Τεχνολογικο Εκπαιδευτικο Ίδρυμα Κεντρικης Μακεδονιας - Σερρες
Τμημα Μηχανικων Πληροφορικης

Αριθμητικές Μέθοδοι σε Προγραμματιστικό Περιβάλλον

Matlab - Εργαστηριακος Οδηγος

Δρ. Δημήτριος Βαρσάμης

22 Οκτωβρίου 2015

Εισαγωγή

Το πακέτο λογισμικού MATLAB (MathWorks Inc.) παρέχει ένα δυναμικό, εύχρηστο και ανοικτό υπολογιστικό περιβάλλον για υλοποίηση επιστημονικών εφαρμογών σε ένα μεγάλο φάσμα γνωστικών πεδίων, όπως στη Γραμμική Άλγεβρα, τη Στατιστική, τα Εφαρμοσμένα Μαθηματικά, την Αριθμητική Ανάλυση, την Επεξεργασία Σημάτων και Εικόνας, τη Θεωρία Ελέγχου. Έχει υλοποιηθεί σε πολλές λειτουργικές πλατφόρμες (όπως Windows, Macintosh OS και Unix) και σε δύο βασικές εκδόσεις, την επαγγελματική και την εκπαιδευτική (student edition).

Το περιβάλλον του MATLAB υποστηρίζει ένα μεγάλο αριθμό ενδογενών λειτουργιών και συναρτήσεων, καθώς και εξωτερικές βιβλιοθήκες (Toolboxes) για εξειδικευμένες περιοχές εφαρμογών. Υποστηρίζει επίσης μία ευέλικτη, απλή και δομημένη γλώσσα προγραμματισμού (script language), η οποία έχει πολλές ομοιότητες με τη γλώσσα C, και παρέχει δυνατότητες εύκολης δημιουργίας, διασύνδεσης και χρήσης βιβλιοθηκών σε κώδικα γραμμένο στη γλώσσα αυτή (εφεξής θα ονομάζεται γλώσσα MATLAB).

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

Οι παρούσες σημειώσεις, οι οποίες αποσκοπούν αφενός μεν στην εξοικείωση του χρήστη με το πρόγραμμα MATLAB, αφετέρου δε στην εφαρμογή αυτού στην Αριθμητική Ανάλυση, αποτελούνται από δύο μέρη: Στο πρώτο μέρος, παρουσιάζονται οι βασικές εντολές, λειτουργίες και χαρακτηριστικά του MATLAB. Συγκεκριμένα, περιγράφονται οι υποστηριζόμενοι τελεστές και πράξεις, οι στοιχειώδεις μαθηματικές συναρτήσεις και οι εντολές που περιλαμβάνει η ενσωματωμένη γλώσσα προγραμματισμού. Ιδιαίτερη έμφαση δίνεται στο λογισμό και τις πράξεις πινάκων. Τέλος δίνονται οι τρόποι σχεδίασης γραφικών παραστάσεων και κατασκευής συναρτήσεων.

Στο δεύτερο μέρος παρουσιάζονται μερικές αντιπροσωπευτικές και απλές εφαρμογές του MATLAB στη Γραμμική Άλγεβρα και τις Αριθμητικές Μεθόδους. Ειδικότερα, παρουσιάζονται όλες οι Αριθμητικές Μέθοδοι που αναπτύσσονται στο εργαστηριακό μέρος του μαθήματος.

Περιεχόμενα

Μέρος Ι

Το λογισμικό Matlab

Κεφάλαιο 1

Εισαγωγή στο Matlab

Το Matlab είναι ένα λογισμικό που απευθύνεται σε επιστήμονες για την έρευνα τους, σε καθηγητές για την εκπαίδευση αλλά και σε φοιτητές. Το μεγάλο πλεονέκτημα του Matlab έναντι των κλασσικών γλωσσών προγραμματισμού είναι ότι λειτουργεί με διερμηνευτή (Interpreter) και όχι με μεταγλωττιστή (Compiler). Έτσι έχουμε την αμεσότητα επικοινωνίας χρήστη με Η/Υ χωρίς να απαιτούνται ιδιαίτερες γνώσεις από τον χρήστη.

1.1 Το υπολογιστικό περιβάλλον του Matlab

Με την ενεργοποίησή του το Matlab εμφανίζει[1] ένα περιβάλλον με διάφορα παράθυρα. Το βασικό παράθυρο είναι το Command Window το οποίο είναι το παράθυρο εντολών του Matlab. Κατά τη διάρκεια της εργασίας μας το περιβάλλον αποθηκεύει τις εντολές που έχουν δοθεί (παράθυρο Command History), όπως και τις μεταβλητές που έχουν δημιουργηθεί (παράθυρο Workspace) και μπορεί να τις ανακαλέσει. Οι εντολές και οι μεταβλητές αυτές αποτελούν το χώρο εργασίας (Workspace) του Matlab.
Η ανάθεση τιμής σε μια μεταβλητή γίνεται με το (=):

```
>> a=3

a =
     3
```

Παρατηρούμε ότι στο Command Window του Matlab γράφουμε την εντολή και με το Enter (↵) μας επιστρέφει το αποτέλεσμα της εντολής. Στην περίπτωση που δεν θέλουμε να εμφανιστεί το αποτέλεσμα της εντολής τοποθετούμε στο τέλος της εντολής το (;).
Προσοχή στην ονοματολογία των μεταβλητών. Μόνο λατινικοί χαρακτήρες, όχι κενά, όχι σύμβολα (επιτρέπεται η κάτω παύλα), να ξεκινά από γράμμα και υπάρχει διαχωρισμός μεταξύ πεζών κεφαλαίων (Case sensitive).

[1] Αναλόγως την έκδοση

Τα ονόματα των μεταβλητών που έχουν δημιουργηθεί εμφανίζονται με την εντολή `who`:

```
>> who

Your variables are:

a
```

Εάν θέλουμε να δούμε περισσότερες πληροφορίες για τις μεταβλητές που έχουν δημιουργηθεί εμφανίζονται με την εντολή `whos`:

```
>> whos
  Name        Size                Bytes  Class     Attributes

  a           1x1                     8  double
```

Για να *καθαρίσουμε* κάποιες από τις μεταβλητές που έχουμε δημιουργήσει, π.χ. τη μεταβλητή `a`, καλούμε την εντολή `clear a`, ενώ για να *καθαρίσουμε* όλες τις μεταβλητές που έχουμε δημιουργήσει καλούμε την `clear`.

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

Μεταβλητή	Τιμή
`ans`	Το αποτέλεσμα κάθε εντολής που δεν εκχωρείται σε μεταβλητή
`pi`	Ο αριθμός π
`eps`	Ο κοντινότερος αριθμός στο 0 (Ανοχή)
`inf`	Άπειρο
`NaN`	Μη-αριθμός (π.χ. 0/0) (Not a Number)
`i` και `j`	Φανταστική μονάδα
`Realmin`	Ο μικρότερος θετικός πραγματικός αριθμός
`Realmax`	Ο μεγαλύτερος θετικός πραγματικός αριθμός

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

```
>> help clear
 CLEAR  Clear variables and functions from memory.
    CLEAR removes all variables from the workspace.
    CLEAR VARIABLES does the same thing.
    CLEAR GLOBAL removes all global variables.
    CLEAR FUNCTIONS removes all compiled M- and MEX-functions.
    .
    .
    .
    .
    See also clearvars, who, whos, mlock, munlock, persistent.
```

```
    Overloaded methods:
       mbcstore/clear
       cgrules/clear
       cgoptimexprgroup/clear
       xregdesign/clear
       cgexprgroup/clear
       xregtable/clear

    Reference page in Help browser
       doc clear
```

Για περισσότερες επιλογές εύρεσης εντολών, συναρτήσεων, παραδειγμάτων και demos, από την εργαλειοθήκη επιλέγουμε το εικονίδιο της βοήθειας και ανοίγουμε το περιβάλλον της βοήθειας.

Ως path καλείται το μονοπάτι εύρεσης, στο οποίο ψάχνει το MATLAB το όνομα μίας εντολής που εισάγεται ή το όνομα ενός εκτελέσιμου αρχείου. Η εντολή `path` εμφανίζει τη λίστα των ορισθέντων μονοπατιών:

```
>> path

  MATLABPATH

  C:\Documents and Settings\AdministratorTα\ έγγραφάμου \MATLAB
  C:\Documents and Settings\AdministratorTα\ έγγραφάμου \My Dropbox\math_auth\matlab
  .
  .
  .
  .
  C:\Program Files\MATLAB\R2009b\toolbox\rtw\targets\xpc\xpcdemos
  C:\Program Files\MATLAB\R2009b\toolbox\rtw\targets\xpc\xpc\xpcmngr
  C:\Program Files\MATLAB\R2009b\toolbox\rtw\targets\xpc\target\kernel\embedded
```

Για να προσθέσουμε ένα μονοπάτι στα τρέχοντα μονοπάτια, δίνουμε την εντολή `addpath(path)`.
Για παράδειγμα, δίνοντας `addpath('C:\temp')`, προσθέτουμε το μονοπάτι `C:\temp` στα τρέχοντα μονοπάτια.
Ενώ, για να αφαιρέσουμε ένα μονοπάτι από τα τρέχοντα μονοπάτια, δίνουμε την εντολή `rmpath(path)`.
Για παράδειγμα, δίνοντας `rmpath('C:\temp')`, αφαιρούμε το μονοπάτι `C:\temp` από τα τρέχοντα μονοπάτια.

Η εντολή `matlabroot` μας επιστρέφει τον κατάλογο εγκατάστασης του MATLAB. Για παράδειγμα

```
>> matlabroot

ans =

C:\Program Files\MATLAB\R2009b
```

Η εντολή `dir` εμφανίζει όλα τα αρχεία του τρέχοντος καταλόγου, ενώ η εντολή `cd` μας επιστρέφει τον τρέχων κατάλογο εργασίας. Η εντολή `what`

εμφανίζει μόνο τα αρχεία του MATLAB (*.m , *.mat).
Για παράδειγμα,

```
>> dir

.                              fig4.png                    rec_typ_d2_par.m
..                             fig5.png                    slides2.m
GUI_v3                         fig7.png                    slides_num_interg.asv
P.mat                          fig8.png                    slides_num_interg_simp.asv
Untitled.m                     fig9.png                    slides_num_interg_simp.m
alg_dif_new.m                  find_plane.m                slides_num_interg_trap.m
alg_dif_new_par.m              fun_aprox.m                 smith_examples.asv
create_matrices_100_100.asv    matrix_abc.m                smith_examples.m
create_p.m                     paral_1.m                   smith_test.mn
dist_1_all.m                   paral_2.m                   smith_test_1.mn
doolittle.m                    paral_3.m                   smith_test_2.mn
dsp_theory_7.m                 paral_3a.m                  smith_test_3.mn
exam2012_ear1_a.m              prime_number.m              smith_test_4.mn
exams_1_a.m                    prime_number_parallel.m     smith_test_mitrouli_2.mn
fig1.png                       rec_typ_d.m                 str.mat
fig2.png                       rec_typ_d1.m                triagular_basis.m
fig3.png                       rec_typ_d1_par.m
fig4.eps                       rec_typ_d2.m
```

```
>> cd

C:\Documents and Settings\AdministratorTα\ έγγραφάμου \MATLAB
```

```
>> what

M-files in the current directory C:\Documents and Settings\AdministratorTα\
    έγγραφάμου \MATLAB

Untitled                  fun_aprox                 rec_typ_d1_par
alg_dif_new               matrix_abc                rec_typ_d2
alg_dif_new_par           paral_1                   rec_typ_d2_par
create_p                  paral_2                   slides2
dist_1_all                paral_3                   slides_num_interg_simp
doolittle                 paral_3a                  slides_num_interg_trap
dsp_theory_7              prime_number              smith_examples
exam2012_ear1_a           prime_number_parallel     triagular_basis
exams_1_a                 rec_typ_d
find_plane                rec_typ_d1

MAT-files in the current directory C:\Documents and Settings\AdministratorTα\
    έγγραφάμου \MATLAB

P        str
```

Οι εντολές `date` και `clock` μας εμφανίζουν την τρέχουσα ημερομηνία και ώρα αντιστοίχως.

Τα περιεχόμενα της οθόνης μπορούμε να τα σβήσουμε με τις εντολές `clc` ή `home`, οι οποίες δεν επηρεάζουν της μεταβλητές.

Η εμφάνιση των αριθμών διέπεται από ορισμένους κανόνες. Εκτός κι αν ορισθεί διαφορετικά, αν ένα αποτέλεσμα είναι ακέραιος εμφανίζεται ως ακέραιος. Παρόμοια, αν το αποτέλεσμα είναι πραγματικός, εμφανίζεται με 5 ψηφία. Αν τα σημαντικά ψηφία του αποτελέσματος είναι έξω από αυτό το όριο, το αποτέλεσμα εμφανίζεται σε μορφή αριθμού κινητής υποδιαστολής.

Ο χρήστης όμως μπορεί να ορίσει τη μορφή του αποτελέσματος με την εντολή `format`, η οποία έχει τις ακόλουθες μορφές:

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

Η εντολή `format` χωρίς κάποια άλλη προέκταση μας εμφανίζει τους αριθμούς στην μορφή `short`, η οποία είναι η προεπιλεγμένη μορφή (default).

1.2 Τελεστές και ειδικά σύμβολα

Οι υποστηριζόμενοι τελεστές και ειδικά σύμβολα με τη σημασία τους παρατίθενται παρακάτω.

- Αριθμητικοί Τελεστές

Χαρακτήρας	Λειτουργία
`+`	Πρόσθεση
`-`	Αφαίρεση
`*`	Πολλαπλασιασμός αριθμών ή πινάκων
`.*`	Πολλαπλασιασμός πινάκων ανά στοιχείο
`^`	Ύψωση σε δύναμη αριθμού ή πίνακα
`.^`	Ύψωση σε δύναμη πίνακα ανά στοιχείο
`\`	Διαίρεση από τα αριστερά
`/`	Διαίρεση από τα δεξιά
`./`	Διαίρεση πινάκων ανά στοιχείο

- Συγκριτικοί Τελεστές

Χαρακτήρας	Λειτουργία
`==`	Έλεγχος ισότητας
`~=`	Έλεγχος μη ισότητας
`< , >, >=, <=`	Συγκριτικοί τελεστές

- Λογικοί Τελεστές

Χαρακτήρας	Λειτουργία
`~`	Λογικό NOT
`||`	Λογικό OR
`&&`	Λογικό AND
`xor`	Αποκλειστικό OR

- Ειδικά σύμβολα

Χαρακτήρας	Λειτουργία
:	Άνω και κάτω τελεία (σύμβολο περιοχής)
( )	Παρενθέσεις
[ ]	Αγκύλες
.	Δεκαδική υποδιαστολή
,	Διαχωριστής στοιχείων
;	Διαχωριστής γραμμών κατά τη δημιουργία πίνακα
%	Εισαγωγή σχολίων
!	Εισαγωγή εντολών στο λειτουργικό σύστημα
'	Μετατροπή αριθμού ή στοιχείων πίνακα σε συζυγείς
=	Απόδοση τιμής

1.2.1 Αριθμητικές πράξεις

Για την εκτέλεση απλών αριθμητικών πράξεων μεταξύ πραγματικών και ακεραίων χρησιμοποιούνται οι συνήθεις τελεστές. Για παράδειγμα:

```
>> 2^3

ans =

     8

>> 2/3

ans =

          0.666666666666667

>> 2\3

ans =

                        1.5

>> mod(2,3)

ans =

     2
```

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

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

Για να μην εμφανίζεται στην οθόνη το αποτέλεσμα μίας εντολής όπως προαναφέραμε, αρκεί να τεθεί στο τέλος της το σύμβολο (`;`).
Οι μεταβλητές στο MATLAB μπορούν να είναι βαθμωτά μεγέθη ή μιγαδικοί, όπως φαίνεται ακολούθως:

```
>> a=2^3

a =

     8

>> b=3^2;
>> a+b

ans =

    17

>> c=2*a-b^2

c =

   -65

>> w=3+4i

w =

                            3 +                         4i

>> w*w

ans =

                           -7 +                        24i

>> w*w'                                (γινόμενο με το συζυγή)

ans =

    25
```

Πολλές από τις πράξεις που μπορούμε να εκτελέσουμε στο MATLAB θα μας οδηγήσουν σε κάποια *απροσδιόριστα* αποτελέσματα:

```
>> 5/0                    (Η διαίρεση με το μηδέν δίνει άπειρο)

ans =

   Inf

>> 0/0                                   (Απροσδιόριστη μορφή)

ans =

   NaN

>> inf/inf                               (Απροσδιόριστη μορφή)

ans =

   NaN

>> sqrt(-9)                              (Μιγαδικό αποτέλεσμα)

ans =

                          0 +                          3i
```

1.2.2 Λογικές πράξεις

Στο MATLAB έχουμε τη δυνατότητα να κάνουμε συγκρίσεις με λογικές εκφράσεις απευθείας στη γραμμή εντολών

```
>> 2>3

ans =

     0

>> 2^3>=3^2

ans =

     0

>> 3-4~=4-3
```

```
ans =

     1

>> 4*4==2*2*2

ans =

     0

>> (2==3)||(4^2-2^4==0)

ans =

     1

>> (2==3)&&(4^2-2^4==0)

ans =

     0

>> ~(2==3)&&(4^2-2^4==0)

ans =

     1

>> (2==3)||~(4^2-2^4==0)

ans =

     0
```

Η τιμή `1` αντιστοιχεί στο **Αληθές** και η τιμή `0` στο **Ψευδές**.

