ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΧΗΜΙΚΩΝ ΜΗΧΑΝΙΚΩΝ
ΥΠΟΛΟΓΙΣΤΙΚΟ ΚΕΝΤΡΟ

Προγραμματισμός με MATLAB – Συνοπτικός Οδηγός

Α. Σπυρόπουλος

Αθήνα, 2016

Το λογισμικό (software) MATLAB αποτελείται από μια υψηλού επιπέδου γλώσσα και ένα διαδραστικό περιβάλλον για αριθμητικούς υπολογισμούς, οπτικοποίηση και προγραμματισμό. Είναι εύκολο στην εκμάθησή του, ευέλικτο και χρήσιμο για τους μηχανικούς.

Το MATLAB χρησιμοποιεί τρία κύρια παράθυρα:
1) **Παράθυρο εντολών:** Χρησιμοποιείται για να εισάγουμε εντολές και δεδομένα.
2) **Παράθυρο γραφικών:** Χρησιμοποιείται για να εμφανίσουμε γραφικές παραστάσεις.
3) **Παράθυρο επεξεργασίας**: Χρησιμοποιείται για να δημιουργήσουμε και να επεξεργαστούμε m-αρχεία (m-files).

Οι σημειώσεις αυτές έχουν γραφτεί ως άσκηση hands-on. Δηλαδή, θα πρέπει να τις διαβάζετε, ενώ κάθεστε μπροστά στον υπολογιστή σας και να εφαρμόζετε στην πράξη τις εντολές που περιγράφονται.

1 Παράθυρο εντολών

1.1 Το MATLAB ως "calculator"

Αφού ξεκινήσει το MATLAB στο παράθυρο εντολών εμφανίζεται το σύμβολο της γραμμής εντολών (command prompt):

```
>>
```

στο σημείο αυτό το MATLAB περιμένει να γράψουμε τις εντολές οι οποίες θα εκτελεστούν σειριακά γραμμή προς γραμμή. Για παράδειγμα αν γράψουμε:

```
>> 10+3 (και πατήσουμε το ENTER)
```

το MATLAB θα "απαντήσει":

```
ans =

    13
```

Παρατηρούμε ότι το MATLAB ορίζει μια προκαθορισμένη μεταβλητή με το όνομα `ans` η οποία θα φιλοξενήσει το αποτέλεσμα. Για παράδειγμα αν στη συνέχεια γράψουμε:

```
>>  ans+5 (και πατήσουμε το ENTER)
```

το MATLAB θα "απαντήσει":

```
ans =

    18
```

Το MATLAB **εκχωρεί** (assign) το αποτέλεσμα στη μεταβλητή `ans`. Αν θέλουμε να ορίσουμε μια άλλη μεταβλητή πρέπει να γράψουμε για παράδειγμα:

```
>> y=10+3
```

το MATLAB θα εκχωρήσει στη μεταβλητή **`y`** την τιμή 13 και θα "απαντήσει":

```
y =

    13
```

ΠΑΡΑΤΗΡΗΣΕΙΣ

Αν στο τέλος της προηγούμενης εντολής προσθέσουμε το σύμβολο **`;`** τότε το MATLAB εκχωρεί την τιμή του στην μεταβλητή **`y`** αλλά δεν τυπώνει το αποτέλεσμα.

Το MATLAB χρησιμοποιεί τη διάκριση πεζών-κεφαλαίων (case sensitive). Για παράδειγμα μια μεταβλητή με το όνομα **`result`** είναι διαφορετική από τη μεταβλητή **`Result`**

Δυο ή περισσότερες εντολές μπορούν να γραφτούν στην ίδια γραμμή αρκεί να χωρίζονται με το σύμβολο **`;`** ή το **`,`**

ΜΕΤΕΒΛΗΤΕΣ

Μεταβλητή ονομάζεται ένα μέγεθος του οποίου η τιμή μπορεί να αλλάζει κατά τη διάρκεια εκτέλεσης μιας εντολής ή ενός συνόλου εντολών (πρόγραμμα).

ΚΑΝΟΝΕΣ ΟΝΟΜΑΣΙΑΣ ΜΕΤΑΒΛΗΤΩΝ

Το όνομα μια μεταβλητής πρέπει να:
1) αρχίζει με γράμμα του αγγλικού αλφαβήτου,
2) περιέχει μόνο γράμματα του αγγλικού αλφαβήτου, αριθμούς ή το σύμβολο της κάτω παύλας (underscore),
3) μην ταυτίζεται με τα ονόματα των συναρτήσεων ή των μεταβλητών (προκαθορισμένες μεταβλητές) του MATLAB.

ΠΡΟΣΟΧΗ: Στο MATLAB υπάρχει διάκριση μεταξύ κεφαλαίων και πεζών (case sensitive)

Άσκηση: Δοκιμάστε να γράψετε **`x=4,a=5`** και στη συνέχεια, σε νέα γραμμή εντολών, **`x=4;a=5;`**. Τι παρατηρείτε;

Στους επόμενους πίνακες δίνονται οι αριθμητικοί τελεστές και ορισμένες από τις μαθηματικές συναρτήσεις που χρησιμοποιούνται στο MATLAB.

Όπως και σε κάθε άλλη γλώσσα προγραμματισμού, οι μεταβλητές που βλέπετε στους πίνακες πρέπει να αντιστοιχούν σε αριθμούς. Δηλαδή σε κάθε μεταβλητή πρέπει να έχει εκχωρηθεί μια συγκεκριμένη τιμή.

