 1 & 4 \end{bmatrix}$ τότε $\det(A) = \begin{vmatrix} -2 & 3 \\ 1 & 4 \end{vmatrix} = (-2)4 - 3 \cdot 1 = -8 - 3 = -11$

Εάν $A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$ τότε η **ορίζουσά του πίνακα** ισούται με

$$\det(A) = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11}\det(M_{11}) - a_{12}\det(M_{12}) + a_{13}\det(M_{13})$$

Για το στοιχείο a_{ij} η M_{ij} (έλασσων ορίζουσα του στοιχείου) είναι η ορίζουσα του πίνακα που προκύπτει εάν από τον A αφαιρέσουμε την i γραμμή και την j στήλη.

$$\det(A) = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11}\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12}\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13}\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}$$

Εάν $A = \begin{bmatrix} 0 & 1 & 5 \\ 3 & -6 & 9 \\ 2 & 6 & 1 \end{bmatrix}$ τότε

$$\det(A) = \begin{vmatrix} 0 & 1 & 5 \\ 3 & -6 & 9 \\ 2 & 6 & 1 \end{vmatrix} = 0\begin{vmatrix} -6 & 9 \\ 6 & 1 \end{vmatrix} - 1\begin{vmatrix} 3 & 9 \\ 2 & 1 \end{vmatrix} + 5\begin{vmatrix} 3 & -6 \\ 2 & 6 \end{vmatrix} = 0 + 15 + 150 = 165$$

Εδώ αναπτύξαμε την ορίζουσα ως προς την πρώτη γραμμή. Θα μπορούσαμε να την αναπτύξουμε ως προς οποιαδήποτε άλλη γραμμή ή στήλη. Θα πρέπει όμως να θυμόμαστε ότι το πρόσημο του γινομένου κάθε στοιχείου ή με την ελάσσονα ορίζουσα του είναι το αποτέλεσμα $(-1)^{i+j}$ όπου i, j η γραμμή και η στήλη του στοιχείου αντίστοιχα. Το ανάπτυγμα της ορίζουσας ως προς τη δεύτερη στήλη είναι:

$$\det(A) = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = -a_{12}\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{22}\begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix} - a_{32}\begin{vmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{vmatrix}$$

Και για το παράδειγμά μας

$$\det(A) = \begin{vmatrix} 0 & 1 & 5 \\ 3 & -6 & 9 \\ 2 & 6 & 1 \end{vmatrix} = -1\begin{vmatrix} 3 & 9 \\ 2 & 1 \end{vmatrix} + (-6)\begin{vmatrix} 0 & 5 \\ 2 & 1 \end{vmatrix} - 6\begin{vmatrix} 0 & 5 \\ 3 & 9 \end{vmatrix} = 15 + 60 + 90 = 165\,.$$

Με παρόμοιο τρόπο αναπτύσσουμε τις ορίζουσες τετραγωνικών πινάκων με μεγαλύτερη διάσταση. **Οι ιδιάζοντες πίνακες έχουν μηδενική ορίζουσα.**

Η συνάρτηση του MATLAB που υπολογίζει την ορίζουσα ενός πίνακα είναι η **det( )**. Η συνάρτηση **inv( )** υπολογίζει τον αντίστροφο πίνακα, εφόσον αυτός υπάρχει. Επειδή ο υπολογισμός του αντίστροφου γίνεται με προσεγγιστικές μεθόδους, εάν ο αντίστροφος δεν υπάρχει, συνήθως η συνάρτηση επιστρέφει απάντηση πίνακα μαζί με ένα προειδοποιητικό μήνυμα. Μία δοκιμή μπορεί να αποδείξει πόσο ανακριβής μπορεί να είναι ο υπολογισμός του αντίστροφου.

Για τον $A=\begin{bmatrix} 1 & 2 & -1 \\ 1 & 1 & -2 \\ -2 & -1 & 1 \end{bmatrix}$ ισχύει $A^{-1}=\begin{bmatrix} -\frac{1}{4} & -\frac{1}{4} & -\frac{3}{4} \\ \frac{3}{4} & -\frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & -\frac{3}{4} & -\frac{1}{4} \end{bmatrix}$ ενώ ο πίνακας $B=\begin{bmatrix} 1 & 2 & 0 \\ 2 & 4 & 3 \\ 0 & 0 & 1 \end{bmatrix}$ δεν αντιστρέφεται. Ας το δούμε στο MATLAB με τις παρακάτω εντολές.

```
>>clear all
>> a=[1 2 -1; 1 1 -2; -2 -1 1];
>> det(a)
>> inv(a)
>> inv(a)*a
```

ans = 4

ans =
-0.2500 -0.2500 -0.7500
0.7500 -0.2500 0.2500
0.2500 -0.7500 -0.2500

Αποτέλεσμα ο μοναδιαίος.

Ας δούμε τις ανάλογες εντολές για τον ιδιάζοντα πίνακα b.

```
>> b=[1 2 0; 2 4 3; 0 0 1];
>> det(b)
>> inv(b)
>> inv(b)*b
```

ans = 0

Warning: Matrix is singular to working precision.
ans =
Inf Inf Inf
Inf Inf Inf
Inf Inf Inf

Warning: Matrix is singular to working precision.
ans =
NaN NaN NaN
NaN NaN NaN
NaN NaN NaN

Υπολογισμός αντίστροφου με τη χρήση του επαυξημένου πίνακα στο MATLAB

Για εκπαιδευτικούς και μόνο λόγους θα υλοποιήσουμε στο MATLAB τον αλγόριθμο που μάθαμε στη θεωρία. Σύμφωνα με αυτόν τον αλγόριθμο, εάν θέλουμε να υπολογίσουμε τον αντίστροφο ενός πίνακα Α, θεωρούμε τον επαυξημένο πίνακα $[A \mid I]$ και εφαρμόζουμε σε αυτόν στοιχειώδεις γραμμοπράξεις που μετατρέπουν τον A σε ανηγμένο κλιμακωτό πίνακα K. Εάν ακολουθήσουμε αυτή τη διαδικασία ο $[A \mid I]$ έχει μετατραπεί σε έναν πίνακα της μορφής $[K \mid B]$

- Αν $K = I$, τότε ο A είναι αντιστρέψιμος και $A^{-1} = B$.
- Αν $K \neq I$, τότε ο A δεν είναι αντιστρέψιμος.

Ένας πίνακας ονομάζεται **ανηγμένος κλιμακωτός** όταν

Α) οι μηδενικές γραμμές αν υπάρχουν βρίσκονται μετά τις μη μηδενικές στο τέλος (κάτω μέρος) του πίνακα.

Β) Το **οδηγό στοιχείο** κάθε γραμμής (πρώτο μη μηδενικό στοιχείο της) βρίσκεται τουλάχιστον μία θέση δεξιότερα από τον οδηγό της προηγούμενης και έχει την τιμή 1.

Γ) κάθε στήλη που περιέχει οδηγό στοιχείο έχει όλα τα άλλα στοιχεία της μηδενικά.

Ένα παράδειγμα ανηγμένου κλιμακωτού πίνακα είναι ο ακόλουθος:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & . & . & 0 & 0 \\ 0 & 1 & 0 & 0 & . & . & 0 & 0 \\ \vdots & 0 & 0 & 1 & . & . & 0 & 0 \\ \vdots & \vdots & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 \end{bmatrix}$$

Ας δοκιμάζουμε με αυτόν τον τρόπο να υπολογίσουμε, αν υπάρχουν τους, αντιστρόφους των πινάκων:

$$A = \begin{pmatrix} 1 & 0 & 2 \\ 2 & -1 & 3 \\ 4 & 1 & 8 \end{pmatrix} \quad \text{και} \quad B = \begin{pmatrix} 1 & 2 & 0 \\ 2 & 4 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

Θα χρησιμοποιήσουμε την **rref( )** η οποία μας επιστρέφει την ανηγμένη κλιμακωτή μορφή ενός πίνακα. Θα εφαρμόσουμε αυτήν την εντολή στον επαυξημένο πίνακα $[A \mid I]$ που υλοποιούμε με τη μορφή [A,eye(3)] όπου δημιουργούμε έναν block πίνακα.

```
>> clear all;
>>A=[1 0 2; 2 -1 3; 4 1 8]
>>AI=[A,eye(3)]
>>AA= rref(AI)
```

```
AI =
     1     0     2     1     0     0
     2    -1     3     0     1     0
     4     1     8     0     0     1
```

```
AA =
     1     0     0   -11     2     2
     0     1     0    -4     0     1
     0     0     1     6    -1    -1
```

Μετά τις γραμμοπράξεις το αριστερό μέρος του επαυξημένου πίνακα (γραμμές 1..3) είναι ο μοναδιαίος και το δεξί (γραμμές 4..6) ο αντίστροφος.

```
>>inva=AA(:,4:6)
```

```
inva =
   -11     2     2
    -4     0     1
     6    -1    -1
```

Για τον δεύτερο πίνακα, ο οποίος είναι ιδιάζοντας, έχουμε:

```
>> B=[1 2 0; 2 4 0; 0 0 1]
>>BI=[B,eye(3)]
>>BB= rref(BI)
```

```
BB =
    1.0000    2.0000         0         0    0.5000         0
         0         0    1.0000         0         0    1.0000
         0         0         0    1.0000   -0.5000         0
```

Παρατηρούμε ότι μετά τις γραμμοπράξεις, σύμφωνα με τα όσα έχουμε δει στη θεωρία, το αριστερό μέρος του πίνακα (οι τρεις πρώτες στήλες) δεν είναι ο μοναδιαίος οπότε και ο πίνακας δεν αντιστρέφεται.

Άσκηση για εξάσκηση:

Δίνονται οι πίνακες:

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

Για τους ακόλουθους υπολογισμούς :

α) Υπολογίστε (εάν αυτό είναι δυνατό) τους $A+B$, $A+C$, $3A$, AB, BA, AC, CB και τον πίνακα $AA^T - I$.

β) Να υπολογιστεί πίνακας ο οποίος να έχει ως το (i, j)- στοιχείο το (i, j)- στοιχείο του πίνακα A υψωμένο στο τετράγωνο και πολλαπλασιασμένο με το (i, j)- στοιχείο του πίνακα B μειωμένο κατά 2.

γ) Να υπολογιστεί η ορίζουσα των πινάκων A, B, C και να υπολογιστούν οι A^{-1}, B^{-1}, C^{-1} όπου είναι δυνατό να γίνουν οι υπολογισμοί.

δ) Υπολογίστε την ορίζουσα του πίνακα $A^3 - I$.

Όπου δεν γίνονται οι υπολογισμοί να αιτιολογήσετε γιατί δεν γίνονται. (Ως I θεωρείται ο 3x3 μοναδιαίος, A^T ο ανάστροφος του πίνακα A και A^{-1} ο αντίστροφος του πίνακα A)

Γεωμετρική Σημασία Γραμμικών Συστημάτων

Είναι γνωστό ότι μπορούμε να αναπαραστήσουμε γεωμετρικά τα συστήματα με δύο ή τρεις αγνώστους και τη λύση τους. Κάθε εξίσωση με δύο αγνώστους αντιστοιχεί σε μία ευθεία του επιπέδου ενώ κάθε εξίσωση με τρεις αγνώστους αντιστοιχεί σε ένα επίπεδο του χώρου. Η λύση του συστήματος αποτελείται από το σύνολο των σημείων τομής των γραμμών (ή επιπέδων αντίστοιχα) που συνθέτουν το σύστημα των εξισώσεων.

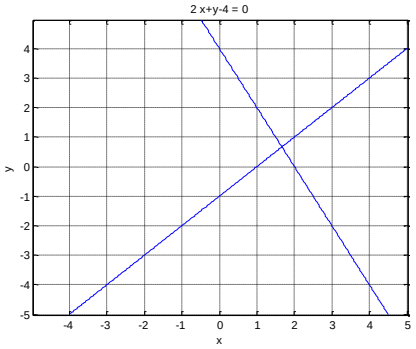
Συστήματα 2x2.

Για τη γραφική αναπαράσταση των ευθειών (και άλλων καμπυλών στο επίπεδο) θα χρησιμοποιήσουμε τη συνάρτηση γραφικών **ezplot( )**. Στην μορφή σύνταξης που θα χρησιμοποιήσουμε αυτήν την εντολή, ο τύπος της συνάρτησης που θέλουμε να σχεδιάσουμε εισάγεται ως κείμενο μέσα σε εισαγωγικά. Κάθε εξίσωση τη λύνουμε ώστε να ισούται με μηδέν στη μορφή *f(x,y)=0* και χρησιμοποιούμε τον τύπο *f(x,y)* στην κλήση της ezplot. Μετά τη συνάρτηση ακολουθεί το πεδίο στο οποίο θα γίνει το γράφημα μέσα σε αγκύλες όπου τα όρια χωρίζονται με κόμμα. Η συνάρτηση clf καθαρίζει το παράθυρο των γραφικών από προηγούμενα γραφικά και ότι ρυθμίσεις έχουμε ορίσει για τα γραφικά μας, η grid on εμφανίζει το πλέγμα στο γράφημα. Τέλος η hold κρατάει το τρέχων γράφημα ώστε το επόμενο να εμφανιστεί στους ίδιους άξονες. Η διαδικασία αυτή τερματίζεται όταν πληκτρολογήσουμε hold off.

Όταν σχεδιάσουμε διαδοχικά τις ευθείες που αντιστοιχούν στις εξισώσεις του ακόλουθου συστήματος

$$\begin{matrix} x - y = 1 \\ 2x + y = 4 \end{matrix}$$

διατηρώντας το γράφημα της πρώτης με τη hold, βλέπουμε το σημείο τομής που είναι και η λύση του συστήματος.




```
>> clear all
>> clf
>> ezplot('x-y-1',[-5 ,5])
>> grid on
>> hold
>> ezplot('2*x+y-4',[-5, 5])
```

Όταν σχεδιάσουμε τις ευθείες που αντιστοιχούν στις εξισώσεις του συστήματος $\begin{matrix} x - y = -1 \\ x - y = 1 \end{matrix}$ βλέπουμε ότι σε αυτήν την περίπτωση οι ευθείες είναι παράλληλες οπότε δεν έχει λύση το σύστημα.

```
>> clear all
>> clf
>> ezplot('x-y-1',[-5, 5])
>> grid on
>> hold
>> ezplot('x-y+1',[-5 ,5])
```

Ενώ όταν σχεδιάσουμε τις ευθείες που αντιστοιχούν στις εξισώσεις του συστήματος $\begin{matrix} -6x + 3y = -6 \\ 2x - y = 2 \end{matrix}$

βλέπουμε ότι σε αυτήν την περίπτωση οι ευθείες ταυτίζονται οπότε έχουμε άπειρες λύσεις για το σύστημα.

```
>> clear all
>> clf
>> ezplot('-6*x+3*y+6',[-5, 5])
>> grid on
>> hold
>> ezplot('2*x-y-2',[-5 ,5])
```

Συστήματα 3x3.

Έστω ότι θέλουμε να παρουσιάσουμε γραφικά τη λύση του συστήματος: $\begin{aligned} x + y + z &= 1 \\ x + 2y + z &= 0 \\ x + y - z &= -1 \end{aligned}$

Αφαιρώντας την πρώτη και την τρίτη εξίσωση βλέπουμε ότι το *z=1*. Άρα από την πρώτη έχουμε *x+y=0* και από τη δεύτερη *y=-1*. Οπότε η λύση είναι *x=1,y=-1, z=1* και τα τρία επίπεδα θα τέμνονται σε ένα σημείο.

Για τη γραφική παράσταση επιπέδων ή επιφανειών στο χώρο χρησιμοποιείται η **ezsurf( )**, η οποία μπορεί να εμφανίζει επιφάνειες στον τρισδιάστατο χώρο. Στην μορφή σύνταξης που θα χρησιμοποιήσουμε αυτήν την εντολή, για το συγκεκριμένο σύστημα, ο τύπος της συνάρτησης που θέλουμε να σχεδιάσουμε εισάγεται ως κείμενο μέσα σε εισαγωγικά. Κάθε εξίσωση την λύνουμε ως προς τον ένα άγνωστο π.χ. για το *z* στην μορφή *z=f(x,y)* και χρησιμοποιούμε τον τύπο *f(x,y)* στην κλήση της ezsurf. Παρόμοια μπορούμε να λύσουμε ως προς άλλο άγνωστο. Μετά τη συνάρτηση ακολουθεί το πεδίο στο οποίο θα γίνει το γράφημα μέσα σε αγκύλες όπου τα όρια χωρίζονται με κόμμα.

```
>> clear all
>> clf
>> ezsurf('1-x-y',[-5,5])
>> hold
>> ezsurf('-2*y-z',[-5,5])
>> ezsurf('x+y-1',[-5,5])
```

Παρατηρούμε ότι τα τρία επίπεδα τέμνονται σε ένα σημείο.

Έστω ότι θέλουμε να παρουσιάσουμε γραφικά τη λύση του συστήματος: $\begin{aligned} x + y + z &= 1 \\ x + 2y + z &= 0 \\ x + y + z &= -2 \end{aligned}$

Άμεσα μπορούμε να συμπεράνουμε ότι το σύστημα είναι ασυμβίβαστο, οπότε τα τρία επίπεδα δεν θα πρέπει να έχουν κοινά σημεία.

```
>> clear all
>> clf
>> ezsurf('1-x-y',[-5,5])
>> hold
>> ezsurf('-2*y-z',[-5,5])
>> ezsurf('-2-x-y',[-5,5])
```

Έστω ότι τώρα θέλουμε να παρουσιάσουμε γραφικά τη λύση του συστήματος: $\begin{aligned} x + y + z &= 1 \\ x + y &= 1 \\ x + y - z &= 1 \end{aligned}$

Αφαιρώντας την πρώτη και την τρίτη εξίσωση βλέπουμε ότι το *z=0*. Οπότε οι τρεις εξισώσεις ταυτίζονται σε μία στην *x+y=1*. Η λύση του συστήματος είναι η *y=1- x, z=0*. Δηλαδή η λύση είναι η ευθεία *y=1- x* στο επίπεδο *z=0*.

Επειδή τη δεύτερη εξίσωση δεν μπορούμε να τη λύσουμε ως προς z, θα επιλέξουμε να εκτελέσουμε την **ezsurf( )**, στην σύνταξή της με την οποία εμφανίζει επιφάνειες όταν γνωρίζουμε την παραμετρική τους παράσταση των συντεταγμένων των σημείων τους $(x(t), y(t), z(t))$. Ο τύπος της κάθε συνιστώσας της επιφάνειας, μπορεί να εισαχθεί ως κείμενο μέσα σε εισαγωγικά. Έχοντας λοιπόν την εξίσωση ενός επιπέδου την λύνουμε ως προς ένα από τα x, y, z και βρίσκουμε την παραμετρική παράσταση αυτής της συντεταγμένης ως προς τις άλλες. Στη συνέχεια ορίζουμε την τριάδα (x, y, z) των σημείων του επιπέδου αντικαθιστώντας στην μεταβλητή ως προς την οποία λύσαμε με την παράσταση της. Με αυτόν τον τρόπο

βρίσκουμε ότι για την $1^{η}$ εξίσωση (επίπεδο) ισχύει $z = 1 - x - y$, για την $2^{η}$ $y = 1 - x$ (αφού δεν υπάρχει z σε αυτήν) και για την $3^{η}$ $z = x + y - 1$.

```
>> clear all
>> clf
>> ezsurf('x','y','1-x-y')
>> hold
>> ezsurf('x','1-x','z')
>> ezsurf('x','y','-1+x+y')
```

Στο γράφημα που παίρνουμε, η τομή των τριών επιπέδων (που είναι μία ευθεία) αποτελεί τη λύση του συστήματος των εξισώσεων των επιπέδων.

Επίλυση γραμμικών συστημάτων n εξισώσεων με n αγνώσους στο MATLAB

Ο τελεστής της διαίρεσης (αριστερής και δεξιάς) μπορεί να εφαρμοστεί και ανάμεσα σε πίνακες. Στην ουσία εάν διαιρέσουμε έναν πίνακα A με ένα πίνακα B με την κοινή διαίρεση / τότε πολλαπλασιάζουμε τον A με τον αντίστροφο του B, δηλαδή εκτελούμε $A / B = A \cdot B^{-1}$. Επίσης, εάν διαιρέσουμε έναν πίνακα A με ένα πίνακα B με την διαίρεση \ τότε πολλαπλασιάζουμε τον αντίστροφο του A με τον B, δηλαδή εκτελούμε $A \setminus B = A^{-1} \cdot B$. Ο συγκεκριμένος τελεστής χρησιμοποιείται στην επίλυση γραμμικών συστημάτων.

Ένα γραμμικό σύστημα $n \times n$ μπορεί να γραφεί στη μορφή πινακική $A \cdot x = b$. Εφόσον υπάρχει ο αντίστροφος του πίνακα A, η **λύση** του συστήματος δίνεται από τον τύπο $x = A^{-1}b$. Αυτόν τον τύπο μπορούμε να τον υλοποιήσουμε στο MATLAB με τη χρήση του τελεστή αριστερής διαίρεσης \.

Όταν θέλουμε να λύσουμε στο MATLAB το σύστημα $\begin{matrix} x + y + 2z = 3 \\ 2x + 2y + 3z = 5 \\ x - y = 5 \end{matrix}$

το γράφουμε σε μορφή πινάκων ως : $\underbrace{\begin{pmatrix} 1 & 1 & 2 \\ 2 & 2 & 3 \\ 1 & -1 & 0 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} x \\ y \\ z \end{pmatrix}}_{X} = \underbrace{\begin{pmatrix} 3 \\ 5 \\ 5 \end{pmatrix}}_{B}$

και εκτελούμε τις ακόλουθες εντολές για να πάρουμε τη λύση:

```
>> clear all
>> a=[1 1 2; 2 2 3; 1 -1 0];
>> b=[3 5,5]';
>> x=a\b
```

```
x =
    3
   -2
    1
```

Εναλλακτικά θα μπορούσαμε να χρησιμοποιήσουμε την εντολή:

```
>> x=inv(a)*b
```

Μπορούμε να επαληθεύσουμε το αποτέλεσμα με την απλή εντολή που πρέπει να επιστρέφει **μηδενικό** διάνυσμα ως αποτέλεσμα.

```
>> a*x- b
```

Αν τώρα θέλουμε να λύσουμε το σύστημα: $\begin{matrix} x + y + 2z = 3 \\ 2x + 2y + 4z = 5 \\ x - y = 5 \end{matrix}$

Ορίζουμε τους πίνακες της πινακικής αναπαράστασής του: $\underbrace{\begin{pmatrix} 1 & 1 & 2 \\ 2 & 2 & 4 \\ 1 & -1 & 0 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} x \\ y \\ z \end{pmatrix}}_{X} = \underbrace{\begin{pmatrix} 3 \\ 5 \\ 5 \end{pmatrix}}_{B}$

και στη συνέχεια μπορούμε να προσπαθήσουμε να το λύσουμε.

```
>> clear all
>> a=[1 1 2; 2 2 4; 1 -1 0];
>> b=[3 5,5]';
>> x=a\b
```

```
Warning: Matrix is singular to working
precision.
x =
   NaN
  -Inf
   Inf
```

Το συγκεκριμένο σύστημα δεν έχει λύση αφού η ορίζουσα του πίνακα του είναι 0 (ιδιάζοντας πίνακας).

```
>> det(a)
```

Διερεύνηση και λύση συστημάτων με την rref στο MATLAB

Η εντολή **rref( ),** η οποία μας επιστρέφει την ανηγμένη κλιμακωτή μορφή ενός πίνακα, μπορεί να χρησιμοποιηθεί για να βρούμε τις λύσεις συστημάτων, άσχετα με το εάν ο πίνακας τους είναι τετραγωνικός. Μπορούμε να εφαρμόσουμε αυτήν την εντολή στον επαυξημένο πίνακα $[A \mid b]$ και στη συνέχεια, αφού θεωρήσουμε τον γραμμοισοδύναμο (με τις ίδιες λύσεις συστήματος) επαυξημένο πίνακα, να βρούμε τη λύση.

Για να λύσουμε το πρώτο από τα δύο συστήματα που είδαμε παραπάνω εκτελούμε:

```
>> clear all
>> a=[1 1 2; 2 2 3; 1 -1 0];
>> b=[3 5,5]';
>> epafx=[a,b];
>> rref(epafx)
```

```
ans =

     1     0     0     3
     0     1     0    -2
     0     0     1     1
```

Από τον τελικό επαυξημένο εξάγουμε τη λύση του συστήματος.

Για να λύσουμε το δεύτερο από τα δύο συστήματα που είδαμε παραπάνω εκτελούμε:

```
>> clear all
>> a=[1 1 2; 2 2 4; 1 -1 0];
>> b=[3 5,5]';
>> epafx=[a,b];
>> rref(epafx)
```

```
ans =

     1     0     1     0
     0     1     1     0
     0     0     0     1
```

Από την τελευταία εξίσωση που έχουμε στον τελικό επαυξημένο εξάγουμε το συμπέρασμα ότι το σύστημα είναι ασυμβίβαστο.

$$-x + 2z + w = 0$$

Έστω τώρα ότι θέλουμε να λύσουμε το σύστημα: $y - 2z + w = 0$

$$2x + y - w = 0$$

Το συγκεκριμένο σύστημα τριών εξισώσεων με τέσσερις αγνώστους είναι ομογενές. Αυτό σημαίνει ότι το σύστημα είτε έχει μόνο την μηδενική λύση είτε άπειρες λύσεις. Για να λύσουμε το συγκεκριμένο σύστημα εκτελούμε:

```
>> clear all
>> a=[-1 0 2 1; 0 1 -2 1; 2 1 0 -1];
>> b=[0 0 0]';
>> epafx=[a,b];
>> rref(epafx)
```

```
ans =

     1     0     0    -1     0
     0     1     0     1     0
     0     0     1     0     0
```

Από την τελευταία γραμμή έχουμε $6z = 0 \Rightarrow z = 0$. Οπότε οι δύο πρώτες εξισώσεις είναι οι $x - w = 0, y + w = 0$ από όπου προκύπτουν $x = w, y = -w$ από όπου προκύπτει η μονοπαραμετρική απειρία λύσεων) $[x \quad y \quad z \quad w]^T = [w \quad -w \quad 0 \quad w]^T, x_4 \in \mathbb{R}$.

Άσκηση για εξάσκηση:

Με οποιοδήποτε από γνωστούς τρόπους, λύστε στο MATLAB το σύστημα:

$$x + 3y = -1$$
$$-y + 2z = 0$$
$$5x + 4z = 4$$

Τα πολυώνυμα στο MATLAB

Τα πολυώνυμα στο MATLAB τα χειριζόμαστε με τη χρήση διανυσμάτων (γραμμές ή στήλες) που περιέχουν τους συντελεστές τους. Εάν p είναι ένα διάνυσμα μεγέθους n+1 του οποίου τα στοιχεία είναι οι συντελεστές του πολυωνύμου $p_1 x^n + p_2 x^{n-1} + p_3 x^{n-2} + \ldots + + p_n x + p_{n+1}$. Η **polyval(p,x)** υπολογίζει την τιμή στο x του πολυωνύμου, δηλαδή υπολογίζει την τιμή p(1) *x^(n)+ p(2)*x^(n-1) + ... + p(n)*x + p(n+1). Αν το x είναι πίνακας ή διάνυσμα το πολυώνυμο υπολογίζεται σε όλα τα στοιχεία του x. Για παράδειγμα ας ορίσουμε το πολυώνυμο $p(x) = x^3 - 3x^2 + 4x - 2$ για το οποίο ισχύει $p(0) = -2$ και ότι έχει ρίζες τα $1, 1+i, 1-i$.

```
>> clear all; p=[1,-3,4,-2];
>> polyval(p,0)
>> polyval(p,[1,1+i,1-i])
```

```
ans = -2
```

```
ans =

   0   0   0
```

Ας δούμε και το γράφημά του:

```
>>clf
>> x=-10:0.1:10;
>> plot(x,polyval(p,x))
```

Η **roots(p)** υπολογίζει τις ρίζες του πολυωνύμου του οποίου συντελεστές είναι τα στοιχεία του διανύσματος p . Επίσης η **poly(v)**, όταν το v είναι διάνυσμα, επιστρέφει διάνυσμα που αντιστοιχεί σε πολυώνυμο του οποίου οι ρίζες είναι τα στοιχεία του v. Φυσικά το πολυώνυμο εκφράζεται με τη μορφή διανύσματος των συντελεστών του.