1.3 Βασικές συναρτήσεις

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

Συνάρτηση	Λειτουργία
`sin`	Ημίτονο
`sinh`	Υπερβολικό ημίτονο
`asin`	Αντίστροφο ημίτονο
`asinh`	Αντίστροφο υπερβολικό ημίτονο
`cos`	Συνημίτονο
`cosh`	Υπερβολικό συνημίτονο
`acos`	Αντίστροφο συνημίτονο
`acosh`	Αντίστροφο υπερβολικό συνημίτονο
`tan`	Εφαπτομένη
`tanh`	Υπερβολική εφαπτομένη
`atan`	Αντίστροφη εφαπτομένη
`atanh`	Αντίστροφη Υπερβολική εφαπτομένη
`sec`	Τέμνουσα
`sech`	Υπερβολική τέμνουσα
`asec`	Αντίστροφη τέμνουσα
`asech`	Αντίστροφη υπερβολική τέμνουσα
`csc`	Συντέμνουσα
`csch`	Υπερβολική συντέμνουσα
`acsc`	Αντίστροφη συντέμνουσα
`acsch`	Αντίστροφη υπερβολική συντέμνουσα
`cot`	Συνεφαπτομένη
`coth`	Υπερβολική συνεφαπτομένη
`acot`	Αντίστροφη συνεφαπτομένη
`acoth`	Αντίστροφη υπερβολική συνεφαπτομένη
`exp`	Εκθετική συνάρτηση (e^x)
`log`	Νεπέριος (φυσικός) λογάριθμος (lnx)
`log10`	Δεκαδικός λογάριθμος ($\log(x)$)
`log2`	Λογάριθμος με βάση το 2 ($\log_2(x)$)
`sqrt`	Τετραγωνική ρίζα
`abs`	Απόλυτη τιμή
`angle`	Πρωτεύων όρισμα μιγαδικού
`conj`	Συζυγής μιγαδικού
`imag`	Φανταστικό μέρος μιγαδικού
`real`	Πραγματικό μέρος μιγαδικού
`fix`	Ακέραιο μέρος
`floor`	Κάτω ακέραιο μέρος
`ceil`	Άνω ακέραιο μέρος
`round`	Στρογγυλοποίηση σε ακέραιο
`rem`	Υπόλοιπο διαίρεσης
`sign`	Πρόσημο

Οι βασικές συναρτήσεις καλούνται με το όνομά τους καθορίζοντας μέσα στις παρενθέσεις τα ορίσματα τους.

Παραδείγματα κλήσης βασικών συναρτήσεων

Τριγωνομετρικές συναρτήσεις

```
>> sin(0)

ans =

     0

>> cos(pi/3)                          (Υπολογισμός του cos(π/3))

ans =

     0.5

>> tan(pi/4)                          (Υπολογισμός του tan(π/4))

ans =

     1

>> sin(2)^2+cos(2)^2                  (επαλήθευση ταυτότητας)

ans =

     1
```

Λογαριθμικές και Εκθετικές συναρτήσεις

```
>> exp(1)                                  (Υπολογισμός του e)

ans =

          2.71828182845905

>> log(exp(2))                        (Υπολογισμός του lne^2 = 2)

ans =

     2

>> log10(1000)                      (Υπολογισμός του log 1000 = 3)

ans =

     3
```

```
>> log2(1024)                    (Υπολογισμός του log2 1024 = 10)
ans =
    10
```

Συναρτήσεις για μιγαδικούς αριθμούς

```
>> x=3+4i
x =
     3 +                    4i
>> angle(x)                                   (Όρισμα του x)
ans =
         0.927295218001612
>> abs(x)                                     (Μέτρο του x)
ans =
     5
>> real(x)                          (Πραγματικό μέρος του x)
ans =
     3
>> imag(x)                          (Φανταστικό μέρος του x)
ans =
     4
>> conj(x)                                    (Συζυγής του x)
ans =
     3 -                    4i
```

Διάφορες συναρτήσεις

```
>> x=6.25

x =

                    6.25

>> sqrt(x)

ans =

                     2.5

>> fix(x)                                   (ακέραιο μέρος)

ans =

     6

>> ceil(x)                                (άνω ακέραιο φράγμα)

ans =

     7

>> floor(x)                               (κάτω ακέραιο φράγμα)

ans =

     6

>> round(x)                        (στρογγυλοποίηση σε ακέραιο)

ans =

     6
```

1.4 Ορισμός μαθηματικών συναρτήσεων

Ο πιο απλός τρόπος να ορίσουμε κάποιες συναρτήσεις (κυρίως μαθηματικές) στο MATLAB είναι με τη χρήση του ***inline object***.

Η σύνταξη της εντολής είναι `f=inline('math expression')`. όπου `f` το όνομα της inline συνάρτησης και `math expression` μια μαθηματική έκφραση.

Η κλήση της inline συνάρτησης πραγματοποιείται ως εξής `f(list of value)`.
Για παράδειγμα, έχουμε την συνάρτηση

$$f(x) = x^2 + 3x$$

και θέλουμε να υπολογίσουμε την τιμή της για $x = 10$, για $x = 1, 2, \ldots, 5$ και την τιμή της παράστασης

$$2 \cdot f^2(3) - 5 \cdot f(10)$$

```
>> f=inline('x.^2+3*x')

f =

     Inline function:
     f(x) = x.^2+3*x

>> f(10)

ans =

   130

>> x=1:5

x =

     1     2     3     4     5

>> f(x)

ans =

     4    10    18    28    40

>> 2*f(3)^2-5*f(10)

ans =

    -2
```

Η inline συνάρτηση μπορεί να έχει παραπάνω από μια μεταβλητές. Για παράδειγμα έχουμε την συνάρτηση

$$g(x, y) = x^2 + y^2$$

και θέλουμε να υπολογίσουμε την τιμή της για $x = 2$ και $y = 3$.

```
>> g=inline('x.^2+y.^2')

g =

     Inline function:
     g(x,y) = x.^2+y.^2

>> g(2,3)

ans =

    13
```

Επίσης, στην inline συνάρτηση μπορούμε να χρησιμοποιήσουμε και συναρτήσεις του MATLAB. Για παράδειγμα έχουμε την συνάρτηση

$$h(x) = x^2 + \sin(2x) + e^{x^2+1}$$

και θέλουμε να υπολογίσουμε την τιμή της για $x = 0$.

```
>> h=inline('x.^2+sin(2*x)+exp(x.^2+1)')

h =

     Inline function:
     h(x) = x.^2+sin(2*x)+exp(x.^2+1)

>> h(0)

ans =

          2.71828182845905
```

1.5 Πίνακες και επεξεργασία τους

1.5.1 Ορισμός πινάκων

Οι πίνακες είναι το δομικό στοιχείο του MATLAB[2] και όλες οι μεταβλητές εξ' ορισμού δηλώνονται ως πίνακες.
Τα στοιχεία του πίνακα περικλείονται σε αγκύλες (`[]`). Τα στοιχεία της γραμμής (στήλες) χωρίζονται με κόμμα (`,`) ή κενό ενώ τα στοιχεία της στήλης (γραμμές) χωρίζονται με ερωτηματικό (`;`) ή ⏎ (Enter).

[2]Η ονομασία του MATLAB προέρχεται από τη φράση MATrix LABoratory

Για παράδειγμα, ένα διάνυσμα (μονοδιάστατος πίνακας) μπορούμε να το ορίσουμε ως εξής

- Πίνακας γραμμή
  `x=[1 2 3]` ή
  `x=[1, 2, 3]`
- Πίνακας στήλη
  `x=[1; 2; 3]` ή
  `x=[1` ↵ `2` ↵ `3]`

ενώ ένα δισδιάστατο πίνακα ως εξής

- `A=[1 2 3;4 5 6;7 8 9]` ή
  `A=[1,2,3;4,5,6;7,8,9]` ή
  `A=[1,2,3` ↵ `4,5,6` ↵ `7,8,9]` ή
  `A=[1 2 3` ↵ `4 5 6` ↵ `7 8 9]`

Τα στοιχεία ενός πίνακα αποθηκεύονται κατά στήλες. Ο κενός πίνακας συμβολίζεται με `[]`.

Παράδειγμα 1.5.1. *Δίνονται οι παρακάτω πίνακες*

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

$$x = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}, \quad y = \begin{bmatrix} 5 \\ 6 \\ 7 \end{bmatrix}$$

Να κατασκευαστούν

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

A =

     1     2     3
     4     5     6

>> B=[0 1 0;1 0 1]

B =

     0     1     0
     1     0     1

>> C=[1,2;2,1]
```

```
C =

     1     2
     2     1

>> x=[1,2,3]

x =

     1     2     3

>> y=[5;6;7]

y =

     5
     6
     7
```

♦

Το Matlab μας δίνει τη δυνατότητα να ορίσουμε πίνακες των οποίων τα στοιχεία να είναι ακολουθίες αριθμών.
Για παράδειγμα, μπορούμε να ορίσουμε διανύσματα με στοιχεία αριθμούς που μεταβάλλονται με ένα συγκεκριμένο βήμα, δηλαδή,

- `x=[a:s:b]` ή `x=a:s:b`
- `a` αρχική τιμή, `s` βήμα, `b` τελική τιμή.
- Όταν το βήμα (`s`) παραλείπεται το Matlab το θεωρεί ίσο με τη μονάδα.

```
>> x=1:10

x =

     1     2     3     4     5     6     7     8     9    10

>> x=1:0.1:2

x =

  Columns 1 through 4

     1             1.1             1.2             1.3
```

```
  Columns 5 through 8

   1.4             1.5             1.6             1.7

  Columns 9 through 11
   1.8             1.9               2

>> x=10:-1:1

x =

    10     9     8     7     6     5     4     3     2     1
```

Επίσης, μπορούμε να δημιουργήσουμε ένα διάνυσμα με στοιχεία ένα συγκεκριμένο πλήθος αριθμών που ανήκουν στο διάστημα $[a, b]$ και ισαπέχουν μεταξύ τους, δηλαδή,

- `x=linspace(a,b,n)`
- γραμμικό διάστημα (ακολουθία ισαπέχοντων αριθμών) με `a` πρώτο αριθμό, `b` τελευταίο αριθμό, `n` πλήθος αριθμών.

```
>> x=linspace(1,10,7)

x =

  Columns 1 through 3

                        1                        2.5
                                                   4

  Columns 4 through 6

                      5.5                         7
                                                8.5

  Column 7

                       10
```

1.5.2 Προσπέλαση και διαχείριση πινάκων

Για να προσπελασθεί ένας πίνακας δηλώνονται μέσα σε παρένθεση οι δείκτες του. Η αναφορά σε τμήματα στοιχείων πίνακα γίνεται με χρήση του

συμβόλου (:). Έτσι, με την αναφορά `A(i:k,j:h)` μπορούν να προσπελασθούν τα στοιχεία ενός πίνακα `A` δύο διαστάσεων, τα οποία ανήκουν στις γραμμές `i` έως και `k` και στις στήλες `j` έως και `h`.
Το σύμβολο (:) όταν χρησιμοποιείται μόνο του επιλέγει όλες τις γραμμές ή στήλες.
Για τους πίνακες του παραδείγματος 1.5.1 έχουμε

```
>> A(2,3)

ans =

     6

>> x(3)

ans =

     3

>> A(5)

ans =

     3

>> A(:,3)

ans =

     3
     6

>> B(1:4)

ans =

     0     1     1     0

>> B([1:2],[2:3])

ans =

     1     0
     0     1

>> A([2,1],[3,1])
```

```
ans =

     6     4
     3     1
```

Είναι δυνατό να ενωθούν δύο ή περισσότεροι πίνακες, αρκεί να έχουν τον ίδιο αριθμό γραμμών ή στηλών ανάλογα με τον τρόπο που θα ενωθούν. Για τους πίνακες του παραδείγματος 1.5.1 έχουμε

```
>> [A,B]

ans =

     1     2     3     0     1     0
     4     5     6     1     0     1

>> [A;B]

ans =

     1     2     3
     4     5     6
     0     1     0
     1     0     1

>> [A;x]

ans =

     1     2     3
     4     5     6
     1     2     3

>> [[A;x],y]

ans =

     1     2     3     5
     4     5     6     6
     1     2     3     7
```

1.5.3 Πράξεις με πίνακες

Υποστηρίζονται τρεις κατηγορίες πράξεων σε πίνακες. Οι χρησιμοποιούμενοι τελεστές διακρίνονται και διαφοροποιούνται ανάλογα με την κατη-

γορία. Οι τελεστές είναι οι ακόλουθοι: `+`, `-`, `*`, `^`, `\`, `/`, `.*`, `./`, `.\`, `.^`, `'` και `.'`.

- Πράξεις μεταξύ πινάκων ίδιων διαστάσεων ανά στοιχείο:
  Η πράξη πινάκων ανά στοιχείο εκτελείται μία φορά για κάθε στοιχείο των πινάκων-ορισμάτων και αποθηκεύεται στην αντίστοιχη θέση του πίνακα-αποτελέσματος.
  - Πρόσθεση πινάκων: `A+B`
  - Αφαίρεση πινάκων: `A-B`
  - Πολλαπλασιασμός ανά στοιχείο: `A.*B`
  - Διαίρεση πινάκων ανά στοιχείο: `A./B` ή `B./A`
  - Ύψωση των στοιχείων πίνακα στις δυνάμεις των αντίστοιχων στοιχείων ενός άλλου: `A.^B`
- Πράξεις μεταξύ βαθμωτής μεταβλητής (πραγματικής ή μιγαδικής) και πίνακα:
  - Πρόσθεση βαθμωτού `c` σε (όλα τα στοιχεία) πίνακα: `c+A` ή `A+c`
  - Αφαίρεση πίνακα από βαθμωτό: `c-a`
  - Πολλαπλασιασμός πίνακα με βαθμωτό: `c*A` ή και `c.*A`
  - Διαίρεση πίνακα με βαθμωτό: `A./c` ή και `c.\A`
  - Ύψωση των στοιχείων πίνακα σε μια βαθμωτή δύναμη: `A.^x`
- Σύνθετες πράξεις πινάκων, δηλαδή αυτές που προϋποθέτουν πιο σύνθετους υπολογισμούς:
  - Πολλαπλασιασμός πινάκων (με διαστάσεις που συμφωνούν): `A*B`
  - Αντιστροφή πίνακα: δίνεται από τη συνάρτηση: `inv(A)`
  - Η ορίζουσα πίνακα δίνεται από τη συνάρτηση: `det(A)`
  - Ανάστροφος πίνακας (A^T): `.'`
  - Ερμητιανός πίνακας (Συζυγοανάστροφος A^H): `'`
  - Διαίρεση πινάκων, που έχει το νόημα του πολλαπλασιασμού με τον αντίστροφο πίνακα, εάν αυτός υπάρχει: `A/B` $= A \cdot B^{-1}$, αριστερή διαίρεση, ή `B\A` $= B^{-1} \cdot A$, δεξιά διαίρεση.
  - Ύψωση τετραγωνικού πίνακα σε μια βαθμωτή δύναμη `c`: `A^c`

Για τους πίνακες του παραδείγματος 1.5.1 έχουμε

```
>> A+B

ans =
```

```
     1     3     3
     5     5     7

>> A.*B

ans =

     0     2     0
     4     0     6

>> A./B

ans =

   Inf     2   Inf
     4   Inf     6

>> A*C
??? Error using ==> mtimes
Inner matrix dimensions must agree.

>> C*A

ans =

     9    12    15
     6     9    12

>> A^B
??? Error using ==> mpower
At least one operand must be scalar.

>> A.^B

ans =

     1     2     1
     4     1     6

>> A'

ans =

     1     4
     2     5
```

```
     3     6
```

1.5.4 Σύγκριση πινάκων και λογικοί πίνακες

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

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

F =

     0     1     0
     0     1     0
     0     0     1

>> F=abs(A)>=2

F =

     0     1     0
     0     1     1
     1     0     1

>> A>2

ans =

     0     0     0
     0     0     1
     1     0     0

>> F=abs(A)>B

F =

     0     1     0
     0     0     1
     0     1     0

>> F=abs(A)~=abs(B)
```

```
F =

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

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

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

A =

     0     2     1
     1    -2     4
     4    -1     2

>> F=A>2

F =

     0     0     0
     0     0     1
     1     0     0

>> A(F)

ans =

     4
     4
```

Ο παραπάνω μηχανισμός επιτρέπει να εξάγουμε τα στοιχεία ενός πίνακα, τα οποία επαληθεύουν μια σύγκριση. Για τον σκοπό αυτό δεν έχουμε παρά να δεικτοδοτήσουμε τον πίνακα με την λογική έκφραση που αναπαριστά τη σύγκριση:

```
>> A(A>2)                          (τα στοιχεία του A με A>2)

ans =

     4
     4
```

```
>> A(A>B)                          (τα στοιχεία του A με A>B)

ans =

     2
    -1
     4
```

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

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

Χρησιμοποιώντας τον πίνακα A των προηγούμενων παραδειγμάτων, έχουμε:

```
>> find(A>2)                  (μονοδιάστατες διευθύνσεις του A)

ans =

     3
     8

>> [i,j]=find(A>2)      (τα i και j περιέχουν τους δείκτες
   γραμμών και στηλών)

i =

     3
     2


j =

     1
     3
```

1.5.5 Ειδικοί πίνακες

Για την κατασκευή ειδικών πινάκων υπάρχουν ειδικές συναρτήσεις. Συγκεκριμένα, μπορούμε να κατασκευάσουμε τους εξής τετραγωνικούς και

μη πίνακες:

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

όπου `k` το μέγεθος τετραγωνικού πίνακα, `n` και `m` ο αριθμός γραμμών και στηλών, αντίστοιχα, ενός πίνακα $n \times m$.