Αριθμητικοί τελεστές

Μαθηματική έκφραση	MATLAB
$\alpha + \beta$	`a + b`
$\alpha - \beta$	`a - b`
$\alpha\beta$	`a*b`
$\frac{\alpha}{\beta}$	`a/b`
α^{β}	`a^b`

Η προτεραιότητα των αριθμητικών τελεστών είναι:
^ Υψηλή
* / Μεσαία
+ - Χαμηλή

Για παράδειγμα στην αριθμητική παράσταση:

a = f/h^g + i
η σειρά που θα γίνουν οι πράξεις είναι:

1. Υπολογισμός του h^g και αποθήκευση του αποτελέσματος σε μια προσωρινή μεταβλητή temp_1
2. Υπολογισμός του f/temp_1 και αποθήκευση του αποτελέσματος σε μια προσωρινή μεταβλητή temp_2
3. Υπολογισμός του temp_2 + i και αποθήκευση του αποτελέσματος στη μεταβλητή a

Υπάρχει περίπτωση να αλλάξουμε τη προκαθορισμένη σειρά (προτεραιότητα των τελεστών) που γίνονται οι πράξεις χρησιμοποιώντας παρενθέσεις ( )

Για παράδειγμα στην αριθμητική παράσταση:

a = (f/h)^g + i

η σειρά που θα γίνουν οι πράξεις είναι:

1. Υπολογισμός του f/h και αποθήκευση του αποτελέσματος σε μια προσωρινή μεταβλητή temp_1
2. Υπολογισμός του temp_1^g και αποθήκευση του αποτελέσματος σε μια προσωρινή μεταβλητή temp_2
3. Υπολογισμός του temp_2 + i και αποθήκευση του αποτελέσματος στη μεταβλητή a

Μεταξύ τελεστών της ίδιας προτεραιότητας οι πράξεις γίνονται από αριστερά προς τα δεξιά.

Μαθηματικές συναρτήσεις

Μαθηματική έκφραση	Εντολή
sin(α)	`sin(a)`
cos(α)	`cos(a)`
tan(α)	`tan(a)`
$\sqrt{\alpha}$	`sqrt(a)`
|α|	`abs(a)`
e^{α}	`exp(a)`
ln(α)	`log(a)`
$\log_{10}(\alpha)$	`log10(a)`
$\sin^{-1}(\alpha)$	`asin(a)`
$\cos^{-1}(\alpha)$	`acos(a)`
$\tan^{-1}(\alpha)$	`atan(a)`
Επιστρέφει έναν πίνακα διαστάσεων a×b με στοιχεία τυχαίους αριθμούς κανονικά κατανεμημένους στο διάστημα (0 1]	`rand(a,b)`
Στρογγυλοποίηση στον πλησιέστερο ακέραιο	`round(a)`
Στρογγυλοποίηση στον μικρότερο ακέραιο	`floor(a)`
Στρογγυλοποίηση στον μεγαλύτερο ακέραιο	`ceil(a)`

Το MATLAB χρησιμοποιεί ορισμένες προκαθορισμένες μεταβλητές (predefined variables). Μπορούμε να εκχωρήσουμε δικές μας τιμές στις μεταβλητές αυτές αλλά αυτό μπορεί να οδηγήσει σε **λάθος υπολογισμούς**.

Προκαθορισμένες μεταβλητές

Μεταβλητή	Επεξήγηση
`ans`	Η μεταβλητή αυτή "φιλοξενεί" το αποτέλεσμα ενός υπολογισμού αν δεν το εκχωρήσουμε σε άλλη μεταβλητή.
`pi`	3.1415 . . .
`eps`	2.220446049250313e-16 Η ακρίβεια των υπολογισμών (machine epsilon). Είναι η απόσταση του αριθμού 1 από την αμέσως μεγαλύτερη τιμή. Το eps χαρακτηρίζει τον υπολογιστή στο πεδίο της αριθμητικής ανάλυσης.
`i`	Ο μιγαδικός αριθμός `i` ή `j` με την ιδιότητα $i^2=-1$ ή $j^2=-1$ αντίστοιχα
`j`	
`NaN`	"Not a Number". Για παράδειγμα η εντολή 0/0 έχει ως αποτέλεσμα `NaN`.
`nan`	
`Inf`	Η τιμή του άπειρου (infinity). Για παράδειγμα η εντολή 1/0 έχει ως αποτέλεσμα `Inf`.
`inf`	
`true`	1
`false`	0

Για παράδειγμα, για να ορίσουμε μιγαδικές μεταβλητές μπορούμε να χρησιμοποιήσουμε τα σύμβολα **`i`** ή **`j`**

```
>> x=2+4i
```

το MATLAB θα "απαντήσει":

```
x =

   2.0000 + 4.0000i
```

Για να δούμε την τιμή της μεταβλητής `pi` γράφουμε:

```
>> pi
```

το MATLAB θα "απαντήσει":

```
ans =

    3.1416
```

Αν θέλουμε περισσότερα δεκαδικά ψηφία γράφουμε την εντολή:

```
>> format long
```

και στη συνέχεια

```
>> pi
```

το MATLAB θα "απαντήσει":

```
ans =

   3.141592653589793
```

Στον παρακάτω πίνακα δίνονται οι τιμές για την μεταβλητή `pi` που θα τυπώσει το MATLAB ανάλογα με τον τύπο (type) της εντολής format

`format` *type*

type	Τιμή της `pi`
`short`	`3.1416`
`long`	`3.141592653589793`
`short e`	`3.1416e+00`
`long e`	`3.141592653589793e+00`

Μια μεταβλητή που "φιλοξενεί" μια μόνο τιμή την ονομάζουμε βαθμωτή (scalar).