Για το πολυώνυμο που έχουμε ορίσει στο πάνω παράδειγμα έχουμε:

```
>> roots(p)
>> poly([1,1+i,1-i])
```

ans =
1.0000 + 1.0000i
1.0000 - 1.0000i
1.0000

ans =
1 -3 4 -2

Τα πολυώνυμα πηλίκο και υπόλοιπο της διαίρεσης δύο πολυωνύμων p1,p2 μπορεί να επιστραφεί από την **deconv(p1,p2)** και το γινόμενό τους με την **conv(p1,p2)**.

Για παράδειγμα σχετικά με τη διαίρεση $(x^4 - x^3 + 2x - 3)/(x^2 - 1)$, η οποία έχει πηλίκο $x^2 - x + 1$ και υπόλοιπο $x - 2$ και την οποία κάνουμε αναλυτικά ως εξής:

$$
\begin{array}{r|l}
x^4 - x^3 + 2x - 3 & x^2 - 1 \\
\cline{2-2}
-x^4 + x^2 & x^2 - x + 1 \\
\cline{1-1}
-x^3 + x^2 + 2x - 3 & \\
x^3 - x & \\
\cline{1-1}
x^2 + x - 3 & \\
-x^2 + 1 & \\
\cline{1-1}
x - 2 &
\end{array}
$$

Όπως σε όλες τις διαιρέσεις θα ισχύει:

Διαιρεταίος = Διαιρέτης x Πηλίκο + Υπόλοιπο

δηλαδή θα ισχύει:

$$x^4 - x^3 + 2x - 3 = (x^2 - 1)(x^2 - x + 1) + (x - 2)$$

Για να εκτελέσουμε στο MATLAB πληκτρολογούμε:

$x^4 - x^3 + 2x - 3$

$x^4 - x^3 + 2x - 3$ / $x^2 - 1$

```
>> [phliko,ypoloipo]=deconv([1,-1,0,2,-3],[1,0,-1])
```

Παρατηρούμε ότι βάζουμε 0 ως συντελεστή του x^2 για το πολυώνυμο διαιρετέο και 0 ως συντελεστή του x .

Τα αποτελέσματα είναι:

$x^2 - x + 1$

```
phliko =
     1    -1     1
```

$x^2 - x + 1$

$x - 2$

```
ypoloipo =
     0     0     0     1    -2
```

Φυσικά ικανοποιείται η ταυτότητα της διαίρεσης διαιρετέος=διαιρέτης x πηλίκο + υπόλοιπο, το οποίο υλοποιούμε με την ακόλουθη εντολή:

```
>> conv(phliko,[1,0,-1])+ypoloipo
```

ans =
1 -1 0 2 -3

$x^4 - x^3 + 2x - 3$

$x^2 - 1$

Άσκηση για εξάσκηση:

Διαιρέστε τα πολυώνυμα $p(x) = 2x^3 - 4x^2 + 3x - 2$ και $q(x) = x - 3$ και γράψτε στο χαρτί ποια είναι τα πολυώνυμα πηλίκο και υπόλοιπο.
Στη συνέχεια κάντε το γράφημα του πολυωνύμου $p(x)$ στο διάστημα [-5,5] (όπου έχετε θεωρήσει διαμέριση πλάτους 0.01).

Συμβολικά μαθηματικά στο MATLAB

Οι συμβολικοί υπολογισμοί μπορεί να είναι:

Υπολογισμοί με ακριβή αριθμητική

Υπολογισμοί με μεταβλητή ακρίβεια (μεγαλύτερη της διπλής ακρίβειας 16 ψηφίων με την οποία εργάζεται το MATLAB).

Χειρισμός και υπολογισμοί με μαθηματικά σύμβολα ή συναρτήσεις, λύσεις εξισώσεων, ακριβείς αλγεβρικοί και διάφοροι άλλοι ακριβείς υπολογισμοί στους οποίους παίρνουμε αποτελέσματα όπως αυτά που θα είχαμε αν τους κάναμε στο χαρτί.

Το **Matlab Symbolic Toolbox** δίνει τη δυνατότητα να χειριστούμε συμβολικές ποσότητες και να εκτελέσουμε συμβολικούς υπολογισμούς, υπολογισμούς με ακριβή αριθμητική και υπολογισμούς με μεταβλητή ακρίβεια (μεγαλύτερη της διπλής ακρίβειας 16 ψηφίων με την οποία εργάζεται το MATLAB). Το Matlab Symbolic Toolbox παρέχεται δωρεάν με τη φοιτητική έκδοση του MATLAB. Για την πλήρη έκδοση θα πρέπει να αγοραστεί ξεχωριστά. Οι συμβολικοί υπολογισμοί δεν υποστηρίζονται από τον κλώνο του MATLAB την Octave.

Όπως είναι γνωστό ο υπολογιστής χρησιμοποιεί αριθμητική κινητής υποδιαστολής με 16 ψηφία ακρίβεια στο MATLAB. Θεωρώντας συμβολικές ποσότητες (με την εντολή **sym** ή με την παραπλήσια **syms**) κάνουμε ακριβείς πράξεις χρησιμοποιώντας την ακριβή άλγεβρα και αριθμητική που θα χρησιμοποιούσαμε κάνοντας αυτές τις πράξεις στο χαρτί. Τέλος, έχουμε τη δυνατότητα να εκτελούμε αριθμητικές πράξεις με ακρίβεια μεγαλύτερη των 16 ψηφίων (με τη χρήση της εντολής **vpa**) ή να μετατρέπουμε συμβολικές ή μεταβλητής ακρίβειας αριθμητικές ποσότητες σε ποσότητες διπλής ακρίβειας (με την εντολή **double**). Στην συνέχεια θα παρουσιάσουμε μόνο τη δήλωση και τον χειρισμό συμβολικών ποσοτήτων και μερικές από τις συμβολικές δυνατότητες συμβολικών αλγεβρικών πράξεων.

Για παράδειγμα υπολογίζουμε το $\frac{1}{4}\,\frac{2}{3}=\frac{1}{6}$ με αριθμητικούς και συμβολικούς υπολογισμούς.

```
>> 1/4*2/3
```

ans =

0.1667

```
>> sym('1/4')*sym('2/3')
```

ans =1/6

Το $\cos\frac{\pi}{2}$ συμβολικά υπολογίζεται ακριβώς μηδέν ενώ αριθμητικά με έναν πολύ μικρο αριθμο.

```
>> cos(pi/2)
```

ans= 6.1232e-017

```
>> cos(sym('pi')/2)
```

ans =0

Για τον υπολογισμό της $\sqrt{2}$ με 30 ψηφία ακρίβεια πληκτρολογούμε:

```
>> vpa('sqrt(2)',30)
```

ans = 1.41421356237309504880168872421

```
>>format long
>> double(ans)
```

ans = 1.414213562373095

```
>> sqrt(2)
```

ans = 1.414213562373095

Όπως αναφέραμε συμβολικές μεταβλητές ποσότητες μπορούμε να δηλώσουμε με τη χρήση της εντολής δηλωτικής εντολής syms. Εναλλακτικά η εκτέλεση της εντολής sym μετατρέπει το όρισμά της σε συμβολικό αντικείμενο. Οι εντολές **whos** και **class** μπορούν να μας δείξουν τον τύπο των μεταβλητών. Μεταβλητές που προέρχονται από πράξεις άλλων συμβολικών μεταβλητών είναι και αυτές συμβολικές. Δείτε την εφαρμογή των παραπάνω εκτελώντας τις ακόλουθες εντολές:

```
>> x=sym('x');
>> syms y z
>> class(z)
>> whos
```

Δείτε επίσης τα περιεχόμενα του Workspace.

Παρατήρηση:

Η εντολή

```
>> syms y z
```

είναι ισοδύναμη με τις δύο ακόλουθες εντολές:

```
>> y=sym('y');
>> z=sym('z');
```

Γνωρίζουμε τώρα πια ότι το MATLAB είναι ένα περιβάλλον το οποίο είναι φτιαγμένο για να εργάζεται με πίνακες. Οι βασικοί τελεστές +,-,*,/,^ πινάκων (και διανυσμάτων) εφαρμόζονται και δίνουν αποτελέσματα εφόσον οι διαστάσεις των πινάκων (και διανυσμάτων) το επιτρέπει. Οι τελεστές .* , ./, .^ εφαρμόζουν την πράξη στοιχείο προς στοιχείο. Όλοι αυτοί οι τελεστές εφαρμόζονται είτε όταν οι πίνακες περιέχουν αμιγώς αριθμητικές ποσότητες είτε όταν ορίζονται ως συμβολικές ποσότητες.

Για παράδειγμα, αν θεωρήσουμε τους ακόλουθους πίνακες

$$A=\begin{pmatrix}1&2&3\\2&3&4\\3&4&5\end{pmatrix},\ B=\begin{pmatrix}2&1&0\\0&2&0\\0&0&x\end{pmatrix}$$

από τους οποίους ο δεύτερος είναι συμβολικός και θέλουμε να υπολογίσουμε τα

α) $A+B$

β) $3B$

γ) AB, BA

δ) την ορίζουσα των A, B και τους A^{-1}, B^{-1}

τότε πληκτρολογούμε

```
>> clear all
>> syms x
>> a=[1 2 3;2 3 4;3 4 5];
>> b=[2 1 0; 0 2 0; 0 0 x];
>> a+b
>> 3*b
>> a*b
>> b*a
>> det(b)
>> det(a)
>> inv(a)
>> inv(b)
```

```
ans =
 [ 3,  3,  3]
 [ 2,  5,  4]
 [ 3,  4, 5+x]
```

```
ans =
 [ 2,  5, 3*x]
 [ 4,  8, 4*x]
 [ 6, 11, 5*x]
```

```
ans =
 [ 4,  7, 10]
 [ 4,  6,  8]
 [ 3*x, 4*x, 5*x]
```

```
ans =4*x
```

```
ans =0
```

```
ans =
 [ 1/2, -1/4,  0]
 [  0,  1/2,  0]
 [  0,   0,  1/x]
```

Τα αποτελέσματα πράξεων πινάκων με συμβολικά αντικείμενα με αριθμητικούς πίνακες είναι συμβολικά αντικείμενα και όχι πίνακες με αριθμητικά στοιχεία.
Συμβολικά μπορούν να λυθούν και συστήματα. Για παράδειγμα, έστω ότι θέλουμε να λυθεί το παρακάτω σύστημα :

$$ax + y + z = 1$$

$$x + ay + z = a$$

$$x + y + az = a^2$$

και να γίνει πλήρης διερεύνηση για τις τιμές του a αρχικά δηλώνουμε τις συμβολικές μεταβλητές. Στη συνέχεια λύνουμε με την εντολή **solve( )** το σύστημα. Η solve έχει ως πρώτα ορίσματα τα αριστερά μέλη της ισότητας των εξισώσεων αφού έχουμε μεταφέρει όλες τις ποσότητες σε αυτά. Στη συνέχεια ως ορίσματα παραθέτονται οι άγνωστοι ως προς τους οποίους θέλουμε να λύσουμε το σύστημα .

```
>> clear all
>> syms a x y z
>> s=solve(a*x+y+z-1,x+a*y+z-a,x+y+a*z-a^2,x,y,z)
```

```
s =
    x: [1x1 sym]
    y: [1x1 sym]
    z: [1x1 sym]
```

Η απάντηση στην παραπάνω εντολή είναι ένα συμβολικό αντικείμενο το s το οποίο συμπεριλαμβάνει τις εκφράσεις των x,y,z που ικανοποιούν το σύστημα. Για να δούμε τις συμβολικές εκφράσεις των λύσεων, αλλά να τις χειριστούμε, αναθέτοντας τις, ως τιμές σε κάποιες μεταβλητές για παράδειγμα, πληκτρολογούμε:

```
>> s.x
>> s.y
>> s.z
```

```
ans = -(a+1)/(a+2)
```

```
ans = 1/(a+2)
```

```
ans = (1+a^2+2*a)/(a+2)
```

Η **factor** παραγοντοποιεί συμβολικές εκφράσεις οπότε η διερεύνηση είναι τώρα εύκολη.

```
>> factor(s.z)
```

```
ans = (a+1)^2/(a+2)
```

Συμβολική λύση μπορούμε να έχουμε και με τη χρήση του τελεστή \.

```
>> clear all
>> syms a real
>> am=[a 1 1; 1 a 1;1 1 a];b=[1 a a^2]';
>> s=am\b
```

```
s =
  -(a+1)/(a+2)
    1/(a+2)
  (a+1)^2/(a+2)
```

Από τα παραπάνω συμπεραίνουμε ότι αν $a \neq -2$ έχουμε την παραπάνω λύση. Εάν $a = -2$ δεν υπάρχει λύση ενώ εάν $a = -1$ έχουμε απειρία λύσεων.

Ένας εναλλακτικός τρόπος να κάνουμε διερεύνηση είναι, αφού δηλώσουμε τη συμβολική μεταβλητή a ως πραγματική, να ορίσουμε τους πίνακες και να ζητήσουμε, όπως κάναμε και στους πίνακες με αριθμητικά στοιχεία, με τη χρήση της **rref( )** να γίνουν οι κατάλληλες γραμμοπράξεις στον επαυξημένο πίνακα ώστε να μας επιστραφεί η ισοδύναμη ανηγμένη κλιμακωτή μορφή του (reduced row echelon form). Η **rref( )** μπορεί, όπως είδαμε, να χρησιμοποιηθεί και για αριθμητικούς πίνακες (δηλαδή πίνακες με μόνο αριθμητικά στοιχεία).

```
>> clear all
>> syms a real
>> am=[a 1 1; 1 a 1;1 1 a];
>> b=[1 a a^2]';
>>factor(rref([am,b]))
```

```
ans =
[      1,      0,      0, -(1+a)/(a+2)]
[      0,      1,      0,     1/(a+2)]
[      0,      0,      1, (1+a)^2/(a+2)]
```

Τέλος η εντολή αντιστροφής πινάκων **inv( )** εφαρμόζεται και σε συμβολικούς πίνακες, όπως βλέπουμε για τον πίνακα του παραπάνω συστήματος.

```
>> syms a
>> am=[a 1 1; 1 a 1;1 1 a];
>> inv(am)
>> monad=inv(am)*am
```

```
ans =
 [ (a+1)/(a^2+a-2),   -1/(a^2+a-2),   -1/(a^2+a-2)]
[   -1/(a^2+a-2), (a+1)/(a^2+a-2),   -1/(a^2+a-2)]
[   -1/(a^2+a-2),   -1/(a^2+a-2), (a+1)/(a^2+a-2)]
```

```
[(a+1)/(a^2+a-2)*a-2/(a^2+a-2), (a+1)/(a^2+a-2)-1/(a^2+a-2)*a-1/(a^2+a-2), (a+1)/(a^2+a-2)-1/(a^2+a-2)*a-1/(a^2+a-2)]
[ (a+1)/(a^2+a-2)-1/(a^2+a-2)*a-1/(a^2+a-2),          (a+1)/(a^2+a-2)*a-2/(a^2+a-2), (a+1)/(a^2+a-2)-1/(a^2+a-2)*a-1/(a^2+a-2)]
[ (a+1)/(a^2+a-2)-1/(a^2+a-2)*a-1/(a^2+a-2), (a+1)/(a^2+a-2)-1/(a^2+a-2)*a-1/(a^2+a-2),          (a+1)/(a^2+a-2)*a-2/(a^2+a-2)]
```

Αν απλοποιήσουμε το αποτέλεσμα με την εντολή **simplify**( ), παρατηρήστε ότι τελικά το τελευταίο αποτέλεσμα είναι ο αναμενόμενος μοναδιαίος. Αυτό το βλέπουμε εάν εκτελέσουμε:

```
>> simplify(monad)
```

```
ans =
 [ 1, 0, 0]
[ 0, 1, 0]
[ 0, 0, 1]
```

Μία ανάλογη εντολή με την οποία απλοποιούμε συμβολικές εκφράσεις είναι η **simple( )** η οποία, αφού εκτελέσει και εμφανίσει διάφορες διαδικασίες απλοποίησης, επιστρέφει την πιο απλή (σε έκταση) μορφή.

```
>> simple(monad)
```

Ας δούμε κάποιες ακόμη εφαρμογές με συμβολικούς αλγεβρικούς υπολογισμούς. Το ανάπτυγμα μιας αλγεβρικής παράστασης δίνεται από την συνάρτηση **expand( ).** Η **sym2poly( )** μετατρέπει το συμβολικό πολυώνυμο σε πολυώνυμο διανυσματικής αναπαράστασης. Η pretty( ) εμφανίζει το αποτέλεσμα με μια πιο μαθηματική μορφή ώστε να είναι καλύτερα αναγνώσιμο.

Με τις ακόλουθες εντολές θα υπολογίσουμε το ανάπτυγμα του $(x+1)^5$.

```
>> clear all; syms x a b c
>> expand((x+1)^5)
>> pretty(ans)
>> sym2poly(ans)
```

ans=x^5+5*x^4+10*x^3+10*x^2+5*x+1

ans = 1 5 10 10 5 1

$x^5 + 5x^4 + 10x^3 + 10x^2 + 5x + 1$

Για να παραγοντοποιηθεί το πολυώνυμο $(a-b)^3 + (b-c)^3 + (c-a)^3$ εκτελούμε:

```
>> factor((a-b)^3+(b-c)^3+(c-a)^3)
```

ans =-3*(c-a)*(b-a)*(b-c)

Για να βρεθεί η λύση της πολυωνυμικής εξίσωσης $3(x+3)^2 + 5(x+5)^2 = 8(x+8)^2$ πληκτρολογούμε

```
>> p=solve(3*(x+3)^2+5*(x+5)^2-8*(x+8)^2,x)
```

Όπως έχει αναφερθεί, τα ορίσματα της εντολής **solve( )** είναι η πολυωνυμική έκφραση που θέλουμε να εξισώσουμε με το 0 και ο άγνωστος ως προς τον οποίο θα γίνει η λύση. Η απάντηση είναι η μοναδική λύση $x = -6$.

Αυτό φαίνεται όταν κάνουμε τις πράξεις και φέρουμε σε πιο απλή μορφή την έκφραση $3(x+3)^2 + 5(x+5)^2 - 8(x+8)^2$.

```
>>simple(3*(x+3)^2+5*(x+5)^2-8*(x+8)^2)
```

ans = -60*x-360

Τέλος, για να βρεθεί η λύση της πολυωνυμικής εξίσωσης $3(x+3)^2 + 5(x+5)^2 = 0$ πληκτρολογούμε:

```
>> p=solve(3*(x+3)^2+5*(x+5)^2,x)
```

p =
-17/4+1/4*i*15^(1/2)
-17/4-1/4*i*15^(1/2)

Παρατηρούμε ότι έχουμε δύο ρίζες τις οποίες μπορούμε να πάρουμε ως στοιχεία του διανύσματος λύσης:

```
>> p(1)
>> p(2)
```

Τέλος, με τη χρήση της εντολής **[a ,b]=numden(p)** μπορούμε να εκχωρήσουμε στη μεταβλητή **a** τον αριθμητή και στην μεταβλητή **b** τον παρονομαστή μίας ρητής συμβολικής έκφρασης **p**. Για παράδειγμα έστω ότι δίνεται η ρητή έκφραση

$$p = \frac{x^4 + x^3 + 2x + 2}{x^5 - x^4 + x^3 - x^2}$$

Για τον αριθμητή έχουμε:

$$x^4 + x^3 + 2x + 2 = x^4 + 2x + x^3 + 2 = x(x^3 + 2) + (x^3 + 2) = (x^3 + 2)(x + 1)$$

Για τον παρονομαστή:

$$x^5 - x^4 + x^3 - x^2 = x^4(x-1) + x^2(x-1) = \left(x^4 + x^2\right)(x-1) = x^2\left(x^2 + 1\right)(x-1)$$

Έστω ότι θέλουμε, με τη χρήση των δυνατοτήτων του MATLAB, να βρούμε τις πραγματικές ρίζες του αριθμητή και του παρονομαστή. Μπορούμε στον ορισμό της συμβολικής ποσότητας *x* να περιορίσουμε τις τιμές της στους πραγματικούς αριθμούς. Στη συνέχεια να εκχωρήσουμε σε μεταβλητές τον αριθμητή

και τον παρονομαστή της ρητής έκφρασης και να βρούμε τις ρίζες τους. Στο MATLAB θα εκτελέσουμε τις ακόλουθες εντολές:

```
>> clear all
>> syms x real
>> p=(x^4+x^3+2*x+2)/(x^5-x^4+x^3-x^2);
>> [pn,pd]=numden(p);
>> denom=factor(pd)
>> rdenom=solve(pd,x)
```

```
denom =
-x^2*(x - 1)*(x^2 + 1)
```

```
rdenom = 0
         0
         1
```

Από την παραγοντοποίηση παρατηρούμε ότι οι ρίζες του παρονομαστή είναι το 0 (διπλή), το 1 και οι δύο μιγαδικές +i,-i. Επειδή, με τον ορισμό της έχουμε περιορίσει την ποσότητα *x* στους πραγματικούς αριθμούς, η επίλυση μας επιστρέφει τις πραγματικές ρίζες του παρονομαστή. Για τον αριθμητή οι ανάλογες εντολές θα δώσουν αποτελέσματα που χρειάζονται περαιτέρω χειρισμό.

```
>> num=factor(pn)
>> rnum=solve(pn,x)
>> simplify(rnum)
```

```
num =
-(x + 1)*(x^3 + 2)
```

```
rnum =              -1
((-2)^(1/3)*3^(1/2)*i)/2 - (-2)^(1/3)/2
```

```
rnum =              -1
                 -2^(1/3)
```

Από την παραγοντοποίηση παρατηρούμε ότι οι ρίζες του παρονομαστή είναι το -1 και τρεις ρίζες (δύο μιγαδικές και μία πραγματική) του παράγοντα $x^3 + 2$. Επειδή, με τον ορισμό της έχουμε περιορίσει την ποσότητα *x* στους πραγματικούς αριθμούς, αναμέναμε η επίλυση να μας επιστρέψει τις πραγματικές ρίζες του αριθμητή. Αυτό κάνει, ωστόσο χρειάζεται η απλοποίηση του αποτελέσματος για να δούμε την πιο απλή μορφή της ρίζας $-\sqrt[3]{2}$.

Μαθηματική Ανάλυση στο MATLAB

Αθροίσματα και γινόμενα

Αθροίσματα και γινόμενα αριθμητικών ποσοτήτων μπορούν να υπολογιστούν στο MATLAB αθροίζοντας ή πολλαπλασιάζοντας αντίστοιχα τα στοιχεία διανυσμάτων που περιέχουν τις ποσότητες. Οι συναρτήσεις **sum( )** και η **prod( )** μπορούν να χρησιμοποιηθούν για να υπολογίσουμε τις ακόλουθες ποσότητες:

$$\sum_{k=1}^{100} k \qquad \prod_{k=1}^{100} k \qquad \sum_{k=1}^{100} k^2 \qquad \prod_{k=1}^{100} k^2$$

```
>> clear all
>> x=1:100;
>> sum(x)
>> prod(x)
>> sum(x.^2)
>> prod(x.^2)
```

```
5050
```

```
9.3326e+15
```

```
338350
```

```
Inf
```

Παρατηρούμε ότι η τελευταία ποσότητα είναι τελικά τόσο μεγάλη που ξεπερνά τα όρια των αριθμών που χειρίζεται το MATLAB και θεωρείται άπειρη.
Εναλλακτικά μπορούμε να χρησιμοποιήσουμε τις συμβολικές δυνατότητες του περιβάλλοντος. Η εντολή **symsum( )** υπολογίζει συμβολικά αθροίσματα και η **subs(p,a,b)** αντικαθιστά σε μία συμβολική παράσταση **p** μία συμβολική ποσότητα **a** με μία τιμή (αριθμητική ή συμβολική) **b**. Οι παρακάτω εντολές υλοποιούν τον υπολογισμό των

$$\sum_{k=1}^{100} k^2 \quad \text{και} \quad \sum_{k=1}^{n} k^2$$

με συμβολικό τρόπο:

```
>> syms k n
>> symsum(k^2,1,100)
>> symsum(k^2,1,n)
>> pretty(ans)
>> subs(ans,n,100)
```

1/3*(n+1)^3-1/2*(n+1)^2+1/6*n+1/6

338350

Άπειρα Αθροίσματα

Τα άπειρα αθροίσματα $\sum_{n=1}^{\infty} a_n$ τα ονομάζουμε σειρές στα Μαθηματικά. Ένα τέτοιο άθροισμα είτε συγκλίνει, όταν το όριο $\lim_{n\to\infty} s_n = a$ όπου $s_n = \sum_{i=1}^{n} a_i$ είναι το μερικό άθροισμα, είτε απειρίζεται.

Για να κατανοήσουμε τον ορισμό της σύγκλισης με τη χρήση συμβολικών υπολογισμών θα υπολογίσουμε το μερικό άθροισμα της σειράς

$$\sum_{n=1}^{\infty} \left(\frac{1}{n} - \frac{1}{n+1}\right)$$

δηλαδή την ποσότητα

$$s_n = \sum_{k=1}^{n} \left(\frac{1}{k} - \frac{1}{k+1}\right)$$

Και θα εξετάσουμε την τιμή της όταν το n τείνει στο άπειρο.

```
>> clear all
>> syms k n
>> symsum(1/k-1/(k+1),1,n)
>> simplify(ans)
```

Παρατηρούμε ότι η τιμή

$$\lim_{n\to\infty} s_n = \lim_{n\to\infty} \sum_{i=1}^{n} a_i = \lim_{n\to\infty} \left(\frac{1}{1} - \frac{1}{2} + \frac{1}{2} - \frac{1}{3} + \frac{1}{3} - \frac{1}{4} + \frac{1}{4} - \ldots - \frac{1}{n} + \frac{1}{n} - \frac{1}{n+1}\right) = \lim_{n\to\infty} \left(1 - \frac{1}{n+1}\right) = \lim_{n\to\infty} \frac{n}{n+1} = 1$$

οπότε και η σειρά συγκλίνει στο 1. Πράγματι, βλέπουμε ότι η ακόλουθη εντολή το πιστοποιεί.

```
>> symsum(1/n-1/(n+1),1,Inf)
```

Υπάρχει εκτεταμένη θεωρία στην Μαθηματική Ανάλυση σχετικά με τη σύγκλιση και την εύρεση της τιμής μίας σειράς.
Εδώ, ενδεικτικά, αναφέρουμε τη γνωστή γεωμετρική σειρά, όπου

$$\sum_{n=1}^{\infty} r^n = \frac{r}{1-r}, \quad \text{όταν } |r| < 1$$

και τα ακόλουθα δύο αθροίσματα:

$$\sum_{n=1}^{\infty} \frac{1}{n!} = e \qquad \sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$

Για το δεύτερο εύκολα βλέπουμε ότι ισχύει:

```
>> symsum(1/n^2,1,Inf)
```

Ορισμός συμβολικών συναρτήσεων.