Παραδείγματα Ειδικών πινάκων

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

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

» ones(sizeA())
ans =
         1     1     1
         1     1     1
         1     1     1

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

```
         1     1     1     1     1     1
         1     1     1     1     1     1

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

1.5.6 Χρήσιμες συναρτήσεις πινάκων

Μερικές ιδιαίτερα χρήσιμες συναρτήσεις, που απλουστεύουν το χειρισμό πινάκων, είναι:

Συνάρτηση	Λειτουργία
`size(A)`	επιστρέφει το μέγεθος του πίνακα (πλήθος γραμμών και στηλών).
`size(A,1)`	επιστρέφει το πλήθος των γραμμών του πίνακα A
`size(A,2)`	επιστρέφει το πλήθος των στηλών του πίνακα A.
`length(v)`	επιστρέφει τον αριθμό των στοιχείων ενός διανύσματος v.
`length(A)`	επιστρέφει την μεγαλύτερη από τις διαστάσεις ενός πίνακα $m \times n$.
`flipud(A)`	επιστρέφει το συμμετρικό ενός πίνακα ως προς τις γραμμές.
`fliplr(A)`	επιστρέφει το συμμετρικό ενός πίνακα ως προς τις στήλες.
`rot90(A)`	περιστρέφει ένα πίνακα κατά 90 μοίρες.
`reshape(A,m,n)`	μετασχηματίζει ένα πίνακα διαστάσεων $i \times j$ σε πίνακα διαστάσεων $m \times n$.
`diag(v)`	κατασκευάζει διαγώνιο πίνακα με κύρια διαγώνιο ίση με το διάνυσμα v.
`diag(A)`	Για πίνακα διαστάσεων $m \times n$, εξάγει την κύρια διαγώνιο του σε ένα πίνακα στήλη.
`trace(A)`	επιστρέφει το ίχνος (άθροισμα διαγώνιων στοιχείων) ενός πίνακα δύο διαστάσεων.
`triu(A)`	Άνω τριγωνικός πίνακας που αντιστοιχεί στον πίνακα .
`tril(A)`	Κάτω τριγωνικός πίνακας που αντιστοιχεί στον πίνακα .
`sum(v)`	Για ένα διάνυσμα v υπολογίζει το άθροισμα των στοιχείων του.
`sum(A)`	Για έναν πίνακα δύο διαστάσεων , επιστρέφει έναν πίνακα-γραμμή με τα αθροίσματα των στηλών του.
`sum(A,1)`	Για έναν πίνακα δύο διαστάσεων , επιστρέφει έναν πίνακα-γραμμή με τα αθροίσματα των στηλών του.
`sum(A,2)`	Για έναν πίνακα δύο διαστάσεων A, επιστρέφει έναν πίνακα-στήλη με τα αθροίσματα των γραμμών του.

Παραδείγματα συναρτήσεων πινάκων

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

ans =

     4    -1     2
     1    -2     4
     0     2     1

>> fliplr(A)

ans =

     1     2     0
     4    -2     1
     2    -1     4

>> rot90(A)

ans =

     1     4     2
     2    -2    -1
     0     1     4

>> v=[3 15 -10];
>> diag(v)

ans =

     3     0     0
     0    15     0
     0     0   -10

>> diag(A)

ans =

     0
    -2
     2

>> sum(A)
```

```
ans =

     5    -1     7

>> sum(A,2)

ans =

     3
     3
     5
```

1.6 Πολυώνυμα

Τα πολυώνυμα αναπαρίστανται ως διανύσματα (πίνακες-γραμμή), που περιέχουν τους συντελεστές κατά φθίνουσα διάταξη. Οι μηδενικοί όροι πρέπει να λαμβάνονται υπόψη, θέτοντας 0 στις αντίστοιχες θέσεις. Για παράδειγμα, το πολυώνυμο

$$p(x) = 2x^3 + 5x - 6$$

ορίζεται ως

```
p=[2 0 5 -6]
```

Οι πράξεις των πολυωνύμων υλοποιούνται ως εξής:

Πράξη	Λειτουργία
+, -	Άθροισμα και διαφορά, υποστηρίζεται από τις αντίστοιχες πράξεις μεταξύ διανυσμάτων. Όταν τα πολυώνυμα δεν είναι του ίδιου βαθμού, θα πρέπει το πολυώνυμο του μικρότερου βαθμού να συμπληρώνεται στην αρχή με τον κατάλληλο αριθμό μηδενικών.
`conv(p,q)`	Πολλαπλασιασμός πολυωνύμων.
`[q,r]=deconv(a,b)`	Διαίρεση πολυωνύμων, όπου `q` είναι το πηλίκο και `r` το υπόλοιπο της διαίρεσης.

Οι βασικές λειτουργίες των πολυωνύμων επιτελούνται με τις ακόλουθες συναρτήσεις:

Συνάρτηση	Λειτουργία
`polyval(p,x)`	Υπολογισμός τιμής $p(x)$ του πολυωνύμου.
`r=roots(p)`	Εύρεση ριζών, όπου r είναι ένας πίνακας-στήλη, το οποίο και επιστρέφει τις ρίζες του πολυωνύμου $p(x)$.
`pp=poly(r)`	Κατασκευή πολυωνύμου με δοθείσες ρίζες, όπου r είναι ένας πίνακας-στήλη που περιέχει τις ρίζες του ζητούμενου πολυωνύμου $pp(x)$.
`h=polyder(p)`	Παράγωγος του πολυωνύμου $p(x)$.
`h=polyint(p)`	Ολοκλήρωμα του πολυωνύμου $p(x)$.

Παράδειγμα 1.6.1. *Δίνονται τα πολυώνυμα*

$$p(x) = x^3 + 5x - 6 \quad \text{,και} \quad q(x) = x^2 - x + 3$$

Να υπολογιστούν: $p(1),\ p(5),\ q(10),\ p(x)+q(x),\ p(x)\cdot q(x),\ \dfrac{p(x)}{q(x)},\ p'(x),\ p'(2),\ \int q(x)dx$

```
>> p=[2 0 5 -6];
>> q=[0 2 -1 3];
>> polyval(p,1)                                              (p(1))

ans =

     1

>> polyval(p,5)                                              (p(5))

ans =

   269

>> polyval(q,10)                                             (q(10))

ans =

   193

>> p+q                                   (p(x)+q(x) = 2x^3+2x^2+4x-3)

ans =

     2     2     4    -3

>> conv(p,q)          (p(x)·q(x) = 4x^5-2x^4+16x^3-17x^2+21x-18)

ans =
```

```
     0     4    -2    16   -17    21   -18

>> deconv(p,q)
??? Error using ==> deconv at 21
First coefficient of A must be non-zero.


>> q=[2 -1 3]

q =

     2    -1     3

>> [d,r]=deconv(p,q)                    (p(x)/q(x) = x + 1/2 + (2.5x-7.5)/q(x))

d =

    1.0000    0.5000


r =

         0         0    2.5000   -7.5000

>> polyder(p)                                   (p'(x) = 6x^2 + 5)

ans =

     6     0     5

>> polyval(polyder(p),2)                        (p'(2))

ans =

    29

>> polyint(q)                          (∫ q(x)dx = (2/3)x^3 - (1/2)x^2 + 3x)

ans =

    0.6667   -0.5000    3.0000         0
```

♦

Κεφάλαιο 2

Προγραμματισμός σε Matlab

2.1 Αρχεία Matlab, Script - Function

Έχουμε δυο τρόπους να υλοποιήσουμε προγράμματα σε MATLAB:

- **script**
  Γράφουμε τις εντολές του MATLAB σε ένα m-file το αποθηκεύουμε[1] σε αρχείο με προέκταση .m και *εκτελούμε* αυτό το m-file.

- **function**
  Γράφουμε τις εντολές του MATLAB σε ένα m-file (function) το αποθηκεύουμε[2] σε αρχείο με προέκταση .m και *καλούμε* αυτό το m-file (function) με τα κατάλληλα ορίσματα.

Στα m-file του MATLAB μπορούμε να χρησιμοποιήσουμε τις παρακάτω εντολές:

Εντολή	Λειτουργία
`clear`	καθαρίζει τις μεταβλητές
`clc`	καθαρίζει την οθόνη
`close all`	κλείνει όλα τα γραφικά παράθυρα
`disp(a)`	εμφανίζει το περιεχόμενο της μεταβλητής `a` (Εντολή εξόδου)
`disp('text')`	εμφανίζει το κείμενο `text` (Εντολή εξόδου)
`a=input()`	εισαγωγή τιμών από το πληκτρολόγιο στην μεταβλητή `a` (Εντολή εισόδου)
`fprirnt`	Εντολή εξόδου που λειτουργεί όπως στην C
`% comments`	Σχόλια

2.1.1 Προγραμματισμός με Script

Για τη δημιουργία και εκτέλεση ενός script ακολουθούμε τα παρακάτω βήματα:

[1] Προσοχή στην ονομασία του m-file, ότι ισχύει για την ονοματολογία των μεταβλητών
[2] Προσοχή στην ονομασία του m-file, ότι ισχύει για την ονοματολογία των μεταβλητών

- Ανοίγουμε ένα m-file είτε από το μενού (αναλόγως την έκδοση, είτε m-file είτε script)

  File » New » M-file ή File » New » script

  είτε από το εικονίδιο (New).
  Το νέο παράθυρο του MATLAB που εμφανίζεται είναι ο Editor του MATLAB

- Γράφουμε στον Editor τις εντολές που θα εκτελούσαμε στην γραμμή των εντολών του MATLAB.

- Αποθηκεύουμε το m-file με ένα όνομα (ισχύουν τα ίδια με τις μεταβλητές)

- Εκτελούμε το m-file είτε από το εικονίδιο (Run), είτε από το μενού Debug » Run (F5), είτε στο Command Window γράφοντας το όνομα του αρχείου (χωρίς την προέκταση .m) και ↵ (Enter).

Με τη χρήση των script σε MATLAB, έχουμε καλύτερη διαχείριση και εποπτεία των εντολών, αλλά δεν προσφέρονται για ανάπτυξη μεγάλων προγραμμάτων.

Για παράδειγμα, να γίνει script το οποίο να υπολογίζει την υποτείνουσα ορθογωνίου τριγώνου δοθέντων των δύο κάθετων πλευρών $a = 3$, $b = 4$.
Στον Editor γράφουμε

```
1 a=3
2 b=4
3 y=sqrt(a^2+b^2)
4 disp('Ypoteinousa:')
5 disp(y)
```

Εκτελούμε και στο Command Window έχουμε

```
a =

     3

b =

     4

y =

     5
```

```
Ypoteinousa:
     5
```

Στο προηγούμενο παράδειγμα το MATLAB μας επιστρέφει τις τιμές των εκχωρήσεων. Μπορούμε να μην τις εμφανίσουμε με τη χρήση του συμβόλου (;).
Επίσης, στην αρχή κάθε script πρέπει να ενσωματώνουμε τις εντολές που καθαρίζουν το workspace του MATLAB, την οθόνη και να κλείνουν τα παράθυρα γραφικών (figures) αν υπάρχουν.
Στον Editor γράφουμε

```
clc
clear
a=3;
b=4;
y=sqrt(a^2+b^2);
disp('Ypoteinousa:')
disp(y)
```

Εκτελούμε και στο Command Window έχουμε

```
Ypoteinousa:
     5
```

Στα προηγούμενα παραδείγματα χρησιμοποιούμε την εντολή `disp` δυο φορές. Μια για να εμφανίσουμε κείμενο και μια για να εμφανίσουμε την τιμή της μεταβλητής. Μπορούμε, εναλλακτικά, να χρησιμοποιήσουμε την εντολή `fprirnt`.
Στον Editor γράφουμε

```
clc
clear
a=3;
b=4;
y=sqrt(a^2+b^2);
fprintf('Ypoteinousa: %d\n',y)
```

Εκτελούμε και στο Command Window έχουμε

```
Ypoteinousa: 5
```

Παράδειγμα 2.1.1. *Να γίνει m-file το οποίο να υπολογίζει την υποτείνουσα ορθογωνίου τριγώνου δοθέντων των δύο κάθετων πλευρών.*

Στον Editor γράφουμε

```
clear;
clc;
a=input('dose pleyra')
```

```
4 b=input('dose pleyra')
5 y=sqrt(a^2+b^2)
6 disp('Ypoteinousa:')
7 disp(y)
```

♦

2.1.2 Προγραμματισμός με Function

Για τη δημιουργία και εκτέλεση μιας συνάρτησης (function) ακολουθούμε τα παρακάτω βήματα:

- Ανοίγουμε ένα m-file είτε από το μενού (αναλόγως την έκδοση, είτε m-file είτε function)

  File » New » M-file ή File » New » function

  είτε από το εικονίδιο (New).
  Το νέο παράθυρο του MATLAB που εμφανίζεται είναι ο Editor του MATLAB

- Γράφουμε στον Editor τις εντολές που θα εκτελεί μια συνάρτηση για να μας επιστρέψει το αποτέλεσμα.

- Αποθηκεύουμε το m-file με το όνομα που μας προτρέπει το MATLAB (ισχύουν τα ίδια με τις μεταβλητές)

- Καλούμε την συνάρτηση στο Command Window γράφοντας το όνομα του αρχείου (χωρίς την προέκταση .m) βάζοντας σε παρενθέσεις τα κατάλληλα ορίσματα και ⏎ (Enter).

Η συνάρτηση είναι σε ξεχωριστό αρχείο και μπορεί να κληθεί είτε στο Command Window, είτε από οπουδήποτε άλλο m-file.

Για να ξεχωρίζει το MATLAB οτι το m-file είναι συνάρτηση θα πρέπει να υπάρχει η δήλωση της συνάρτησης, δηλαδή,

- στην πρώτη γραμμή του m-file δηλώνουμε τις εξόδους, το όνομα και τις εισόδους με την δήλωση `function`

- `function [y1,y2,..]=name(x1,x2,..)`
  όπου

  – `[y1,y2,..]` Μεταβλητές εξόδου

  – `name` Όνομα συνάρτησης

  - `(x1,x2,..)` Μεταβλητές εισόδου

- στις γραμμές μετά την δήλωση πρέπει να βάλουμε σχόλια τα οποία αποτελούν κείμενο βοήθειας της συνάρτησης.
-
```
function [y1,y2,..]=name(x1,x2,..)
% help comments
% help comments
command
.
.
.
command % comments
```

Κατά τη συγγραφή μιας συνάρτησης θα πρέπει να προσέξουμε τα εξής:

- Το όνομα με το οποίο καλείται η συνάρτηση είναι το όνομα του m-file και όχι το όνομα της συνάρτησης.
- Το κείμενο βοήθειας είναι απαραίτητο στην συνάρτηση για να μας υποδείξει τα ορίσματα εισόδου και τις μεταβλητές εξόδου καθώς και την λειτουργία της συνάρτησης.

Μια συνάρτηση την καλούμε με το όνομα του m-file και μέσα σε παρενθέσεις βάζουμε τα ορίσματα που θέλουμε.
`name(x1,x2,..)`

Για παράδειγμα, να γίνει συνάρτηση που να δέχεται τις κάθετες πλευρές ενός ορθογωνίου τριγώνου και να επιστρέφει την υποτείνουσα.
Στον Editor γράφουμε

```
function y=ypot(a,b)
%ypologismos ypoteinousas
%y      ypoteinousa
%a, b   kathetes pleyres
y=sqrt(a^2+b^2);
%
```

Στο Command Window καλούμε την συνάρτηση με ορίσματα $a = 3$, $b = 4$ και έχουμε

```
>> ypot(3,4)

ans =

     5
```

ενώ με ορίσματα $a = 12$, $b = 5$ έχουμε

```
>> ypot(12,5)

ans =

    13
```

Παράδειγμα 2.1.2. *Να γίνει συνάρτηση που να δέχεται έναν πίνακα (μονοδιάστατο) και να επιστρέφει το μέσο όρο των στοιχείων του πίνακα.*

Στον Editor γράφουμε

```
1 function y=mop(a)
2 %Calculation of the mean of a vector
3 % a   a vector
4 % y   the mean
5 y=sum(a)/length(a);
```

Στο Command Window καλούμε την συνάρτηση και έχουμε

```
>>a=[2 3 4 5];
>>mop(a)

ans=

     3.5
```

♦

Παράδειγμα 2.1.3. *Να γίνει συνάρτηση που να δέχεται έναν πίνακα (μονοδιάστατο) και να επιστρέφει το μέγιστο και το ελάχιστο των στοιχείων του πίνακα.*

Στον Editor γράφουμε

```
1 function [y,z]=pin(a)
2 %Calculation of the max and min of a vector
3 % a   a vector
4 % y   the max element
5 % z   the min element
6 y=max(a);
7 z=min(a);
```

Στο Command Window καλούμε την συνάρτηση και έχουμε

```
>>a=[3 5 7 9];
>>[y1,y2]=pin(a)
```

```
y1=

       9

y2=

       3
```

♦

2.2 Βασικές δομές σε Matlab

Το MATLAB παρέχει δυνατότητες προγραμματισμού, με τη βοήθεια μίας γλώσσας που έχει πολλές ομοιότητες με τη γλώσσα C.

2.2.1 Δομές Επιλογής

Εντολή `if`

Η εντολή `if` έχει την ακόλουθη σύνταξη:

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

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

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

Οι εντολές μεταξύ `else` και `end` εκτελούνται όταν η <έκφραση> είναι ψευδής.
Η <έκφραση> μπορεί να είναι μια λογική μεταβλητή ή μια συνθήκη (παράσταση που μας επιστρέφει λογικό αποτέλεσμα).

Για την κατασκευή σύνθετων δομών `if` χρησιμοποιείται η εντολή `elseif`.