Χρήσιμες εντολές

Εντολή	Επεξήγηση
`who`	Τυπώνει την τρέχουσα λίστα με τις μεταβλητές που είναι σε χρήση.
`whos`	Τυπώνει την τρέχουσα λίστα με τις μεταβλητές που είναι σε χρήση με επιπλέον πληροφορίες.
`clear`	"Καταστρέφει" όλες τις μεταβλητές που είναι σε χρήση.
`clear a b`	"Καταστρέφει" τις μεταβλητές `a` και `b`
`help` *`command`*	*`command`* = εντολή ή συνάρτησης του MATLAB. Για παράδειγμα αν γράψουμε: `help format` θα πάρουμε πληροφορίες για τη χρήση και το αποτέλεσμα της εντολής `format`.
`lookfor '`*`keyword`*`'`	*`keyword`* = μια λέξη κλειδί. Για παράδειγμα αν γράψουμε: `lookfor 'newton'` θα πάρουμε πληροφορίες για όλες τις εντολές ή συναρτήσεις της MATLAB που έχουν στην περιγραφή τους τη λέξη newton.
`why`	?

1.2 Arrays - Διανύσματα και πίνακες

Οι arrays, σε αντίθεση με τις scalars, είναι μεταβλητές που μπορούν να φιλοξενήσουν περισσότερες από μία τιμές.

Για παράδειγμα μπορούμε να ορίσουμε μια **μονοδιάστατη array - διάνυσμα**, με όνομα a, αν γράψουμε:

```
>> a = [1 2 3 4]

a =

   1   2   3   4
```

Οι αγκύλες [ ] (brackets) χρησιμοποιούνται για να δώσουμε τιμές στις arrays. Με την παραπάνω εντολή δημιουργείτε ένα διάνυσμα με 1 γραμμή και 4 στήλες (1x4). **Συνήθως στη γραμμική άλγεβρα όταν μιλάμε για διανύσματα εννοούμε τις arrays με μια στήλη.**
Για παράδειγμα μπορούμε να ορίσουμε ένα διάνυσμα με όνομα b που θα έχει 4 γραμμές και 1 στήλη (4x1) αν γράψουμε:

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

b =

     1
     2
     3
     4
```

ή αν χρησιμοποιήσουμε τον **τελεστή (operator) αναστροφής '**

```
>> b=[1 2 3 4]'

b =

     1
     2
     3
     4
```

Από εδώ και πέρα θα χρησιμοποιούνται οι εξής ορισμοί:
Διάνυσμα γραμμής διάστασης n: Μονοδιάστατη array με 1 γραμμή και *n* στήλες
Διάνυσμα διάστασης n: Μονοδιάστατη array με *n* γραμμές και 1 στήλη

Για να ορίσουμε **διδιάστατες arrays - πίνακες** γράφουμε (στους πίνακες δίνουμε συνήθως όνομα με κεφαλαία):

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

A =

     1     2
     3     4
     5     6
```

Με την παραπάνω εντολή δημιουργήθηκε ο πίνακας A με 3 γραμμές και 2 στήλες ή πιο απλά 3x2

Αν γράψουμε την εντολή whos θα πάρουμε πληροφορίες για τα διανύσματα a, b και για τον πίνακα A που φτιάξαμε:

```
>> whos
  Name        Size                Bytes  Class     Attributes

  A           3x2                    48  double
  a           1x4                    32  double
  b           4x1                    32  double
```

Μπορούμε να δούμε την τιμή επιλεγμένων στοιχείων ενός διανύσματος ή πίνακα χρησιμοποιώντας το όνομα της μεταβλητής και τις παρενθέσεις **( )**. Για παράδειγμα αν γράψουμε:

```
>> A(3,2)

ans =

     6
```

βλέπουμε την τιμή που έχει το στοιχείο του πίνακα A που βρίσκεται στην 3η γραμμή και 2η στήλη.

Χρήσιμες συναρτήσεις

Εντολή	Επεξήγηση
`zeros(m,n)`	Δημιουργεί πίνακα με *m* γραμμές και *n* στήλες με όλα τα στοιχεία του ίσα με το 0
`ones(m,n)`	Δημιουργεί πίνακα με *m* γραμμές και *n* στήλες με όλα τα στοιχεία του ίσα με το 1

1.3 Ο τελεστής : (colon operator)

Εντολή	Επεξήγηση

m:*n*	Δημιουργεί αριθμούς από το *m* μέχρι και το *n* με βήμα 1
	Για παράδειγμα:
	`>> i=1:5`
	
	`i =`
	
	`1 2 3 4 5`
	Τη μεταβλητή i μπορούμε να τη δούμε ως διάνυσμα γραμμής διάστασης 5

m:*s*:*n*	Δημιουργεί αριθμούς από το *m* μέχρι και το *n* με βήμα *s*
	Για παράδειγμα:
	`>> x=2:0.5:4`
	
	`x =`
	
	`2.0000 2.5000 3.0000 3.5000 4.0000`

:	Μπαλαντέρ (wildcard)

<u>Παράδειγμα του τελεστή : ως wildcard</u>
Αν θέλουμε να δούμε όλα τα στοιχεία μιας στήλης ή μιας γραμμής του πίνακα Α χρησιμοποιούμε το σύμβολο :