Μπορούμε να ορίσουμε μαθηματικές συναρτήσεις στο MATLAB με συμβολικές μεταβλητές στις οποίες θα είναι δυνατό να εφαρμόζονται συμβολικές πράξεις και εντολές για συμβολικές παραστάσεις και συναρτήσεις. Για να το πετύχουμε αυτό θα πρέπει να ορίσουμε πρώτα τη συμβολική μεταβλητή της μαθηματικής συνάρτησης (είτε τις συμβολικές μεταβλητές) και στη συνέχεια να ορίσουμε τη συνάρτηση με τη χρήστη της **inline**. Σε αυτήν την περίπτωση δεν χρειάζεται να βάλουμε τον τύπο της συνάρτησης σε εισαγωγικά και να χρησιμοποιήσουμε την **vectorize** για να μπορεί να εφαρμοστεί η συνάρτηση σε διανύσματα. Η γραφική παράσταση των συμβολικών μαθηματικών συναρτήσεων γίνεται με τη χρήση της **ezplot**.

Ας δούμε τι ακόλουθο παράδειγμα. Έστω ότι θέλουμε να ορίσουμε τις συναρτήσεις $f(x) = \frac{1}{x+1}$, $g(x) = x^3 - 1$ να βρούμε διάφορες τιμές τους και να κάνουμε τη γραφική τους παράσταση.

```
>> clear all
>> syms x real
>> f=inline(1/(x+1))
>> g=inline(x^3-1)
>> f(2)
>> g([1;2])
>> ezplot(f(x),[-3,3])
>> figure
>> ezplot(g(x),[-3,3])
```

Σύνθεση συναρτήσεων και αντίστροφη συνάρτησης.

Από την ανάλυση είναι γνωστό ότι για δύο πραγματικές συναρτήσεις $f: A \to B$ και $g: C \to D$ μπορούμε να ορίσουμε τη σύνθεσή τους $h = g \circ f: A \to D$ η οποία έχει τύπο $h(x) = g(f(x))$. Για τον ορισμό της συνάρτησης h το πεδίο τιμών της f και το πεδίο ορισμού της g πρέπει να έχουν κοινά στοιχεία και ίσως είναι απαραίτητο να θεωρηθούν περιορισμοί στα σύνολα *A* και *D*.

Επίσης μία συνάρτηση λέγεται συνάρτηση 1-1 όταν $f(x_1) = f(x_2) \Rightarrow x_1 = x_2$ και επί όταν $f(A) = B$. Συναρτήσεις που έχουν αυτές τις δύο ιδιότητες αντιστρέφονται. Δηλαδή μπορούμε να βρούμε συνάρτηση $f^{-1}: B \to A$, την οποία ονομάζουμε αντίστροφη συνάρτηση, για την οποία να ισχύει $f^{-1}(f(x)) = x$. Οι γραφικές παραστάσεις μίας συνάρτησης και της αντίστροφής της είναι συμμετρικές ως προς την ευθεία $y = x$.

Για παράδειγμα για τις γνωστές τριγωνομετρικές συναρτήσεις ισχύει:

$$\sin(\Theta) = \alpha \Leftrightarrow \arcsin(\alpha) = sin^{-1}(\alpha) = \Theta$$

$$cos(\Theta) = \alpha \Leftrightarrow \arccos(\alpha) = cos^{-1}(\alpha) = \Theta$$

και για την εκθετική συνάρτηση $f(x) = e^x$, εφόσον ισχύει $e^{ln(x)} = x$, η αντίστροφη συνάρτηση της είναι η $f^{-1}(x) = ln(x)$.

Για τις συναρτήσεις που ορίσαμε παραπάνω $f(x)=\frac{1}{x+1}$, $g(x)=x^3-1$ έχουμε:

Το πεδίο ορισμού της f είναι φανερά το.

$$D_f=\mathbb{R}-\{-1\}=(-\infty,\ -1)\cup(-1,\ +\infty)$$

Το πεδίο τιμών της f ορίζεται ως:

$$R_f=\left\{y\in ¡\ :\ \text{για τα οποία η εξίσωση ως προς } x,\ y=\frac{1}{x+1}\text{έχει μία τουλάχιστον λύση στο } D_f\right\}$$

Δηλαδή ζητάμε τη λύση του συστήματος:

$$\left.\begin{array}{l} y=\dfrac{1}{x+1} \\ x\neq -1 \end{array}\right\}\Rightarrow \left.\begin{array}{l} x=\dfrac{1}{y}-1 \\ x\neq -1 \end{array}\right\}\Rightarrow \left.\begin{array}{l} x=\dfrac{1-y}{y} \\ x\neq -1 \end{array}\right\}\Rightarrow \left.\begin{array}{l} x=\dfrac{1-y}{y} \\ \dfrac{1-y}{y}\neq -1 \end{array}\right\}\Rightarrow \left.\begin{array}{l} x=\dfrac{1-y}{y} \\ 1\neq 0 \end{array}\right\}$$

Το οποίο έχει νόημα μόνο όταν $y\neq 0$, οπότε πεδίο τιμών της f είναι το

$$R_f=¡\ -\{0\}=(-\infty,0)\cup(0,+\infty)$$

Η συνάρτηση είναι 1-1 διότι $f(x_1)=f(x_2)\Rightarrow \frac{1}{x_1-1}=\frac{1}{x_2-1}\Rightarrow x_1=x_2$ και είναι επί του πεδίου τιμών της.

Μπορούμε λοιπόν να ορίσουμε την αντίστροφη συνάρτηση $f^{-1}: R_f=¡\ -\{0\}\to D_f=¡\ -\{-1\}$

$$y=f(x)=\frac{1}{x+1}\Rightarrow x=\frac{1}{y}-1=f^{-1}(y)\ .$$

Το πεδίο ορισμού της g είναι φανερά το $D_g=\mathbb{R}$.

Το πεδίο τιμών της g ορίζεται ως:

$$R_g=\left\{y\in ¡\ :\ \text{για τα οποία η εξίσωση ως προς } x,\ y=x^3-1\ \text{έχει μία τουλάχιστον λύση στο } D_g\right\}$$

Φανερά αυτή η εξίσωση έχει λύση για κάθε $y\in ¡$ διότι

$$x^3-1=y\Leftrightarrow x=\begin{cases}\sqrt[3]{y+1} & \text{όταν } y\geq -1\\ \sqrt[3]{-(y+1)} & \text{όταν } y<-1\end{cases}$$

οπότε $R_g=¡$.

(Παρατήρηση: Στους πραγματικούς αριθμούς αρνητική ρίζα με οποιοδήποτε εκθέτη δεν ορίζεται.)

Η συνάρτηση g είναι 1-1 διότι $g(x_1)=g(x_2)\Rightarrow x_1^3-1=x_2^3-1\Rightarrow x_1^3=x_2^3\Rightarrow x_1=x_2$ και φανερά επί του $R_g=¡$ οπότε ορίζεται η αντίστροφος της $g^{-1}: D_{g^{-1}}=R_G=¡\to R_{g^{-1}}=D_g=¡$ η οποία έχει τύπο

$$g^{-1}(x)=\begin{cases}\sqrt[3]{x+1} & \text{όταν } x\geq -1\\ \sqrt[3]{-(x+1)} & \text{όταν } x<-1\end{cases}.$$

Για την σύνθεση $(f\circ g)(x)$ έχουμε: $(f\circ g)(x)=f(g(x))=f(x^3-1)=\frac{1}{(x^3-1)+1}=\frac{1}{x^3}$

Για την σύνθεση $(g\circ f)(x)$ έχουμε:

$$(g\circ f)(x)=(g(f(x))=g\left(\frac{1}{x+1}\right)=\left(\frac{1}{x+1}\right)^3-1=\frac{1-(x+1)^3}{(x+1)^3}=-\frac{x^3+3\cdot x^2+3\cdot x}{(x+1)^3}$$

Η χρήση των συμβολικών δυνατοτήτων του MATLAB μας επιτρέπει να βρούμε την αντίστροφη συνάρτησης με την συνάρτηση **finverse( )** ή τον τύπο της σύνθετης συνάρτησης με τη χρήση της εντολής **compose( )**. **Προσοχή**, αφού οριστεί η μεταβλητή x ως συμβολική, στην **inline( )** ο τύπος της συνάρτησης δε χρειάζεται εισαγωγικά. Και σε αυτήν την περίπτωση μπορούμε να χρησιμοποιήσουμε την **ezplot( )**.

Ας δούμε και τους τύπους και τα γραφήματα καθεμίας από τις f, g με την αντίστροφή της στο ίδιο γράφημα. Τις αντίστροφες συναρτήσεις f^{-1}, g^{-1} μπορούμε να τις ορίσουμε μέσα σε inline ώστε να τις χειριζόμαστε και αυτές ως αντικείμενα συναρτήσεων. Με την εντολή **close( )** μπορούμε να κλείσουμε μεμονωμένα, ή και όλα (με την μορφή που την εκτελούμε παρακάτω) τα ανοικτά παράθυρα γραφικών.

```
>> close('all')
>> finv=inline(finverse(f(x)))
>> figure
>> ezplot(f(x),[-3,3])
>> hold
>> ezplot(finv(x),[-3,3])
```

Μπορούμε να παρατηρήσουμε ότι οι γραφικές παραστάσεις των g, g^{-1} είναι (όπως είναι αναμενόμενο) συμμετρικές ως προς την ευθεία *y=x*.

```
>> ezplot(x,[-3,3])
>> hold off
```

Στον υπολογισμό της αντιστρόφου συνάρτησης της *g(x)*, το MATLAB αδυνατεί να κάνει την ανάλυση της αντίστροφης συνάρτησης σε κλάδους που κάναμε παραπάνω και για αυτό μας επιστρέφει ως αντίστροφη τον ένα κλάδο και κατάλληλο μήνυμα ότι δεν υπάρχει μοναδική αντίστροφος.

```
>> ginv= inline(finverse(g(x)))
>> figure
>> ezplot(g(x),[-3,3])
>> hold
>> ezplot(ginv(x),[-3,3])
```

Ας δούμε και τους τύπους και τα γραφήματα των συναρτήσεων που προκύπτει από τη σύνθεσή τους $f \circ g$ και $g \circ f$.

```
>> close('all')
>> z=compose(f(x),g(x))
>> pretty(z)
>> figure
>> ezplot(z,[-3,3])
>> figure
>> h=inline(simplify(compose(g(x),f(x))))
>> figure
>> ezplot(h(x),[-3,3])
```

Χειρισμός ορίων στο MATLAB.

Για να υπολογίσουμε όρια στο Matlab θα πρέπει να βασιστούμε στις συμβολικές δυνατότητες του περιβάλλοντος. Αυτό σημαίνει ότι θα πρέπει αρχικά να δηλώσουμε με τη χρήση της εντολής **syms** τις συμβολικές ποσότητες που θα περιέχονται στα όρια μας. Η εντολή με τη χρήση της οποίας υπολογίζουμε όρια είναι η **limit**. Στον πίνακα που ακολουθεί παρουσιάζεται η σύνταξή της.

Μαθηματική Έκφραση	Υλοποίηση στο Matlab
$\lim_{x\to 0} f(x)$	**limit(f,0) ή limit(f(x),0)**
$\lim_{x\to a} f(x)$	**limit(f,x,a) ή limit(f(x),x,a)**
$\lim_{x\to a-} f(x)$	**limit(f,x,a, 'left') ή limit(f(x),x ,a, 'left')**
$\lim_{x\to a+} f(x)$	**limit(f,x,a, 'right') ή limit(f(x),x, a, 'right')**

Στον παραπάνω πίνακα το x είναι η ανεξάρτητη μεταβλητή ως προς την οποία βρίσκουμε το όριο και θα πρέπει να έχει ορισθεί ως συμβολική ποσότητα. Η f μπορεί να είναι μία συμβολική ποσότητα ως προς x ή να έχει ορισθεί πριν ως μία συμβολική μαθηματική συνάρτηση f(x) με τη χρήση της inline. Για όρια στο άπειρο (ή μείον άπειρο) το a αντικαθιστάται με το inf (ή με το –inf αντίστοιχα). Η limit μπορεί να εφαρμοστεί και σε διάνυσμα συναρτήσεων. Κατά τη χρήση της limit χωρίς την εμφάνιση της παραμέτρου x το MATLAB θεωρεί ότι μεταβλητή είναι η x και υπολογίζει το όριο.

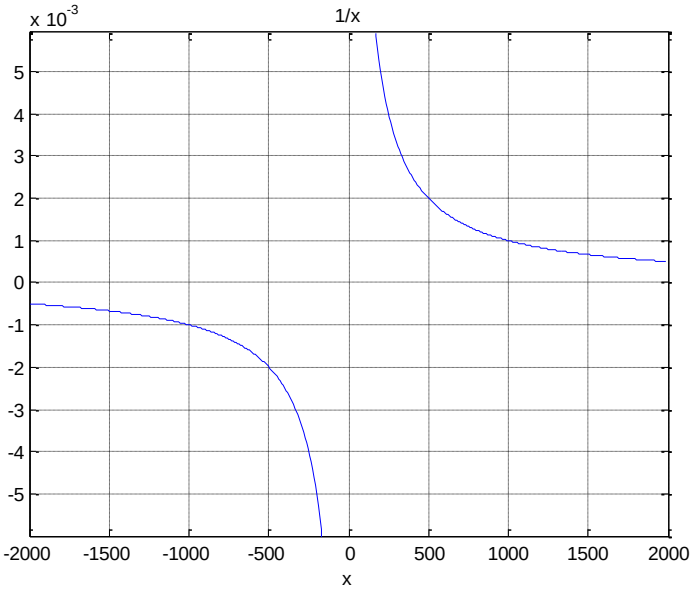
Ακολούθως θα υπολογίσουμε τα όρια

$$\lim_{x\to 0}\frac{\sin(x)}{x}=1 \qquad \lim_{x\to -2}\frac{x^3-3x+2}{x^2-2x-8}\overset{0/0}{=}\lim_{x\to -2}\frac{3x^2-3}{2x-2}=-\frac{3}{2}$$

αφού ορίσουμε τη συμβολική ποσότητα x, που θα χειριστούμε. (το πρώτο είναι γνωστό όριο και το δεύτερο μπορούμε να το υπολογίσουμε με τη χρήση του κανόνα L'Hospital)

```
>> clear all
>> syms x
>> limit(sin(x)/x,x,0)
>> p=inline((x^3-3*x+2)/(x^2-2*x-8))
>> pretty(p(x))
>> limit(p(x),x,-2)
```

Επίσης θα μελετήσουμε τη συμπεριφορά της $f(x)=\frac{1}{x}$



```
>> clear all
>> syms x
>> limit(1/x,x,0)
>> limit(1/x,x,0,'right')
>> limit(1/x,x,0,'left')
>> limit(1/x,x,inf)
>> limit(1/x,x,-inf)
```

Τέλος θα υπολογίσουμε την παράγωγο της $f(x) = \cos(x)$ με τη χρήση του ορισμού της παραγώγου:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

με τις ακόλουθες εντολές.

```
>> clear all
>> syms x h
>> g=inline(cos(x))
>> limit((g(x+h)-g(x))/h,h,0)
```

Ο υπολογισμός παραγώγων στο MATLAB.

Για να υπολογίσουμε παραγώγους συναρτήσεων στο MATLAB μπορούμε πάλι να βασιστούμε στις συμβολικές δυνατότητες του εργαλείου Symbolic Math Toolbox του περιβάλλοντος. Αυτό σημαίνει ότι θα πρέπει αρχικά να δηλώσουμε με τη χρήση της εντολής **syms** τις συμβολικές ποσότητες που θα περιέχονται στις παραγώγους που θα υπολογίσουμε. Η εντολή με τη χρήση της οποίας υπολογίζουμε παράγωγους είναι η **diff**. Στον πίνακα που ακολουθεί παρουσιάζεται η σύνταξή της.

Μαθηματική Έκφραση	**Υλοποίηση στο Matlab**
$\frac{df(x)}{dx}$	**diff(f,x) ή diff(f(x),x)**
$\frac{d^k f(x)}{dx^k}$	**diff(f,x,k) ή diff(f(x),x,k)**

Στον παραπάνω πίνακα το x είναι η ανεξάρτητη μεταβλητή ως προς την οποία παραγωγίζουμε και θα πρέπει να έχει ορισθεί ως συμβολική ποσότητα. Η f μπορεί να είναι μία συμβολική ποσότητα ως προς x ή να έχει ορισθεί πριν ως μία συμβολική μαθηματική συνάρτηση f(x) με τη χρήση της inline. Η diff( ) μπορεί να εφαρμοστεί και σε διάνυσμα συναρτήσεων. Κατά τη χρήση της diff χωρίς την εμφάνιση της παραμέτρου x το MATLAB θεωρεί ότι μεταβλητή είναι η x και υπολογίζει το όριο.

Για εφαρμογή θα υπολογίσουμε την πρώτη και τη δεύτερη παράγωγο της συνάρτησης

$$f(x) = \cos(x^2 + 1)$$

αφού ορίσουμε τις συμβολικές ποσότητες που θα χειριστούμε.

```
>> clear all
>> syms x n
>> f=inline(cos(x^2+1))
>> diff(f(x),x)
>> pretty(ans)
>> diff(f(x),x,2)
>> pretty(ans)
```

Εφαρμογή: Εύρεση ακρότατων και σημείων καμπής.

Θα μελετήσουμε ως προς τα ακρότατα και τα σημεία καμπής τη συνάρτηση

$$f(x) = (x+1)^2 (x-1)^2$$

Θυμίζουμε ότι εάν για ένα σημείο x_0 ισχύει $\left.\frac{df}{dx}\right|_{x=x_0} = \left.\frac{d^2 f}{dx^2}\right|_{x=x_0} = \ldots = \left.\frac{d^n f}{dx^n}\right|_{x=x_0} = 0$ και $\left.\frac{d^{n+1} f}{dx^{n+1}}\right|_{x=x_0} \neq 0$.

Εάν το **n είναι άρτιος** τότε το σημείο είναι **σημείο καμπής**. Εάν το **n είναι περιττός** τότε αν $\left.\frac{d^{n+1} f}{dx^{n+1}}\right|_{x=x_0} > 0$ τότε το σημείο είναι **σημείο τοπικού ελαχίστου**, ενώ αν $\left.\frac{d^{n+1} f}{dx^{n+1}}\right|_{x=x_0} < 0$ το σημείο είναι **σημείο τοπικού μεγίστου**.

Αρχικά ορίζουμε το x ως συμβολική ποσότητα και τη συνάρτηση *f(x)*.

```
>> clear all
>> clf
>> syms x
>> f=inline((x-1)^2*(x+1)^2)
```

Θα μας χρειαστεί η πρώτη, η δεύτερη και η τρίτη παράγωγος της συνάρτησης, την οποία υπολογίζουμε με την diff.

```
>> df=diff(f(x),x)
```

df =2*(-1+x)*(x+1)^2+2*(-1+x)^2*(x+1)

```
>> ddf=diff(f(x),x,2)
```

ddf =2*(x+1)^2+8*(-1+x)*(x+1)+2*(-1+x)^2

```
>> dddf=diff(f(x),x,3)
```

dddf =24*x

Στο επόμενο βήμα βρίσκουμε τα κρίσιμα σημεία, δηλαδή τις ρίζες της πρώτης παραγώγου:

```
>> s=solve(df,x)
```

s =
0
-1
1

Αντικαθιστώντας τις ρίζες που βρήκαμε (3 στην περίπτωσή μας) στη δεύτερη παράγωγο βρίσκουμε, από το πρόσημο της τιμής της δεύτερης παραγώγου στα σημεία αυτά, ότι το 0 είναι σημείο τοπικού μεγίστου και τα 1 και -1 ολικού ελαχίστου. Το μηδέν ως μοναδικό τοπικό μέγιστο είναι και ολικό.

```
>> subs(ddf,x,s(1))
```

-4<0 το 0 τοπικού μεγίστου

```
>> subs(ddf,x, s(2))
```

8>0 το 0 τοπικού ελαχίστου

```
>> subs(ddf,x,s(3))
```

8>0 το 0 τοπικού ελαχίστου

Βρίσκοντας τις ρίζες της δεύτερης παραγώγου και αντικαθιστώντας τις στην τρίτη παράγωγο βρίσκουμε ότι τα σημεία $\pm\frac{\sqrt{3}}{3}$ αποτελούν σημεία καμπής.

```
>> ds=solve(ddf,x)
```

ds =
1/3*3^(1/2)
-1/3*3^(1/2)

```
>> subs(dddf,x,ds(1))
```

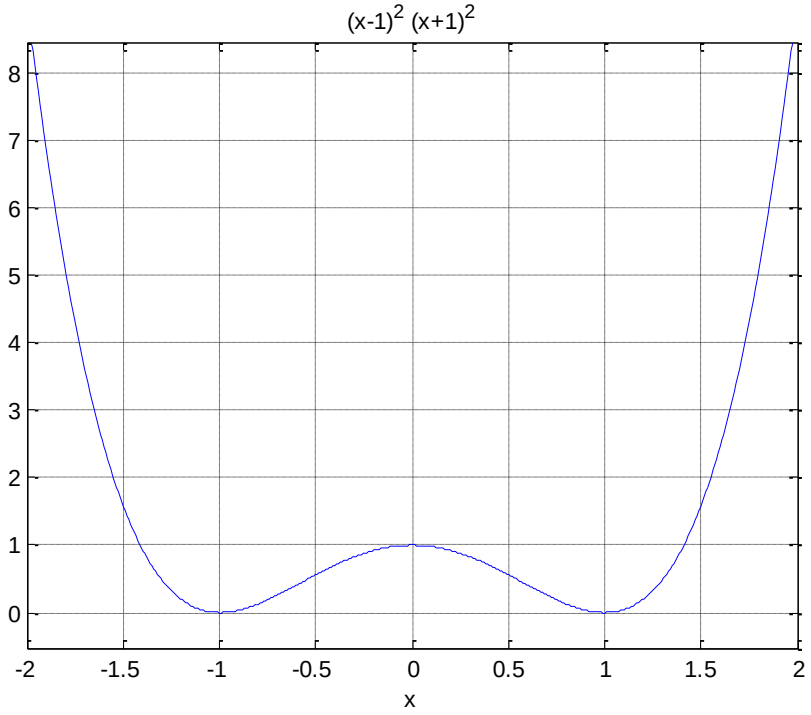
8*3^(1/2) διάφορο του 0 οπότε σημείο καμπής

```
>> subs(dddf,x,ds(2))
```

-8*3^(1/2) διάφορο του 0 οπότε σημείο καμπής

Όλα τα παραπάνω συμπεράσματα πιστοποιούνται και από το γράφημα της συνάρτησης.

```
>> ezplot(f(x),[-2,2])
>> grid on
```



Εφαρμοσμένο παράδειγμα. Ένα πρόβλημα ελαχίστου.

Ένα κλειστό κυλινδρικό δοχείο με κυκλική βάση έχει χωρητικότητα 64 cm^3. Θα βρούμε τις διαστάσεις του ώστε το ποσό του μετάλλου που χρειάζεται για τα τοιχώματά του να είναι ελάχιστο.

Για να ελαχιστοποιήσουμε το ποσό του μετάλλου αρκεί να ελαχιστοποιήσουμε την επιφάνεια του κυλινδρικού δοχείου.

Έστω r, h, V, E η ακτίνα της βάσης, το ύψος, ο όγκος και το εμβαδό της επιφάνειας του δοχείου αντίστοιχα. Από γνωστούς τύπους έχουμε ότι:

$$V = \pi r^2 h \Leftrightarrow \pi r^2 h = 64 \Leftrightarrow h = h(r) = \frac{64}{\pi r^2}$$

$$E(r) = 2\pi rh + 2\pi r^2 = 2\pi r \frac{64}{\pi r^2} + 2\pi r^2 = \frac{128}{r} + 2\pi r^2, r > 0$$

Το κρίσιμο σημείο (σημείο που μηδενίζεται η πρώτη παράγωγος) της συνάρτησης του εμβαδού βρίσκεται λύνοντας την:

$$E'(r) = 0$$

Στο MATLAB υπολογίζουμε την παράγωγο και λύνουμε την εξίσωση. Ορίζουμε την πρώτη και τη δεύτερη παράγωγο ως συμβολική συνάρτηση.

```
>> clear all
>> syms r real
>> E=inline(128/r+2*pi*r^2)
>> h=inline(64/(pi*r^2))
>> DE=inline(diff(E(r),r))
>> DDE=inline(diff(E(r),r,2))
>> s=solve(DE(r),r)
```

```
s =
 2/pi^(1/3)*2^(2/3)
```

Η φύση του προβλήματος μας επιβάλει το ότι μιγαδικές ρίζες δεν έχουν νόημα μιας και δεν μπορούμε να έχουμε μιγαδικές διαστάσεις. Για αυτό το λόγο, ορίσαμε τη συμβολική ποσότητα *r* ως πραγματικό αριθμό και έτσι οι μιγαδικές ρίζες της πρώτης παραγώγου δεν υπολογίζονται. Θα εξετάσουμε τώρα, μελετώντας την τιμή της παράγωγου $2^{ης}$ τάξης στο κρίσιμο σημείο, αν το σημείο αυτό είναι θέση ολικού ελαχίστου.

```
>> DDE(s)
```

```
ans = 12 pi
```

Παρατηρούμε ότι $E''(r) > 0, \forall r > 0$. Επομένως $r = 2\sqrt[3]{\frac{4}{\pi}}$ είναι θέση ολικού ελαχίστου. Για την τιμή αυτή της ακτίνας το ύψος είναι:

```
>> h(s)
```

```
ans =

4/pi^(1/3)*2^(2/3)
```

Ορισμένο ολοκλήρωμα στο MATLAB

Ο υπολογισμός του ορισμένου ολοκληρώματος

$$\int_a^b f(x)dx$$

μπορεί να γίνει με δύο τρόπους.

Ο πρώτος είναι χρησιμοποιώντας τις συμβολικές δυνατότητες του MATLAB και τη συνάρτηση συμβολικού (ακριβούς) υπολογισμού ολοκληρώματος **int( )**. Στην εντολή αυτή ως ορίσματα έχουμε την προς ολοκλήρωση συνάρτηση, τη μεταβλητή ολοκλήρωσης και τα άκρα. Οι δυνατότητες υπολογισμού ολοκληρωμάτων με αυτήν τη συνάρτηση δεν είναι απεριόριστες.