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

Παράδειγμα 2.2.1. *Δίνεται ο παρακάτω τύπος*

$$a_n = \begin{cases} n-1 & n<0 \\ n^2-1 & 0 \le n \le 10 \\ \frac{n}{3}-2 & n>10 \end{cases}$$

Να γραφεί πρόγραμμα το οποίο να υπολογίζει τις τιμές του a_n.

Στον Editor γράφουμε

```
1  clc
2  clear
3  n=input('Give n: ');
4  if n<0
5      a=n-1;
6  elseif n<=10
7      a=n^2-1;
8  else
9      a=n/3-2;
10 end
11 disp(a)
```

♦

2.2.2 Δομές Επανάληψης

Εντολή `for`

Η εντολή `for` έχει την ακόλουθη σύνταξη:

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

όπου το `<πεδίο τιμών>` είναι ένα διάνυσμα (μονοδιάστατος πίνακας), δηλαδή, στην πιο απλή μορφή το διάνυσμα μπορεί να οριστεί με τον παρακάτω τρόπο

- `x=a:s:b`
- `a` αρχική τιμή, `s` βήμα, `b` τελική τιμή.
- Όταν το βήμα (`s`) παραλείπεται το MATLAB το θεωρεί ίσο με τη μονάδα.

Παράδειγμα 2.2.2. *Να βρεθούν οι 10 πρώτες τιμές της ακολουθίας (αναδρομικός τύπος) που δίνεται από τον τύπο*

$$a_n = 3 \cdot a_{n-1} + 8$$

με αρχική τιμή $a_1 = 1$.

Στον Editor γράφουμε

```
clear
clc
a(1)=1;
for i=2:10
    a(i)=3*a(i-1)+8;
end
disp(a')
```

Εκτελούμε και στο Command Window έχουμε

```
           1
          11
          41
         131
         401
        1211
        3641
       10931
       32801
       98411
```

♦

Παράδειγμα 2.2.3. *Δίνεται ο παρακάτω τύπος*

$$y(n) = n^2 + 3 \cdot n - 1$$

Να βρεθούν οι τιμές των $y(1)$, $y(5)$, $y(10)$, $y(100)$ *και* $y(-10)$.

Στον Editor γράφουμε

```
clc
clear
for n=[1,5,10,100,-10]
    y=n^2+3*(n-1)
end
```

Εκτελούμε και στο Command Window έχουμε

```
y =

     1

y =

    37

y =

   127

y =

       10297

y =

    67
```

♦

Στο παράδειγμα 2.2.2 χρησιμοποιούμε την απλή δεικτοδότηση `i=2:10` κατά την οποία η μεταβλητή της επανάληψης (`i`) παίρνει όλες τις τιμές από 2 έως 10 με βήμα 1.
Αντιθέτως, στο παράδειγμα 2.2.3 χρησιμοποιούμε την δεικτοδότηση `n=[1,5,10,100,-10]` κατά την οποία η μεταβλητή της επανάληψης (`n`) παίρνει τις τιμές του πίνακα.

Εντολή `while`

Η εντολή `while` έχει την ακόλουθη σύνταξη:

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

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

Παράδειγμα 2.2.4. *Να βρεθούν οι τιμές της ακολουθίας (αναδρομικός τύπος)* a_n *για τις οποίες να ισχύει η σχέση*

$$|a_n| < 10000$$

Οι τιμές της ακολουθίας a_n *δίνονται από τον τύπο*

$$a_n = 3 \cdot a_{n-1} + 8$$

με αρχική τιμή $a_1 = 1$.

Στον Editor γράφουμε

```
clear
clc
a(1)=1;
i=1;
while abs(a(i))<10000
    i=i+1;
    a(i)=3*a(i-1)+8;
end
k=1:(length(a)-1);
out=[k', a(k)']
```

Εκτελούμε και στο Command Window έχουμε

```
out =

           1           1
           2          11
           3          41
           4         131
           5         401
           6        1211
           7        3641
```

Ένα ισοδύναμο πρόγραμμα με την χρήση διακόπτη (`done`) είναι το παρακάτω

```
clear
clc
a(1)=1;
i=2;
done=0;
while done==0
```

```
    a(i)=3*a(i-1)+8;
    if abs(a(i))<10000
        i=i+1;
    else
        done=1;
    end
end
k=1:(length(a)-1);
out=[k', a(k)']
```

♦

Παράδειγμα 2.2.5. *Να βρεθούν οι τιμές της ακολουθίας (αναδρομικός τύπος)* a_n *για τις οποίες να ισχύει η σχέση*

$$|a_n - a_{n-1}| < 10^6$$

Οι τιμές της ακολουθίας a_n *δίνονται από τον τύπο*

$$a_n = 3 \cdot a_{n-1} + 8$$

με αρχική τιμή $a_1 = 1$.

Στον Editor γράφουμε

```
clear
clc
a(1)=1;
i=2;
done=0;
while done==0
    a(i)=3*a(i-1)+8;
    if abs(a(i)-a(i-1))<10^6
        i=i+1;
    else
        done=1;
    end
end
k=1:(length(a)-1);
out=[k', a(k)']
```

Εκτελούμε και στο Command Window έχουμε

```
out =

           1           1
           2          11
           3          41
```

```
     4         131
     5         401
     6        1211
     7        3641
     8       10931
     9       32801
    10       98411
    11      295241
    12      885731
```

♦

Μία αποδοτική προγραμματιστική τακτική στο MATLAB είναι η χρήση εντολών που χρησιμοποιούν διανύσματα αντί των συμβατικών ανακυκλώσεων `while` και `for`. Η προσέγγιση αυτή είναι ταχύτερη, αν και απαιτεί μεγαλύτερη μνήμη.

Εντολή `break`

Η εντολή `break` τερματίζει την εκτέλεση ενός βρόχου. Στην περίπτωση που ο βρόχος που περικλείει την εντολή break βρίσκεται στο εσωτερικό ενός ή περισσότερων βρόχων, τότε τερματίζεται η εκτέλεση του πλέον εσωτερικού.
Το παράδειγμα 2.2.5 με τη χρήση της εντολής `break` γράφεται

```
1  clear
2  clc
3  a(1)=1;
4  i=2;
5  while (1)
6      a(i)=3*a(i-1)+8;
7      if abs(a(i)-a(i-1))<10^6
8          i=i+1;
9      else
10         break;
11     end
12 end
13 k=1:(length(a)-1);
14 out=[k', a(k)']
```

Άλλες εντολές τερματισμού

Η εντολή `error` τερματίζει την εκτέλεση ενός προγράμματος και εμφανίζει ένα μήνυμα λάθους ενώ η εντολή `return` τερματίζει την εκτέλεση μίας συνάρτησης και επιστρέφει την τιμή αυτής.

Κεφάλαιο 3

Γραφικά στο Matlab

Υποστηρίζεται ένα πλούσιο σύνολο συναρτήσεων για τη σχεδίαση δισδιάστατων και τρισδιάστατων γραφικών παραστάσεων. Στη συνέχεια περιγράφονται οι αντίστοιχες εντολές.

3.1 Εντολή `plot`

3.1.1 Γραφική Παράσταση Συνάρτησης

Για τη δημιουργία γραφικής παράστασης μιας συνάρτησης χρησιμοποιούμε την εντολή `plot(x,y,'parameters')`. Πρώτα ορίζουμε τα διανύσματα `x` και `y`.

- `x` είναι το διάνυσμα των τετμημένων των σημείων της συνάρτησης.
- `y` είναι το διάνυσμα των τεταγμένων των σημείων της συνάρτησης.

Το όρισμα `'parameters'` είναι προαιρετικό με το οποίο καθορίζουμε το χρώμα, το σχήμα των σημείων και το είδος της γραμμής[1].

Παράδειγμα 3.1.1. *Να γίνει η γραφική παράσταση της συνάρτησης* $f(x) = e^x + 5x - 13$ *στο διάστημα* $[-5, 5]$.

Σε Matlab θα έχουμε

- ορίζουμε την συνάρτηση $f(x)$
  ```
  f=inline('exp(x)+5*x-13')
  ```
- δημιουργούμε τα διανύσματα `x` και `y`
  ```
  x=-5:0.01:5;
  y=f(x);
  ```
- καλούμε την συνάρτηση `plot` με τα κατάλληλα ορίσματα
  ```
  plot(x,y)
  ```

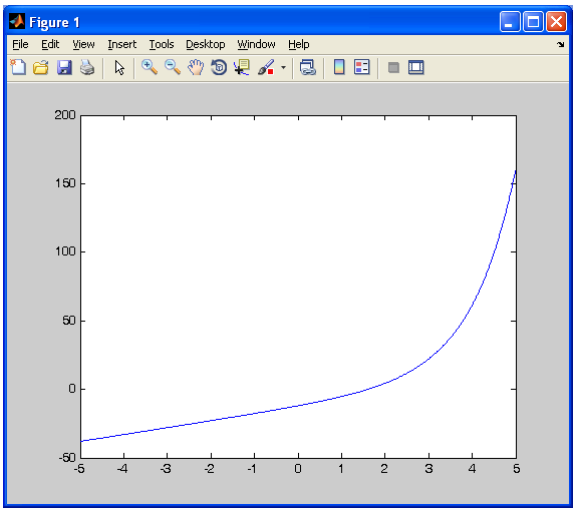
[1] Περισσότερες πληροφορίες στη βοήθεια του Matlab

- εναλλακτικά μπορούμε να κάνουμε και το εξής
  `plot(x,f(x))`

Επομένως, στον Editor γράφουμε

```
1 f=inline('exp(x)+5*x-13');
2 x=-5:0.01:5;
3 plot(x, f(x))
```

το εκτελούμε και μας επιστρέφει το παρακάτω figure



♦

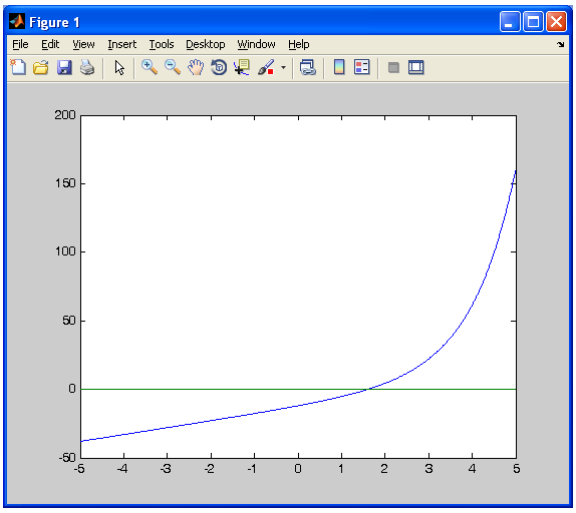
Έπειτα, μπορούμε να προσθέσουμε και κάποιες άλλες εντολές για την καλύτερη παρουσίαση της γραφικής παράστασης.

Εντολή	Λειτουργία
`title('some text')`	Εισαγωγή τίτλου στο figure
`xlabel('some text')`	Ετικέτα στον οριζόντιο άξονα
`ylabel('some text')`	Ετικέτα στον κατακόρυφο άξονα
`text(x,y,'some text')`	Κείμενο μέσα στο figure
`legend('name1','name2')`	εμφανίζουμε ένα υπόμνημα στο γράφημα μας

Στις γραφικές παραστάσεις των συναρτήσεων χρειάζεται να αποτυπωθεί και ο άξονας $x'x$, επομένως, στην συνάρτηση `plot(x,f(x))` προσθέτουμε το ζεύγος διανυσμάτων που αναπαριστά τον $x'x$ (`x, zeros(1, length(x))`).
Στον Editor γράφουμε

```
1 f=inline('exp(x)+5*x-13');
2 x=-5:0.01:5;
3 plot(x, f(x), x, zeros(1,length(x)));
```

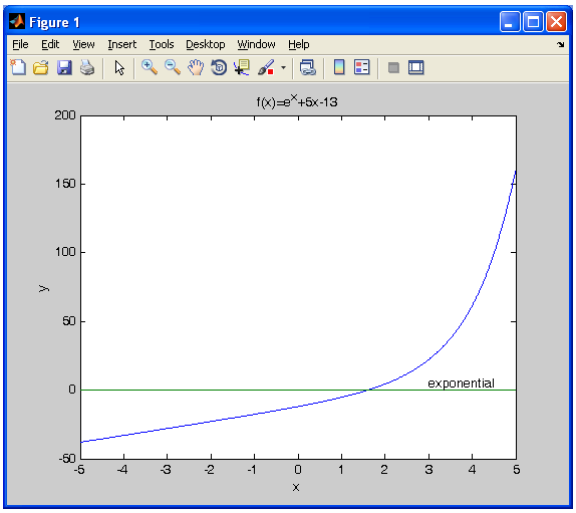
το εκτελούμε και μας επιστρέφει το παρακάτω figure



Επιπλέον, στο figure μπορούμε να προσθέσουμε πληροφορίες όπως φαίνεται παρακάτω
Στον Editor γράφουμε

```
1 f=inline('exp(x)+5*x-13');
2 x=-5:0.01:5;
3 plot(x, f(x), x, zeros(1,length(x)));
4 title('f(x)=e^x+5x-13');
5 xlabel('x');
6 ylabel('y');
7 text(3,5,'exponential');
```

το εκτελούμε και μας επιστρέφει το παρακάτω figure



3.1.2 Γραφική Παράσταση Σημείων

Για τη δημιουργία γραφικής παράστασης ενός συνόλου διακριτών σημείων χρησιμοποιούμε την εντολή `plot(x,y,'parameters')`.
Πρώτα ορίζουμε τα διανύσματα `x` και `y`.

- `x` είναι το διάνυσμα των τετμημένων των σημείων.
- `y` είναι το διάνυσμα των τεταγμένων των σημείων.

Στο όρισμα `'parameters'` στο οποίο ορίζουμε το χρώμα, το σχήμα των σημείων και το είδος της γραμμής, πρέπει να ορίσουμε υποχρεωτικά το σχήμα των σημείων και να μην ορίσουμε το είδος της γραμμής.

Παράδειγμα 3.1.2. *Δίνεται ο παρακάτω πίνακας τιμών*

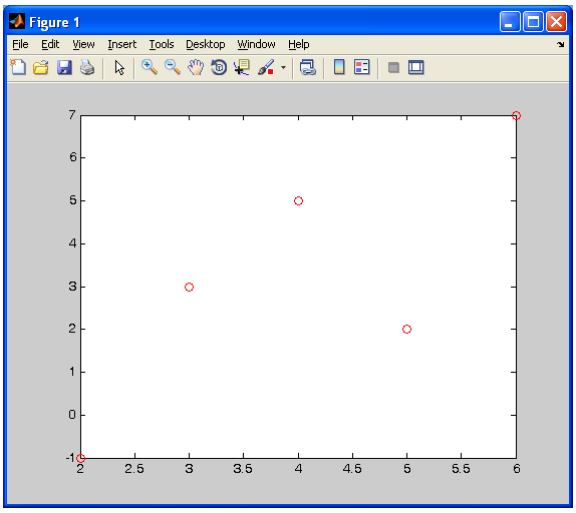
X	2	3	4	5	6
Y	-1	3	5	2	7

Να γίνει η γραφική παράσταση των παραπάνω σημείων.

Στον Editor γράφουμε

```
1 x=[2 3 4 5 6];
2 y=[-1 3 5 2 7];
3 plot(x,y,'ro');
```

το εκτελούμε και μας επιστρέφει το παρακάτω figure



3.2 Εντολή `subplot`

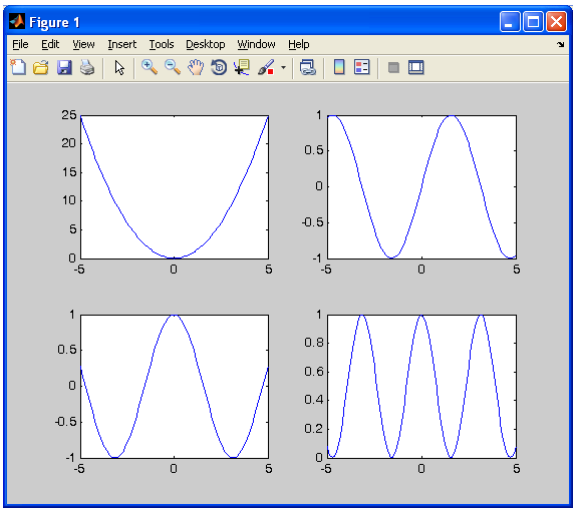
Με την εντολή `subplot` μπορούμε μια εικόνα να την χωρίσουμε σε μικρότερες εικόνες έτσι ώστε σε μια εικόνα να καταχωρήσουμε περισσότερα από ένα γραφήματα. Η σύνταξη της είναι η εξής `subplot(n,m,a)`

- `n` και `m` ορίζουν τη διάταξη των υπο-γραφημάτων ως ένα δισδιάστατο πίνακα (n γραμμές και m στήλες)
- `a` ο αριθμός του γραφήματος που είναι ενεργό

δηλαδή, γράφοντας `subplot(2,2,1)` θα δημιουργηθούν 4 γραφήματα με διάταξη 2×2 και είναι ενεργό το πρώτο υπο-γράφημα.
Για παράδειγμα, οι εντολές

```
x=-5:0.1:5;
y1=x.^2;
y2=sin(x);
y3=cos(x);
y4=cos(x).^2;
subplot(2,2,1)
plot(x,y1)
subplot(2,2,2)
plot(x,y2)
subplot(2,2,3)
plot(x,y3)
subplot(2,2,4)
plot(x,y4)
```

μας επιστρέφουν το παρακάτω figure



Μέρος ΙΙ

Αριθμητικές Μέθοδοι (Εργαστηριακές Σημειώσεις)

Κεφάλαιο 1

Επίλυση μη γραμμικών εξισώσεων

1.1 Επαναληπτική μέθοδος - Εργαστήριο 4

1.1.1 Επαναληπτική Μέθοδος

Υπολογισμός μιας ρίζας της εξίσωσης $f(x) = 0$ με χρήση ενός αναδρομικού τύπου της μορφής

$$x_n = g(x_{n-1})$$

Για να δημιουργήσουμε τον αναδρομικό τύπο πρέπει η εξίσωση $f(x) = 0$ να γραφεί με τη μορφή $x = g(x)$.
Για παράδειγμα, θέλουμε να βρούμε την ρίζα της εξίσωσης

$$x^2 - 2 = 0$$

Δημιουργούμε τον αναδρομικό τύπο

$$x^2 - 2 = 0 \Rightarrow x^2 + 2x = 2 + 2x \Rightarrow x(x + 2) = 2 + 2x \Rightarrow x = \frac{2 + 2x}{x + 2}$$

Επομένως, ο αναδρομικός τύπος είναι

$$x_n = \frac{2 + 2x_{n-1}}{2 + x_{n-1}}$$

Παράδειγμα 1.1.1 (Επαναληπτική Μέθοδος). *Να βρεθούν οι 10 πρώτες τιμές του x που δίνονται από τον τύπο*

$$x_n = \frac{2 + 2x_{n-1}}{2 + x_{n-1}}$$

με αρχική τιμή $x_1 = 1$.