Για παράδειγμα αν θέλουμε να δούμε όλα τα στοιχεία της 2ης στήλης του πίνακα Α που έχει οριστεί ως:

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

A =

     1     2
     3     4
     5     6
```

γράφουμε:

```
>> A(:,2)

ans =

     2
     4
     6
```

1.4 Η συνάρτηση `linspace`

Η συνάρτηση `linspace` δημιουργεί διανύσματα γραμμής με προκαθορισμένο πλήθος στοιχείων μεταξύ δυο τιμών (αρχική και τελική τιμή). Σε αντίθεση με τον τελεστή `:` που δημιουργεί και αυτός διανύσματα γραμμής μεταξύ δυο τιμών με προκαθορισμένο βήμα.

Για παράδειγμα για να δημιουργήσουμε ένα διάνυσμα γραμμής διάστασης 10 (10 στοιχεία) στο διάστημα [0 1] γράφουμε την εντολή:

```
>> x=linspace(0,1,10)

x =

  Columns 1 through 6

         0    0.1111    0.2222    0.3333    0.4444    0.5556

  Columns 7 through 10

    0.6667    0.7778    0.8889    1.0000
```

Αν παραλειφθεί ο αριθμός των σημείων τότε το MATLAB δημιουργεί 100 σημεία.

1.5 Πράξεις μεταξύ των arrays

Το MATLAB για να κάνει πράξεις μεταξύ των arrays χρησιμοποιεί τους ορισμούς της γραμμικής άλγεβρας. Αν θέλουμε να αλλάξουμε αυτή τη συμπεριφορά και να κάνουμε πράξεις στοιχείο με στοιχείο (element by element), πρέπει πριν από τον τελεστή της μαθηματικής πράξης να βάλουμε το σύμβολο της τελείας .

Παράδειγμα

Έστω

a: βαθμωτή μεταβλητή

x,y: διανύσματα διάστασης *m*

A: πίνακας διάστασης *nxm*

Εντολή	Επεξήγηση
`y'*x`	Εσωτερικό γινόμενο (inner product)
`x*y'`	Εξωτερικό γινόμενο (outer product)
`A*x`	Πολλαπλασιασμός πίνακα με διάνυσμα
`A\x`	$A^{-1}x$
`x+y`	Πρόσθεση μεταξύ διανυσμάτων
`a*x`	Πολλαπλασιασμός βαθμωτής μεταβλητής με διάνυσμα
`A/a`	Διαίρεση πίνακα με βαθμωτή μεταβλητή

Έστω τα διανύσματα x,y διάστασης m=4:

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

x =

     1
     2
     3
     4

>> y=[5;6;7;8]

y =

     5
     6
     7
     8
```

Αν θέλουμε να υπολογίσουμε το εσωτερικό γινόμενο των διανυσμάτων αυτών γράφουμε:

```
>> y'*x
```

```
ans =

    70
```

ή

```
>> x'*y

ans =

    70
```

Αν γράψουμε:

```
>> x*y
```

θα εμφανιστεί το μήνυμα σφάλματος (error message):

```
Error using  *
Inner matrix dimensions must agree.
```

Αυτό συμβαίνει γιατί ο πολλαπλασιασμός μεταξύ των διανυσμάτων με διαστάσεις `mx1 * mx1` δεν ορίζεται στη γραμμική άλγεβρα. Το ίδιο συμβαίνει και για τον πολλαπλασιασμό μεταξύ πινάκα με πίνακα και πίνακα με διάνυσμα.

Αν στο προηγούμενο παράδειγμα θέλουμε να πολλαπλασιάσουμε τα διανύσματα x, y στοιχείο προς στοιχείο, τότε πρέπει να χρησιμοποιήσουμε το σύμβολο `.` πριν από το `*`

```
>> x.*y

ans =

     5
    12
    21
    32
```

Οι περισσότερες μαθηματικές συναρτήσεις του MATLAB όπως οι `sqrt, abs, sin, acos, tanh, exp` αν χρησιμοποιηθούν με όρισμα array κάνουν υπολογισμούς στοιχείο προς στοιχείο.

```
>> sqrt(x)

ans =
```

```
    1.0000
    1.4142
    1.7321
    2.0000
```

Αν υπάρχει και αντίστοιχος ορισμός στην γραμμική άλγεβρα για τις συναρτήσεις αυτές και θέλουμε το MATLAB να κάνει τον υπολογισμό με βάση αυτόν τον ορισμό τότε πρέπει να χρησιμοποιηθεί το γράμμα **m** μετά το όνομα της συνάρτησης. Για παράδειγμα:

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

A =

     1     2     3
     4     5     6
     7     8     9

>> sqrt(A)


ans =

    1.0000    1.4142    1.7321
    2.0000    2.2361    2.4495
    2.6458    2.8284    3.0000