Για εφαρμογή θα υπολογίσουμε το

$$\int_0^{\pi} \sin(x)dx$$

Πρώτα συμβολικά. Αρχικά ορίζουμε το x ως συμβολική ποσότητα και τη συνάρτηση *f(x)*.

```
>> clear all
>> syms x
>> f=inline(sin(x))
```

Στη συνέχεια εκτελούμε:

```
>> int(f(x),x,0,pi)
```

Εναλλακτικά, μπορούμε να υπολογίσουμε το ολοκλήρωμα με τη συνάρτηση **quad( )**. Η συνάρτηση αυτή υπολογίζει με προσεγγιστικές μεθόδους το ολοκλήρωμα και δέχεται ως παραμέτρους τη συνάρτηση **που έχει οριστεί με inline** και τα άκρα της ολοκλήρωσης. **Προσοχή, επειδή η συνάρτηση δεν πρέπει να έχει ορισθεί ως συμβολική μαθηματική συνάρτηση ο τύπος της θα πρέπει να μπει σε εισαγωγικά.** Το αποτέλεσμα αποτελεί **προσέγγιση της τιμής** του ολοκληρώματος. Για το ίδιο ολοκλήρωμα έχουμε:

```
>> clear all
>> f=inline('sin(x)')
>> quad(f,0,pi)
```

Αόριστο ολοκλήρωμα στο MATLAB

Ο υπολογισμός του αόριστου ολοκληρώματος

$$\int f(x)dx$$

μπορεί να γίνει **μόνο** χρησιμοποιώντας τις συμβολικές δυνατότητες του MATLAB και τη συνάρτηση συμβολικού (ακριβούς) υπολογισμού ολοκληρώματος **int( )**. Οι δυνατότητες υπολογισμού ολοκληρωμάτων με αυτήν τη συνάρτηση δεν είναι απεριόριστες.

Για εφαρμογή θα υπολογίσουμε τα

$$\int \sin(x)dx \qquad \int \frac{x^2}{\sqrt{4-x^2}}dx \qquad \int \frac{x^3}{\sqrt{x^2+25}}dx .$$

με τη χρήση των εντολών:

```
>> clear all
>> syms x
>> int(sin(x),x)
>> int(x^2/sqrt(4-x^2),x)
>> int(x^3/sqrt(x^2+25),x)
```

Γενικευμένα Ολοκληρώματα

Έστω ότι η *f(t)* μία πραγματική ορισμένη στο διάστημα $a \le t < \infty$. Τότε το ολοκλήρωμα

$$\int_a^{\infty} f(t)dt = \lim_{b\to\infty}\int_a^b f(t)dt$$

Ονομάζεται **γενικευμένο ολοκλήρωμα** (πρώτου είδους) της *f(t).* Αν το όριο υπάρχει και είναι πραγματικός αριθμός τότε λέμε ότι το γενικευμένο ολοκλήρωμα υπάρχει ή **συγκλίνει.** π.χ.

$$\int_1^{\infty}\frac{dt}{t^2} = \lim_{b\to\infty}\int_1^b\frac{dt}{t^2} = \lim_{b\to\infty}\left[-\frac{1}{t}\right]_1^b = -\lim_{b\to\infty}\frac{1}{b}+1 = 1$$

Αν το όριο δεν υπάρχει ή απειρίζεται τότε λέμε ότι το γενικευμένο ολοκλήρωμα **αποκλίνει.**

Για παράδειγμα:

$$\int_1^{\infty}\frac{dt}{t} = \lim_{b\to\infty}\int_1^b\frac{dt}{t} = \lim_{b\to\infty}\left[\ln t\right]_1^b = \lim_{b\to\infty}\ln b = +\infty$$

ή

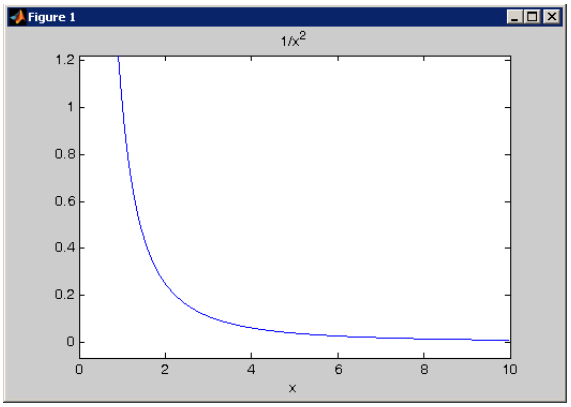
$$\int_0^{\infty}\cos t dt = \lim_{b\to\infty}\int_0^b\cos t dt = \lim_{b\to\infty}\left[\sin t\right]_0^b = \lim_{b\to\infty}\sin b$$

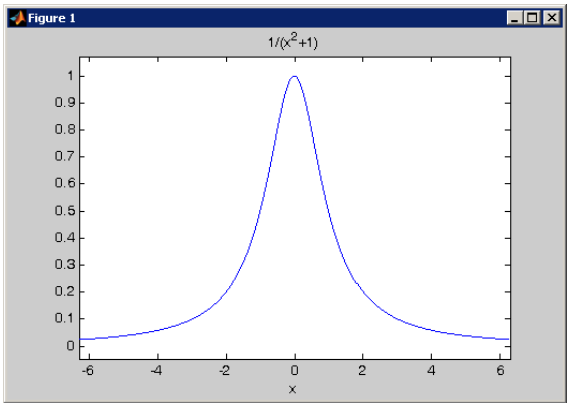
το οποίο δεν υπάρχει επειδή στην περίπτωση όπου το *t* απειρίζεται η συνάρτηση *sin* κυμαίνεται μεταξύ 1 και -1.

Ένα γενικευμένο ολοκλήρωμα α' είδους μπορεί να επίσης τη μορφή $\int_{-\infty}^{a} f(x)dx$. Υπάρχουν και γενικευμένα ολοκληρωματα β' είδους της μορφής, $\int_{a+}^{b} f(x)dx$ ή $\int_{a}^{b-} f(x)dx$ όπου η συνάρτηση δεν ορίζεται στο σημεία α+ ή στο b-. Τέλος, ένα γενικευμένο ολοκλήρωμα γ' είδους μπορεί να έχει τη μορφή $\int_{a+}^{\infty} f(x)dx$

Ως εφαρμογή, θα υπολογίσουμε με την εντολή **int( )** τα γενικευμένα ολοκληρώματα

$$\int_0^{\infty}\frac{1}{x^2}dx \qquad \int_1^{\infty}\frac{1}{x^2}dx \qquad \int_{-\infty}^{\infty}\frac{1}{x^2+1}dx .$$






```
>> clear all
>> syms x
>> int(1/x^2,x,0,inf)
>> int(1/x^2,x,1,inf)
>> int(1/(x^2+1),x,-inf,inf)
```

inf

1

pi

Το αποτέλεσμα του πρώτου ολοκληρώματος μας λέει ότι το εμβαδό της περιοχής, που οριοθετείται από τους άξονες xx' και yy' και την καμπύλη στο πρώτο τεταρτημόριο, δεν είναι φραγμένο και απειρίζεται. Ενώ το δεύτερο αποτέλεσμα μας λέει ότι το εμβαδό της περιοχής, που οριοθετείται από τον άξονα xx', την ευθεία x=1 και την καμπύλη στο πρώτο τεταρτημόριο είναι φραγμένο και τείνει στο 1. Τέλος για την συνάρτηση $f(x)=\frac{1}{x^2+1}$ το εμβαδό της περιοχής, που οριοθετείται από τον άξονα xx' και την καμπύλη ισούται με π .

Οι σειρές Taylor και Maclaurin

Είναι γνωστό στη Μαθηματική Ανάλυση ότι αν η συνάρτηση f είναι απείρως παραγωγήσιμη με συνεχείς παραγώγους στην περιοχή ενός πραγματικού αριθμού α τότε η συνάρτηση μπορεί να γραφεί ως σειρά

$$f(x)=f(a)+\frac{(x-a)}{1!}f^{(1)}(a)+\frac{(x-a)^2}{2!}f^{(2)}(a)+\mathrm{L}\ +\frac{(x-a)^n}{n!}f^{(n)}(a)+\ldots=\sum_{n=1}^{\infty}\frac{(x-a)^n}{n!}f^{(n)}(a)$$

η οποία ονομάζεται σειρά **Taylor** της συνάρτησης με κέντρο το α.
Αν α=0 τότε το ανάπτυγμα ονομάζεται και ανάπτυγμα σε σειρά Maclaurin.

$$f(x)=f(0)+\frac{x}{1!}f^{(1)}(0)+\frac{x^2}{2!}f^{(2)}(0)+\mathrm{L}\ +\frac{x^n}{n!}f^{(n)}(0)+\ldots.$$

Για παράδειγμα για την $f(x)=\sin(x)$ η σειρά Maclaurin είναι η ακόλουθη:

$$\left\{\begin{array}{l} f(x)=\sin(x)\Rightarrow\ f(0)=\sin(0)=0 \\ f^{(1)}(x)=\cos(x)\Rightarrow\ f^{(1)}(0)=\cos(0)=1 \\ f^{(2)}(x)=-\sin(x)\Rightarrow\ f^{(2)}(0)=-\sin(0)=0 \\ f^{(3)}(x)=-\cos(x)\Rightarrow\ f^{(3)}(0)=-\cos(0)=-1 \\ f^{(4)}(x)=\sin(x)\Rightarrow\ f^{(4)}(0)=\sin(0)=0 \\ f^{(5)}(x)=\cos(x)\Rightarrow\ f^{(5)}(0)=\cos(0)=1 \end{array}\right\}\Rightarrow \sin(x)=0+\frac{x}{1}-\frac{x^3}{3!}+\frac{x^5}{5!}\mathrm{L}$$

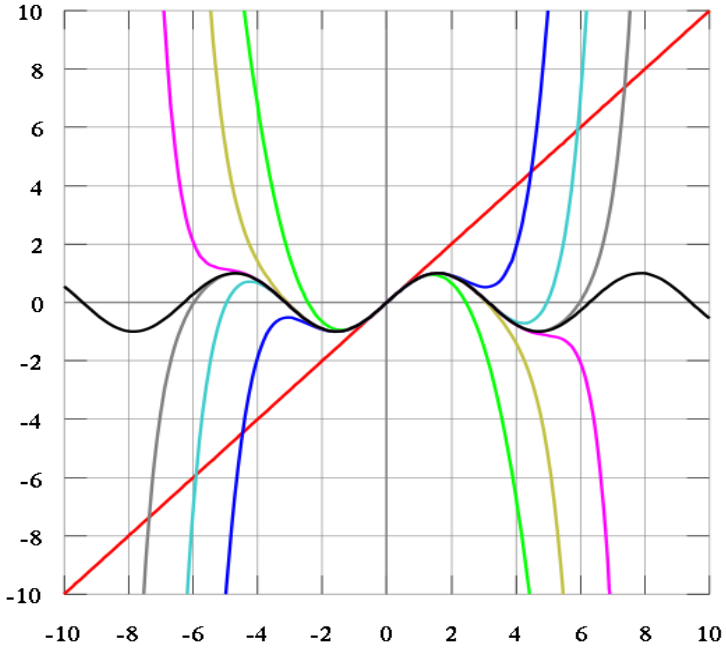
Οι σειρές αυτές μας δίνουν τη δυνατότητα να προσεγγίσουμε συναρτήσεις με πολυώνυμα βαθμού n εάν αποκόψουμε τη σειρά στους n πρώτους όρους.

$$f(x) \approx f(a) + \frac{(x-a)}{1!} f^{(1)}(a) + \frac{(x-a)^2}{2!} f^{(2)}(a) + \mathrm{L} + \frac{(x-a)^n}{n!} f^{(n)}(a)$$

ή

$$f(x) \approx f(0) + \frac{x}{1!} f^{(1)}(0) + \frac{x^2}{2!} f^{(2)}(0) + \mathrm{L} + \frac{x^n}{n!} f^{(n)}(0)$$

Η προσέγγιση της y=sin(x) στο 0 με πολυώνυμα Taylor βαθμού 1,3,5,7,9,11,13 φαίνεται στο παρακάτω σχήμα (πηγή Wikipedia):



Για να προσεγγίσουμε μία συνάρτηση με ανάπτυγμα Taylor ή Maclaurin στο MATLAB θα πρέπει να βασιστούμε στις συμβολικές δυνατότητες του περιβάλλοντος. Αυτό σημαίνει ότι θα πρέπει αρχικά να δηλώσουμε ως συμβολικές, με τη χρήση της εντολής **syms( ),** τις μεταβλητές ή τη συνάρτηση. Η εντολή με τη χρήση της οποίας υπολογίζουμε τα αναπτύγματα είναι η **Taylor( ) ή ισοδύναμα η taylor( )**. Το αποτέλεσμα της Taylor είναι ένα πολυώνυμο και όχι μία άπειρη σειρά. Στον πίνακα που ακολουθεί παρουσιάζεται η σύνταξή της. Όταν η μεταβλητή της συνάρτησης είναι το x, η παράμετρος x στην κλίση της μπορεί να παραληφθεί. Η παράμετρος a καθορίζει το σημείο ως προς το οποίο αναπτύσσουμε. Όταν παραλείπουμε το σημείο ως προς το οποίο θα γίνει το ανάπτυγμα της σειράς τότε θεωρείται ότι αυτό είναι το 0 και το αποτέλεσμα είναι πολυώνυμο Maclaurin. **Το n καθορίζει πόσους όρους θα έχει το ανάπτυγμα, οπότε το αποτέλεσμα θα είναι ένα πολυώνυμο n-1 βαθμού.**

Μαθηματική Έκφραση	**Υλοποίηση στο Matlab**
$\sum_{k=0}^{n-1} (x-a)^k \frac{f^{(k)}(a)}{k!}$	**taylor(f, x,n,a) ή taylor(f,n,a)**
$\sum_{k=0}^{n-1} x^k \frac{f^{(k)}(0)}{k!}$	**taylor(f,x,n) ή taylor(f,n)**
$\sum_{k=0}^{5} x^k \frac{f^{(k)}(0)}{k!}$	**taylor(f,x) ή taylor(f)**

Θα αναπτύξουμε την `sin(x)` με κέντρο το 1 και το 0 διαδοχικά. Θυμηθείτε ότι:

$$\sin(x) = 0 + 1x + 0\frac{x^2}{2!} - 1\frac{x^3}{3!} + 0\frac{x^4}{4!} + \frac{x^5}{5!} + 0\frac{x^6}{6!} - \frac{x^7}{7!} + 0\frac{x^8}{8!} + \frac{x^9}{9!}\text{L} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!}\text{L}$$

```
>> clear all
>> syms x
>> f=inline(sin(x))
>> p=taylor(f(x),x,7)
>> pretty(p)
>> p=taylor(f(x),x,6)
>> pretty(p)
>> p=taylor(f(x),x,8)
>> pretty(p)
>> p=taylor(f(x),x,5,1)
>> pretty(p)
```

p =x-1/6*x^3+1/120*x^5

p =x-1/6*x^3+1/120*x^5

p = x-1/6*x^3+1/120*x^5-1/5040*x^7

p = sin(1) - (sin(1)*(x - 1)^2)/2 + (sin(1)*(x - 1)^4)/24 + cos(1)*(x - 1) - (cos(1)*(x - 1)^3)/6

Εάν δεν καθορίσουμε τον βαθμό του πολυωνύμου το περιβάλλον επιστρέφει το πολυώνυμο πέμπτου βαθμού (δηλαδή πολυώνυμο με 6 όρους).

```
>> pretty(taylor(sin(x)))
```

p =x-1/6*x^3+1/120*x^5

Ένα παράδειγμα συνδυασμού στο MATLAB.

Τα αναπτύγματα Taylor μπορούν να χρησιμοποιηθούν στον υπολογισμό ορίων ή ακόμη και για προσέγγιση τιμών ορισμένων ολοκληρωμάτων συναρτήσεων. Για να δούμε μία τέτοια εφαρμογή, χρησιμοποιώντας το ανάπτυγμα σε σειρά Taylor με κέντρο το 1 (δηλαδή δυνάμεων του $x-1$) για την συνάρτηση $\ln x$ μπορούμε να υπολογίσουμε το όριο $\lim_{x\to 1}\frac{\ln x}{x-1}$. Με αναλυτικό τρόπο στο χαρτί θα κάναμε τις ακόλουθες πράξεις:

$$\lim_{x\to 1}\frac{\ln(x)}{x-1} = \lim_{x\to 1}\frac{(x-1) - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} - \frac{(x-1)^4}{4} + \ldots}{x-1} = \lim_{x\to 1}\left(1 - \frac{(x-1)}{2} + \frac{(x-1)^2}{3} - \frac{(x-1)^3}{4} + \ldots\right) = 1$$

Στο MATLAB τώρα, αρχικά υπολογίζουμε το όριο συμβολικά ώστε να γνωρίζουμε ποια τιμή αναμένουμε να υπολογίσουμε:

```
>> clear all
>> syms x
>> limit(log(x)/(x-1),x,1)
```

Στη συνέχεια το ανάπτυγμα Taylor.

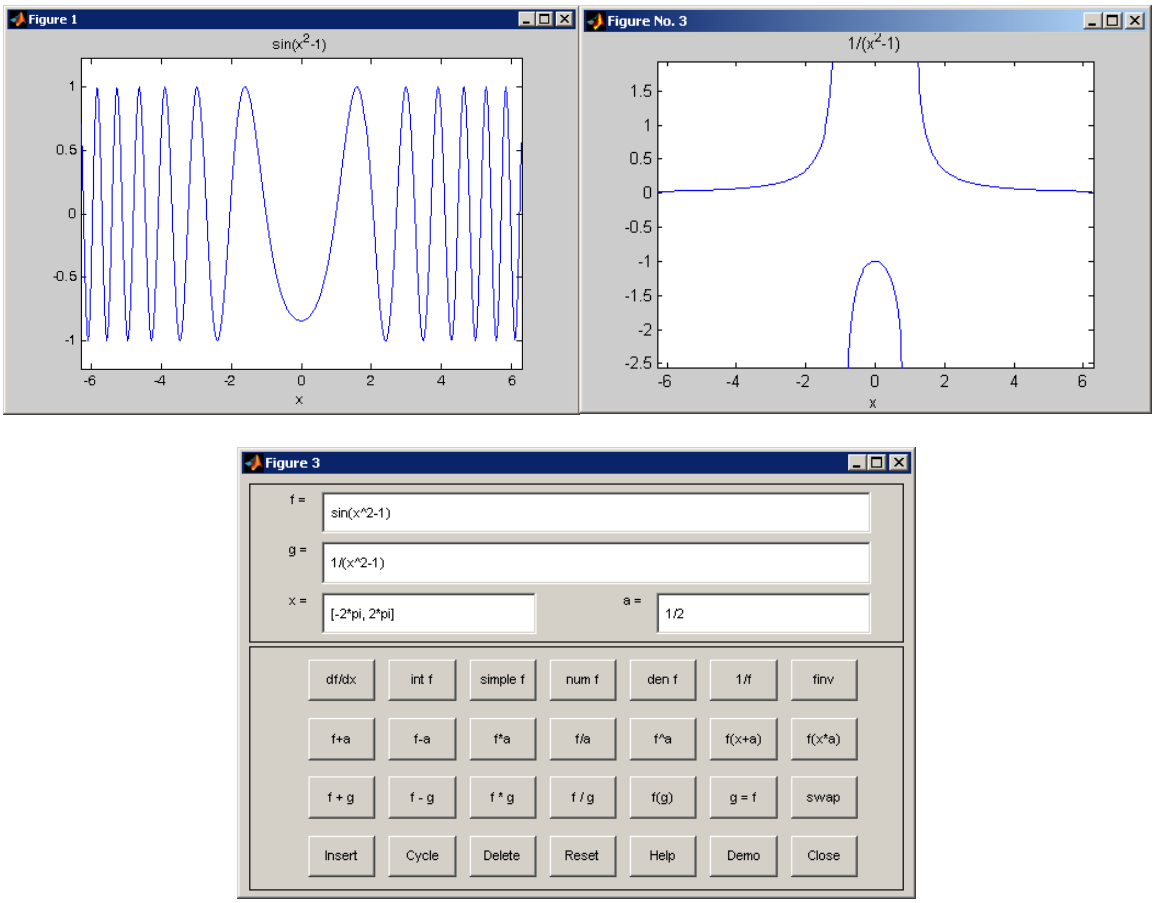
```
>> p=Taylor(log(x),7,1)
>> q=simple(p/(x-1))
>> pretty(q)
```

Η τιμή και το όριο του αναπτύγματος στο x=1 υπολογίζονται με τις εντολές:

```
>> subs(q,x,1)
>> limit(q,x,1)
```

Τα περιβάλλοντα funtool και taylortool

Το Symbolic Math Toolbox που μας δίνει μία ένα απλό γραφικό περιβάλλον, το **funtool**, με το οποίο μπορούμε να διερευνήσουμε τα όσα είδαμε παραπάνω με τη χρήση πλήκτρων. Η χρήση του είναι ιδιαίτερα απλή. Απλά εκτελούμε την εντολή **funtool** και με πολύ απλό και κατανοητό τρόπο μπορούμε να μελετήσουμε, μεταξύ άλλων, τη συμπεριφορά συναρτήσεων, της σύνθεσής τους, των παραγώγων τους, των αντιστρόφων τους και

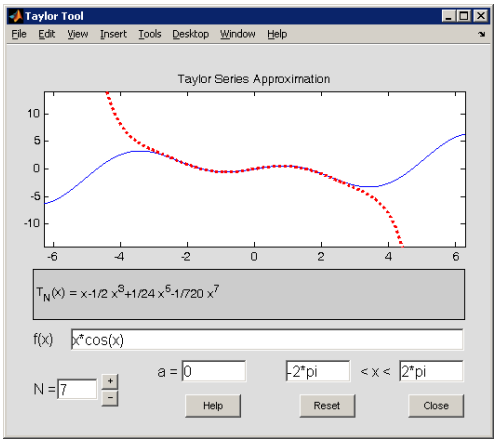


Ανάλογα με το funtool υπάρχει και το taylortool με το οποίο μπορούμε να διερευνήσουμε την προσέγγιση με πολυώνυμα Taylor.

Η εντολή που εκτελούμε είναι η

```
>> taylortool
```

Και το περιβάλλον το ακόλουθο:



Άσκηση για εξάσκηση:

Σας δίνεται η ακόλουθη συνάρτηση:

$$f(x) = \ln\left((x-1)^2\right) - x$$

Α) Καθαρίστε τη μνήμη από τις όποιες τιμές μεταβλητών υπάρχουν ενεργές και το παράθυρο γραφικών. Ορίστε τη μεταβλητή *x* ως συμβολική.

Β) Ορίστε τη συνάρτηση με τη χρήση της inline και κάντε το γράφημά της *f* στο διάστημα *[-3,3]*. Βρείτε τα σημεία τομής της συνάρτησης με τον άξονα *xx'*.

Γ) Εξετάστε εάν το MATLAB μπορεί να υπολογίσει την αντίστροφή της f . Μπορείτε από το γράφημά της f να το αιτιολογήσετε;

Δ) Υπολογίστε την πρώτη και τη δεύτερη παράγωγό της *f* .

Ε) Βρείτε και χαρακτηρίστε τα τοπικά ακρότατά της *f* .

ΣΤ) Υπολογίστε το ολοκλήρωμα $\int_2^5 f(x)dx.$

Ζ) Ορίστε ως συμβολική συνάρτηση *p(x)* το πολυώνυμο Taylor βαθμού 4 της συνάρτησης *f* με κέντρο το σημείο 2. Αναπτύξτε το σε δυνάμεις του x. Βρείτε τις ρίζες του και επαληθεύστε ότι αυτές ικανοποιούν την *p(x)=0*.

Περισσότερα για τα γραφήματα

Η πιο απλή περίπτωση γραφικής αναπαράστασης των στοιχείων ενός διανύσματος είναι η ***plot(y)*** όπου τα στοιχεία του διανύσματος **y** σχεδιάζονται σε συνδυασμό με τους δείκτες των στοιχείων του.

```
>> y=[1.2,2.4, 3.2, 4.0, 5.6, 6.3, 7.1, 8.4];
>> plot(y)
```

Όπως έχουμε ήδη δει οι επόμενες εντολές σχεδιάζουν τα ζεύγη τιμών δύο διανυσμάτων.

```
>> x=[1.3, 2.5, 3.2, 4.1, 5.3, 6.2, 7.4, 8.2, 9.4];
>> y=[2.3, 3.7, 4.5, 5.3, 7.6, 5.5, 4.2, 3.4, 2.1];
>> plot(x,y)
```

Ο εξ ορισμού τρόπος σχεδιασμού στο MATLAB μπορεί να αλλάξει αν την εντολή ***plot(x, y)***, την αντικαταστήσουμε με την ***plot(x, y, string)***, όπου η ***string*** συνδυάζει μέχρι τρία στοιχεία τα οποία ελέγχουν το χρώμα, το σύμβολο και τον τύπο της γραμμής σχεδιασμού. Για παράδειγμα, η εντολή ***plot(x,y,'r*--')*** προσδιορίζει ότι σε κάθε σημείο $x(i), y(i)$ θα τοποθετηθεί ένα κόκκινο αστέρι και ότι τα σημεία θα συνδέονται με μία διακεκομμένη γραμμή. Η εντολή ***plot(x,y,'b+')*** προσδιορίζει ότι ο σχεδιασμός των σημείων $x(i), y(i)$ θα γίνει με την χρήση ενός σταυρού χρώματος μπλε χωρίς να συνδέονται με κάποια γραμμή. Οι βασικές επιλογές για το σύμβολο, το είδος της γραμμής και το χρώμα, οι οποίες μπορούν να χρησιμοποιηθούν με την εντολή ***plot*** περιέχονται στους παρακάτω πίνακες.

Σύμβολο	**Χρώμα γραμμής**	**Είδος γραμμής**
(o) Κύκλος	(**r**) Κόκκινο	(-) Συμπαγής γραμμή
(*) Αστέρι	(**g**) Πράσινο	(--) Διακεκομμένη γραμμή
(×) Σταυρός	(**b**) Μπλε	(**:**) *Dotted* γραμμή
(+) Συν	(**c**) Κυανό	(**-.**) *Dash-dot* γραμμή
(*s*) Τετράγωνο	(**m**) Μπορντό	
(*d*) Διαμάντι	(**y**) Κίτρινο	
(>) Δεξιά γωνία	(**k**) Μαύρο	
(<) Αριστερή γωνία	(**w**) Λευκό	

```
>> figure
>> plot(x,y,'r*--')
>> figure
>> plot(x,y,'gs-.')
>> b=[1.2, 2.0, 3.2, 4.1, 5.2, 5.8, 8.3];
>> c=[0.8, 1.5, 2.4, 3.6, 4.3, 5.4, 7.2];
>> figure
>> plot(x,y,'gs--',b,c,'rd-.')
```

Με την εντολή **figure** ανοίγουμε ένα νέο παράθυρο γραφικών όπου θα εμφανιστεί το γράφημα της επόμενης εντολής σχεδίασης. Με την εντολή **clf** καθαρίζουμε από το περιεχόμενό του το τελευταίο παράθυρο γραφικών.

```
>> clf
```

Με την εντολή **close all** κλείνουμε όλα τα ανοικτά παράθυρα γραφημάτων.

```
>> close all
```

Οι επιλογές ***LineWidth*** και ***MarkerSize*** μεταβάλουν το πάχος της γραμμής και του συμβόλου. Οι τιμές που επιλέγουμε είναι μετρώνται σε στιγμές (1 στιγμή αντιστοιχεί στο 1/72 της ίντσας).

```
>> plot(x,y,'ms--','LineWidth',5,'MarkerSize',10)
```

Αν μία εντολή σχεδίασης εκτελεστεί και στην συνέχεια ακολουθήσει μία άλλη το νέο γράφημα είτε θα αντικαταστήσει το γράφημα της προηγούμενης, ή θα επικαθίσει στο παλιό γράφημα. Το τι θα γίνει εξαρτάται από την κατάσταση στην οποία βρίσκεται η συνάρτηση ***hold***. Πληκτρολογώντας ***hold on*** προκαλούμε τα επόμενα γραφήματα να επικάθονται στο τρέχον, ενώ η εντολή ***hold off*** ορίζει ότι κάθε νέο γράφημα αντικαθιστά το τρέχον. Η εξ' ορισμού κατάσταση αντιστοιχεί στην εντολή ***hold off***.

```
>> close all
>> plot(x,y,'r*--')
>> hold on
```

Δείτε το γράφημα που εμφανίστηκε.

```
>> plot(b,c,'rd-.')
>> hold off
```

Δείτε το γράφημα ξανά.