- Στον Editor γράφουμε

```
clear
clc
x(1)=1;
for i=2:10
    x(i)=(2+2*x(i-1))/(2+x(i-1));
end
k=1:length(x);
out=[k', x(k)']
```

- Εκτελούμε και στο Command Window έχουμε

```
out =

                    1                    1
                    2    1.33333333333333
                    3                  1.4
                    4    1.41176470588235
                    5    1.41379310344828
                    6    1.41414141414141
                    7    1.41420118343195
                    8    1.41421143847487
                    9    1.41421319796954
                   10    1.41421349985132
```

♦

1.1.2 Ακρίβεια Δεκαδικών ψηφίων

- Μια αριθμητική μέθοδος μας επιστρέφει σε κάθε βήμα μια προσέγγιση της λύσης.

- Αν η αριθμητική μέθοδος συγκλίνει, τότε η μέθοδος προσεγγίζει την λύση.

- Η ακρίβεια δεκαδικών ψηφίων μας δείχνει πόσο καλά προσεγγίζει η μέθοδος την λύση σε σχέση με το πλήθος των δεκαδικών ψηφίων.

- Η ακρίβεια δεκαδικών ψηφίων δίνεται από τον τύπο

$$|x_n - x_{n-1}| < \frac{1}{2}10^{-k}$$

όπου k το πλήθος των δεκαδικών ψηφίων.

Για να υπολογίσουμε την αριθμητική ακρίβεια σε δεκαδικά ψηφία, λύνουμε ως προς k και έχουμε

$$|x_n - x_{n-1}| < \frac{1}{2}10^{-k} \Rightarrow 2 \cdot |x_n - x_{n-1}| < 10^{-k} \Rightarrow$$

$$\log\left(2 \cdot |x_n - x_{n-1}|\right) < \log\left(10^{-k}\right) \Rightarrow \log\left(2 \cdot |x_n - x_{n-1}|\right) < -k \Rightarrow$$

$$k < -\log\left(2 \cdot |x_n - x_{n-1}|\right)$$

Παράδειγμα 1.1.2 (Ακρίβεια Δεκαδικών ψηφίων). *Να βρεθούν οι 10 πρώτες τιμές του x που δίνονται από τον τύπο*

$$x_n = \frac{2 + 2x_{n-1}}{2 + x_{n-1}}$$

με αρχική τιμή $x_1 = 1$. Να βρεθεί η ακρίβεια σε δεκαδικά ψηφία που έχουν οι τιμές x_{10} και x_8.

- Στον Editor γράφουμε

```
clear
clc
x(1)=1;
for i=2:10
    x(i)=(2+2*x(i-1))/(2+x(i-1));
end
k=1:length(x);
out=[k', x(k)']
```

- Εκτελούμε και στο Command Window έχουμε

```
out =

                    1                    1
                    2    1.33333333333333
                    3                  1.4
                    4    1.41176470588235
                    5    1.41379310344828
                    6    1.41414141414141
                    7    1.41420118343195
                    8    1.41421143847487
                    9    1.41421319796954
                   10    1.41421349985132
```

- Για την ακρίβεια του x_{10}, στο Command Window γράφουμε

```
>> -log10(2*abs(out(10,2)-out(9,2)))

ans =

          6.21913310223154
```

- επειδή
$$k < -\log\left(2 \cdot |x_{10} - x_9|\right) = 6.21913310223154$$
θα έχουμε
$$k = 6$$
δηλαδή, η τιμή x_{10} έχει ακρίβεια 6 δεκαδικών ψηφίων.

- Για την ακρίβεια του x_8, στο Command Window γράφουμε

```
>> -log10(2*abs(out(8,2)-out(7,2)))

ans =

          4.68803252210793
```

- επειδή
$$k < -\log\left(2 \cdot |x_8 - x_7|\right) = 4.68803252210793$$
θα έχουμε
$$k = 4$$
δηλαδή, η τιμή x_8 έχει ακρίβεια 4 δεκαδικών ψηφίων.

♦

1.1.3 Ακρίβεια Σημαντικών ψηφίων

- Μια αριθμητική μέθοδος μας επιστρέφει σε κάθε βήμα μια προσέγγιση της λύσης.
- Αν η αριθμητική μέθοδος συγκλίνει, τότε η μέθοδος προσεγγίζει την λύση.
- Η ακρίβεια σημαντικών ψηφίων μας δείχνει πόσο καλά προσεγγίζει η μέθοδος την λύση σε σχέση με το πλήθος των σημαντικών ψηφίων.
- Η ακρίβεια σημαντικών ψηφίων δίνεται από τον τύπο
$$\left|\frac{x_n - x_{n-1}}{x_{n-1}}\right| < \frac{1}{2}10^{-(k-1)}$$
όπου k το πλήθος των σημαντικών ψηφίων.

Για να υπολογίσουμε την αριθμητική ακρίβεια σε σημαντικά ψηφία, λύνουμε ως προς k και έχουμε

$$\left|\frac{x_n - x_{n-1}}{x_{n-1}}\right| < \frac{1}{2}10^{-(k-1)} \Rightarrow 2 \cdot \left|\frac{x_n - x_{n-1}}{x_{n-1}}\right| < 10^{-(k-1)} \Rightarrow$$
$$\log\left(2 \cdot \left|\frac{x_n - x_{n-1}}{x_{n-1}}\right|\right) < \log\left(10^{-(k-1)}\right) \Rightarrow$$
$$\log\left(2 \cdot \left|\frac{x_n - x_{n-1}}{x_{n-1}}\right|\right) < -(k-1) \Rightarrow$$
$$k - 1 < -\log\left(2 \cdot \left|\frac{x_n - x_{n-1}}{x_{n-1}}\right|\right) \Rightarrow$$
$$k < -\log\left(2 \cdot \left|\frac{x_n - x_{n-1}}{x_{n-1}}\right|\right) + 1$$

Παράδειγμα 1.1.3 (Ακρίβεια Σημαντικών ψηφίων). *Να βρεθούν οι 10 πρώτες τιμές του x που δίνονται από τον τύπο*

$$x_n = \frac{2 + 2x_{n-1}}{2 + x_{n-1}}$$

με αρχική τιμή $x_1 = 1$. Να βρεθεί η ακρίβεια σε σημαντικά ψηφία που έχουν οι τιμές x_{10} και x_8.

- Στον Editor γράφουμε

```
clear
clc
x(1)=1;
for i=2:10
    x(i)=(2+2*x(i-1))/(2+x(i-1));
end
k=1:length(x);
out=[k', x(k)']
```

- Εκτελούμε και στο Command Window έχουμε

```
out =

                    1                    1
                    2    1.33333333333333
                    3                  1.4
                    4    1.41176470588235
                    5    1.41379310344828
                    6    1.41414141414141
                    7    1.41420118343195
                    8    1.41421143847487
                    9    1.41421319796954
                   10    1.41421349985132
```

- Για την ακρίβεια του x_{10}, στο Command Window γράφουμε

```
>> -log10(2*abs((out(10,2)-out(9,2))/out(9,2)))+1

ans =

          7.36964798815789
```

- επειδή

$$k < -\log\left(2 \cdot \left|\frac{x_{10} - x_9}{x_9}\right|\right) + 1 = 7.36964798815789$$

θα έχουμε

$$k = 7$$

δηλαδή, η τιμή x_{10} έχει ακρίβεια 7 σημαντικών ψηφίων.

- Για την ακρίβεια του x_8, στο Command Window γράφουμε

```
>> -log10(2*abs((out(8,2)-out(7,2))/out(7,2)))+1

ans =

           5.8385437184424
```

- επειδή

$$k < -\log\left(2 \cdot \left|\frac{x_8 - x_7}{x_7}\right|\right) + 1 = 5.8385437184424$$

θα έχουμε

$$k = 5$$

δηλαδή, η τιμή x_8 έχει ακρίβεια 5 σημαντικών ψηφίων.

♦

1.1.4 Επαναληπτική Μέθοδος με κριτήριο τερματισμού

Παράδειγμα 1.1.4 (Επαναληπτική Μέθοδος με κριτήριο τερματισμού). *Να βρεθούν οι τιμές των x_n μέχρι να ισχύει το κριτήριο τερματισμού*

$$|x_n - x_{n-1}| < \frac{1}{2}10^{-4}$$

δηλαδή, η προσεγγιστική λύση να έχει ακρίβεια τεσσάρων δεκαδικών ψηφίων.
Οι τιμές των x_n δίνονται από τον τύπο

$$x_n = \frac{2 + 2x_{n-1}}{2 + x_{n-1}}$$

με αρχική τιμή $x_1 = 1$.

- Στον Editor γράφουμε

```
clear
clc
x(1)=1;
i=2;
done=0;
while done==0
    x(i)=(2+2*x(i-1))/(2+x(i-1));
    if abs(x(i)-x(i-1))<(1/2*10^-4)
        done=1;
    else
        i=i+1;
    end
end
k=1:length(x);
out=[k', x(k)']
```

- Εκτελούμε και στο Command Window έχουμε

```
out =

                    1                           1
                    2           1.33333333333333
                    3                         1.4
                    4           1.41176470588235
                    5           1.41379310344828
                    6           1.41414141414141
                    7           1.41420118343195
                    8           1.41421143847487
```

♦

Άσκηση 1.1.1.
Δίνεται ο τύπος

$$x_n = \frac{3 + x_{n-1}}{1 + x_{n-1}}$$

1. *Να βρεθούν οι 20 πρώτες τιμές του* x *που δίνονται από τον παραπάνω τύπο με αρχική τιμή* $x_1 = 2$.
2. *Να βρεθεί η ακρίβεια σε δεκαδικά ψηφία που έχουν οι τιμές* x_{15} *και* x_{20}.
3. *Να βρεθεί η ακρίβεια σε σημαντικά ψηφία που έχουν οι τιμές* x_{15} *και* x_{20}.

1.2 Μέθοδος Διχοτόμησης - Εργαστήριο 5

- Υπολογισμός μιας ρίζας της εξίσωσης $f(x) = 0$ στο διάστημα $[a, b]$ οπού ισχύει $f(a) \cdot f(b) < 0$.
- Είσοδος
  - Η συνάρτηση f
  - Το διάστημα $[a, b]$
  - Η ακρίβεια σε δεκαδικά ψηφία (tol)
  - Ο μέγιστος αριθμός επαναλήψεων
- Έξοδος
  - Η προσεγγιστική ρίζα
  - Μήνυμα αποτυχίας

1.2.1 Μέθοδος Διχοτόμησης - Αλγόριθμος

ΕΙΣΟΔΟΣ: $f(x)$, a, b, tol, n

ΒΗΜΑ 1 Αν $f(a) \cdot f(b) < 0$ πήγαινε στο βήμα 2
Διαφορετικά
ΕΞΟΔΟΣ: Δεν ισχύει το Θ. Bolzano στο αρχικό διάστημα, τερμάτισε

ΒΗΜΑ 2 Θέσε $i = 1$

ΒΗΜΑ 3 Όταν $i <= n$ εκτέλεσε τα βήματα $3 - 6$

ΒΗΜΑ 4 Θέσε $x = \dfrac{a+b}{2}$

ΒΗΜΑ 5 Αν $f(x) = 0$ ή $\dfrac{b-a}{2} < tol$ τότε
ΕΞΟΔΟΣ: το x είναι η λύση, τερμάτισε

ΒΗΜΑ 6 Αν $f(a) \cdot f(x) > 0$ τότε
Θέσε $a = x$
Διαφορετικά
Θέσε $b = x$

ΒΗΜΑ 7 Θέσε $i = i + 1$

ΒΗΜΑ 8 ΕΞΟΔΟΣ: Η μέθοδος εξάντλησε όλες τις επαναλήψεις, τερμάτισε

- Η μέθοδος διχοτόμησης πρέπει να εφαρμόζεται σε διαστήματα στα οποία υπάρχει ακριβώς μια ρίζα.

- Τον παραπάνω αλγόριθμο αν τον εφαρμόσουμε σε διάστημα με καμία ή 2 ή 4 ρίζες (γενικά άρτιο αριθμό ριζών) θα σταματήσει από το Βήμα 1.

- Τον παραπάνω αλγόριθμο αν τον εφαρμόσουμε σε διάστημα με 3 ή 5 ρίζες (γενικά περιττό αριθμό ριζών) θα βρει μια από όλες τις ρίζες.

1.2.2 Μέθοδος Διχοτόμησης - Υλοποίηση σε Matlab

- Υλοποίηση της μεθόδου Διχοτόμησης σε συνάρτηση MATLAB με `while` και `break`

```
function bisection(f,a,b,tol,n)
if f(a)*f(b)>0.0
   error('function has same sign at end points')
end
i = 1;
while i<=n
    x=(a+b)/2;
    if f(x)==0 || (b-a)/2<tol
        disp('The solution found')
        disp(x)
        break;
    end
    if f(a)*f(x)>0
        a=x;
    else
        b=x;
    end
    i = i + 1;
end
```

- Υλοποίηση της μεθόδου Διχοτόμησης σε συνάρτηση MATLAB με `for` και `break`

```
function bisection(f,a,b,tol,n)
if f(a)*f(b)>0.0
   error('function has same sign at end points')
end
for i=1:n
    x=(a+b)/2;
    if f(x)==0 || (b-a)/2<tol
        disp('The solution found')
        disp(x)
        break;
    end
```

```
    if f(a)*f(x)>0
        a=x;
    else
        b=x;
    end
end
```

- Υλοποίηση της μεθόδου Διχοτόμησης σε συνάρτηση MATLAB με `while` και διακόπτη (`done`)

```
function bisection(f,a,b,tol,n)
if f(a)*f(b)>0.0
   error('function has same sign at end points')
end
i = 1;
done=0;
while i<=n && done==0
    x=(a+b)/2;
    if f(x)==0 || (b-a)/2<tol
        disp('The solution found')
        disp(x)
        done=1;
    else
        i = i + 1;
    end
    if f(a)*f(x)>0
        a=x;
    else
        b=x;
    end
end
```

1.2.3 Μέθοδος Διχοτόμησης - Βελτίωση προγράμματος

- Είσοδος παραμέτρων από τον χρήστη - Δημιουργία συνάρτησης με έξοδο και χρήση μεταβλητών - πινάκων.
  `function out=bisect(f, a, b, tol, n)`
  - Η συνάρτηση $f(x)$
  - Το διάστημα $[a, b]$
  - Η ακρίβεια σε δεκαδικά ψηφία (tol)
  - Ο μέγιστος αριθμός επαναλήψεων (n)
- Περισσότερες πληροφορίες στην έξοδο

- Αριθμό βημάτων
- Νέο διάστημα σε κάθε βήμα
- Τιμή της ρίζας και της συνάρτησης $f(x)$ σε κάθε βήμα

- Υλοποίηση της μεθόδου Διχοτόμησης σε συνάρτηση MATLAB

```
function out=bisect(f, a, b, tol, n)
if f(a)*f(b)>0.0
   error('function has same sign at end points')
end
a(1)=a;
b(1)=b;
i=1;
while i<=n
    x(i)=(a(i)+b(i))/2;
    if f(x(i))==0 || (b(i)-a(i))/2<tol
        break;
    end
    if f(a(i))*f(x(i))>0
        a(i+1)=x(i);
        b(i+1)=b(i);
    else
        a(i+1)=a(i);
        b(i+1)=x(i);
    end
    i=i+1;
end
if i>n
    k=1:n;
else
    k=1:i;
end
out=[k', a(k)', b(k)', x(k)', f(x(k))'];
```

Παράδειγμα 1.2.1. *Να βρεθεί η ρίζα της συνάρτησης*

$$f(x) = x^3 - 2x - 5$$

με τη μέθοδο Διχοτόμησης, στο διάστημα $[1, 3]$ *με ακρίβεια* 4 *δεκαδικών ψηφίων και με μέγιστο αριθμό επαναλήψεων* 50.