>> sqrtm(A)

ans =

   0.4498 + 0.7623i   0.5526 + 0.2068i   0.6555 - 0.3487i
   1.0185 + 0.0842i   1.2515 + 0.0228i   1.4844 - 0.0385i
   1.5873 - 0.5940i   1.9503 - 0.1611i   2.3134 + 0.2717i
```

Έστω a μια μονοδιάστατη array. Μπορούμε να πάρουμε πληροφορίες για τα στοιχεία της array, χρησιμοποιώντας τις παρακάτω συναρτήσεις:

Μαθηματικές συναρτήσεις

MATLAB	Αποτέλεσμα
`sum(a)`	Άθροισμα των στοιχείων της array a
`prod(a)`	Γινόμενο των στοιχείων της array a
`min(a)`	Το μικρότερο στοιχείο της array a
`max(a)`	Το μεγαλύτερο στοιχείο της array a
`mean(a)`	Μέση τιμή των στοιχείων της array a
`sort(a)`	Ταξινόμηση των στοιχείων κατά αύξουσα σειρά της array a
`length(a)`	Το πλήθος των στοιχείων της array a

1.6 Η συνάρτηση `input`

Με τη συνάρτηση αυτή προτρέπουμε (prompt) τον χρήστη να πληκτρολογήσει μια τιμή η οποία θα εκχωρηθεί σε αντίστοιχη μεταβλητή. Για παράδειγμα:

```
>> clear
>> a=input('Give me a value: ')
Give me a value: 5

a =

     5
```

1.7 Η συνάρτηση `disp`

Με τη συνάρτηση αυτή μπορούμε εμφανίσουμε στην οθόνη την τιμή μιας μεταβλητής:

```
>> clear
>> x=10;
>> disp(x)
    10
```

ή να εμφανίσουμε ένα μήνυμα στον χρήστη:

```
>> disp('MATLAB is fun')
MATLAB is fun
```

1.8 Η συνάρτηση `fprintf:`

Η συνάρτηση αυτή μας δίνει πλήρη έλεγχο στον τρόπο με τον οποίο θα εμφανίσουμε στην οθόνη μηνύματα ή/και μεταβλητές. Η σύνταξη της έχει την μορφή:

```
fprintf('format', x, y, . . .)
```

όπου format είναι η μορφή με την οποία θέλουμε να εμφανίσουμε τις μεταβλητές x, y, . . .

format	Επεξήγηση
`%d`	Ακέραια μορφή
`%e`	Εκθετική μορφή με μικρό e
`%E`	Εκθετική μορφή με κεφαλαίο E
`%f`	Δεκαδική μορφή
`%g`	Σημαντικά ψηφία (Significant digits)
`\n`	Νέα γραμμή
`\t`	Tab

Παράδειγμα

```
>> fprintf('The value of pi is: %e \n',pi)
The value of pi is: 3.141593e+00
```

```
>> fprintf('The value of pi is: %f \n',pi)
The value of pi is: 3.141593
```

Μπορούμε επίσης να ορίσουμε το πλήθος των δεκαδικών ψηφίων καθώς και το πλήθος των σημαντικών ψηφίων που θα τυπωθούν.

Για παράδειγμα αν θέλουμε τα τυπώσουμε την τιμή του pi με τρία σημαντικά ψηφία γράφουμε:

```
>> fprintf('The value of pi is: %.3g \n',pi)
```

```
The value of pi is: 3.14
```
Ενώ αν θέλουμε να τυπώσουμε την τιμή του pi με τρία δεκαδικά ψηφία γράφουμε:

```
>> fprintf('The value of pi is: %.3f \n',pi)
The value of pi is: 3.142
```

Τέλος μπορούμε να ορίσουμε και το ελάχιστο πλήθος των χαρακτήρων (πλήθος ψηφίων + το σύμβολο **.** για τους δεκαδικούς) που θα τυπωθούν:

```
>> fprintf('The value of pi is: %7.3f \n',pi)
The value of pi is:   3.142
```

1.9 Δημιουργία και προσπέλαση αρχείων

Το MATLAB έχει τη δυνατότητα να διαβάσει και να γράψει αρχεία δεδομένων. Η μια προσέγγιση περιλαμβάνει έναν ειδικό τύπο (δυαδικό αρχείο - binary file), που ονομάζεται **mat-file**, το οποίο είναι ειδικά σχεδιασμένο για το περιβάλλον του MATLAB. Τέτοια αρχεία δημιουργούνται και είναι προσπελάσιμα με τις εντολές **save** και **load**.

Η εντολή **save** χρησιμοποιείται για την δημιουργία αρχείου το οποίο μπορεί να περιέχει όλες ή μέρος των μεταβλητών που έχουν οριστεί. Η σύνταξή της είναι:

```
save filename var1 var2 . . .
```

όπου `filname` το όνομα του αρχείου που θα δημιουργηθεί (`filname`.mat) ,
και `var1 var2` τα ονόματα των μεταβλητών που θέλουμε να αποθηκεύσουμε στο αρχείο.
Αν παραλείψουμε τα ονόματα των μεταβλητών τότε όλες οι μεταβλητές που έχουμε ορίσει αποθηκεύονται στο αρχείο `filname`.mat

Η εντολή **load** ανακτά τις μεταβλητές που έχουμε αποθηκεύσει σε ένα αρχείο. Η σύνταξή της είναι:

```
load filename var1 var2 . . .
```

Αν παραλείψουμε τα ονόματα των μεταβλητών τότε ανακτώνται όλες οι μεταβλητές που έχουμε αποθηκεύσει στο αρχείο `filname`.mat

Παράδειγμα

```
>> clear
>> x=3

x =

     3
```

```
>> A=rand(3)

A =

    0.6998    0.7670    0.8703
    0.8605    0.7036    0.1624
    0.0761    0.8925    0.6426

>> save myvars
>> clear
>> A
Undefined function or variable 'A'.

>> load myvars x
>> A
Undefined function or variable 'A'.