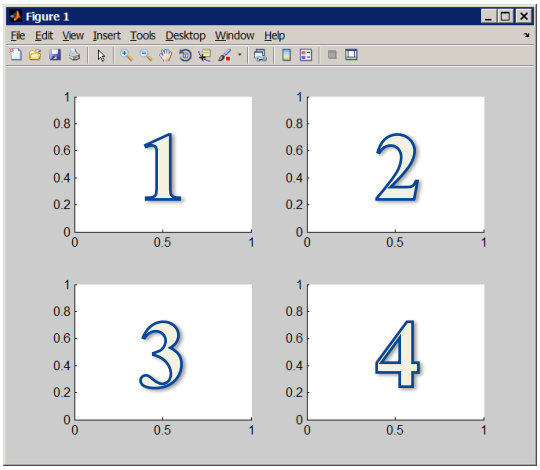
Το MATLAB μας δίνει την δυνατότητα να προσθέσουμε πλέγμα σε κάποιο γράφημα. Η εντολή για την εισαγωγή είναι ***grid on*** ή απλά ***grid*** και συνήθως ακολουθεί την εντολή *plot*. Αν θέλουμε να απενεργοποιήσουμε το πλέγμα χρησιμοποιούμε την εντολή ***grid off***.

```
>> x=(0:0.25:5*pi)';
>> y=sin(x);
>> z=cos(x);
>> plot(x,y, '--',x,z,':')
>> grid on
```

Οι συναρτήσεις ***title***, ***xlabel***, ***ylabel*** μπορούν να χρησιμοποιηθούν για να τοποθετήσουμε σε ένα γράφημα τίτλο στο επάνω μέρος και συμβολισμό στους άξονες. Η συνάρτηση ***box off*** μετακινεί το τετράγωνο περίγραμμα από το γράφημα και αφήνει μόνο τους άξονες. Το επόμενο παράδειγμα είναι ενδεικτικό.

```
>> box off
>> xlabel('x')
>> ylabel('f(x)')
>> title('Plot of sin(x) and cos(y)')
>> grid off
```

Τέλος, η εντολή ***subplot*** μας επιτρέπει να τοποθετήσουμε περισσότερα από ένα γραφήματα σε μορφή πλέγματος στο ίδιο παράθυρο γραφικών. Το συντακτικό της εντολής γράφεται με την μορφή ***subplot(m, n, p)*** και χωρίζει το παράθυρο σε έναν $m \times n$ πίνακα χωρίων γραφικών, έχοντας κάθε ένα τους δικούς του άξονες. Τα γραφήματα αριθμούνται από αριστερά προς τα δεξιά σε κάθε γραμμή και η αρίθμηση συνεχίζει στην από κάτω γραμμή του πλέγματος. Για παράδειγμά η ***subplot(2, 2, 1), subplot(2, 2, 2),…*** τοποθετεί το επόμενο γράφημα που δημιουργεί η plot στο $1^{ο}$, $2^{ο}$,…. από τα τέσσερα γραφήματα μίας διάταξης 2×2 γραφημάτων, όπως φαίνεται στην αρίθμηση παρακάτω.



Στο ακόλουθο παράδειγμα δημιουργούμε ένα πίνακα χωρίων γραφικών με μία γραμμή και δύο στήλες.

```
>> clf
>> clear all
>> x=(0:0.25:5*pi)';
>> y=sin(x);
>> z=cos(x);
>> subplot(1,2,1)
>> plot(x,y, '--')
>> xlabel('x')
>> ylabel('f(x)')
>> title('Plot of sin(x)')
>> subplot(1,2,2)
>> plot(x,z,':')
>> xlabel('x')
>> ylabel('f(x)')
>> title('Plot of cos(y)')
```

Προγραμματισμός στο MATLAB

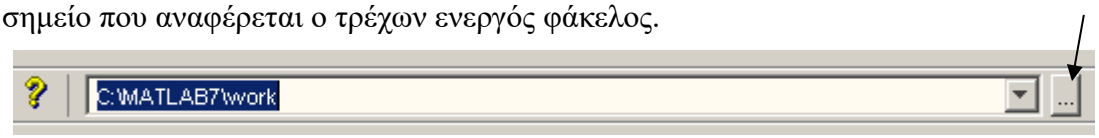
Αρχεία εντολών Scripts

Τα scripts είναι αρχεία κειμένου που περιέχουν εντολές του MATLAB. Το όνομα τους πρέπει να έχει επέκταση .m (π.χ. name.m) το οποίο δεν πρέπει να είναι όνομα εντολής ή άλλου αρχείου. Εκτελούνται είτε μέσω του παραθυρικού περιβάλλοντος του editor του MATLAB (κατάλογος επιλογών Debug -> Run) είτε από το prompt πληκτρολογώντας απλά το όνομα τους (π.χ. name) ακολουθούμενο από enter. Το .m αρχείο θα πρέπει να βρίσκεται στον ενεργό φάκελο. Με την κλήση του script εκτελούνται οι εντολές που περιέχει. Τα scripts είναι κατάλληλα για επίλυση προβλημάτων που απαιτούν μεγάλο αριθμό εντολών. Για να δημιουργήσουμε ένα .m αρχείο μπορούμε να επιλέξουμε File->New-> M-File. Η επιλογή αυτή ανοίγει τον συντάκτη (editor) του MATLAB όπου μπορούμε να πληκτρολογήσουμε τις όποιες εντολές. Εντολές που ξεκινούν από το σύμβολο % θεωρούνται **σχόλια**.

Για να κατασκευάσετε ένα script που θα εμφανίζει το γράφημα της sin(x)*exp(-x) στο [0, 2π] ακολουθείστε τα παρακάτω βήματα.

Βήμα 1

Με τη βοήθεια του My Computer δημιουργείστε έναν δικό σας φάκελο μέσα στον φάκελο που θα ορίσει ο διδάσκοντας σας. Στη συνέχεια κάντε αυτόν το φάκελο ενεργό. Για να το πετύχετε αυτό πλοηγηθείτε και βρείτε τον συγκεκριμένο φάκελο επιλέγοντας το πλήκτρο που βρίσκεται στο παράθυρο του MATLAB δίπλα στη σημείο που αναφέρεται ο τρέχων ενεργός φάκελος.



Τέλος, δημιουργείστε ένα αρχείο και αποθηκεύστε το στο φάκελό σας με το όνομα plot1.m και περιεχόμενα :

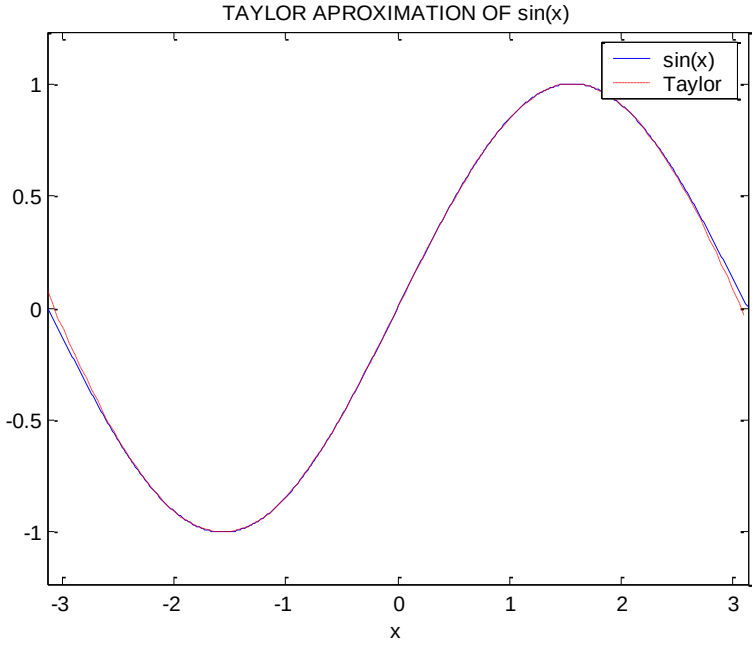
```
% SXOLIA OTI GRAFETAI  EDW DEN EKTELEITAI
% GRAFIMA THS sin(x) KAI TOY POLYWNYMOU TAYLOR
% 7ou VATHMOU (ME OKTO OROUS) STO [-pi, pi].
clear all;
clf;
syms x
f = sin(x);
p = taylor(f,8);
xd = -pi:0.05:pi; yd = subs(p,x,xd);
ezplot(f, [-pi,pi]); hold on;
plot(xd, yd, 'r-.')
title('TAYLOR APROXIMATION OF sin(x)');
legend('sin(x)','Taylor')
```

Βήμα 2

Στο prompt πληκτρολογείτε

```
>> plot1
```

Οι εντολές που βρίσκονται μέσα στο script εκτελούνται η μία μετά την άλλη και τελικά παίρνουμε το ακόλουθο γράφημα:



Η εντολή **legent( )** δημιουργεί τη λεζάντα στην πάνω δεξιά γωνία του γραφήματος.

Γραφήματα σε λογαριθμική κλίμακα

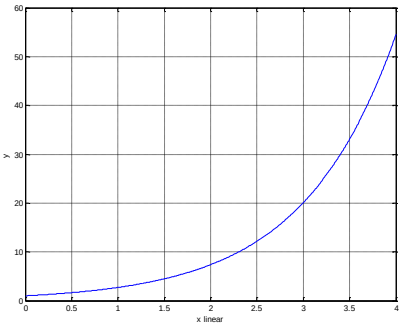
Πολλές φορές είναι χρήσιμο να κάνουμε γραφικές παραστάσεις με λογαριθμική κλίμακα στον ένα ή και στους δύο άξονες. Οι αντίστοιχες συναρτήσεις είναι της μορφής ***semilogx***, ***semilogy***, και ***loglog*** αντίστοιχα, όπου **x** και **y** διανύσματα, οι οποίες αντικαθιστούν την συνάρτηση *plot*.

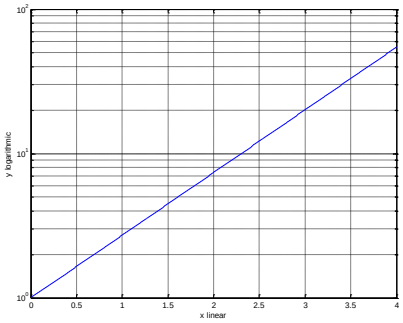
Με τις παρακάτω εντολές γίνεται αρχικά το γράφημα του διανύσματος **y** με κανονική κλίμακα και στη συνέχεια γίνεται το γράφημα με μία $\log_{10}$ κλίμακα και το **x** με μία γραμμική κλίμακα. Ίσες αυξήσεις κατά μήκος του άξονα **y** παριστάνουν πολλαπλάσια δυνάμεων του 10. Έτσι το γράφημα της e^x στους άξονες αυτούς είναι μία ευθεία γραμμή, επειδή η εξίσωση $y = e^x$ μετατρέπεται σε γραμμική αν πάρουμε λογάριθμους και στα δύο μέρη.

$$y = e^x \Leftrightarrow \log_{10} y = x \log_{10} e$$

Δημιουργείστε ένα αρχείο και αποθηκεύστε το στο φάκελό σας με το όνομα logplot.m και περιεχόμενα :

```
close all
clear all
clf
x=0:0.01:4;
plot(x,exp(x))
xlabel('x linear'), grid
ylabel('y')
figure
semilogy(x, exp(x)), grid
xlabel('x linear')
ylabel('y logarithmic')
```





Στο prompt πληκτρολογείτε

```
>> logplot
```

Δομές Επανάληψης και Επιλογής στο Matlab

Λογικές Εκφράσεις

Λογική έκφραση είναι μία έκφραση η οποία μπορεί να πάρει τιμή True ή False. Το MATLAB παριστάνει τις τιμές **true και false με τους αριθμούς 1 και 0 αντίστοιχα**. Μία τέτοια έκφραση πρώτα από όλα μπορεί να είναι μία απλή λογική συνθήκη με την οποία συγκρίνουμε, με τη χρήση ενός τελεστή σύγκρισης δύο ποσότητες ή εκφράσεις και αποτελείται από τρία μέρη :

σκέλος Α τελεστής σύγκρισης σκέλος Β

Οι κυριότεροι τελεστές σύγκρισης είναι οι ακόλουθοι:

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

Στους τελεστές που αποτελούνται από περισσότερα από ένα σύμβολα δεν μπαίνει ανάμεσα τους κενός χαρακτήρας.

Υποθέτουμε ότι στην απλή μεταβλητή x έχει εκχωρηθεί μία τιμή π.χ. x=2. Τότε μπορούμε να κάνουμε βρούμε τις τιμές για διάφορες λογικές έκφραση σ' αυτή σύμφωνα με τις παρακάτω εντολές:

```
>> clear all;
>> x=2;
>> x==2, x~=2, x>2, x<2, x>=2, x<=2
```

Μία λογική έκφραση μπορεί ακόμη να είναι ένας λογικός έλεγχος. Μερικοί από τους λογικούς ελέγχους είναι οι ακόλουθοι.

Ελεγχος	**Τιμή True αν :**
isequal(a,b)	Οι πίνακες ή τα διανύσματα ίσα
isempty(a)	Ο πίνακας δεν έχει στοιχεία
isnan(a)	Το όρισμα είναι NaN
isinf(a)	Το όρισμα είναι Inf

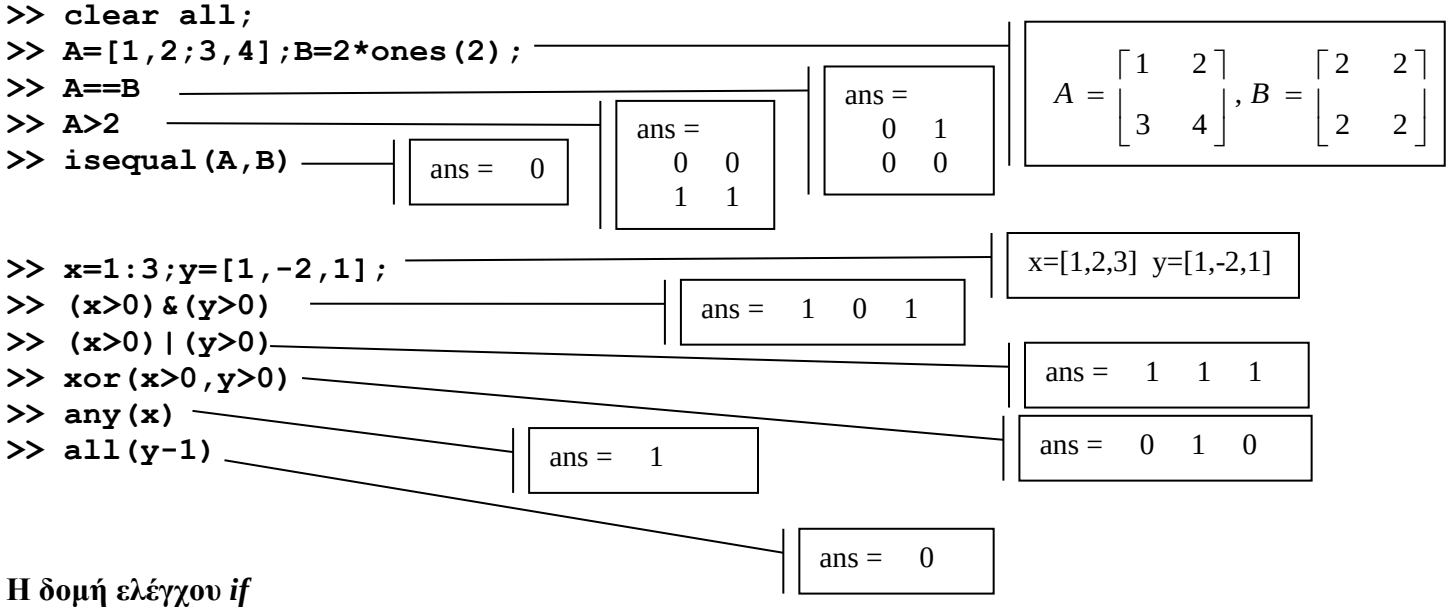
Μία λογική έκφραση μπορεί τέλος να είναι μία σύνθετη λογική συνθήκη. Οι σύνθετες λογική συνθήκες είναι λογικές εκφράσεις που αποτελούνται από απλές συνθήκες και λογικές πράξεις που μπορούμε να εφαρμόσουμε σε αυτές. Τις χρησιμοποιούμε όταν θέλουμε να εκφράσουμε σύνθετες λογικές προτάσεις. Δηλαδή, χρησιμοποιούμε τις σύνθετες λογικές συνθήκες όταν η λογική πρόταση που θέλουμε να εκφράσουμε δεν μπορεί να γραφεί ως μια απλή σύγκριση.

Οι λογικές πράξεις τελούνται όταν ένας λογικός τελεστής δρα μεταξύ δύο λογικών εκφράσεων (ή πάνω σε μία για τη λογική άρνηση) Οι κυριότεροι λογικοί τελεστές είναι οι ακόλουθοι:

&	Λογικός τελεστής **και**
|	Λογικός τελεστής **είτε**
~	Λογικός τελεστής **άρνησης**
xor( )	Λογικός τελεστής **Αποκλειστικής Διάζευξης**
all	**Αληθής** αν όλα τα στοιχεία διανύσματος είναι **μη μηδενικά**
any	**Αληθής** αν κάποιο από τα στοιχεία διανύσματος είναι **μη μηδενικό**

Αν και υπάρχει καθορισμένη προτεραιότητα στους λογικούς τελεστές καλό είναι την προτεραιότητα των πράξεων να την καθορίζουμε εμείς με παρενθέσεις.

Οι λογικές συγκρίσεις ανάμεσα σε πίνακες (διανύσματα) ιδίου μεγέθους επιστρέφουν έναν λογικό πίνακα (διάνυσμα) με στοιχεία το αποτέλεσμα της ανά στοιχείο σύγκρισης. Ένας τέτοιος πίνακας έχει τιμή αληθή όταν όλα τα στοιχεία του είναι μονάδες. Στο ακόλουθο παράδειγμα εφαρμόζουμε τους τελεστές συσχέτισης σε πίνακες και διανύσματα, και τα αντίστοιχα αποτελέσματα είναι πίνακες με στοιχεία το αποτέλεσμα της συσχέτισης κάθε στοιχείου του πίνακα ή του διανύσματος. Στο τέλος βλέπουν την εφαρμογή των any και all.



Η δομή ελέγχου *if*

Η πιο απλή εκδοχή της εντολής *if* είναι της μορφής:

***if* { σχέση }**
{ εντολές }
end.

Μπορούμε να δημιουργήσουμε περισσότερο σύνθετες δομές επιλογής με τη χρήση της σύνθετης εντολής *if*:

if **{ *σχέση* }**
{ *εντολές*}
elseif **{ *σχέση* }**
{ *εντολές* }
else
{ *εντολές*}
end

Ως εφαρμογή θα δημιουργήσουμε ένα script το οποίο ανάλογα με την επιλογή μας θα εμφανίζει κάποιο σχετικό μήνυμα. Πληκτρολογήστε τις παρακάτω εντολές σε ένα script με όνομα **testif.m**, αποθηκεύστε το

στον ενεργό φάκελό σας και εκτελέστε το. Εάν σας εμφανιστούν σφάλματα κατά την εκτέλεσή του βρείτε τα και διορθώστε τα. Εκτελέστε το script τρεις φορές δίνοντας τιμές 1,2 και 3 στη μεταβλητή m.

Σε αυτό το κείμενο χρησιμοποιούμε και κάποιες από τις διαθέσιμες εντολές εισόδου εξόδου του Matlab. Η εντολή **input( )** είναι μία εντολή εισόδου με την οποία αποδίδουμε τιμές με το πληκτρολόγιο σε μεταβλητές κατά την εκτέλεσή της. Το κείμενο που γράφουμε σε εισαγωγικά εμφανίζεται ώστε να γνωρίζουμε τι περιμένει ο υπολογιστής να πληκτρολογήσουμε. Με την εντολή εξόδου **disp( )** εμφανίζουμε κείμενα ή περιεχόμενα μεταβλητών στην οθόνη. Για μια πιο ενδιαφέρουσα παρουσίαση των αποτελεσμάτων στην οθόνη μπορούμε να εφαρμόσουμε την εντολή fprintf η οποία ομοιάζει με την αντίστοιχη της γλώσσας C. Η σύνταξή της εντολής αυτής έχει την μορφή: fprintf (φόρμα εκτύπωσης, λίστα εκφράσεων). Γράφοντας την εντολή: fprintf('%5.2f \n',sqrt(3)) ο χαρακτήρας % σηματοδοτεί την έναρξη ενός τύπου εκτύπωσης κρατώντας ένα πεδίο 5 θέσεων εκ των οποίων οι τρεις θέσεις είναι μετά την υποδιαστολή. Το \n εξαναγκάζει την επόμενη εμφάνιση στην οθόνη να γίνει σε μια γραμμή παρακάτω. Ο Η εντολή **mod(a,2)** επιστρέφει το υπόλοιπο της ακεραίας διαίρεσης του a με το 2. Εάν το αποτέλεσμα είναι 0 τότε ο αριθμός είναι άρτιος, εάν είναι 1 τότε ο αριθμός είναι περιττός σε διαφορετική περίπτωση ο αριθμός δεν είναι ακέραιος.

```
m=input('Dwste enan arithmo ');
if mod(m,2) == 1,
  fprintf('H epilogh sas einai %5.2f \n pou einai perittos',m)
elseif mod(m,2) == 0,
     fprintf('H epilogh sas einai %5.2f \n pou einai artios',m)
else
    disp('Den dwsate akeraio arithmo')
end
```

Ο βρόγχος επανάληψης for

Μία είναι η εντολή for η οποία εκτελεί διαδοχικά τις εντολές του βρόγχου για κάθε τιμή του διανύσματος, με σύνταξη:

Στο Matlab η σύνταξη της εντολής for είναι:

for* { *μεταβλητή* } = { "*μετρητής*" (*διάνυσμα-γραμμή*) }, { *εντολές* } *end

Ακολουθούν δύο χαρακτηριστικά ισοδύναμα παραδείγματα τα οποία απλά εμφανίζουντις κυβικές δυνάμεις της τιμής που λαμβάνει σε κάθε επανάληψη ο μετρητής :

1ος τρόπος:

```
>> for k=[1 3 5 7]
         disp(k^3)
   end
```

2ος τρόπος:

```
>> for k=1:2:7
      disp(k^3)
   end
```

Ο βρόγχος επανάληψης while

Μία άλλη εντολή επανάληψης είναι η εντολή **while.** Παρόμοια με τις άλλες γλώσσες προγραμματισμού, η συγκεκριμένη εντολή εκτελεί διαδοχικά τις εντολές του βρόγχου επανάληψης όσο ισχύει η συνθήκη επανάληψης. Η σύνταξή της είναι η ακόλουθη:

while <συνθήκη>

Εντολές

end

Είναι γνωστό ότι $1+2+3+\cdots+\nu=\frac{\nu(\nu+1)}{2}$. Εάν θέλουμε να υπολογίσουμε, με τη χρήση της εντολής while, το άθροισμα των αριθμών από 1 έως το 100000 (το αποτέλεσμα σύμφωνα με τον τύπο ισούται με $\frac{100000(100000+1)}{2}=50005000$) πληκτρολογούμε:

```
>> num=0;athroisma=0;
>> while num<10000,
     num= num+1;
     athroisma = athroisma +num;
   end
>> disp(num),disp(athroisma)
```

Αποδοτικό MATLAB

Η χρήση παραδοσιακών προγραμματιστών τεχνικών στο MATLAB δεν είναι πάντα η αποδοτικότερη επιλογή. Το περιβάλλον του MATLAB έχει τη δυνατότητα να χειρίζεται διανύσματα και πίνακες με έναν ιδιαίτερα ταχύ τρόπο. Ο προγραμματισμός με τη χρήση της εντολής for, για παράδειγμα, σε μερικές περιπτώσεις θεωρείται μη αποδοτικός. Ως παράδειγμα θα υπολογίσουμε το άθροισμα των τετραγώνων των στοιχείων ($\sum_i x_i^2$) ενός διανύσματος με 500000 τυχαία αριθμούς. Πρώτα με τη χρήση της for:

```
>> clear all
>> n=5e5;x=rand(n,1);
>> tic;s=0;for i=1:n,s=s+x(i)^2;end;toc
```

Στη συνέχεια με τη χρήση διανυσματικών πράξεων και συναρτήσεων του MATLAB.

```
>> clear all
>> n=5e5;x=rand(n,1);
>> tic;s1=sum(x.^2);toc
```

Με τις **tic** και **toc** μετράμε το χρόνο που χρειάζεται σε κάθε περίπτωση το MATLAB. Η πρώτη ενεργοποιεί το χρονόμετρο και η δεύτερη το τερματίζει επιστρέφοντας τον χρόνο που κατέγραψε από την κλίση της tic. Η διαφορά των χρόνων υπολογισμού φανερώνει ότι οι παραδοσιακές τεχνικές δεν είναι πάντα αποδοτικές στο MATLAB. Θυμίζουμε ότι 5e5=5×10^5.

Αρχεία συναρτήσεων functions

Οι συναρτήσεις στο MATLAB είναι m-αρχεία με πρώτη γραμμή:

function [y1, y2,...]=fname(x1, x2,...)

Τα **y1, y2,..** είναι οι **τυπικές παράμετροι** εξόδου.

Τα **x1, x2,...** είναι οι **τυπικές παράμετροι** εισόδου.

fname είναι το όνομα της συνάρτησης και fname.m το όνομα του αρχείου στο οποίο είναι αποθηκευμένη η function.

Η εκτέλεση μίας εντολής με μορφή

[z1, z2,...]= fname(w1, w2,...)

Καλεί τη συνάρτηση με **πραγματικές παραμέτρους** εισόδου w1,w2,…που αντιστοιχούν στις x1,x2,…και αποδίδει τις **πραγματικές παραμέτρους** εξόδου z1,z2,… που αντιστοιχούν στις y1, y2,...

Ισοδύναμη κλήση είναι η ακόλουθη:

[z1, z2,...]= feval('fname', w1, w2,...)

Για τις function ισχύει:

- Όταν η τυπική παράμετρος εξόδου είναι μία δεν χρειάζονται οι αγκύλες.
- Όταν οι πραγματικές παράμετροι εξόδου είναι μία ή καμία (ή λιγότερες από τις τυπικές) από τότε η κλήση επιστέφει τιμή στην πρώτη παράμετρο εξόδου (ή σε αντίστοιχο αριθμό τυπικών).
- **Σχόλια** εισάγονται μετά το σύμβολο %.
- Με την εντολή **help fname** εμφανίζονται τα πρώτα σχόλια που έχουμε εισάγει στο fname.m.
- Οι μεταβλητές που εμφανίζονται μέσα στη συνάρτηση είναι **τοπικές**.
- Το όνομα της συνάρτησης και το όνομα του αρχείου πρέπει να είναι το **ίδιο**.
- Κάθε συνάρτηση μπορεί να καλεί άλλες συναρτήσεις ή τον εαυτό της.
- Οι τυπικές παράμετροι εξόδου θα πρέπει να παίρνουν κάποια τιμή μέσα στις εντολές της συνάρτησης διαφορετικά επιστρέφουν κενές τιμές.