Σε MATLAB θα έχουμε

- ορίζουμε την συνάρτηση $f(x)$
  `f=inline('x.^3-2*x-5')`

- καλούμε την συνάρτηση `bisect` με τα κατάλληλα ορίσματα `out=bisect(f, 1, 3, 1/2*10^-4, 50)`

δηλαδή,

```
>> f=inline('x.^3-2*x-5')

f =

     Inline function:
     f(x) = x.^3-2*x-5

>> out=bisect(f, 1, 3, 1/2*10^-4, 50)

out =

      1                   1                   3                   2                        -1
      2                   2                   3                 2.5                     5.625
      3                   2                 2.5                2.25                  1.890625
      4                   2                2.25               2.125                 0.345703125
      5                   2               2.125              2.0625            -0.351318359375
      6              2.0625               2.125             2.09375         -0.008941650390625
      7             2.09375               2.125            2.109375          0.166835784912109
      8             2.09375            2.109375           2.1015625         0.0785622596740723
      9             2.09375           2.1015625          2.09765625         0.0347142815589905
     10             2.09375          2.09765625          2.09570312         0.0128623321652412
     11             2.09375         2.095703125         2.0947265625       0.00195434782654047
     12             2.09375        2.0947265625        2.09423828125      -0.00349514919798821
     13       2.09423828125        2.0947265625       2.094482421875     -0.000770775208366103
     14       2.094482421875       2.0947265625      2.0946044921875      0.000591692672969657
     15       2.094482421875      2.0946044921875     2.09454345703125    -8.95646760454838e-005
     16     2.09454345703125      2.0946044921875     2.09457397460938      0.000251058146290006
```

Από τον πίνακα `out` τον οποίο επιστρέφει η συνάρτηση `bisect` παρατηρούμε τα εξής:

- Για τον υπολογισμό της ρίζας εκτελέστηκαν 16 επαναλήψεις (πρώτη στήλη)
- Το αριστερό άκρο του διαστήματος $a_{16} = 2.09454345703125$ (δεύτερη στήλη)
- Το δεξιό άκρο του διαστήματος $b_{16} = 2.0946044921875$ (τρίτη στήλη)
- Η προσεγγιστική τιμή της ρίζας είναι $x_{16} = 2.09457397460938$ (τέταρτη στήλη)
- Η τιμή της συνάρτησης είναι $f(x_{16}) = 0.000251058146290006$ (πέμπτη στήλη)

- Επομένως, η λύση στο πρόβλημα είναι $x_{16} = 2.09457397460938$

Αν καλέσουμε την `bisect` στο διάστημα $[4, 6]$ θα έχουμε

```
>> f=inline('x.^3-2*x-5')

f =
```

```
     Inline function:
     f(x) = x.^3-2*x-5

>> out=bisect(f, 4, 6, 1/2*10^-4, 50)
??? Error using ==> bisect at 3
function has same sign at end points
```

Δεν μας επιστρέφει αποτέλεσμα διότι δεν ισχύει το Θ. Bolzano

♦

Παράδειγμα 1.2.2. *Να βρεθεί η ρίζα της συνάρτησης*

$$f(x) = x - \ln(|x| + 1) - 2$$

με τη μέθοδο Διχοτόμησης, με ακρίβεια 4 *δεκαδικών ψηφίων και με μέγιστο αριθμό επαναλήψεων* 50*.*

- Για να βρούμε τη ρίζα της συνάρτησης θα πρέπει πρώτα να βρούμε ένα κατάλληλο διάστημα με την βοήθεια της γραφικής παράστασης της συνάρτησης.

- Σχεδιάζουμε την την γραφική παράσταση της συνάρτησης στο διάστημα $[-10, 10]$ και επιλέγουμε το κατάλληλο διάστημα[1].

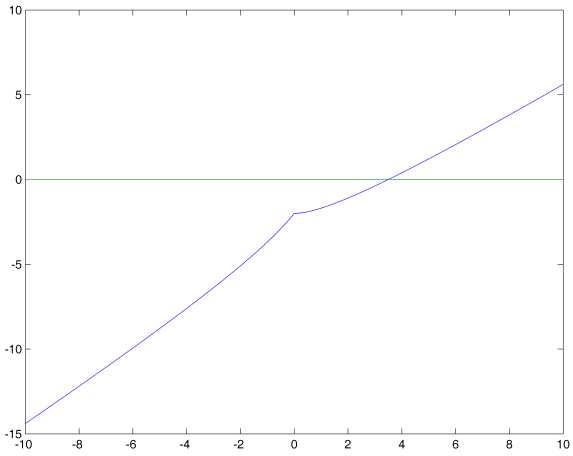
- Από την γραφική παράσταση της συνάρτησης επιλέγουμε το διάστημα $[3, 4]$.

Σε MATLAB θα έχουμε

- ορίζουμε την συνάρτηση $f(x)$
  `f=inline('x-log(abs(x)+1)-2')`

- σχεδιάζουμε την συνάρτηση $f(x)$ και τον άξονα $x'x$
  ```
  x=-10:0.001:10;
  plot(x, f(x), x, zeros(1,length(x)))
  ```

[1] Στα πλαίσια του εργαστηρίου κατάλληλο διάστημα θα θεωρούμε το διάστημα $[a, a+1]$ με $a \in \mathbb{Z}$



- καλούμε την συνάρτηση `bisect` με τα κατάλληλα ορίσματα `out=bisect(f, 3, 4, 1/2*10^-4, 50)`

- η λύση είναι $x_{15} = 3.50521850585938$

♦

1.2.4 Πλήθος Επαναλήψεων - Ακρίβεια

- Το πλήθος των επαναλήψεων (n) της μεθόδου Διχοτόμησης συνδέεται με την ακρίβεια της λύσης σε δεκαδικά ψηφία (k) με τον τύπο

$$\frac{b-a}{2^n} < tol \Longrightarrow \frac{b-a}{2^n} < \frac{1}{2} \cdot 10^{-k}$$

- Επομένως, μπορούμε να βρούμε είτε το n γνωρίζοντας το k, είτε το k γνωρίζοντας το n.

- Για υπολογίσουμε το πλήθος των επαναλήψεων (n) της μεθόδου Διχοτόμησης με ακρίβεια k δεκαδικών ψηφίων στο διάστημα $[a, b]$, λύνουμε ως προς n, επομένως έχουμε

$$\frac{b-a}{2^n} < \frac{1}{2} \cdot 10^{-k} \Longrightarrow \frac{2^n}{b-a} > 2 \cdot 10^{k} \Longrightarrow$$

$$2^n > (b-a) \cdot 2 \cdot 10^{k} \Longrightarrow$$

$$n > \log_2 \left((b-a) \cdot 2 \cdot 10^{k} \right)$$

- Ενώ, για υπολογίσουμε την ακρίβεια των δεκαδικών ψηφίων (k) της μεθόδου Διχοτόμησης αν εκτελεστούν n επαναλήψεις στο διάστημα $[a, b]$, λύνουμε ως προς k, επομένως έχουμε

$$\frac{b-a}{2^n} < \frac{1}{2} \cdot 10^{-k} \Longrightarrow 2 \cdot 10^k < \frac{2^n}{b-a} > \Longrightarrow$$

$$10^k < \frac{2^n}{2 \cdot (b-a)} \Longrightarrow$$

$$k < \log\left(\frac{2^n}{2 \cdot (b-a)}\right)$$

Παράδειγμα 1.2.3 (Πλήθος Επαναλήψεων). *Να βρεθεί το πλήθος των επαναλήψεων που θα εκτελέσει η μέθοδος Διχοτόμησης στο διάστημα* $[1, 3]$ *με ακρίβεια* 4 *δεκαδικών ψηφίων.*

- Σε MATLAB θα έχουμε

```
>> log2((3-1)*2*10^4)
ans =

          15.2877123795494
```

- επειδή $n > \log_2\left((b-a) \cdot 2 \cdot 10^k\right) = \log_2\left((3-1) \cdot 2 \cdot 10^4\right) = 15.2877123795494$
- θα έχουμε $n = 16$.

♦

Παράδειγμα 1.2.4 (Ακρίβεια). *Να βρεθεί η ακρίβεια των δεκαδικών ψηφίων της μεθόδου Διχοτόμησης αν εκτελεστούν* 10 *επαναλήψεις στο διάστημα* $[1, 3]$.

- Σε MATLAB θα έχουμε

```
>> log10(2^10/(2*(3-1)))
ans =

          2.40823996531185
```

- επειδή $k < \log\left(\frac{2^n}{2 \cdot (b-a)}\right) = \log\left(\frac{2^{10}}{2 \cdot (3-1)}\right) = 2.40823996531185$
- θα έχουμε $k = 2$.

♦

Άσκηση 1.2.1. *Δίνεται η συνάρτηση*

$$f(x) = e^x + 5x - 10$$

1. *Να βρεθεί η ρίζα της συνάρτησης με τη μέθοδο Διχοτόμησης, στο διάστημα* $[0, 4]$ *με ακρίβεια* 6 *δεκαδικών ψηφίων και με μέγιστο αριθμό επαναλήψεων* 50*.*

2. *Να βρεθεί το πλήθος των επαναλήψεων που θα εκτελέσει η μέθοδος Διχοτόμησης στο διάστημα* $[0, 4]$ *με ακρίβεια* 10 *δεκαδικών ψηφίων.*

3. *Να βρεθεί η ακρίβεια των δεκαδικών ψηφίων της μεθόδου Διχοτόμησης αν εκτελεστούν* 20 *επαναλήψεις στο διάστημα* $[0, 4]$*.*

4. *Να βρεθεί η ακρίβεια σε δεκαδικά ψηφία που έχουν οι τιμές* x_{10} *και* x_{20}*.*

5. *Να βρεθεί η ακρίβεια σε σημαντικά ψηφία που έχουν οι τιμές* x_{10} *και* x_{20}*.*

1.3 Μέθοδος Newton -Εργαστήριο 6

1.3.1 Μέθοδος Newton

- Υπολογισμός μιας ρίζας της εξίσωσης $f(x) = 0$ με αρχική τιμή x_0.
- Είσοδος
  - Η συνάρτηση f
  - Η παράγωγος συνάρτηση $f'(x)$
  - Η αρχική τιμή x_0
  - Η ακρίβεια σε δεκαδικά ψηφία (tol)
  - Ο μέγιστος αριθμός επαναλήψεων
- Έξοδος
  - Η προσεγγιστική ρίζα
  - Μήνυμα αποτυχίας

1.3.2 Μέθοδος Newton - Αλγόριθμος

ΕΙΣΟΔΟΣ: $f(x)$, $f'(x)$, x_0, tol, n

ΒΗΜΑ 1 Θέσε $i = 2$, $x(1) = x_0$

ΒΗΜΑ 2 Όταν $i <= n$ εκτέλεσε τα βήματα $3 - 5$

ΒΗΜΑ 3 Θέσε $x(i) = x(i-1) - \dfrac{f(x(i-1))}{f'(x(i-1))}$

ΒΗΜΑ 4 Αν $f(x(i)) = 0$ ή $|x(i) - x(i-1)| < tol$ τότε
ΕΞΟΔΟΣ: το $x(i)$ είναι η λύση, τερμάτισε

ΒΗΜΑ 5 Θέσε $i = i + 1$

ΒΗΜΑ 6 ΕΞΟΔΟΣ: Η μέθοδος εξάντλησε όλες τις επαναλήψεις, τερμάτισε

1.3.3 Μέθοδος Newton - Υλοποίηση σε Matlab

- Υλοποίηση της μεθόδου Newton σε συνάρτηση MATLAB

```
function out=newton(f, df, x1, tol, n)
x(1)=x1;
i=2;
while i<=n
    x(i)=x(i-1)-f(x(i-1))/df(x(i-1));
    if f(x(i))==0 || abs(x(i)-x(i-1))<tol
        break;
```

```
    end
    i = i + 1;
end
if i>n
    k=1:n;
else
    k=1:i;
end
out=[k', x(k)', f(x(k))'];
```

Παράδειγμα 1.3.1. *Να βρεθεί η ρίζα της συνάρτησης*

$$f(x) = x^3 - 2x - 5$$

με τη μέθοδο Newton, με αρχική τιμή 1 *με ακρίβεια* 8 *δεκαδικών ψηφίων και με μέγιστο αριθμό επαναλήψεων* 50*.*

Σε MATLAB θα έχουμε

- ορίζουμε την συνάρτηση $f(x)$
  `f=inline('x.^3-2*x-5')`
- ορίζουμε την παράγωγο συνάρτηση $f'(x)$
  `df=inline('3*x.^2-2')` [2]
- καλούμε την συνάρτηση newton με τα κατάλληλα ορίσματα
  `out=newton(f, df, 1, 1/2*10^-8, 50)` δηλαδή,

```
>> f=inline('x.^3-2*x-5')

f =

     Inline function:
     f(x) = x.^3-2*x-5

>> df=inline('3*x.^2-2')

df =

     Inline function:
     df(x) = 3*x.^2-2

>> out=newton(f, df, 1, 1/2*10^-8, 50)

out =

   1                   1                          -6
   2                   7                         324
   3     4.76551724137931          93.6945987125343
```

[2] Την παράγωγο συνάρτηση μπορούμε να την ορίσουμε και με αυτοματοποιημένο τρόπο

```
 4     3.34870275948028          25.8543115461301
 5     2.53159964100251          6.16181457004877
 6     2.17391588493923         0.925899647287488
 7     2.09788368644176        0.0372620055958821
 8     2.09455771585006     6.95840817304116e-005
 9     2.09455148156421     2.44225084600203e-010
10     2.09455148154233    -8.88178419700125e-016
```

Από τον πίνακα `out` τον οποίο επιστρέφει η συνάρτηση `newton` παρατηρούμε τα εξής:

- Για τον υπολογισμό της ρίζας εκτελέστηκαν 10 επαναλήψεις (πρώτη στήλη)
- Η προσεγγιστική τιμή της ρίζας είναι $x_{10} = 2.09455148154233$ (δεύτερη στήλη)
- Η τιμή της συνάρτησης είναι $f(x_{10}) = -8.88178419700125 \times 10^{-016}$ (τρίτη στήλη)
- Επομένως, η λύση στο πρόβλημα είναι $x_{10} = 2.09455148154233$

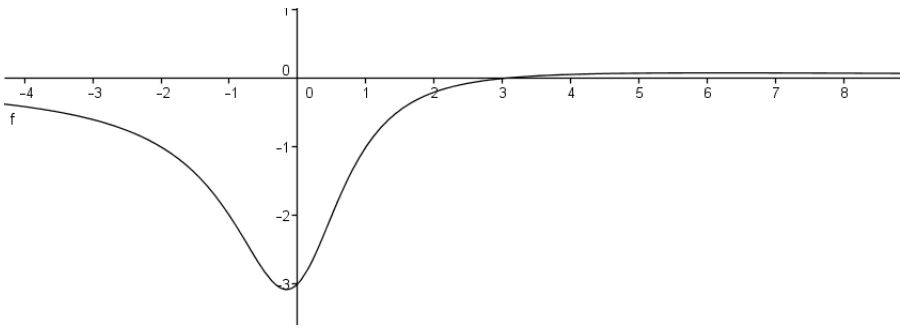
♦

Παράδειγμα 1.3.2 (Ειδική περίπτωση).
Να βρεθεί η ρίζα της συνάρτησης

$$f(x) = \frac{x-3}{x^2+1}$$

με τη μέθοδο Newton, με ακρίβεια 8 *δεκαδικών ψηφίων και με μέγιστο αριθμό επαναλήψεων* 50.

- η παραπάνω συνάρτηση έχει προφανή ρίζα το 3
- έχει ιδιόμορφη γραφική παράσταση



- πλησιάζει στο μηδέν όταν το $x \to \infty$ και $x \to -\infty$

Επομένως, σε MATLAB θα έχουμε

- ορίζουμε την συνάρτηση $f(x)$
  `f=inline('(x-3)./(x.^2+1)')`
- ορίζουμε την παράγωγο συνάρτηση $f'(x)$
  `df=inline(diff(sym(f)))`[3]
- καλούμε την συνάρτηση newton με τα κατάλληλα ορίσματα, με αρχική τιμή 2
  `newton(f, df, 2, 1/2*10^-8, 50)`
  Απάντηση: $x_7 = 3$
- με αρχική τιμή 5
  `newton(f, df, 5, 1/2*10^-8, 50)`
  Απάντηση: $x_{50} = -660095034734379$
  Δηλαδή, Εκτελέστηκε ο μέγιστος αριθμός των επαναλήψεων

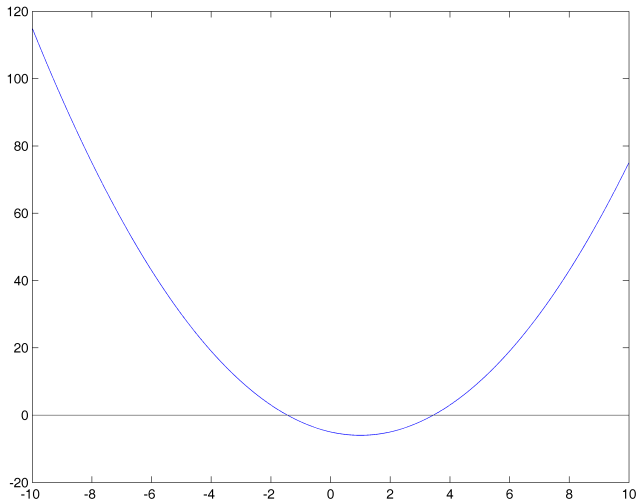
♦

Παράδειγμα 1.3.3. *Να βρεθεί η θετική ρίζα της συνάρτησης*

$$f(x) = x^2 - 2x - 5$$

με τη μέθοδο Newton, με ακρίβεια 8 *δεκαδικών ψηφίων και με μέγιστο αριθμό επαναλήψεων* 50.