>> x

x =

     3

>> load myvars A
>> A

A =

    0.6998    0.7670    0.8703
    0.8605    0.7036    0.1624
    0.0761    0.8925    0.6426
```

Αν θέλουμε τα αρχεία που δημιουργούμε να είναι προσπελάσιμα και από άλλες εφαρμογές εκτός του MATLAB τότε πρέπει να τα αποθηκεύσουμε σε μορφή ASCII (text format). Αυτό γίνεται προσθέτοντας το **-ascii** στην εντολή **save**. Για παράδειγμα:

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

A =

    0.4090    0.5664    0.2768
    0.1580    0.1700    0.9085
    0.6140    0.2747    0.5410
```

```
>> save file.txt A -ascii
>> clear
>> A
Undefined function or variable 'A'.

>> load file.txt
>> A
Undefined function or variable 'A'.

>> file

file =

    0.4090    0.5664    0.2768
    0.1580    0.1700    0.9085
    0.6140    0.2747    0.5410
```

Όπως παρατηρούμε κατά την ανάκτηση των δεδομένων με την εντολή `load`, από ένα αρχείο που είναι σε μορφή ASCII, δημιουργείται μια μεταβλητή που έχει το όνομα του αρχείου αυτού.

Εναλλακτικά θα μπορούσαμε να χρησιμοποιήσουμε την εντολή `load` ως συνάρτηση και να δώσουμε απευθείας τιμή σε μια μεταβλητή:

```
>> A=load('file.txt')

A =

    0.4090    0.5664    0.2768
    0.1580    0.1700    0.9085
    0.6140    0.2747    0.5410
```

2 Παράθυρο γραφικών

Με το MATLAB μπορούμε να κάνουμε την γραφική παράσταση συναρτήσεων γρήγορα και εύκολα.
Έστω ότι θέλουμε να φτιάξουμε την γραφική παράσταση της $f(x)=x^2+2$ στο διάστημα [-2 2]

πρώτα θα δώσουμε τιμές στη μεταβλητή x στο διάστημα [-2 2] με βήμα 0.1

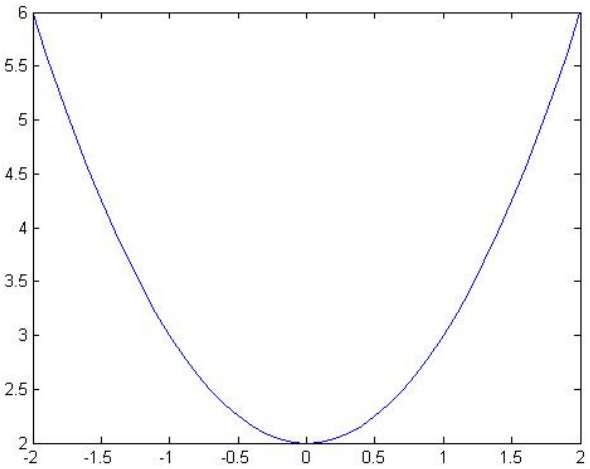
```
>> x=-2:0.1:2;
```

Στη συνέχεια δίνουμε τιμές στη μεταβλητή $y= x^2+2$

```
>> y=x.^2+2;
```

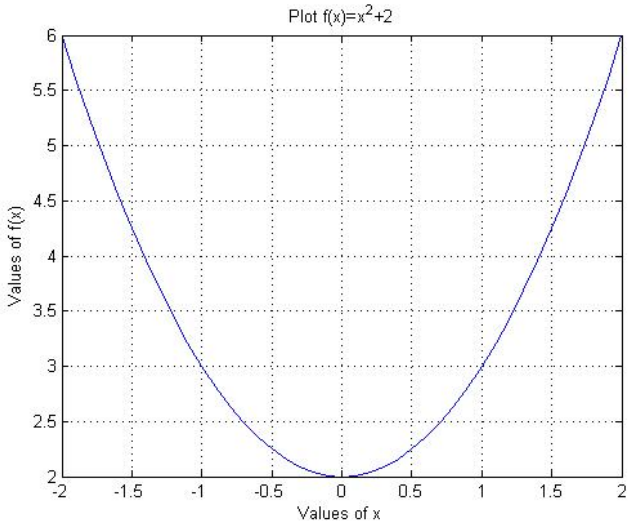
Η γραφική παράσταση θα φτιαχτεί με την εντολή **plot**

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



Μπορούμε να διαμορφώσουμε το διάγραμμα με τις παρακάτω εντολές:

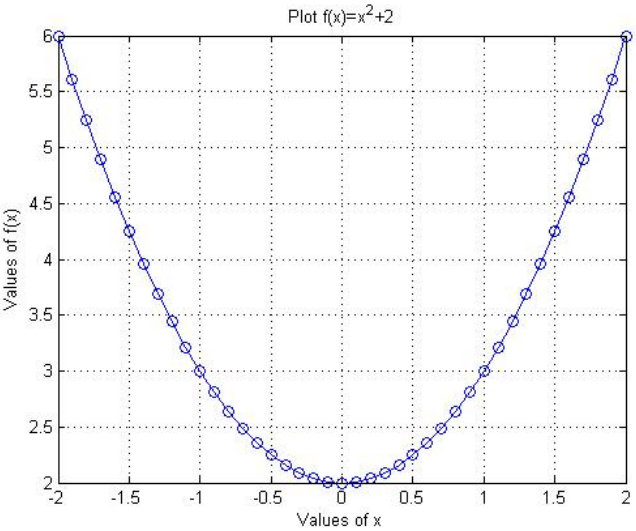
```
>> axis square
>> title('Plot f(x)=x^2+2')
>> xlabel('Values of x')
>> ylabel('Values of f(x)')
>> grid
```