Όλες σχεδόν οι εντολές του MATLAB είναι m-αρχεία.
Για παράδειγμα η **linspace( )** οποία είναι μία **εσωτερική εντολή του MATLAB την οποία μπορούμε να καλέσουμε όποτε θέλουμε.** Με αυτήν μπορούμε να ορίσουμε μία διαμέριση n σημείων σε ένα διάστημα. Δηλαδή η συνάρτηση αυτή μας χωρίζει το διάστημα σε n-1 ίσου πλάτους διαστήματα. Η **linspace( )** είναι μία function με περιεχόμενο:

```
function y = linspace(d1, d2, n)
%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.
%   See also LOGSPACE, :.
%   Copyright 1984-2000 The MathWorks, Inc.
%   $Revision: 5.10 $  $Date: 2000/06/01 16:46:39 $
if nargin == 2
    n = 100;
end
y = [d1+(0:n-2)*(d2-d1)/(n-1) d2];
```

Μπορούμε να πληροφορηθούμε για το τι κάνει με την εντολή

```
>>help linspace
```

Η linspace έχει τρεις τυπικές παραμέτρους εισόδου. Όταν δώσουμε ως όρισμα τα άκρα ενός διαστήματος [d1,d2] και τον αριθμό των σημείων της διαμέρισης μας επιστρέφει ένα διάνυσμα με στοιχεία τα σημεία της διαμέρισης. Η ακόλουθη κλίση θα χωρίσει το διάστημα [0,1] σε 10 ίσου πλάτους διαστήματα και θα επιστρέψει ένα διάνυσμα με τα 11 άκρα αυτών των διαστημάτων.

```
>>y=linspace(0,1,11)
```

Με την κλήση μίας συνάρτησης μία μεταβλητή με όνομα **nargin** λαμβάνει ως τιμή το πλήθος των πραγματικών παραμέτρων εισόδου και μία άλλη με όνομα **nargout** το πλήθος των πραγματικών παραμέτρων εισόδου με το οποίο έγινε η κλήση της συνάρτησης. Με τη χρήση τους μπορούμε να ελέγξουμε κλήσεις οι οποίες γίνονται με μικρότερο αριθμό πραγματικών παραμέτρων εισόδου και εξόδου. Για παράδειγμα εάν εκτελέσουμε

```
>>y=linspace(0,1)
```

η μεταβλητή nargin παίρνει την τιμή 2. Στον κώδικα της συνάρτησης υπάρχει η εντολή

```
if nargin == 2
    n = 100;
end
```

η οποία θέτει τον αριθμό των σημείων της διαμέρισης n ίσο με 100. Η παραπάνω κλίση θα χωρίσει το διάστημα [0,1] σε 99 ίσου πλάτους διαστήματα και θα επιστρέψει ένα διάνυσμα με τα 100 άκρα αυτών των διαστημάτων. (περισσότερα για την if θα δούμε στο μέλλον).

Ανώνυμες functions

Μια ανώνυμη function είναι μια function που δεν είναι αποθηκευμένη σε ένα αρχείο συνάρτησης, αλλά σχετίζεται με μια μεταβλητή της οποίας ο τύπος δεδομένων είναι **function_handle**. Οι ανώνυμες function μπορούν να δέχονται μεταβλητές εισόδου και να επιστρέφουν τιμές σε μία μεταβλητή εξόδου, όπως κάνουν και οι τυπικές functions (στις οποίες όμως μπορούμε να έχουμε περισσότερες από μία εξόδους) . Ωστόσο, μπορούν να περιέχουν μόνο μία εκτελέσιμη εντολή.

Η δήλωση

```
>> myfunction = @(x,y) (x^2 + y^2 + x*y);
```

Δημιουργεί τη συνάρτηση και η κλήση

```
>> a=myfunction(1,2)
```

a =7

Την καλεί και δίνει αποτέλεσμα 7. Οι ανώνυμες functions μπορούν να είναι παράμετροι σε κλήση άλλων funtions μέσα στις οποίες μπορούν να κληθούν να εκτελεστούν με τη βοήθεια της εντολής **feval**.

Για παράδειγμα εάν έχω τη function test

```
function y=test(f,x,y);
y=feval(f,x,y);
```

η κλήση της είναι:

```
>> b=test(myfunction,1,2)
```

b =7

Μία εφαρμογή στους πίνακες υλοποιούμενη με Scripts και functions

Όπως είδαμε στη θεωρία της Γραμμικής Άλγεβρας, εάν πολλαπλασιάσουμε τον πίνακα

$$Q_\vartheta = \begin{bmatrix} \cos\vartheta & -\sin\vartheta \\ \sin\vartheta & \cos\vartheta \end{bmatrix}$$

με το διάνυσμα των συντεταγμένων $\begin{bmatrix} x \\ y \end{bmatrix}$ ενός σημείου του καρτεσιανού επιπέδου τότε παίρνουμε τις συντεταγμένες του σημείου που προκύπτει από την αριστερόστροφη στροφή γύρω από την αρχή των αξόνων (0,0) κατά γωνία ϑ .

Αρχικά δημιουργούμε ένα αρχείο με όνομα **qth.m** με περιεχόμενο μία συνάρτηση **function** με όνομα **qth** η οποία επιστρέφει τον πίνακα αριστερόστροφης στροφής γύρω από την αρχή των αξόνων (0,0) κατά γωνία ϑ .

```
function y=Qth(theta);
y=[cos(theta),-sin(theta);sin(theta),cos(theta)];
```

Αυτό το function θα μπορούμε να καλούμε από ένα script που θα έχει ως περιεχόμενο:

```
clear all
clf
xy1=[1;1];
for i=1:4, xy1(:,i+1)=Qth(pi/2)*xy1(:,i);end
xy2=[1;1];
for i=1:6, xy2(:,i+1)=Qth(pi/3)*xy2(:,i);end
plot(xy1(1,:),xy1(2,:), 's-',xy2(1,:),xy2(2,:), 'd-')
title('Strofh shmeiwn me function');
```

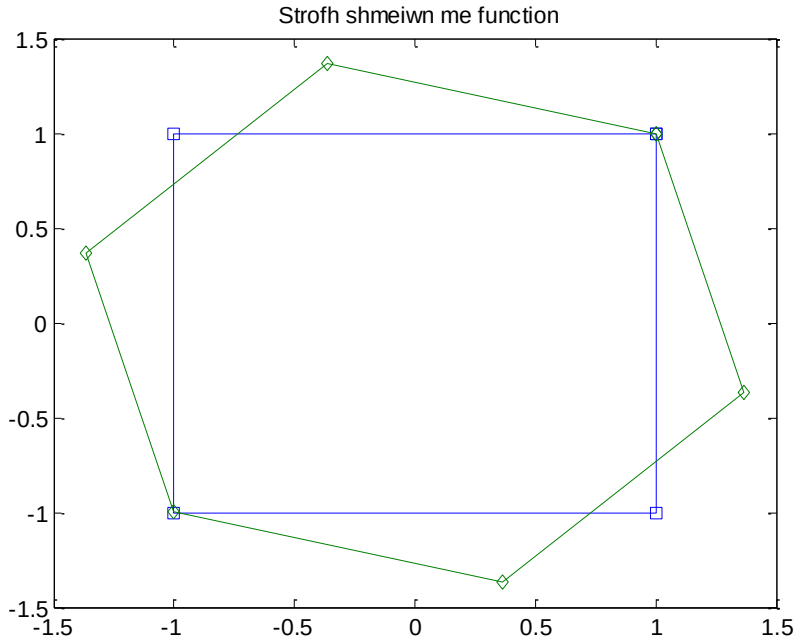
(Πληκτρολογήστε τις παραπάνω εντολές σε ένα script με όνομα **strofi1.m** αποθηκεύστε το στον ενεργό φάκελό σας και εκτελέστε το. Εάν σας εμφανιστούν σφάλματα κατά την εκτέλεσή του βρείτε τα και διορθώστε τα.**)**

Στον πρώτο βρόγχο επανάληψης χρησιμοποιώντας πίνακα στροφής κατά $\pi/2$ ξεκινάμε από το σημείο (1,1) εκτελούμε 4 διαδοχικές στροφές. Είναι αναμενόμενο είναι το τελευταίο σημείο να είναι πάλι το (1,1). Το διάνυσμα xy1 τελικά έχει δύο γραμμές. Η πρώτη έχει τις τετμημένες των σημείων που προκύπτουν από τη διαδικασία και η δεύτερη τις τεταγμένες τους.

Το ίδιο κάνουμε και με πίνακα στροφής κατά $\pi/3$ εκτελούμε 6 διαδοχικές στροφές και αποθηκεύουμε τα σημεία που προκύπτουν στο διάνυσμα xy2 με ανάλογο με το διάνυσμα xy1 τρόπο.

Με την εντολή σχεδίασης plot εμφανίζουμε δύο γραμμές στο γράφημα, μία που ενώνει τα σημεία της πρώτης ενότητας στροφών (στην θέση των οποίων σχεδιάζονται τετραγωνάκια) και μία που ενώνει τα σημεία της δεύτερης στροφής (στην θέση των οποίων σχεδιάζονται ρόμβοι).

Το αποτέλεσμα της εκτέλεσης του script είναι το ακόλουθο παράθυρο γραφικών:



Εναλλακτικά μπορούμε να υλοποιήσουμε τα παραπάνω με τη χρήση της εντολής **inline** χωρίς να χρειάζεται να δημιουργήσουμε function. Ο κώδικας του script γίνεται τώρα

```
clear all
clf
Qtheta=inline('[cos(theta), -sin(theta);sin(theta),cos(theta)]');
xy1=[1;1];
for i=1:4, xy1(:,i+1)=Qtheta(pi/2)*xy1(:,i);end
xy2=[1;1];
for i=1:6, xy2(:,i+1)=Qtheta(pi/3)*xy2(:,i);end
plot(xy1(1,:),xy1(2,:), 's-',xy2(1,:),xy2(2,:), 'd-')
title('Strofh shmeiwn me inline');
```

Πληκτρολογήστε τις παραπάνω εντολές σε ένα script με όνομα **strofi1a.m** (ή τροποποιείστε το προηγούμενο), αποθηκεύστε το στον ενεργό φάκελό σας και εκτελέστε το. Εάν σας εμφανιστούν σφάλματα κατά την εκτέλεσή του βρείτε τα και διορθώστε τα.

Εφαρμογή:

Δίνεται η ακόλουθη εξίσωση της έλλειψης:

$$\frac{x^2}{5^2} + \frac{y^2}{2^2} = 1$$

είναι γνωστό ότι τα σημεία της συγκεκριμένης έλλειψης ικανοποιούν τις παραμετρικές εξισώσεις

$$\begin{cases} x(t) = 5\cos(t) \\ y(t) = 2\sin(t) \end{cases} \quad 0 \le t \le 2\pi$$

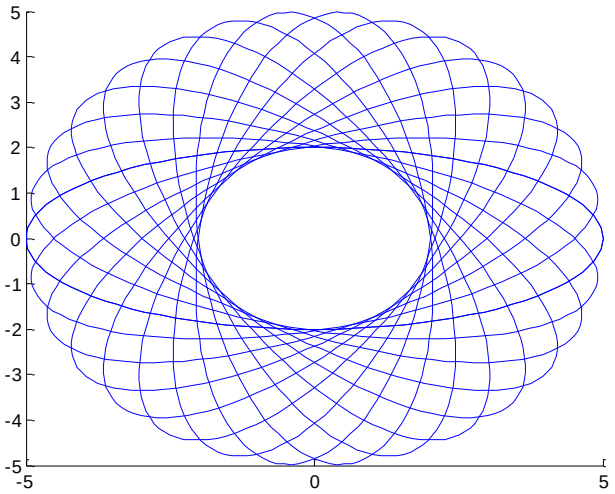
Θέλουμε να δημιουργήσουμε ένα script **strofiellipse.m** το οποίο να σχεδιάζει τη συγκεκριμένη έλλειψη και όλες τις ελλείψεις που προκύπτουν από τη στροφή της κατά τις ακόλουθες γωνίες:

$$\theta_k = \frac{k \cdot 2\pi}{15}, \quad k = 0,1,2,\text{L} \ ,15$$

Το περιεχόμενο του script είναι:

```
clear all
clf
Qtheta=inline('[cos(theta), -sin(theta);sin(theta),cos(theta)]');
t=linspace(0,2*pi,101);
xy0=[5*cos(t);2*sin(t)];
theta=linspace(0,2*pi,16);
hold
for i=theta,
 xy=Qtheta(i)*xy0;
 plot(xy(1,:),xy(2,:))
end
```

Το σχήμα που παίρνουμε ως αποτέλεσμα είναι το ακόλουθο:



Άσκηση

Στο σκληρό δίσκο C: δημιουργείστε ένα φάκελο με όνομα myfiles και κάντε τον ενεργό στο MATLAB. Σε αυτόν τον φάκελο δημιουργείστε ένα script με όνομα askscipt.m το οποίο να κάνει τα ακόλουθα:

α) Να καθαρίζει τη μνήμη από μεταβλητές.

β) Με τη χρήση της εντολής input να διαβάζει έναν πίνακα και να τον εκχωρεί σε μεταβλητή με όνομα mat.

γ) Να εξετάζει εάν ο πίνακας είναι τετραγωνικός και σε μία τέτοια περίπτωση με τη χρήση της fprintf ή της disp να επιστρέφει την κατάλληλο μήνυμα και την τιμή της ορίζουσάς του. Σε αντίθετη περίπτωση να καλεί function με όνομα labmat η οποία να παίρνει ως είσοδο τον πίνακα και να επιστρέφει το μέγιστο στοιχείο του και πίνακα με τις στήλες του αρχικού πίνακα ταξινομημένες.

Διανυσματικοί χώροι στο MATLAB.

Βάση διανυσματικού χώρου, Γραμμική ανεξαρτησία διανυσμάτων και τάξη πίνακα

Έστω ένα σύνολο στοιχείων $\{v_1,...,v_m\}$ ενός δ.χ.. Εάν κάποιο από αυτά τα στοιχεία μπορεί να γραφεί ως γραμμικός συνδυασμός των υπολοίπων τότε λέμε ότι τα στοιχεία $\{v_1,...,v_m\}$ είναι **γραμμικά εξαρτημένα**. Δηλαδή, θα πρέπει να υπάρχουν $\{\lambda_1,...,\lambda_{k-1},\lambda_{k+1}...,\lambda_m\}$ όπου $\lambda_i \in \mathbb{R}$ **όχι όλα μηδενικά** ώστε για κάποιο στοιχείο v_k να ισχύει

$$v_k = \lambda_1 v_1 + ... + \lambda_{k-1} v_{k-1} + \lambda_{k+1} v_{k+1} + ... + \lambda_m v_m .$$

Ισοδύναμα θα μπορούσαμε να πούμε ότι τα $\{v_1,...,v_m\}$ είναι γραμμικά εξαρτημένα όταν η σχέση $\lambda_1 v_1 + ... + \lambda_m v_m = \mathbf{0}$, όπου $\lambda_i \in \mathbb{R}$ έχει (εκτός την προφανή μηδενική) και μία άλλη λύση $\{\lambda_1,...,\lambda_{k-1},\lambda_k,\lambda_{k+1}...,\lambda_m\}$ όπου $\lambda_i \in \mathbb{R}$ **όχι όλα μηδενικά**. Επίσης γνωρίζουμε ότι αν $v_1,...,v_n \in \mathbb{R}^n$ και ορίσουμε Α ως τον $n \times n$ πίνακας του οποίου η στήλη i είναι το $v_i, i = 1,...,n,$ μετασχηματίζοντας με κατάλληλες γραμμοπράξεις τον πίνακα Α σε πίνακα Β κλιμακωτής μορφής μπορώ να συμπεράνω ποια από τα $v_i, i = 1,...,n,$ είναι γραμμικά ανεξάρτητα. Οι στήλες του Β περιέχουν μη μηδενικό οδηγό στοιχείο μας καθορίζουν ποιες αντίστοιχες στήλες του Α αποτελούν γραμμικά ανεξάρτητα διανύσματα.

Για να εξετάσουμε στο MATLAB εάν τα διανύσματα $\eta_1 = [0 \quad 0 \quad 1 \quad 2]^T$, $\eta_2 = [1 \quad 6 \quad -5 \quad -2]^T$, $\eta_3 = [2 \quad 3 \quad -2 \quad 0]^T$ είναι γραμμικά εξαρτημένα ή ανεξάρτητα αρχικά θα ορίσουμε έναν πίνακα με στήλες τα διανύσματα η_1, η_2, η_3.

```
>> clear all
>> A=[0,1,2;0,6,3;1,-5,-2;2,-2,0]
```

Με τη χρήση της **rref( )** μπορούμε να δούμε ότι η τάξη του πίνακα είναι 3 γιατί και οι τρεις στήλες του αποτελέσματος έχουν μη μηδενικό οδηγό στοιχείο, οπότε και οι τρεις στήλες του αρχικού πίνακα είναι γραμμικά ανεξάρτητα διανύσματα.

```
>> rref(A)
```

Έστω ένας mxn πίνακας Α τότε το σύνολο

$$R_A = \{ b \in \mathbb{R}^m : Ax = b \text{ για κάποιο } x \in \mathbb{R}^n \}$$

ονομάζεται **εικόνα του πίνακα Α**. Η διάσταση του χώρου αυτού ονομάζεται **τάξη ή βαθμός (rank)** του πίνακα.

Η τάξη ενός πίνακα είναι ίση με τον αριθμό των μη μηδενικών οδηγών στοιχείων που υπάρχουν στην κλιμακωτή μορφή που μπορούμε να φέρουμε τον πίνακα. Μία βάση του χώρου εικόνα είναι η βάση του διανυσματικού χώρου που προκύπτει εάν θεωρήσουμε τις στήλες του πίνακα (χώρος στηλών) ως διανύσματα. Οπότε, ως βάση παίρνουμε τα διανύσματα (στήλες του αρχικού πίνακα) που αντιστοιχούν σε στήλες με μη μηδενικά οδηγά στοιχεία στην τελική κλιμακωτή μορφή του πίνακα

Η εντολή του MATLAB για τον υπολογισμό της τάξης πίνακα είναι η **rank( )**.

```
>> rank(A)
```

Από τα αποτελέσματα και με βάση όσα έχουμε στην θεωρία συμπεραίνουμε ότι τα διανύσματα είναι γραμμικώς ανεξάρτητα. Αυτό διότι στην ανηγμένη μορφή του πίνακα έχει τρία μη μηδενικά οδηγά στοιχεία ή παρατηρούμε ότι η τάξη του πίνακα είναι 3.

Ένα σύνολο στοιχείων $\{v_1,...,v_m\}$ του V ονομάζεται **βάση** του V αν έχει τις ιδιότητες

a. το $\{v_1,...,v_m\}$ **παράγει** το V, δηλαδή κάθε στοιχείο του V είναι γραμμικός συνδυασμός των $v_1,...,v_m$.

b. αν ισχύει η σχέση $\lambda_1 v_1 + \ldots + \lambda_m v_m = \mathbf{0}$, όπου $\lambda_i \in \mathbb{R}$, τότε αναγκαστικά έχουμε $\lambda_1 = \lambda_2 = \ldots = \lambda_m = 0$. Δηλαδή είναι **γραμμικά ανεξάρτητα**.

Ένας διανυσματικός χώρος μπορεί να έχει παραπάνω από μία βάσεις. **Διάσταση ενός χώρου** (συμβολίζεται $\dim V$) είναι ο αριθμός των γραμμικά ανεξάρτητων διανυσμάτων της βάσης. Η διάσταση του τετριμμένου μηδενικού χώρου $V = \{\mathbf{0}\}$ είναι 0.

Για να βρούμε μία βάση του χώρου που παράγουν τα διανύσματα του παραδείγματός μας θα πρέπει να χρησιμοποιήσουμε την εντολή συμβολικών υπολογισμών **colspace( )**. Για αυτήν την εντολή ο πίνακας θα πρέπει να έχει δηλωθεί ως συμβολικός με τη χρήση της εντολής **sym( )**.

```
>> clear all
>> A=sym([0,1,2;0,6,3;1,-5,-2;2,-2,0])
>> colspace(A)
```

Τα τρία διανύσματα αυτά που αντιστοιχούν στις στήλες του αποτελέσματος παράγουν το χώρο.

Θα κάνουμε τους ίδιους υπολογισμούς και για τα διανύσματα $\eta_1 = [0 \quad 0 \quad 1 \quad 2]^T$, $\eta_2 = [2 \quad 6 \quad -5 \quad -2]^T$, $\eta_3 = [1 \quad 3 \quad -2 \quad 0]^T$.

```
>> clear all
>> A=[0,2,1;0,6,3;1,-5,-2;2,-2,0]
```

Πάλι με τη χρήση της **rref( )** μπορούμε να δούμε ότι η τάξη του πίνακα είναι 2 γεγονός που επιβεβαιώνεται και με τη χρήση της **rank( )**.

```
>> rref(A)
>> rank(A)
```

Εδώ συμπεραίνουμε ότι τα δύο πρώτα διανύσματα είναι γραμμικώς ανεξάρτητα, εφόσον στην ανηγμένη μορφή έχουμε δύο μη μηδενικά οδηγά στοιχεία στην πρώτη και δεύτερη στήλη ή παρατηρούμε ότι η τάξη του πίνακα είναι 2. Οπότε τελικά συμπεραίνουμε ότι τα δύο πρώτα διανύσματα είναι γραμμικά ανεξάρτητα και παράγουν το χώρο.

Για να βρούμε μία βάση του χώρου με γεννήτορες τα διανυσμάτων του παραδείγματός μας θα πρέπει να χρησιμοποιήσουμε πάλι την εντολή συμβολικών υπολογισμών **colspace( )**.

```
>> colspace(sym(A))
```

Τώρα τα δύο αυτά διανύσματα που αντιστοιχούν στις στήλες του αποτελέσματος αυτά παράγουν το χώρο. Παρατηρούμε ότι δεν μας επιστράφηκαν τα δύο πρώτα διανύσματα που συμπεράναμε παραπάνω ότι είναι γραμμικά ανεξάρτητα. Άλλωστε, γνωρίζουμε ότι η βάση ενός διανυσματικού χώρου δεν είναι μοναδική.

Ορθοκανονική βάση στο ¡ ^{n} στο MATLAB.

Το **Ευκλείδιο εσωτερικό γινόμενο** δύο διανυσμάτων $x = [x_1 \quad x_2 \quad x_3]^T$, $y = [y_1 \quad y_2 \quad y_3]^T$ του $\mathbb{R}^3$ ορίζεται ως

$$x \cdot y = x_1 y_1 + x_2 y_2 + x_3 y_3$$

Με βάση ευκλείδιο εσωτερικό γινόμενο ορίζουμε το **μέτρο** διανύσματος $x = [x_1 \quad x_2 \quad \ldots. \quad x_n]^T$

$$\|x\| = \sqrt{x \cdot x} = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}$$

Δύο διανύσματα είναι **κάθετα (ορθογώνια)** μεταξύ τους **εάν και μόνο αν**

$$x \cdot y = 0$$

Μία βάση $\{v_1, \ldots, v_m\}$ ενός διανυσματικού χώρου *V* ονομάζεται **ορθοκανονική** όταν τα διανύσματα είναι ορθογώνια μεταξύ τους ανά δύο $v_1 \cdot v_2 = 0$ και καθένα από αυτά έχουν μέτρο 1.

Ως εφαρμογή θα υπολογίσουμε στο MATLAB μία ορθοκανονική βάση του χώρου που ορίζουν τα διανύσματα $\eta_1 = [1 \quad 1 \quad 1 \quad 1]^T$, $\eta_2 = [0 \quad 1 \quad 1 \quad 1]^T$, $\eta_3 = [0 \quad 0 \quad 1 \quad 1]^T$.

Ορίζουμε τον πίνακα με στήλες τα διανύσματα.

```
>>clear all
>>C=[1,0,0; 1,1,0;1,1,1;1,1,1]
```

Με την εντολή **orth( )** μας επιστρέφεται ένας πίνακας με τα διανύσματα της ορθοκανονικής βάσης.

```
>> OC=orth(C)
```

Τα διανύσματα αυτά, όπως βλέπουμε με τις εντολές που ακολουθούν, είναι κάθετα μεταξύ τους (έχουν εσωτερικό γινόμενο 0) και έχουν μέτρο 1. Στις τρεις πρώτες εντολές εφαρμόζουμε το εσωτερικό γινόμενο μεταξύ των διανυσμάτων και στις επόμενες βρίσκουμε το μέτρο του κάθε διανύσματος.

```
>> OC(:,1)'*OC(:,2)
>> OC(:,2)'*OC(:,3)
>> OC(:,1)'*OC(:,3)
>> sum(OC(:,1).^2)
>> sum(OC(:,2).^2)
>> sum(OC(:,3).^2)
```

Παρατηρούμε ότι τα αποτελέσματα που αναμένουμε μηδέν έχουν μία τάξη του 10^{-16}. Αυτό οφείλεται στο ότι ο υπολογισμός της ορθοκανονικής βάσης γίνεται με προσεγγιστικές μεθόδους και την αριθμητική 16 ψηφίων της μηχανής. Αποτελέσματα με τέτοια ακρίβεια τα θεωρούμε ακριβή.

Ισοδύναμη της εντολής που χρησιμοποιήσαμε για να υπολογίσουμε το μέτρο του διανύσματος είναι και η εντολή **norm(…,2)**.

```
>> norm(OC(:,2),2)
```

Βάση χώρου Πυρήνα και βάση χώρου Εικόνα στο MATLAB.

Μία **σχέση** μεταξύ των στοιχείων δύο συνόλων *A,B* αντιστοιχίζει στοιχεία του *A* με στοιχεία του *B* άλλου μέσω ενός κανόνα που μπορεί να είναι και ένας Μαθηματικός τύπος. Μία σχέση λέγεται **απεικόνιση** όταν αντιστοιχίζει κάθε στοιχείο του *A* με ένα, μοναδικό στοιχείο του *B*.

Έστω δύο διανυσματικοί χώροι *V,W* διαστάσεων *n,m* αντίστοιχα και *f* είναι μία απεικόνιση από το *V* στο *W* για την οποία ισχύουν οι ακόλουθες ιδιότητες:

1. $f(x+y) = f(x) + f(y), \quad \forall x, y \in V$
2. $f(\lambda x) = \lambda f(x), \quad \forall x \in V \text{ και } \lambda \in \mathbb{R}$

Σε μία τέτοια περίπτωση η απεικόνιση ονομάζεται **γραμμική απεικόνιση** ή **γραμμικός μετασχηματισμός.**

Για κάθε γραμμικό μετασχηματισμό ορίζουμε τα ακόλουθα σύνολα:

1. Η **εικόνα** του μετασχηματισμού $\text{Im } f = \{ f(x) : x \in V \} \subseteq W$
2. Τον **πυρήνα** του μετασχηματισμού $Kerf = \{ x \in V : f(x) = \mathbf{0} \} \subseteq V$

Τα δύο αυτά σύνολα είναι διανυσματικοί χώροι, διανυσματικοί υπόχωροι των *V* και *W* αντίστοιχα. Για τις διαστάσεις τους, που ονομάζονται **τάξη** και **μηδενικότητα** του μετασχηματισμού, ισχύει :

$$\dim V = \dim(\text{Im } f) + \dim(Kerf)$$

Έστω ένας **γραμμικός μετασχηματισμός** $f:\mathbb{R}^n \to \mathbb{R}^m$, όπου θεωρούμε τους δ.χ. $\mathbb{R}^n, \mathbb{R}^m$ με τις κανονικές τους βάσεις, τότε υπάρχει μοναδικός πίνακας $m\ x\ n\ \ A$, τέτοιος ώστε:

$$f(x) = Ax$$

Ο πίνακας αυτός ονομάζεται **πίνακας (αναπαράστασης) του μετασχηματισμού (ή της απεικόνισης)**.