- Για να βρούμε την θετική ρίζα της συνάρτησης θα πρέπει πρώτα να βρούμε μια κατάλληλη αρχική τιμή με την βοήθεια της γραφικής παράστασης της συνάρτησης.
- Σχεδιάζουμε την την γραφική παράσταση της συνάρτησης στο διάστημα $[-10, 10]$ και επιλέγουμε την κατάλληλη αρχική τιμή.



[3] Υπολογίζει την παράγωγο μιας inline function και μας επιστρέφει inline function

- Από την γραφική παράσταση της συνάρτησης επιλέγουμε αρχική τιμή το 4.
- ορίζουμε την συνάρτηση $f(x)$
  `f=inline('x.^2-2*x-5')`
- ορίζουμε την παράγωγο συνάρτηση $f'(x)$
  `df=inline(diff(sym(f)))`
- καλούμε την συνάρτηση newton με τα κατάλληλα ορίσματα
  `newton(f, df, 4, 1/2*10^-8, 50)`
  Απάντηση: $x_6 = 3.44948974278318$

♦

Άσκηση 1.3.1. *Δίνεται η συνάρτηση*

$$f(x) = e^x + 5x - 10$$

1. *Να βρεθεί η ρίζα της συνάρτησης με τη μέθοδο Newton, με αρχική τιμή $x_0 = 0$ με ακρίβεια 12 δεκαδικών ψηφίων και με μέγιστο αριθμό επαναλήψεων 50.*
2. *Να βρεθεί η ακρίβεια σε δεκαδικά ψηφία που έχουν οι τιμές x_6 και x_7.*
3. *Να βρεθεί η ακρίβεια σε σημαντικά ψηφία που έχουν οι τιμές x_6 και x_7.*

Κεφάλαιο 2

Παρεμβολή

2.1 Πολυωνυμική Παρεμβολή - Εργαστήριο 7

- Υπολογισμός Πολυωνύμου Παρεμβολής n βαθμού που διέρχεται από $n+1$ σημεία.
- Είσοδος
  - τα $n+1$ σημεία
- Έξοδος
  - το πολυώνυμο παρεμβολής n βαθμού

2.1.1 Γενική Μέθοδος

Έστω τα σημεία

$$(x_i, y_i) \quad i = 0, 1, \ldots, n$$

- Υπολογισμός του πίνακα Vandermonde

$$V = \begin{bmatrix} x_0^n & x_0^{n-1} & \cdots & x_0 & 1 \\ x_1^n & x_1^{n-1} & \cdots & x_1 & 1 \\ \vdots & \vdots & \ddots & & \vdots \\ x_n^n & x_n^{n-1} & \cdots & x_n & 1 \end{bmatrix}$$

- Υπολογισμός των συντελεστών του πολυωνύμου με βάση τον τύπο

$$p = V^{-1} \cdot y$$

οπού y το διάνυσμα στήλη των τεταγμένων των σημείων.

Παράδειγμα 2.1.1. *Έστω τα σημεία $A(1,1)$, $B(2,3)$ και $\Gamma(3,2)$.*
Να βρεθεί το πολυώνυμο παρεμβολής το οποίο διέρχεται από τα παραπάνω σημεία (με τη γενική μέθοδο).

- Υπολογισμός του πίνακα Vandermonde

$$V = \begin{bmatrix} 1^2 & 1^1 & 1 \\ 2^2 & 2^1 & 1 \\ 3^2 & 3^1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 4 & 2 & 1 \\ 9 & 3 & 1 \end{bmatrix}$$

- Υπολογισμός των συντελεστών του πολυωνύμου, σε MATLAB θα έχουμε

```
y=[1; 3; 2];
V=[1, 1, 1; 4, 2, 1; 9, 3, 1];
p=inv(V)*y
```

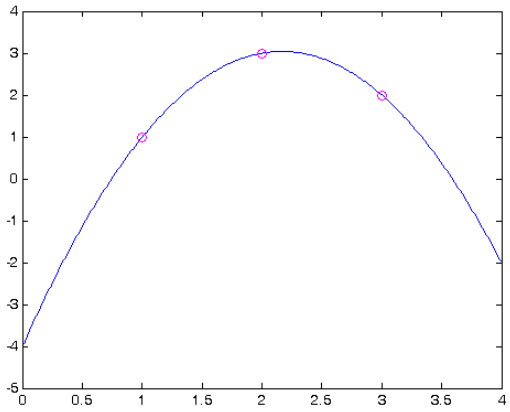
- Οι παραπάνω εντολές μας επιστρέφουν

```
p=
    -1.5
     6.5
    -4
```

- τα οποία είναι οι συντελεστές του πολυωνύμου, δηλαδή,

$$p(x) = -1.5x^2 + 6.5x - 4$$

Δηλαδή, το πολυώνυμο $p(x)$ διέρχεται από τα παραπάνω σημεία όπως φαίνεται στη γραφική παράσταση



♦

Άσκηση 2.1.1. *Έστω τα σημεία* $A(0,-4)$, $B(1,-2)$, $\Gamma(3,14)$ *και* $\Delta(4,40)$.

1. *Να βρεθεί το πολυώνυμο παρεμβολής το οποίο διέρχεται από τα παραπάνω σημεία (με τη γενική μέθοδο).*

   - *Απάντηση:* $\left(V = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 27 & 9 & 3 & 1 \\ 64 & 16 & 4 & 1 \end{bmatrix}, \quad p(x) = x^3 - 2x^2 + 3x - 4 \right)$

2.1.2 Πολυωνυμική Παρεμβολή - Συνάρτηση `interp1`

- Η πολυωνυμική παρεμβολή μπορεί να υλοποιηθεί με τη συνάρτηση `interp1`.
- `interp1(x,y,xi)`
  - `x,y` οι τετμημένες και οι τεταγμένες των δοθέντων σημείων
  - `xi` η παρεμβαλλόμενη τιμή
  - Η παραπάνω σύνταξη υλοποιεί γραμμική παρεμβολή
- Η σύνταξη `interp1(x,y,xi,'cubic')` υλοποιεί παρεμβολή με πολυώνυμα τρίτου βαθμού (κυβική παρεμβολή
- Η σύνταξη `interp1(x,y,xi,'spline')` υλοποιεί παρεμβολή με κυβικά splines

Παράδειγμα 2.1.2. *Έστω τα σημεία* $A(1,1)$, $B(2,3)$ *και* $\Gamma(3,2)$. *Να βρεθούν οι τιμές του* y

- *για* $x = 1.5$ *με γραμμική παρεμβολή*
- *για* $x = 2.9$ *με γραμμική παρεμβολή, με κυβική παρεμβολή και με παρεμβολή με κυβικά splines*

Σε MATLAB θα έχουμε

```
>> x=[1, 2, 3];
>> y=[1, 3, 2];
>> interp1(x,y,1.5)

ans =
     2

>> interp1(x,y,2.9)

ans =
                        2.1

>> interp1(x,y,2.9,'cubic')

ans =
                     2.2305

>> interp1(x,y,2.9,'spline')

ans =
                      2.235
```

♦

Άσκηση 2.1.2. *Δίνεται ο παρακάτω πίνακας τιμών*

x	2	3	4	5	6
y	-1	3	5	2	7

Να βρεθούν οι τιμές του y

1. *για* $x = 2.7$ *με γραμμική παρεμβολή*
   - *Απάντηση:* $y = 1.8$
2. *για* $x = 5.8$ *με γραμμική παρεμβολή και με κυβική παρεμβολή*
   - *Απάντηση:* $y = 6$ *και* $y = 5.328$

2.1.3 Πολυωνυμική Παρεμβολή - Συνάρτηση `polyfit`

- Εύρεση πολυωνύμου παρεμβολής με τη συνάρτηση `polyfit`.
- `polyfit(x,y,n)`
  - `x,y` οι τετμημένες και οι τεταγμένες των δοθέντων σημείων
  - `n` ο βαθμός του πολυωνύμου παρεμβολής
  - Στην περίπτωση που ο βαθμός `n` δεν είναι ίσος με το πλήθος των σημείων -1 , τότε η συνάρτηση υλοποιεί προσέγγιση.

Παράδειγμα 2.1.3. *Έστω τα σημεία* $A(1,1)$, $B(2,3)$ *και* $\Gamma(3,2)$.
Να βρεθεί το πολυώνυμο παρεμβολής που διέρχεται από τα παραπάνω σημεία.

Σε MATLAB θα έχουμε

- ```
  x=[1, 2, 3];
  y=[1, 3, 2];
  p=polyfit(x,y,2)
  ```
- Οι παραπάνω εντολές μας επιστρέφουν

```
p =
        -1.5          6.5           -4
```

- τα οποία είναι οι συντελεστές του πολυωνύμου, δηλαδή,

$$p(x) = -1.5x^2 + 6.5x - 4$$

♦
```

2.1.4 Polynomial Functions

- `polyval(p,x)` υπολογισμός της τιμής του πολυωνύμου `p` για την δοθείσα τιμή `x`
- `polyder(p)` υπολογισμός της παραγώγου συνάρτησης (πολυώνυμο) του πολυωνύμου `p`
- `polyint(p)` υπολογισμός της αρχικής συνάρτησης (πολυώνυμο) του πολυωνύμου `p`
- `conv(p1,p2)` υπολογισμός του γινομένου των πολυωνύμων `p1` και `p2`

Παράδειγμα 2.1.4. *Για το πολυώνυμο του προηγούμενου παραδείγματος να βρεθούν οι παρακάτω παραστάσεις*

- $p(1.7)$
- $p'(2)$

Σε MATLAB θα έχουμε

- από το προηγούμενο παράδειγμα

```
>> x=[1, 2, 3];
>> y=[1, 3, 2];
>> p=polyfit(x,y,2)

p =
       -1.5         6.5          -4

>> polyval(p,1.7)

ans =
           2.715

>> dp=polyder(p)

dp =
              -3              6.5

>> polyval(dp,2)

ans =
         0.499999999999999
```

♦

Άσκηση 2.1.3. *Έστω τα σημεία* $A(0,-4)$, $B(1,-2)$, $\Gamma(3,14)$ *και* $\Delta(4,40)$.

1. *Να βρεθεί το πολυώνυμο παρεμβολής το οποίο διέρχεται από τα παραπάνω σημεία.*

   - *Απάντηση:* $p(x) = x^3 - 2x^2 + 3x - 4$

2. *Να υπολογιστεί η τιμή* $p''(2)$*.*

   - *Απάντηση:* $p''(2) = 8$

Κεφάλαιο 3

Αριθμητική Ολοκλήρωση

3.1 Αριθμητική Ολοκλήρωση - Εργαστήριο 8

- Με την αριθμητική ολοκλήρωση υπολογίζεται η τιμή ενός ορισμένου ολοκληρώματος.
- Η αριθμητική ολοκλήρωση μπορεί να εφαρμοστεί είτε σε συνάρτηση είτε σε ένα σύνολο σημείων που προέρχονται από δειγματοληψία.
- Στην πρώτη προσέγγιση έχουμε
  - Είσοδος
    * τη συνάρτηση
    * το διάστημα
    * το πλήθος των υποδιαστημάτων που θα εφαρμοστεί η αριθμητική ολοκλήρωση
  - Έξοδος
    * η τιμή του ορισμένου ολοκληρώματος
- Στην δεύτερη προσέγγιση έχουμε
  - Είσοδος
    * τις τετμημένες και τις τεταγμένες των σημείων
  - Έξοδος
    * η τιμή του ορισμένου ολοκληρώματος

3.1.1 Κανόνας Τραπεζίου

Έστω μια συνάρτηση f. Η τιμή του ορισμένου ολοκληρώματος

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

προσεγγίζεται από τον κανόνα του τραπεζίου σύμφωνα με τον τύπο

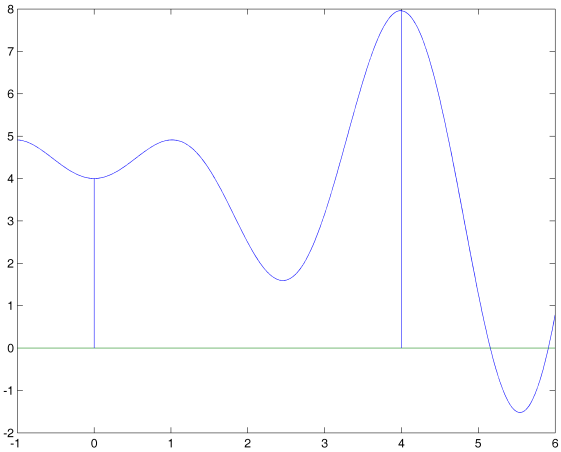
$$I = \frac{h}{2}\left(f_0 + 2 \cdot f_1 + \ldots + 2 \cdot f_{n-1} + f_n\right) = \frac{h}{2}\left(f_0 + 2 \cdot \sum_{i=1}^{n-1} f_i + f_n\right)$$

όπου n το πλήθος των υποδιαστημάτων που θα εφαρμοστεί ο τύπος και

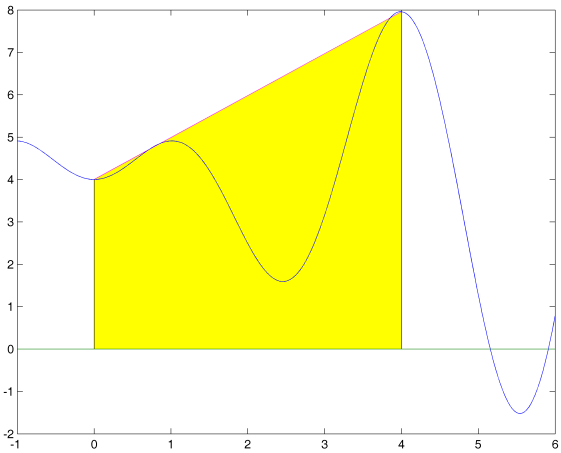
$$h = \frac{b-a}{n}$$

το βήμα των ισαπεχόντων σημείων.

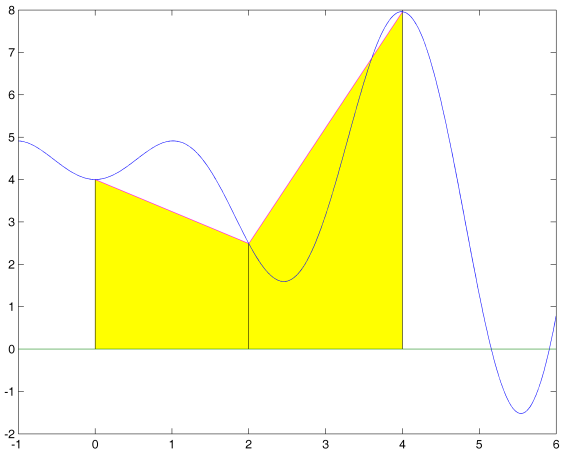
Γραφικά έχουμε τα εξής:



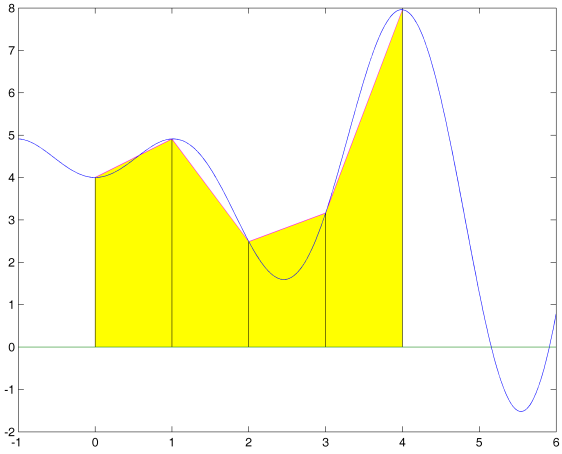
Υπολογισμός του ολοκληρώματος $\int_0^4 f(x)dx$



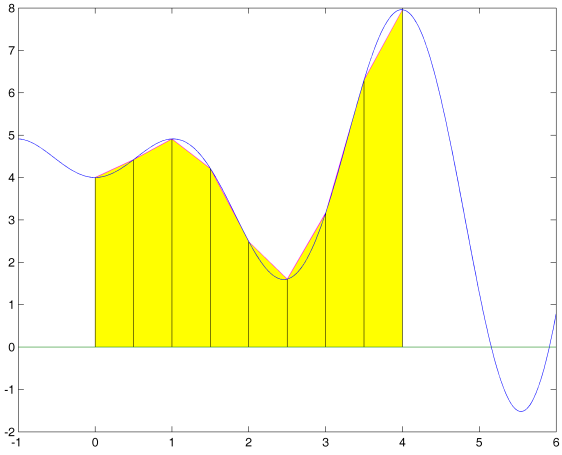
Κανόνας Τραπεζίου για $n = 1$



Κανόνας Τραπεζίου για $n = 2$



Κανόνας Τραπεζίου για $n = 4$



Κανόνας Τραπεζίου για $n = 8$

3.1.2 Κανόνας Τραπεζίου - Υλοποίηση

- Υλοποίηση πρώτης προσέγγισης σε συνάρτηση MATLAB

```
function I=trapeziou(f,a,b,n)
h=(b-a)/n;
S=f(a);
for i=1:n-1
   x=a+h*i;
   S=S+2*f(x);
end
S=S+f(b);
I=h*S/2;
```

Παράδειγμα 3.1.1. *Να υπολογιστεί η τιμή του ορισμένου ολοκληρώματος*

$$I = \int_0^4 (x \cdot \sin(2x) + 4)dx$$

με τον κανόνα του τραπεζίου για $n = 1,\ n = 2,\ n = 4,\ n = 100,\ n = 1000.$

Σε MATLAB θα έχουμε

```
>> f=inline('x.*sin(2*x)+4')

f =

     Inline function:
     f(x) = x.*sin(2*x)+4

>> trapeziou(f,0,4,1)

ans =
          23.9148659729871

>> trapeziou(f,0,4,2)

ans =
          16.9302230052618

>> trapeziou(f,0,4,4)

ans =
          16.5361624348598


>> trapeziou(f,0,4,100)
```

```
ans =
          16.5383163693361

>> trapeziou(f,0,4,1000)

ans =
          16.5383393964196
```

♦

Παράδειγμα 3.1.2. *Να υπολογιστεί η τιμή του ορισμένου ολοκληρώματος*

$$I = \int_0^4 (x \cdot \sin(2x) + 4)dx$$

με τον κανόνα του τραπεζίου για $n = 500$ *και να βρεθεί η ακρίβεια του αποτελέσματος σε δεκαδικά ψηφία.*

Σε MATLAB θα έχουμε

- ```
  >> f=inline('x.*sin(2*x)+4')

  f =

       Inline function:
       f(x) = x.*sin(2*x)+4

  >> I1=trapeziou(f,0,4,500)

  I1 =

              16.53833869789

  >> I2=trapeziou(f,0,4,499)

  I2 =

            16.5383386941534
  ```

- συγκρίνουμε τις τιμές ως προς τα δεκαδικά ψηφία τους σύμφωνα με τον τύπο

  $$|x_n - x_{n-1}| < \frac{1}{2}10^{-k} \Longrightarrow k < -\log\left(2 \cdot |x_n - x_{n-1}|\right)$$

  όπου k το πλήθος των δεκαδικών ψηφίων.
  ```

- ```
  >> -log10(2*abs(I1-I2))

  ans =

             8.1264968625927
  ```

  επειδή

  $$k < -\log\left(2 \cdot |I_{500} - I_{499}|\right) = 8.1264968625927$$

  θα έχουμε

  $$k = 8$$

  Επομένως, η προσεγγιστική λύση I_{500} έχει ακρίβεια 8 δεκαδικών ψηφίων.