Αν ξαναδώσουμε την εντολή plot για να δημιουργήσουμε ένα καινούργιο διάγραμμα τότε το MATLAB θα αντικαταστήσει το προηγούμενο με το καινούργιο.
Αν θέλουμε να σχεδιάσουμε με την εντολή plot πάνω στο ίδιο διάγραμμα τότε πρέπει να γράψουμε πρώτα την εντολή

```
>> hold on
```

Για παράδειγμα αν θέλουμε να σχεδιάσουμε το παρακάτω διάγραμμα το οποίο αποτελείτε από την προηγούμενη γραφική παράσταση και ταυτόχρονα έχουν σχεδιαστεί και τα σημεία (x,y) τότε πρέπει να γράψουμε μετά την εντολή grid του προηγούμενου παραδείγματος:




```
>> hold on
```

```
>> plot(x,y,'o')
>> hold off
```

Στον παρακάτω πίνακα φαίνονται οι προσδιορισμοί για το χρώμα, το σύμβολο και τον τύπο της γραμμής που μπορούμε να δώσουμε στην εντολή plot

Colors		Symbols		Line Types	
Blue	b	Point	.	Solid	-
Green	g	Circle	o	Dotted	:
Red	r	X-mark	x	Dashdot	-.
Cyan	c	Plus	+	Dashed	--
Magenta	m	Star	*		
Yellow	y	Square	s		
Black	k	Diamond	d		
White	w	Triangle(down)	v		
		Triangle(up)	^		
		Triangle(left)	<		
		Triangle(right)	>		
		Pentagram	p		
		Hexagram	h		

Μεταξύ των προσδιορισμών του παραπάνω πίνακα μπορούν να γίνουν και συνδυασμοί. Για παράδειγμα αν θέλαμε η γραμμή του διαγράμματος να ήταν κόκκινη, διακεκομμένη και ταυτόχρονα να υπάρχουν και τα σημεία σημειωμένα με το σύμβολο του τετραγώνου, τότε σε μια εντολή θα γράφαμε:

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

Αν έχουμε δημιουργήσει ένα διάγραμμα (Figure 1 όπως το ονομάζει το MATLAB) και θέλουμε να δημιουργήσουμε ένα δεύτερο τότε πριν από την εντολή plot πρέπει να γράψουμε την εντολή **`figure(2)`**

Άσκηση

Να λυθεί με τη βοήθεια του MATLAB η παρακάτω άσκηση:

Ένα σώμα αφήνεται τη χρονική στιγμή t_o=0, να πέσει ελεύθερα από ύψος H=45m από το έδαφος. Αν g=9.81 m/s^2 και η αντίσταση του αέρα θεωρηθεί αμελητέα, ζητούνται:

1. Να βρεθεί η τιμή της ταχύτητας και το ύψος από το έδαφος τη χρονική στιγμή t_1=2s.
2. Ποια χρονική στιγμή και με ποια ταχύτητα το σώμα φτάνει στο έδαφος;
3. Να γίνουν τα διαγράμματα σε συνάρτηση με το χρόνο:

   α) της μετατόπισης του σώματος

   β) της απόστασης του σώματος από το έδαφος (ύψος)

1. Αρχικά ορίζουμε τις μεταβλητές H, g και t1:

```
>> clear
>> H=45;
>> g=9.81;
>> t1=2;
```

Η ταχύτητα τη χρονική στιγμή t_1=2s βρίσκεται με την εντολή:

```
>> u1=g*t1

u1 =

   19.6200
```

και το ύψος από το έδαφος h1:

```
>> h1=H-(1/2)*g*t1^2

h1 =

   25.3800
```

2.

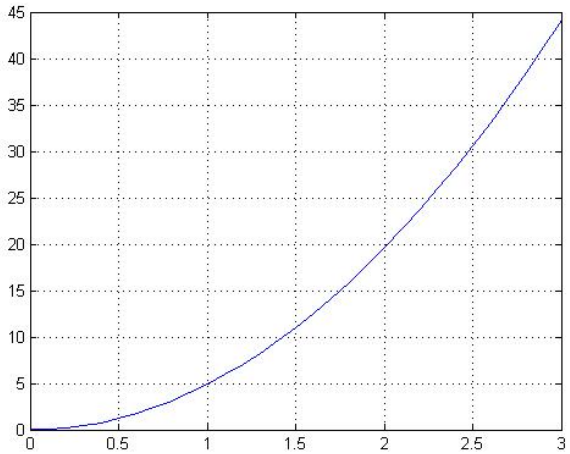
```
>> t2=sqrt(2*H/g)

t2 =

    3.0289
```

```
>> u2=g*t2

u2 =

   29.7136
```

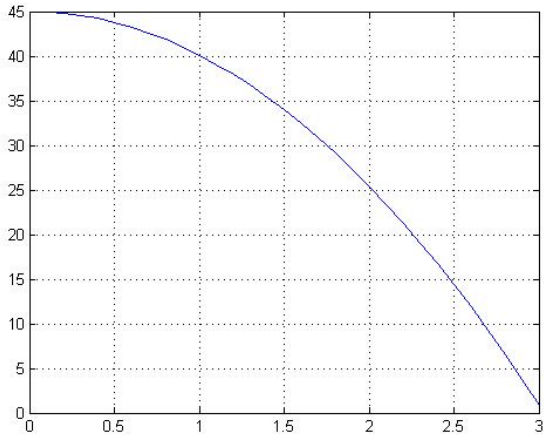
3.