Δίνεται η γραμμική απεικόνιση:

$$f:\mathbb{R}^3 \to \mathbb{R}^2 : [x, y, z]^T \to [2x + y - z, 3x - 2y + 4z]^T$$

Θα βρούμε τον πίνακά της ως προς τις κανονικές βάσεις του πεδίου ορισμού και του πεδίου τιμών της. Στη συνέχεια θα βρούμε τις βάσεις του πυρήνα Kerf και της εικόνας Imf της f. Θα ακολουθήσουμε τη διαδικασία επίλυσης με το χέρι και σε κάθε βήμα θα βλέπουμε επίσης πως μπορούμε να κάνουμε τη λύση με το MATLAB.

Για τις κανονικές βάσεις $\{[1,0,0]^T, [0,1,0]^T, [0,0,1]^T\}$ και $\{[1,0]^T, [0,1]^T\}$ των $\mathbb{R}^3$ και $\mathbb{R}^2$ αντίστοιχα, παρατηρούμε ότι:

$$f([1,0,0]^T) = [2\cdot 1 + 0 - 0,\ 3\cdot 1 - 2\cdot 0 + 4\cdot 0]^T = [2,3]^T = 2\cdot[1,0]^T + 3\cdot[0,1]^T$$

$$f([0,1,0]^T) = [2\cdot 0 + 1 - 0,\ 3\cdot 0 - 2\cdot 1 + 4\cdot 0]^T = [1,-2]^T = 1\cdot[1,0]^T - 2\cdot[0,1]^T$$

$$f([0,0,1]^T) = [2\cdot 0 + 0 - 1,\ 3\cdot 0 - 2\cdot 0 + 4\cdot 1]^T = [-1,4]^T = -1\cdot[1,0]^T + 4\cdot[0,1]^T$$

Επομένως, ο πίνακας της f ως προς τις προηγούμενες βάσεις είναι ο $\begin{bmatrix} 2 & 1 & -1 \\ 3 & -2 & 4 \end{bmatrix}$

Ορίζουμε στο MATLAB τον παραπάνω πίνακα ως συμβολική ποσότητα με όνομα A και ως αριθμητικό πίνακα με όνομα AA.

```
>>clear all

>> A=sym([2 1 -1;3 -2 4])

>> AA=[2 1 -1;3 -2 4]
```

Για τον πυρήνα της f έχουμε:

$$[x,y,z]^T \in Kerf \Leftrightarrow f([x,y,z]^T) = [0,0]^T \Leftrightarrow [2x + y - z, 3x - 2y + 4z]^T = [0,0]^T \Leftrightarrow$$

$$\Leftrightarrow \left\{ \begin{array}{l} 2x + y - z = 0 \\ 3x - 2y + 4z = 0 \end{array} \right\} \Leftrightarrow \left\{ \begin{array}{l} z = 2x + y \\ 3x - 2y + 4(2x + y) = 0 \end{array} \right\}$$

$$\Leftrightarrow \left\{ \begin{array}{l} z = 2x + y \\ 11x + 2y = 0 \end{array} \right\} \Leftrightarrow \left\{ \begin{array}{l} z = 2x - \frac{11}{2}x \\ y = \frac{-11}{2}x \end{array} \right\} \Leftrightarrow \left\{ \begin{array}{l} z = \frac{-7}{2}x \\ y = \frac{-11}{2}x \end{array} \right., x \in \mathbb{R}$$

Έτσι τα στοιχεία του Kerf έχουν τη μορφή:

$$\left[x, \frac{-11}{2}x, -\frac{7}{2}x\right]^T = x\cdot\left[1, \frac{-11}{2}, \frac{-7}{2}\right]^T,$$

με το διάνυσμα $\left[1, \frac{-11}{2}, \frac{-7}{2}\right]^T$ να αποτελεί βάση. Πρόκειται, δηλαδή για έναν υπόχωρο του $\mathbb{R}^3$ διάστασης ένα. Στο MATLAB τώρα, για το συμβολικό πίνακα A μπορούμε να χρησιμοποιήσουμε τη συνάρτηση **null()** για να υπολογίσουμε τη βάση του Kerf.

```
>>n=null(A)
```

Η συνάρτηση null( ) εφαρμόζεται και σε μη συμβολικούς αριθμητικούς πίνακες (π.χ. τον AA) και το αποτέλεσμά της είναι μία (ορθοκανονική) βάση του Kerf, όπως φαίνεται από την εντολή norm (που στην περίπτωση αυτή είναι ένα).

```
>> nn=null(AA)
>> norm(nn,2)
```

Παρατηρούμε το αποτέλεσμα δεν είναι το ίδιο. Ωστόσο είναι εύκολο, διαιρώντας τα δύο αποτελέσματα στοιχείο προς στοιχείο, να διαπιστώσουμε ότι το διάνυσμα της βάσης που επιστρέφει είναι στην ουσία το ίδιο μιας και τα n και nn είναι συγγραμικά. **Θυμηθείτε ότι δύο διανύσματα είναι συγγραμμικά όταν ένα μπορεί να γραφεί ως γινόμενο αριθμού επί το άλλο.**

```
>> double(n)./nn
```

Η double όπως έχουμε πει μετατρέπει το συμβολικό διάνυσμα n σε διάνυσμα με στοιχεία αριθμούς κινητής υποδιαστολής.

Σε μία άλλη έκφρασή της η null( ) μπορεί να επιστρέψει μη ορθοκανονική βάση. Θυμίζουμε ότι μία βάση ονομάζεται ορθοκανονική όταν τα διανύσματά της έχουν μέτρο 1 και είναι κάθετα μεταξύ τους.
Περισσότερα για το τι είναι μέτρο και καθετότητα θα δούμε παρακάτω.

Αντίστοιχα, για την εικόνα Imf της γραμμικής απεικόνισης f έχουμε:

$$\vec{v} \in \mathrm{Im}\ f \Leftrightarrow \vec{v} = f([x,y,z]^T) \Leftrightarrow \vec{v} = [2x+y-z, 3x-2y+4z]^T \Leftrightarrow$$
$$\Leftrightarrow \vec{v} = [2x, 3x]^T + [y, -2y]^T + [-z, 4z]^T \Leftrightarrow$$
$$\Leftrightarrow \vec{v} = x\cdot[2,3]^T + y\cdot[1,-2]^T + z\cdot[-1,4]^T$$

Πρόκειται δηλαδή για τον υπόχωρο του $\mathbb{R}^2$ που παράγεται από τα διανύσματα $[2,3]^T$, $[1,-2]^T$, $[-1,4]^T$. Επειδή όμως κάθε ζευγάρι από αυτά είναι γραμμικά ανεξάρτητα, ο χώρος που παράγουν είναι ένας υπόχωρος του $\mathbb{R}^2$ διάστασης 2, δηλαδή ο ίδιος ο $\mathbb{R}^2$. Έτσι Imf $=\mathbb{R}^2$ και ως βάση της μπορούμε να θεωρήσουμε οποιοδήποτε ζευγάρι από τα $[2,3]^T$, $[1,-2]^T$, $[-1,4]^T$ ή, **γενικότερα, οποιοδήποτε ζευγάρι μη συγγραμμικών, και άρα γραμμικά ανεξάρτητων διανυσμάτων, του** $\mathbb{R}^2$. Όπως είπαμε παραπάνω, δύο διανύσματα είναι συγγραμμικά όταν ένα μπορεί να γραφεί ως γινόμενο αριθμού επί το άλλο.

Για την εικόνα Imf στο MATLAB μπορούμε να χρησιμοποιήσουμε τη collspace( ) η οποία εφαρμόζεται μόνο σε συμβολικούς πίνακες, όπως έχουμε αναφέρει. Στην περίπτωση αυτή ως βάση επιστρέφεται η κανονική βάση, κάτι που είναι σύμφωνο με τα όσα αναφέρουμε παραπάνω.

```
>> colspace(A)
```

Χαρακτηριστικά μεγέθη και εφαρμογές στο MATLAB.

Έστω A ένας nxn πίνακας με πραγματικά στοιχεία. Ο πραγματικός ή μιγαδικός αριθμός λ είναι **ιδιοτιμή** του πίνακα A εάν και μόνο εάν

$$p(\lambda) = \det(A - \lambda I) = 0$$

Το πολυώνυμο $p(\lambda)$ ονομάζεται **χαρακτηριστικό πολυώνυμο**, η παραπάνω εξίσωση, **χαρακτηριστική εξίσωση** και οι ιδιοτιμές και τα ιδιοδιανύσματα ενός πίνακα **χαρακτηριστικά μεγέθη**.

Εάν $A = \begin{bmatrix} a_{11} & \text{K} & a_{nn} \\ \text{M} & \text{O} & \text{M} \\ a_{n1} & \text{L} & a_{nn} \end{bmatrix}$ τότε $p(\lambda) = \det(A - \lambda I) = \begin{vmatrix} a_{11} - \lambda & a_{12} & \text{K} & a_{1n} \\ a_{21} & a_{22} - \lambda & \text{K} & a_{2n} \\ & \text{M} & \text{O} & \text{M} \\ a_{n1} & a_{n2} & \text{L} & a_{nn} - \lambda \end{vmatrix}$

Το χαρακτηριστικό πολυώνυμο ενός πίνακα A είναι ένα πολυώνυμο βαθμού n

$$p(\lambda) = (-1)^n [\lambda^n + b_{n-1}\lambda^{n-1} + \text{L} \ + b_1\lambda + b_0]$$

οπότε έχει n στο σύνολο πραγματικές και μιγαδικές ρίζες, οπότε και ιδιοτιμές n στο σύνολο πραγματικές ή μιγαδικές ιδιοτιμές.

Η πολλαπλότητα κάθε ρίζας του χαρακτηριστικού πολυωνύμου ονομάζεται **αλγεβρική πολλαπλότητα** της αντίστοιχης ιδιοτιμής.

Έστω λ ιδιοτιμή του nxn πίνακα A και έστω

$$E_\lambda = \{v: \ Av = \lambda v\}$$

Το σύνολο αυτό είναι ένας υπόχωρός του $\mathbb{C}^n$ (ο χώρος με τα διανύσματα με μιγαδικά στοιχεία, και καλείται **ιδιόχωρος** του πίνακα A που αντιστοιχεί στην ιδιοτιμή λ.

Η διάσταση $\dim E_\lambda$ του ιδιοχώρου μίας ιδιοτιμής καλείται **γεωμετρική πολλαπλότητα** της ιδιοτιμής και ισούται με την μηδενικότητα του πίνακα A-λI . Η γεωμετρική πολλαπλότητα είναι μικρότερη ή ίση από την αλγεβρική πολλαπλότητα της ιδιοτιμής.

Έστω A ένας nxn πίνακας με πραγματικά στοιχεία, τότε ικανοποιεί το χαρακτηριστικό του πολυώνυμο:

$$p(A) = (-1)^n [A^n + b_{n-1}A^{n-1} + \text{L} \ + b_1 A + b_0 I] = 0$$

Από αυτό το θεώρημα όταν $b_0 \neq 0$,δηλαδή το 0 μηδέν δεν είναι ιδιοτιμή του πίνακα οπότε και $\det(A) \neq 0$

, τότε ο αφού ο πίνακας αντιστρέφεται, μπορούμε να βρούμε τον αντίστροφο του A

$$(-1)^n [A^n + b_{n-1}A^{n-1} + \text{L} \ + b_1 A + b_0 I] = 0 \Leftrightarrow b_0 I = -A^n - b_{n-1}A^{n-1} - \text{L} \ - b_1 A \Leftrightarrow$$

$$A^{-1} = \frac{1}{b_0}\left(-A^{n-1} - b_{n-1}A^{n-2} - \text{L} \ - b_2 A - b_1 I\right)$$

Στο επόμενο παράδειγμα, θα βρούμε τις ιδιοτιμές πίνακα. και τα αντίστοιχα ιδιοδιανύσματα. Θα πιστοποιήσουμε την ισχύ του θεωρήματος Cayley-Hamilton και θα βρούμε τον αντίστροφο του πίνακα με χρήση του θεωρήματος Cayley-Hamilton.

Ορίζουμε τον πίνακα

```
>> clear all
>>format long
>> A=[1 1 -2; -1 2 1; 0 1 -1]
```

Η συνάρτηση **poly( )** όταν εφαρμόζεται σε πίνακα επιστρέφει τους συντελεστές του χαρακτηριστικού πολυωνύμου του πίνακα. Τις ρίζες του μπορούμε να τη βρούμε με την εντολή **roots( ).**

```
>> p=poly(A)
>> roots(p)
```

$$p(\lambda) = \lambda^3 - 2\lambda^2 + \lambda - 2 = (\lambda - 1)(\lambda - 2)(\lambda + 1)$$

Στο ίδιο συμπέρασμα φθάνουμε με τη χρήση του ορισμού.

```
>> syms x
>> pp=Expand(det(x*eye(3)-A))
>> solve(pp)
```

Ωστόσο, και η εντολή **eig( )** υπολογίζει απευθείας τις ιδιοτιμές.

```
>> eig(A)
```

Μπορούμε με την ίδια εντολή, με τη σύνταξη που ακολουθεί, να λαμβάνουμε ως στήλες ενός πίνακα τα ιδιοδιανύσματα που αντιστοιχούν στις ιδιοτιμές και έναν διαγώνιο πίνακα με διαγώνια στοιχεία τις αντίστοιχες ιδιοτιμές.

```
>> [idio,d]=eig(A)
```

Επίσης εύκολα μπορούμε να πιστοποιήσουμε ότι ισχύει το θεώρημα Cayley-Hamilton.

```
>> A^3-2*A^2-A+2*eye(3)
```

Με βάση το θεώρημα Cayley-Hamilton $A^{-1} = -\frac{1}{2}A^2 + A + \frac{1}{2}I$. Κάτι που μπορούμε να το πιστοποιήσουμε με τη χρήση της **inv( )**.

```
>> AINV=-1/2 *A^2+A+1/2 * eye(3)
>> inv(A)
```

Διαγωνοποίηση πίνακα στο MATLAB.

Ένας *n*x*n* πίνακας *A* καλείται **διαγωνοποιήσιμος** εάν υπάρχει διαγώνιος *n*x*n* πίνακας D όμοιος με τον *A.* Δηλαδή υπάρχει αντιστρέψιμος *n*x*n* πίνακας P ώστε

$$D=P^{-1}AP.$$

Ένας *n*x*n* πίνακας *A* είναι διαγωνοποιήσιμος εάν και μόνο εάν έχει *n* γραμμικά ανεξάρτητα ιδιοδιανύσματα. **Δηλαδή είναι διαγωνοποιήσιμος εάν και μόνο εάν η γεωμετρική πολλαπλότητα της κάθε ιδιοτιμής είναι ίση με την αλγεβρική πολλαπλότητά της.**

Εάν $\lambda_1, \lambda_2, ..., \lambda_n$ οι *n* ιδιοτιμές και $v_1, v_2, ..., v_n$ τα αντίστοιχα γραμμικά ανεξάρτητα ιδιοδιανύσματα τότε:

$$D = \begin{bmatrix} \lambda_1 & 0 & . & . & . & 0 & 0 \\ 0 & \lambda_2 & . & . & . & 0 & 0 \\ . & 0 & . & & & & . \\ . & . & . & . & & & . \\ . & . & & . & . & & . \\ 0 & . & . & . & 0 & \lambda_{n-1} & 0 \\ 0 & 0 & . & . & . & 0 & \lambda_n \end{bmatrix}$$

και ο πίνακας P έχει ως στήλες τα αντίστοιχα ιδιοδιανύσματα $P = [v_1, v_2, ..., v_n]$ και ισχύει $D=P^{-1}AP$.

Εφόσον ισχύει $D=P^{-1}AP$ ισχύει και $A=PDP^{-1}$ οπότε και

$$A^k = \left(PDP^{-1}\right)^k = \underbrace{PDP^{-1}PDP^{-1}\cdots PDP^{-1}}_{\kappa \text{ φορές}} = PD^kP^{-1}$$

Στη συνέχεια θα εξετάσουμε εάν διαγωνοποιούνται οι ακόλουθοι πίνακες:

i) $A = \begin{bmatrix} 1 & -3 & 3 \\ 3 & -5 & 3 \\ 6 & -6 & 4 \end{bmatrix}$ ii) $B = \begin{bmatrix} 4 & -4 & 0 & 0 \\ 4 & -5 & 0 & 0 \\ 0 & 2 & 3 & -2 \\ -2 & 4 & 2 & -1 \end{bmatrix}$ iii) $C = \begin{bmatrix} 2 & 1 & 0 \\ 0 & 1 & -1 \\ 0 & 2 & 4 \end{bmatrix}$

i) Ορίζουμε τον πίνακα και υπολογίζουμε τις ιδιοτιμές και τα διδιοδιανύσματα με την **eig( ).**

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

```
>> [p,d]=eig(A)
```

Εφόσον ο πίνακας έχει τρεις διαφορετικές ιδιοτιμές συμπεραίνουμε ότι ο πίνακας διαγωνοποιείται.

Οι πίνακες που μας δίνει η eig( ) ικανοποιούν την σχέση της διαγωνοποίησης $P^{-1}AP = D$.

```
>> inv(p)*A*p
```

Δεν πρέπει να ξεχνάμε ότι η eig( ) υπολογίζει προσεγγιστικά τις ιδιοτιμές και τα ιδιοδιανύσματα. Σε αυτό οφείλονται και οι πολύ μικρές αποκλίσεις από τα αναμενόμενα, με βάση τη θεωρία, αποτελέσματα.

ii)

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

```
>> [p,d]=eig(B)
```

Παρατηρούμε ότι πίνακας έχει δύο διακεκριμένες ιδιοτιμές με αλγεβρική πολλαπλότητα 2. Παρατηρούμε εύκολα ότι σε κάθε ιδιοτιμή αντιστοιχεί ένα ιδιοδιάνυσμα (μπορεί να εμφανίζονται δύο στον πίνακα p αλλά αυτά είναι είτε ίδια ή ανάλογα). Δηλαδή κάθε ιδιοτιμή έχει γεωμετρική πολλαπλότητα 1 διαφορετική από την αλγεβρική της συνεπώς ο πίνακας δεν διαγωνοποιείται.

iii)

```
>> C= [ 2 1 0; 0 1 -1; 0 2 4]
```

```
>> [p,d]=eig(C)
```

```
>> format long
```

```
>> p
```

Παρατηρούμε ότι πίνακας έχει δύο διακεκριμένες ιδιοτιμές η μια με αλγεβρική πολλαπλότητα 2 και η άλλη με αλγεβρική πολλαπλότητα 1. Παρατηρούμε εύκολα ότι σε κάθε ιδιοτιμή αντιστοιχεί ένα ιδιοδιάνυσμα (μπορεί να εμφανίζονται δύο στον πίνακα p για την μία από αυτές αλλά αυτά είναι είτε ίδια ή ανάλογα). Δηλαδή η μία ιδιοτιμή έχει γεωμετρική πολλαπλότητα 1 διαφορετική από την αλγεβρική της συνεπώς ο πίνακας δεν διαγωνοποιείται.

Διαφορικές Εξισώσεις

Ορισμοί:

Μία **συνήθης διαφορική εξίσωση (δ.ε.)** είναι μία εξίσωση στην οποία εκτός από την άγνωστη συνάρτηση $y=y(x)$ εμφανίζονται και παράγωγοί της. Δηλαδή η γενική μορφή μιας διαφορικής εξίσωσης είναι:

$$F(x,y,y',y'',...,y^{(n)})=0$$

ή

$$F\left(x, y, \frac{dy}{dx}, \frac{d^2y}{dx^2}, ..., \frac{d^ny}{dx^n}\right) = 0 .$$

Ως **τάξη** της δ.ε. λέμε την ανώτερη παράγωγο που εμφανίζεται στην εξίσωση. Μία διαφορική εξίσωση ονομάζεται **γραμμική** εάν μπορεί να γραφεί στη μορφή:

$$a_n(x)\frac{d^ny}{dx^n} + a_{n-1}(x)\frac{d^{n-1}y}{dx^{n-1}} + \text{L} \quad + a_1(x)\frac{dy}{dx} + a_0(x)y = F(x)$$

Εάν ισχύει $F(x)=0$ η γραμμική δ.ε. ονομάζεται **ομογενής,** διαφορετικά ονομάζεται **μη ομογενής**. Στην περίπτωση που τα $\alpha_0(x),\alpha_1(x),...,\alpha_n(x)$ είναι αριθμοί (και όχι συναρτήσεις του x) τότε λέμε ότι είναι μία γραμμική δ.ε. με σταθερούς συντελεστές.

Παραδείγματα:

Πρώτης τάξης γραμμική με σταθερούς συντελεστές ομογενής :

$$\frac{dy}{dx} = 5y \text{ ή } y' = 5y \Leftrightarrow y' - 5y = 0$$

Πρώτης τάξης γραμμική με μη σταθερούς συντελεστές ομογενής :

$$\frac{dy}{dx} = 5xy \text{ ή } y' = 5xy \Leftrightarrow y' - 5xy = 0$$

Πρώτης τάξης γραμμική με σταθερούς συντελεστές μη ομογενής :

$$3\frac{dy}{dx} - \sin(x) = 0 \text{ ή } 3y' - \sin(x) = 0 \Leftrightarrow 3y' = \sin(x)$$

Πρώτης τάξης μη γραμμική $3\frac{dy}{dx} - \sin(y) = 0 \text{ ή } 3y' - \sin(y) = 0$ λόγω του $\sin(y)$.

Τρίτης τάξης μη γραμμικές:

$$\left(\frac{d^3y}{dx^3}\right)^2 + \left(\frac{d^2y}{dx^2}\right)^3 - \frac{dy}{dx} = e^x \text{ ή } \left(y^{(3)}\right)^2 + \left(y''\right)^3 - y' = e^x ,$$

$$\frac{d^3y}{dx^3} + \frac{d^2y}{dx^2}\frac{dy}{dx} = \sin(x) \text{ ή } y^{(3)} + y''y = \sin(x)$$

Ονομάζουμε ως **λύση ή ολοκλήρωμα** της δ.ε. τη συνάρτηση $y=y(x)$ όταν αυτή και οι παράγωγοί της (που εννοείται ότι υπάρχουν) ικανοποιούν την εξίσωση της δ.ε..

Παράδειγμα:

Δείξτε ότι για οποιεσδήποτε τιμές των παραμέτρων C_1,C_2 η συνάρτηση $y = C_1 \cos x + C_2 \sin x$ αποτελεί λύση της δ.ε.

$$\frac{d^2y}{dx^2} + y = 0 \text{ ή } y'' + y = 0$$

$$y = C_1 \cos x + C_2 \sin x \Rightarrow \frac{dy}{dx} = -C_1 \sin x + C_2 \cos x \Rightarrow \frac{d^2y}{dx^2} = -C_1 \cos x - C_2 \sin x$$

Φανερά $\frac{d^2y}{dx^2} + y = \left(C_1 \cos x + C_2 \sin x\right) + \left(-C_1 \cos x - C_2 \sin x\right) = 0$

Μπορεί να δειχθεί ότι ο παραπάνω τύπος δίνει όλες τις πιθανές λύσεις της συγκεκριμένης δ.ε..

Παράδειγμα:

Εξετάστε εάν η $y(x) = 2e^{3x} - 5e^{5x}$ αποτελεί λύση της δ.ε. με τύπο

$$\frac{d^2y}{dx^2} - 7\frac{dy}{dx} + 12y = 0 \text{ ή } y'' - 7y' + 12y = 0$$

Έχουμε $y(x) = 2e^{3x} - 5e^{5x} \Rightarrow \frac{dy}{dx} = 6e^{3x} - 25e^{5x} \Rightarrow \frac{d^2y}{dx^2} = 18e^{3x} - 125e^{5x}$

Οπότε

$$\frac{d^2y}{dx^2} + -7\frac{dy}{dx} + 12y = 18e^{3x} - 125e^{5x} - 7\left(6e^{3x} - 25e^{5x}\right) + 12\left(2e^{3x} - 5e^{5x}\right) =$$

$$= (18 - 42 + 24)e^{3x} + (-125 + 175 - 60)e^{5x} = 0e^{3x} - 10e^{5x} \neq 0$$

Άρα η συγκεκριμένη *y(x)* δεν αποτελεί λύση της δ.ε..

Μια λύση η οποία δίνει όλες τις λύσεις μίας δ.ε ονομάζεται **γενική λύση** ή **γενικό ολοκλήρωμα** της δ.ε.. Το να λύσουμε μία δ.ε. σημαίνει το να βρούμε τη γενική της λύση. Η γενική λύση μίας δ.ε. τάξης *n* αναμένεται να περιέχει *n* παραμέτρους (αυθαίρετες σταθερές), όπως στο πρώτο παράδειγμά μας η λύση της δευτέρας τάξης δ.ε. έχει δύο παραμέτρους, τα C_1,C_2.

Μία λύση που προκύπτει εάν στη γενική λύση δώσουμε συγκεκριμένες τιμές σε όλες τις παραμέτρους που περιέχει, τότε ονομάζεται **μερική λύση** ή **ολοκληρωτική καμπύλη** της δ.ε..

Μία λύση δ.ε. η οποία δεν περιέχει παραμέτρους (αυθαίρετες σταθερές) και η οποία δεν προκύπτει από κάποια γενική λύση (δίνοντας τιμές στις παραμέτρους) ονομάζεται **ιδιάζουσα λύση**.

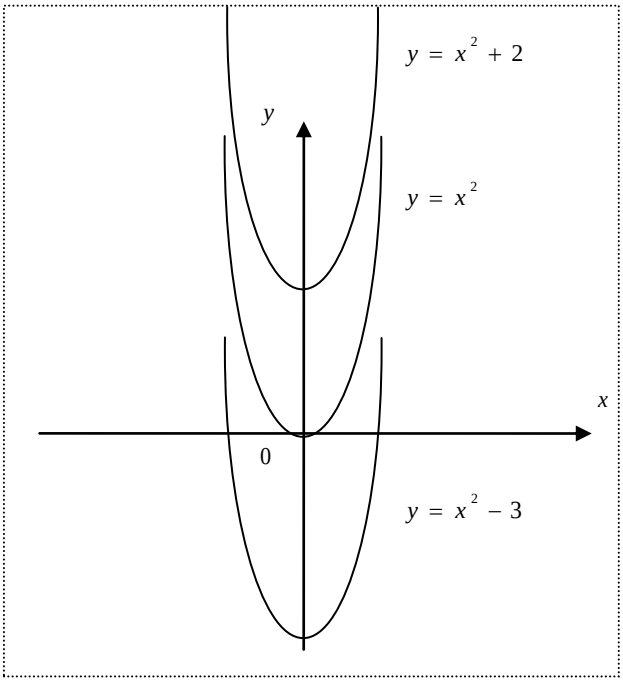
Για να επιλέξουμε τις τιμές των παραμέτρων και να βρούμε μία μερική λύση από μία γενική, θα πρέπει να μας δοθούν ανεξάρτητες μεταξύ τους συνθήκες που θα ικανοποιεί η μερική λύση.

Εάν οι συνθήκες που μας δίνονται έχουν την μορφή **αρχικών συνθηκών** δηλαδή $y(\alpha)=\beta_0, y'(\alpha)= \beta_1, y''(\alpha)= \beta_2,\ldots,y^{(n)}= \beta_n$ για καθορισμένη τιμή $x=\alpha$, τότε το πρόβλημα εύρεσης της μερικής λύσης ονομάζεται **πρόβλημα αρχικών τιμών**.