♦

Άσκηση 3.1.1. *Δίνεται η συνάρτηση*

$$f(x) = e^{x^2}$$

1. *Να υπολογιστεί η τιμή του ορισμένου ολοκληρώματος*

   $$I = \int_0^3 e^{x^2} dx$$

   με τον κανόνα του τραπεζίου για $n = 100$.

   - *Απάντηση (1448.18921586593)*

2. *Να βρεθεί η ακρίβεια του αποτελέσματος σε δεκαδικά ψηφία.*

   - *Απάντηση (0 δ.ψ.)*

3.1.3 Κανόνας Simpson

Έστω μια συνάρτηση f. Η τιμή του ορισμένου ολοκληρώματος

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

προσεγγίζεται από τον κανόνα του Simpson σύμφωνα με τον τύπο

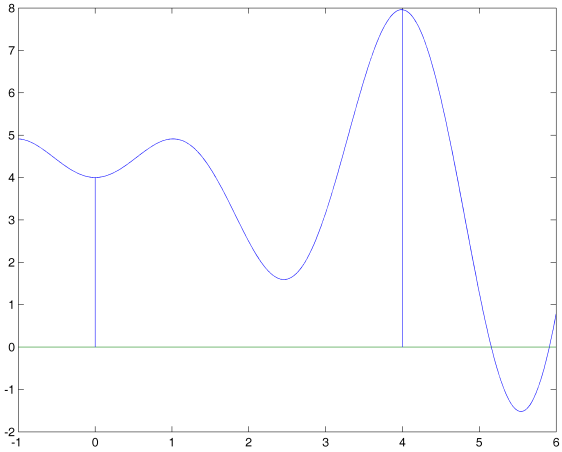
$$\begin{aligned} I &= \frac{h}{3}\left(f_0 + 4 \cdot f_1 + 2 \cdot f_2 + \ldots + 2 \cdot f_{2n-2} + 4 \cdot f_{2n-1} + f_n\right) \\ &= \frac{h}{3}\left(f_0 + 4 \cdot \sum_{i=1}^{n} f_{2i-1} + 2 \cdot \sum_{i=1}^{n-1} f_{2i} + f_n\right) \end{aligned}$$
  ```

όπου n το πλήθος των υποδιαστημάτων που θα εφαρμοστεί ο τύπος και

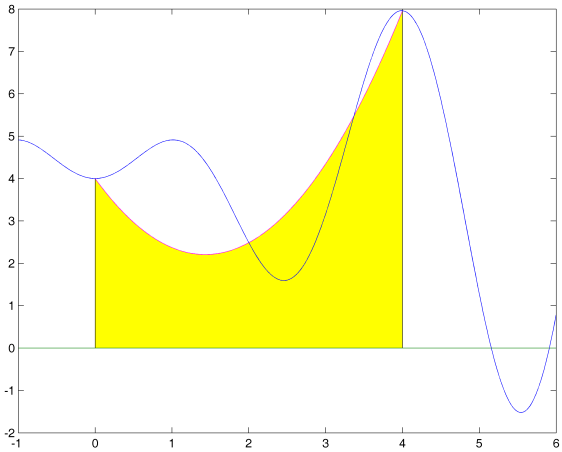
$$h = \frac{b-a}{2n}$$

το βήμα των ισαπέχοντων σημείων.

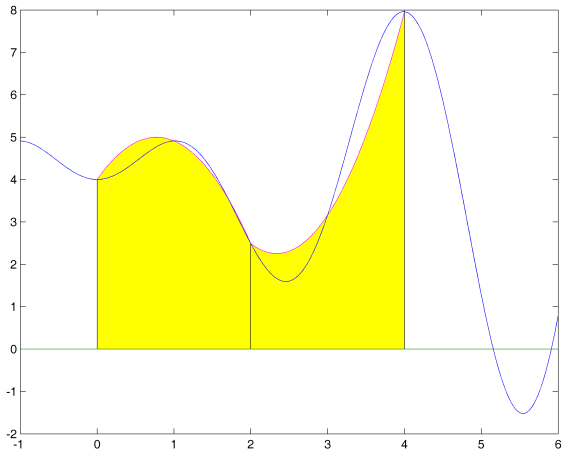
Γραφικά έχουμε τα εξής:



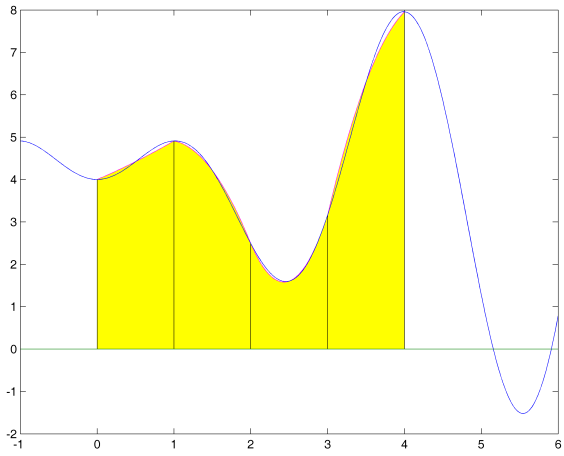
Υπολογισμός του ολοκληρώματος $\int_0^4 f(x)dx$



Κανόνας Simpson για $n = 1$



Κανόνας Simpson για $n = 2$



Κανόνας Simpson για $n = 4$

3.1.4 Κανόνας Simpson - Υλοποίηση

- Υλοποίηση πρώτης προσέγγισης σε συνάρτηση MATLAB

```
function I=simpson(f,a,b,n)
h=(b-a)/(2*n);
S=f(a);
for i=1:n
    x=a+h*(2*i-1);
    S=S+4*f(x);
end
for i=1:n-1
    x=a+h*(2*i);
    S=S+2*f(x);
end
```

```
S=S+f(b);
I=h*S/3;
```

Παράδειγμα 3.1.3. *Να υπολογιστεί η τιμή του ορισμένου ολοκληρώματος*

$$I = \int_0^4 (x \cdot \sin(2x) + 4)dx$$

με τον κανόνα του Simpson για $n = 1,\ n = 2,\ n = 4,\ n = 100,\ n = 1000.$

Σε MATLAB θα έχουμε

```
>> f=inline('x.*sin(2*x)+4')

f =

     Inline function:
     f(x) = x.*sin(2*x)+4


>> simpson(f,0,4,1)

ans =
          14.6020086826867

>> simpson(f,0,4,2)

ans =
          16.4048089113925

>> simpson(f,0,4,4)

ans =
          16.5350927538544


>> simpson(f,0,4,100)

ans =
           16.538339622856

>> simpson(f,0,4,1000)

ans =
          16.5383396292724
```

♦

Παράδειγμα 3.1.4. *Να υπολογιστεί η τιμή του ορισμένου ολοκληρώματος*

$$I = \int_0^4 (x \cdot \sin(2x) + 4)dx$$

με τον κανόνα του Simpson για $n = 500$ *και να βρεθεί η ακρίβεια του αποτελέσματος σε δεκαδικά ψηφία.*

Σε MATLAB θα έχουμε

- ```
  >> f=inline('x.*sin(2*x)+4')

  f =

       Inline function:
       f(x) = x.*sin(2*x)+4

  >> I1=simpson(f,0,4,500)

  I1 =

            16.5383396292628

  >> I2=simpson(f,0,4,499)

  I2 =

            16.5383396292627
  ```

- συγκρίνουμε τις τιμές ως προς τα δεκαδικά ψηφία τους σύμφωνα με τον τύπο

$$|x_n - x_{n-1}| < \frac{1}{2}10^{-k} \Longrightarrow k < -\log(2 \cdot |x_n - x_{n-1}|)$$

όπου k το πλήθος των δεκαδικών ψηφίων.

- ```
  >> -log10(2*abs(I1-I2))

  ans =

            12.6712885414875
  ```

επειδή

$$k < -\log(2 \cdot |I_{500} - I_{499}|) = 12.6712885414875$$

θα έχουμε

$$k = 12$$

Επομένως, η προσεγγιστική λύση I_{500} έχει ακρίβεια 12 δεκαδικών ψηφίων.

♦

Άσκηση 3.1.2. *Δίνεται η συνάρτηση*

$$f(x) = e^{x^2}$$

1. *Να υπολογιστεί η τιμή του ορισμένου ολοκληρώματος*

$$I = \int_0^3 e^{x^2} dx$$

με τον κανόνα του Simpson για $n = 100$.

- *Απάντηση (1444.54569643943)*

2. *Να βρεθεί η ακρίβεια του αποτελέσματος σε δεκαδικά ψηφία.*

- *Απάντηση (3 δ.ψ.)*

Άσκηση 3.1.3. *Δίνεται η συνάρτηση*

$$f(x) = e^{x^2}$$

1. *Να υπολογιστεί η τιμή του ορισμένου ολοκληρώματος*

$$I = \int_0^3 e^{x^2} dx$$

με τον κανόνα του Τραπεζίου και του Simpson για $n = 500$.

- *Απάντηση:*
  - *Μέθοδος Τραπεζίου: 1444.69097472801*
  - *Μέθοδος Simpson: 1444.54512381156*

2. *Να βρεθεί η ακρίβεια των αποτελεσμάτων σε δεκαδικά ψηφία.*

- *Απάντηση:*
  - *Μέθοδος Τραπεζίου: 2.93170469768348 (2 δ.ψ)*
  - *Μέθοδος Simpson: 7.8305603204051 (7 δ.ψ)*

3. *Ποια είναι η πιο ακριβής μέθοδος;*

- *Απάντηση: Η μέθοδος Simpson*

Κεφάλαιο 4

Λύσεις Ασκήσεων

4.1 Λύση της Άσκησης 1.1.1

1. Στον Editor γράφουμε

```
clear
clc
x(1)=2;
for i=2:20
    x(i)=(3+x(i-1))/(1+x(i-1));
end
k=1:length(x);
out=[k', x(k)']
```

Εκτελούμε και στο Command Window έχουμε

```
>>out =

          1                         2
          2          1.66666666666667
          3                      1.75
          4          1.72727272727273
          5          1.73333333333333
          6          1.73170731707317
          7          1.73214285714286
          8          1.73202614379085
          9          1.73205741626794
         10          1.73204903677758
         11          1.73205128205128
         12          1.73205068043172
         13          1.73205084163518
         14          1.73205079844084
         15          1.73205081001473
         16          1.73205080691351
```

```
        17          1.73205080774448
        18          1.73205080752182
        19          1.73205080758149
        20          1.7320508075655
```

2. Στο Command Window έχουμε για x_{15} και x_{20}

```
>>  -log10(2*abs(out(15,2)-out(14,2)))

ans =

          7.63549073463583

>> -log10(2*abs(out(20,2)-out(19,2)))

ans =

           10.495227442818
```

Άρα, 7 και 10 δεκαδικά ψηφία αντίστοιχα.

3. Στο Command Window έχουμε για x_{15} και x_{20}

```
>> -log10(2*abs((out(15,2)-out(14,2))/out(14,2)))+1

ans =

           8.8740513597069

>> -log10(2*abs((out(20,2)-out(19,2))/out(19,2)))+1

ans =

           11.733788070181
```

Άρα, 8 και 11 σημαντικά ψηφία αντίστοιχα.

4.2 Λύση της Άσκησης 1.2.1

1. Στο Command Window έχουμε

```
>> f=inline('exp(x)+5*x-10')

f =

     Inline function:
     f(x) = exp(x)+5*x-10

>> out=bisect(f, 0, 4, 1/2*10^-6, 50)

out =

     1                  0                  4                  2         7.38905609893065
     2                  0                  2                  1        -2.28171817154095
     3                  1                  2                1.5         1.98168907033806
     4                  1                1.5               1.25       -0.259657042538159
     5               1.25                1.5              1.375        0.830076722920577
     6               1.25              1.375             1.3125        0.277950737941104
     7               1.25             1.3125            1.28125      0.00738833627217517
     8               1.25            1.28125           1.265625       -0.126567138776458
     9           1.265625            1.28125          1.2734375       -0.0596984445940976
    10          1.2734375            1.28125          1.27734375       -0.0261824215879454
    11         1.27734375            1.28125           1.279296875     -0.00940389788415175
    12        1.279296875            1.28125          1.2802734375     -0.00100949628660274
    13       1.2802734375            1.28125          1.28076171875     0.00318899091319835
    14       1.2802734375          1.28076171875          1.280517578125    0.00108964006958701
    15       1.2802734375          1.280517578125         1.2803955078125   4.004508383737e-005
    16       1.2802734375         1.2803955078125         1.28033447265625 -0.000484732302886925
    17    1.28033447265625        1.2803955078125         1.28036499023438 -0.000222345284953462
    18    1.28036499023438        1.2803955078125         1.28038024902344 -9.11505194203244e-005
    19    1.28038024902344        1.2803955078125         1.28038787841797 -2.5552822508601e-005
    20    1.28038787841797        1.2803955078125         1.28039169311523  7.24610448443741e-006
    21    1.28038787841797       1.28039169311523          1.2803897857666 -9.15336555706858e-006
    22     1.2803897857666       1.28039169311523         1.28039073944092 -9.53632170563878e-007
    23    1.28039073944092       1.28039169311523         1.28039121627808  3.14623574837469e-006
```

2. Στο Command Window για $k = 10$ έχουμε

```
>> log2((4-0)*2*10^10)

ans =

          36.2192809488736
```

Άρα, θα εκτελεστούν $n = 37$ επαναλήψεις.

3. Στο Command Window για $n = 20$ έχουμε

```
>> log10(2^20/(2*(4-0)))

ans =

          5.11750992628768
```

Άρα, θα έχουμε ακρίβεια $k = 5$ δεκαδικών ψηφίων.

4. Στο Command Window έχουμε για x_{10} και x_{20}

```
>>  -log10(2*abs(out(10,4)-out(9,4)))

ans =

          2.10720996964787

>> -log10(2*abs(out(20,4)-out(19,4)))

ans =

          5.11750992628768
```

Άρα, 2 και 5 δεκαδικά ψηφία αντίστοιχα.

5. Στο Command Window έχουμε για x_{10} και x_{20}

```
>> -log10(2*abs((out(10,4)-out(10,4))/out(9,4)))+1

ans =

           3.21218760440396

>> -log10(2*abs((out(20,4)-out(19,4))/out(19,4)))+1

ans =

           6.2248514802625
```

Άρα, 3 και 6 σημαντικά ψηφία αντίστοιχα.

4.3 Λύση της Άσκησης 1.3.1

1. Στο Command Window έχουμε

```
>> f=inline('exp(x)+5*x-10')

f =

     Inline function:
     f(x) = exp(x)+5*x-10

>> df=inline(diff(sym(f)))

df =
```

```
     Inline function:
     df(x) = exp(x)+5

>> out=newton(f, df, 0, 1/2*10^-12, 50)

out =

     1                   0                          -9
     2                 1.5            1.98168907033806
     3     1.29099830677453          0.0914065359044152
     4     1.28041445503942        0.000202955171054953
     5     1.28039085047017      1.00238750633252e-009
     6     1.28039085035359     -1.77635683940025e-015
     7     1.28039085035359      1.77635683940025e-015
```

2. Στο Command Window έχουμε για x_6 και x_7

```
>> -log10(2*abs(out(6,2)-out(5,2)))

ans =

           9.63233407896853

>> -log10(2*abs(out(7,2)-out(6,2)))

ans =

            15.352529778863
```

Άρα, 9 και 15 δεκαδικά ψηφία αντίστοιχα.

3. Στο Command Window έχουμε για x_{10} και x_{20}

```
>> -log10(2*abs((out(6,2)-out(5,2))/out(5,2)))+1

ans =

           10.7396766410319

>> -log10(2*abs((out(7,2)-out(6,2))/out(6,2)))+1

ans =

           16.4598723408869
```

Άρα, 10 και 16 σημαντικά ψηφία αντίστοιχα.

Αλφαβητικό Ευρετήριο