Εάν οι συνθήκες που μας δίνονται έχουν την μορφή **συνοριακών συνθηκών** δηλαδή για μιας δευτέρας τάξης εξίσωση για παράδειγμα θέλουμε η λύση να ικανοποιεί τις $y(a)=\beta_0$ και $y(b)= \beta_1$, σε ένα διάστημα $[a,b]$ τότε το πρόβλημα εύρεσης της μερικής λύσης ονομάζεται **πρόβλημα συνοριακών τιμών**.

Υπάρχουν θεωρήματα που καθορίζουν πότε υπάρχει η γενική λύση μιας δ.ε. αλλά η ύπαρξη λύσης δεν θα μας απασχολήσει.

Για να κατανοήσουμε τι είναι γενική και τι μερική λύση ας δούμε το ακόλουθο απλό παράδειγμα: Είναι φανερό ότι η διαφορική εξίσωση $y' = 2x$ έχει γενική λύση την $y(x) = x^2 + c$, διότι εάν την παραγωγίσουμε και την αντικαταστήσουμε στην διαφορική εξίσωση την ικανοποιεί. Η γενική αυτή λύση είναι μία μονοπαραμετρική οικογένεια λύσεων δηλαδή, περιέχει μία παράμετρο κάτι που αναμέναμε μιας η διαφορική εξίσωση είναι πρώτης τάξης. Είναι εύκολο να σχεδιάσουμε αυτή την οικογένεια λύσεων μιας και αποτελείται από απλές παραβολές οι οποίες μετατοπίζονται ανάλογα με την τιμή της παραμέτρου c .



Καθεμία από αυτές τις παραβολές αποτελούν μερική λύση της διαφορικής εξίσωσης οι οποίες ικανοποιούν ένα πρόβλημα αρχικών τιμών. Η αρχική συνθήκη καθορίζει την τιμή του c, πότε και ποια από τις μερικές λύσεις λύνει το συγκεκριμένο πρόβλημα. Για παράδειγμα εάν η αρχική μας συνθήκη είναι η $y(0) = 2$ τότε η παραβολή $y(x) = x^2 + 2$ αποτελεί τη (μερική) λύση της διαφορικής εξίσωσης που ικανοποιεί το συγκεκριμένο πρόβλημα αρχικών τιμών $y' = 2x, \quad y(0) = 2$.

Λύση διαφορικών εξισώσεων στο MATLAB

Η λύση διαφορικών εξισώσεων και συστημάτων διαφορικών εξισώσεων στο MATLAB μπορεί να γίνει χρησιμοποιώντας τις συμβολικές δυνατότητες του MATLAB και τη συνάρτηση συμβολικού (ακριβούς) υπολογισμού λύσεων **dsolve( )**.

Η γενική μορφή της σύνταξή της είναι η ακόλουθη:

dsolve('eq1,eq2,...', 'cond1,cond2,...', 'v')

όπου :

- 'v' είναι το σύμβολο της ανεξάρτητης μεταβλητής ως προς την οποία παραγωγίζονται οι διάφορες ποσότητες. Όταν δεν αναφέρεται στην κλίση το σύμβολο το σύμβολο της ανεξάρτητης μεταβλητής, θεωρείται ως ανεξάρτητη μεταβλητή το t .
    - Για παράδειγμα εάν έχουμε την εξίσωση $\frac{dy}{dx} = 2x$ ή $y' = 2x$ (την οποία είδαμε λίγο παραπάνω) η ανεξάρτητη ποσότητα είναι η $'x'$, εξαρτημένη το $y(x)$ και η λύση της δ.ε. είναι της μορφής $y(x) = x^2 + 2$.
    - Ωστόσο μπορεί να θεωρήσουμε και τη διαφορική εξίσωση $x' = 2 - x$ όπου η ανεξάρτητη ποσότητα να είναι το $'t'$, εξαρτημένη το $x(t)$ και η λύση της δ.ε. είναι της μορφής $x(t) = C_1 e^{-t} + 2$.

- 'eq1,eq2,...' είναι οι διαφορικές εξισώσεις που επιθυμούμε να λύσουμε. Η μορφή με την οποία εισάγουμε είναι η μορφή ισότητας όπου η πρώτη παράγωγος συμβολίζεται με D, η δεύτερη με D2 κ.ο.κ. ακολουθούμενη από το σύμβολο ως της εξαρτημένης μεταβλητής που παραγωγίζεται.
    - Για την $y' = 2x$ η εξίσωση γράφεται 'Dy=2*x'.
    - Για την $x' = 2 - x$ η εξίσωση γράφεται 'Dx=2-x'
- 'cond1,cond2,...' είναι οι αρχικές συνθήκες που ισχύουν όταν δεν θέλουμε τη γενική λύση αλλά κάποια μερική λύση της διαφορικής εξίσωσης.

Χρησιμοποιώντας τη συνάρτηση **dsolve( )** θα υπολογίσουμε τη γενική λύση της διαφορικής εξίσωσης $x' = 2 - x$.

```
>> clear all
>> syms x  t
>> s=dsolve('Dx=2-x', 't')
```

Εάν τώρα θεωρήσουμε το πρόβλημα αρχικών τιμών $y' = 2x, \quad y(0) = 2$ και θέλουμε να σχεδιάσουμε και την συγκεκριμένη μερική λύση πληκτρολογούμε:

```
>> clear all
>> syms x  y
>> s1=dsolve('Dy=2*x', 'y(0)=2 ', 'x')
>> ezplot(s1)
```

Επίσης μπορούμε να λύσουμε και συστήματα διαφορικών εξισώσεων, όπως το ακόλουθο:

$$x' = y$$

$$y' = x$$

Όπου η λύση περιγράφεται ως συνάρτηση του t.

```
>> clear all
>> syms x  y t
>> d=dsolve('Dx=y,Dy=x')
>> solx=d.x
>> soly=d.y
```

Ακολούθως λύνουμε ένα πρόβλημα δευτέρας τάξης με συνοριακές συνθήκες:

$$y'' = -y + t, \quad y(0)=1, y(1)=2$$

```
>> clear all
>> syms  y  t
>> d=dsolve('D2y=-y+t', 'y(0)=1, y(1)=2', 't')
>> ezplot(d)
```

Είναι εύκολο αντικαθιστώντας τη λύση που βρήκαμε στον τύπο της διαφορικής εξίσωσης να διαπιστώσουμε ότι την ικανοποιεί:

```
>> diff(d,2)+d-t
```

Και τέλος λύνουμε ένα πρόβλημα δευτέρας τάξης με αρχικές συνθήκες:

$$y'' = -y + t, \quad y(0)=1, y'(0)=2$$

```
>> clear all
>> syms  y  t
>> d=dsolve('D2y=-y+t', 'y(0)=1, Dy(0)=2', 't')
```

Είναι εύκολο πάλι αντικαθιστώντας τη λύση που βρήκαμε στον τύπο της διαφορικής εξίσωσης να διαπιστώσουμε ότι την ικανοποιεί:

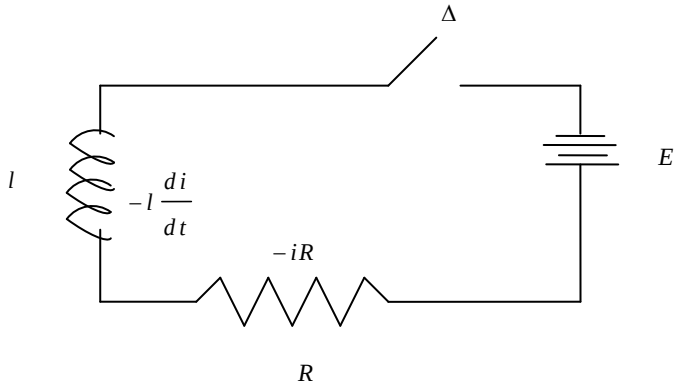
```
>> diff(d,2)+d-t
```

Παρατηρήστε ότι τα δύο τελευταία προβλήματα (ένα αρχικών τιμών και ένα συνοριακών τιμών με την ίδια διαφορική εξίσωση) έχουν διαφορετικές λύσεις.

Το MATLAB δεν μπορεί να λύσει αναλυτικά όλες τις διαφορικές εξισώσεις. Εάν το περιβάλλον αποτύχει στην εύρεση αναλυτικής λύσης επιστρέφει κενή λύση και κατάλληλο μήνυμα προειδοποίησης ή την πιο κατάλληλη ολοκληρωτική ή άλλη μορφή της λύσης. Για τέτοιες διαφορικές εξισώσεις θα πρέπει να επιζητήσουμε την προσεγγιστική λύση τους.

Αριθμητική (Προσεγγιστική) Λύση Προβλημάτων Αρχικών Τιμών 1ης τάξης

Έστω ένα κύκλωμα το οποίο αποτελείται από μία πηγή ηλεκτρεργετικής δύναμης *E (Volt)*, η οποία μπορεί να είναι σταθερή ή να εξαρτάται από το χρόνο δηλαδή *E=E(t),* πηνίο αυτεπαγωγής *l (Herny)*, ωμική αντίσταση *R (Ohm)* και διακόπτη *Δ*, συνδεδεμένα σε σειρά.



Τη χρονική στιγμή *t=0* ο διακόπτης είναι κλειστός κλείνει και ζητείται να προσδιοριστεί η τιμή του ρεύματος *i=i(t)* που αρχίζει να διαρρέει στο κύκλωμα. Εφαρμόζουμε τον πρώτο νόμο του Kirchhoff ο οποίος μας λέει ότι η ηλεκτρεγερτική δύναμη ισοφαρίζει κάθε χρονική στιγμή την πτώση τάσης στο πηνίο $l\frac{di}{dt}$ και την πτώση τάσης στην αντίσταση iR , δηλαδή:

$$E - iR - l\frac{di}{dt} = 0 \Leftrightarrow \frac{di}{dt} = -\frac{R}{l}i + \frac{E}{l},$$

Εάν θεωρήσουμε ότι τη χρονική στιγμή t=0 ισχύει *i(0)=C* αυτή η εξίσωση αποτελεί μία διαφορική εξίσωση που μπορεί να γραφεί ως $\frac{di}{dt} = f(t,i) = ai(t) + e(t)$ όπου $a = -\frac{R}{l}$ και $e(t) = \frac{E}{l}$.

Στη γενική μορφή, αυτή είναι μία διαφορική εξίσωση πρώτης τάξης μη ομογενής με σταθερούς συντελεστές της μορφής $\frac{di}{dt} - ai(t) = e(t)$ με $i(0) = C$. Είναι δηλαδή ένα πρόβλημα αρχικών τιμών πρώτης τάξης με σταθερούς όρους.

Εφαρμοσμένο Παράδειγμα 1:

Για παράδειγμα εάν το κύκλωμα RL αποτελείται από μία πηγή ηλεκτρεργετικής δύναμης *E=4cos(2t) Volt* σταθερή πηνίο αυτεπαγωγής *l=2 Herny,* ωμική αντίσταση *R=8 Ohm* και διακόπτη *Δ* και τη χρονική στιγμή t=0 ισχύει *i(0)=2.*

Το πρόβλημα που έχουμε να λύσουμε είναι της μορφής:

$$\frac{di}{dt} = -4i(t) + 2\cos(2t), \quad i(0)=2$$

Θα μπορούσαμε να λύσουμε αυτό το πρόβλημα με διάφορους τρόπους (π.χ. τη μέθοδο μετασχηματισμού Laplace ή άλλες αναλυτικούς τρόπους λύσης στο χαρτί) και θα καταλήγαμε σε μία «αναλυτική» λύση (συνεχής) της μορφής $i(t) = \ldots$ η οποία ικανοποιεί την εξίσωση και την αρχική συνθήκη.

Στο MATLAB μπορούμε να λύσουμε το πρόβλημα προσεγγιστικά (ή αριθμητικά όπως λέμε). Η μέθοδος ode45 που θα ζητήσουμε να εφαρμόσει το περιβάλλον **θα επιστρέψει ένα διάνυσμα με στοιχεία τιμές που προσεγγίζουν τη λύση σε διάφορες χρονικές στιγμές και ένα διάνυσμα με τις χρονικές αυτές στιγμές.** Το βήμα του χρόνου στο οποίο υπολογίζει τη λύση δεν είναι σταθερό. Το συγκεκριμένο

πρόγραμμα παίρνει πυκνότερη ή αραιότερη διαμέριση του χρόνου ανάλογα με τη μορφή που έχει το πρόβλημα.

Για να λύσουμε «αριθμητικά» το παραπάνω πρόβλημα για $0 \le t \le 15$ πρώτα πρέπει πρώτα να το γράψετε στη μορφή $\frac{dy}{dt} = f(t, y)$ δηλαδή:

$$\frac{dy}{dt} = -4\,y(t) + 2\cos(2t), \quad y(0)=2$$

Για λόγους σύμβασης εδώ αντί του $i(t)$ χρησιμοποιούμε το πιο συνηθισμένο $y(t)$. Στον ενεργό φάκελό μας (πρέπει να έχετε έναν, κάντε τον ενεργό διαφορετικά δημιουργείστε ένα νέο έχουμε δει πως) δημιουργούμε σε ένα m-αρχείο με όνομα myfunc1.m την function που δέχεται ως είσοδο τα $t, y(t)$ και επιστρέφει την τιμή $-4\,y(t) + 2\cos(2t)$ που αποτελεί την παράγωγο του $y(t)$.

```
function ydot = myfunc1(t,y)
% myfunc1
ydot = -4*y+2*cos(2*t);
```

Στη συνέχεια δημιουργείστε αρχείο εντολών script με όνομα solveode1.m με περιεχόμενα:

```
% Lynei th diaforikh
clear all;
clf;
tspan=[0,15];
y0=[2];
[t,y]=ode45('myfunc1',tspan,y0);
plot(t,y,'r.')
xlabel('t')
ylabel('i(t)')
grid on
hold
syms tt  yy
s1=dsolve('Dyy=-4*yy+2*cos(2*tt)','yy(0)=2','tt')
ezplot(s1,[0,10])
maxerror=max(abs(y-subs(s1,t)))
```

Αρχικοποίηση του προβλήματος

Προσεγγιστική λύση τουρ προβλήματος. Επιστρέφει σε ένα διάνυσμα y προσέγγιση της λύσης σε μία διαμέριση t του χρόνου tspan.

Γράφημα της προσεγγιστικής λύσης με κόκκινες τελείες.

Εύρεση και γράφημα της ακριβούς λύσης (μπλε γραμμή) με τις συμβολικές δυνατότητες του MATLAB. Προσέξτε την αλλαγή μεταβλητών σε tt και yy μιας και παραπάνω χρησιμοποιούμε τις t,y.

Υπολογισμός της μέγιστης τιμής του απόλυτου σφάλματος.

Με την πρώτη εντολή (μετά το σχόλιο και τον καθαρισμό τιμών μεταβλητών και παραθύρου γραφικών) ορίζουμε ένα διάνυσμα, το tspan, με τα άκρα του διαστήματος στο οποίο θα πάρουμε προσεγγίσεις της λύσης και με τη δεύτερη την αρχική συνθήκη. Στη συνέχεια καλούμε τη μέθοδο που θα λύσει το πρόβλημα με ορίσματα το όνομα του function που επιστρέφει την παράγωγο $f(t, y)$, το διάστημα της λύσης και την αρχική τιμή. Η μέθοδος που επιλέγουμε εδώ είναι η **ode45**, που είναι κατάλληλη για γενικού τύπου διαφορικές εξισώσεις, αλλά υπάρχουν και άλλες διαθέσιμες που μπορεί να επιλεγούν ανάλογα με τη φύση του προβλήματος. Στη συνέχεια σχεδιάζουμε τη προσεγγιστική λύση στα σημεία της διαμέρισης του διαστήματος λύσης tspan. Στη συνέχεια, και για λόγους σύγκρισης αφού το συγκεκριμένο πρόβλημα είναι σχετικά απλό και λύνεται συμβολικά, υπολογίζουμε την ακριβή λύση και τη σχεδιάζουμε στο ίδιο γράφημα. Εφόσον γνωρίζουμε τη λύση μπορούμε να υπολογίσουμε την μέγιστη τιμή του απόλυτου σφάλματος της προσεγγιστικής λύσης στα σημεία που προσεγγίστηκε η λύση. Για να εκτελεστούν όλα αυτά γράφουμε:

```
>> solveode1
```

Στο γράφημα που παίρνουμε από την εκτέλεση του script μπορούμε να δούμε την προσέγγιση της λύσης στα σημεία της διαμέρισης (κόκκινες τελείες) του διαστήματος λύσης που επέλεξε η μέθοδος που υλοποιεί η **ode45** αλλά και την ακριβή λύση (μπλε γραμμή). Αλλού τα σημεία αυτά είναι πιο πυκνά ενώ

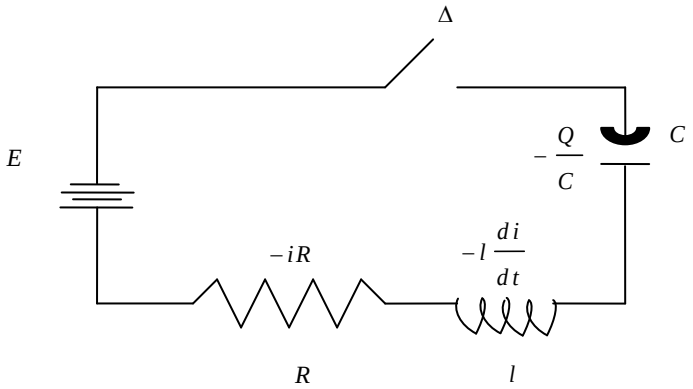
αλλού πιο αραιά ανάλογα με τη μορφή της λύσης. Στο παράθυρο εντολών εμφανίζεται το μέγιστο απόλυτο σφάλμα **maxerror =5.3104e-004** .

Σημείωση:
Θα μπορούσαμε να μην είχαμε δημιουργήσει ξεχωριστό function το οποίο να υλοποιούσε το δεξί μέρος της διαφορικής εξίσωσης, αλλά να το ορίσουμε μέσα στο script με την inline όπως φαίνεται στον κώδικα που ακολουθεί, και είναι διαφορετικός μόνο σε δύο εντολές:

```
% Lynei th diaforikh
clear all;
clf;
tspan=[0,15];
y0=[2];
myfunc=inline(vectorize('-4*y+2*cos(2*t).*(t>=0)'));
[t,y]=ode45(myfunc,tspan,y0);
plot(t,y,'r.')
xlabel('t')
ylabel('i(t)')
grid on
hold
syms tt  yy
s1=dsolve('Dyy=-4*yy+2*cos(2*tt)', 'yy(0)=2 ', 'tt')
ezplot(s1,[0,10])
```

Αριθμητική επίλυση διαφορικής εξίσωσης 2ας τάξης

Έστω ένα κύκλωμα το οποίο αποτελείται από μία πηγή ηλεκτρεργετικής δύναμης *E (Volt)*, η οποία μπορεί να είναι σταθερή ή να εξαρτάται από το χρόνο δηλαδή *E=E(t),* πυκνωτή χωρητικότητας *C (Farad)*, πηνίο αυτεπαγωγής *l (Herny),* ωμική αντίσταση *R (Ohm)* και διακόπτη *Δ*, συνδεδεμένα σε σειρά.



Εφαρμόζουμε τον πρώτο νόμο του Kirchhoff ο οποίος μας δίνει

$$E-\frac{1}{C}\cdot Q-l\frac{di}{dt}-R\cdot i=0\Rightarrow \frac{di}{dt}+\frac{1}{lC}Q+\frac{R}{l}\cdot i=\frac{E}{l},$$

Από $i=\frac{dQ}{dt}\Rightarrow \frac{di}{dt}=\frac{d^2Q}{dt^2}$ καταλήγουμε στην διαφορική εξίσωση:

$$\frac{d^2Q}{dt^2}+\frac{1}{lC}Q+\frac{R}{l}\cdot\frac{dQ}{dt}=\frac{E}{l}$$

Αυτή είναι μία διαφορική εξίσωση της μορφής $y''(t) + ay'(t) + by(t) = e(t)$, όπου $y(t) = Q(t), \quad a = \frac{1}{lC}$,

$b = \frac{R}{l}$ και $e(t) = \frac{E}{l}$. Όταν το συνδυάσουμε με δύο αρχικές τιμές $\mathrm{Q}(0) = \mathrm{C}_1, \mathrm{Q}'(0) = \mathrm{C}_2$ τότε γίνεται ένα πρόβλημα αρχικών τιμών δευτέρας τάξης με σταθερούς όρους.

Εφαρμοσμένο Παράδειγμα 2:

Για παράδειγμα εάν το κύκλωμα RLC αποτελείται από μία πηγή ηλεκτρεργετικής δύναμης *E=6cos(t) Volt* σταθερή, πυκνωτή χωρητικότητας *C=0.125 Farad*, πηνίο αυτεπαγωγής *l=2 Herny,* ωμική αντίσταση *R=10 Ohm* και διακόπτη *Δ*. Έστω ότι $Q(0) = 2, Q'(0) = -5$, τότε η διαφορική εξίσωση που έχουμε να λύσουμε είναι η

$$\frac{d^2Q}{dt^2} + 5 \cdot \frac{dQ}{dt} + 4Q(t) = 3\cos(t), \qquad Q(0) = 2, Q'(0) = -5$$

ή ισοδύναμα

$$\frac{d^2Q}{dt^2} = -5 \cdot \frac{dQ}{dt} - 4Q(t) + 3\cos(t), \qquad Q(0) = 2, Q'(0) = -5$$

Θεωρώντας ως $y_1(t) = Q(t)$ τη λύση της δ.ε. και ως $y_2(t) = \frac{dQ}{dt}$ την παράγωγό της, γράφουμε την παραπάνω εξίσωση στη μορφή ενός συστήματος δύο διαφορικών εξισώσεων πρώτης τάξης:

$$\begin{bmatrix} \frac{dy_1}{dt} \\ \frac{dy_2}{dt} \end{bmatrix} = \begin{bmatrix} y_2(t) \\ -5y_2(t) - 4y_1(t) + 3\cos(t) \end{bmatrix}, \qquad y_1(0) = 2, y_2(0) = -5$$

και στη συνέχεια την λύνουμε προσεγγιστικά («αριθμητικά») όπως κάναμε και στο προηγούμενο παράδειγμα.

Για να λύσουμε «αριθμητικά» το παραπάνω πρόβλημα στο διάστημα $0 \le t \le 20$ δημιουργούμε σε ένα m-αρχείο με όνομα myfunc2.m την παρακάτω function η οποία υλοποιεί το δεξί μέρος της δ.ε.:

```
function ydot = myfunc2(t,y)
% myfunc2
ydot(1,1) = y(2);
ydot(2,1) = -5*y(2)-4*y(1)+3*cos(t);
```

Παρατηρούμε ότι ή έξοδος της συνάρτησης, το διάνυσμα ydot, είναι ένα διάνυσμα στήλη με δύο στοιχεία.

Στη συνέχεια θα πρέπει να δημιουργήσουμε και να εκτελέσουμε ένα αρχείο εντολών script, το οποίο θα ονομάσουμε solveode2.m για το παράδειγμά μας, με περιεχόμενα:

```
% Lynei th diaforikh 2as taxis
clear all;
clf;
tspan=[0,20];
y0=[2,-5];
[t,y]=ode45('myfunc2',tspan,y0);
plot(t,y(:,1), '.')
xlabel('t')
ylabel('Q(t)')
grid on
```

Το διάνυσμα της λύσης y(t) που επιστρέφεται έχει δύο στήλες. Στην πρώτη επιστρέφεται η προσέγγιση της λύσης του προβλήματος και στη δεύτερη η προσέγγιση της παραγώγου της λύσης.

Για τη συγκεκριμένη διαφορική εξίσωση είναι γνωστό ότι η λύση της είναι η

$$\boldsymbol{Q(t) = \frac{1}{2}e^{-t} + \frac{21}{17}e^{-4t} + \frac{9}{34}cos(t) + \frac{15}{34}sin(t),\ \ t \geq 0}$$

Αυτό είναι εύκολο να το διαπιστώσουμε είτε λύνοντας την εξίσωση συμβολικά είτε με τις ακόλουθες εντολές οι οποίες αντικαθιστούν τη λύση στον τύπο της διαφορικής εξίσωσης:

```
>> clear all
>> syms t
>> q=1/2*exp(-t)+21/17*exp(-4*t)+9/34*cos(t)+15/34*sin(t)
>> diff(q,t,2)+5*diff(q,t)+4*q-3*cos(t)
```

Το ακόλουθο script λύνει τη διαφορική και συγκρίνει γραφικά τις τιμές της ακριβούς λύσης (συνεχής πράσινη γραμμή) και της αριθμητικής (με τελίτσες) στο πρώτο γράφημα. Στο δεύτερο γράφημα σχεδιάζει το απόλυτο σφάλμα, δηλαδή απόλυτη τιμή της διαφοράς μεταξύ της προσέγγισης και της τιμής της λύσης σε κάθε σημείο της διαμέρισης. Πληκτρολογήστε το, αποθηκεύστε το με όνομα solveode2a.m και στη συνέχεια εκτελέστε το.

```
% Lynei th diaforikh 2as taxis
clear all;
clf;
tspan=[0,20];
y0=[2,-5];
[t,y]=ode45('myfunc2',tspan,y0);
truesol=1/2*exp(-t)+21/17*exp(-4*t)+9/34*cos(t)+15/34*sin(t);
subplot(2,1,1)
plot(t,y(:,1),'.',t,truesol)
xlabel('t')
ylabel('Q(t)')
grid on
subplot(2,1,2)
plot(t,abs(y(:,1)-truesol));
xlabel('t')
ylabel('Q(t)')
maxerr=max(abs(y(:,1)-truesol))
```

Γράφημα του απόλυτου σφάλματος της προσεγγιστικής λύσης

Μέγιστη τιμή του απόλυτου σφάλματος της προσεγγιστικής λύσης

Παρατηρήστε ότι οι τελίτσες της αριθμητικής λύσης είναι πάνω στη θεωρητική λύση και δείτε στο παράθυρο εντολών
maxerr =

3.6808e-004
το μέγιστο απόλυτο σφάλμα της αριθμητικής λύσης στα σημεία της διαμέρισης.

Άσκηση για εξάσκηση:

Στο φάκελο c:\temp, ή σε οποιονδήποτε άλλο φάκελο που χρησιμοποιείτε για να αποθηκεύετε τα δεδομένα σας, του υπολογιστή σας δημιουργήστε ένα φάκελο με όνομα askode_____ (όπου μετά το am ακολουθεί ο αριθμός μητρώου σας). Εκεί δημιουργήστε όλα τα αρχεία που αφορούν την συγκεκριμένη άσκηση.

Ακολουθώντας τη μεθοδολογία του εφαρμοσμένου παραδείγματος 1, λύστε με τη βοήθεια του MATLAB ακριβώς και προσεγγιστικά, για $0 \leq t \leq 40$, το ακόλουθο πρόβλημα αρχικών τιμών:

$$\frac{dy}{dt} = -0.2\,y(t) + 2e^{-2t}\sin(2t), \quad y(0)=0.01$$

Εμφανίστε το γράφημα της ακριβής και των σημείων προσεγγιστικής λύσης σε ένα γράφημα και υπολογίστε το μέγιστο απόλυτο σφάλμα της προσεγγιστικής λύσης στα σημεία της διαμέρισης που επιστρέφει η `ode45`.