Η μετατόπιση του σώματος δίνεται από τον τύπο $y=\frac{1}{2}gt^2$
και η απόσταση από το έδαφος δίνεται από τον τύπο $h = H - \frac{1}{2}gt^2$

```
>> t=0:0.1:3.0289;
>> y=(1/2)*g*t.^2;
>> plot(t,y)
>> grid
```




```
>> h=H-(1/2)*g*t.^2;
>> figure(2)
>> plot(t,h)
>> grid
```



Παρατηρούμε ότι, και στα 2 διαγράμματα, οι καμπύλες δεν περιέχουν τα σημεία (3.0289,45) για το πρώτο διάγραμμα και (3.0289,0) για το δεύτερο.
Εξηγήστε γιατί συμβαίνει αυτό και πως μπορεί να διορθωθεί;

3 Παράθυρο επεξεργασίας ή παράθυρο προγραμματισμού

Τα m-files παρέχουν έναν εναλλακτικό τρόπο εκτέλεσης εντολών που επεκτείνει σημαντικά τον τρόπο χειρισμού του MATLAB για την επίλυση προβλημάτων. Αντί ο χρήστης να δίνει μια-μια τις προς εκτέλεση εντολές μπορεί να τις ομαδοποιήσει σε ένα m-file για να τις τρέξει όλες μαζί. Η ονοματολογία (m-file) προέρχεται από το γεγονός ότι τα αρχεία αυτά αποθηκεύονται με την επέκταση .m
Τα m-files χωρίζονται σε δύο κατηγορίες: αρχεία script και τα αρχεία συναρτήσεων (functions).

3.1 Αρχεία script

Ένα αρχείο script είναι απλώς μια σειρά από εντολές MATLAB που είναι αποθηκευμένες σε ένα αρχείο. Οι εντολές αυτές μπορούν να εκτελεστούν πληκτρολογώντας το όνομα του αρχείου στο παράθυρο εντολών ή από τις επιλογές στο παράθυρο επεξεργασίας (Debug → Run)

Για παράδειγμα, για να φτιάξουμε ένα m-file με τις εντολές που πρέπει να εκτελεστούν για την επίλυση της προηγούμενης άσκησης κάνουμε τα εξής:

```
>> edit myscript.m
```

με την εντολή αυτή ανοίγει το παράθυρο επεξεργασίας στο οποίο τις εντολές:

```
% This is my first script
clear
H=45;
g=9.81;
t1=2;
u1=g*t1
h1=H-(1/2)*g*t1^2
t2=sqrt(2*H/g)
u2=g*t2
t=0:0.1:3.0289;
y=(1/2)*g*t.^2;
plot(t,y)
grid
h=H-(1/2)*g*t.^2;
figure(2)
plot(t,h)
grid
```

Οι γραμμές που ξεκινούν με το σύμβολο **%** ή ότι βρίσκεται αριστερά του συμβόλου αυτού ορίζετε ως σχόλιο (δεν λαμβάνεται ως εκτελέσιμη εντολή).
Αποθηκεύουμε πρώτα το αρχείο: File → Save και στη συνέχεια εκτελούμε το σύνολο των εντολών είτε από το παράθυρο επεξεργασίας (Debug → Run), είτε από τη γραμμή εντολών γράφοντας το όνομα του αρχείου:

```
>> myscript
```

3.2 Αρχεία συναρτήσεων (functions)

Τα αρχεία συναρτήσεων είναι τα m-files που ξεκινούν με τη λέξη function. Σε αντίθεση με τα αρχεία script, η εκτέλεση τους απαιτεί την είσοδο μεταβλητών (ορίσματα της συνάρτησης) και επιστρέφουν αποτελέσματα στις μεταβλητές εξόδου σε αντιστοιχία με τα functions σε γλώσσες προγραμματισμού όπως η Fortran ή C.

Γενικά ένα function έχει την εξής μορφή:

```
function [outvars] = funcname(inputvars)
  % helpcomments
  % helpcomments
  εντολες → τιμές
  outvars = τιμές
end
```

όπου

`outvars :`	τα ονόματα των μεταβλητών εξόδου χωρισμένα με κόμμα **,** ή κενό
`funcname:`	όνομα της συνάρτησης. **Πρέπει να είναι το ίδιο με το όνομα του αρχείου** Δηλαδή, το m-file πρέπει να αποθηκευτεί ως `funcname.m`
`input:`	τα **ορίσματα (arguments)** της συνάρτησης. Είναι τα ονόματα των μεταβλητών εισόδου χωρισμένα με κόμμα **,**
`helpcomments:`	κείμενο που παρέχει στο χρήστη πληροφορίες σχετικά με το function και τυπώνεται με την εντολή `help funcname`
`εντολές:`	εντολές του MATLAB που υπολογίζουν τις `τιμές` που θα εκχωρηθούν στις μεταβλητές outvars.

Πέρα από τον ρόλο του στην περιγραφή της συνάρτησης, η πρώτη γραμμή των `helpcomments`, που ονομάζεται γραμμή H1, είναι η γραμμή που ερευνάται από την εντολή `lookfor`. Έτσι η γραμμή αυτή πρέπει να περιλαμβάνει λέξεις κλειδιά που σχετίζονται με τις λειτουργίες της συνάρτησης.

Η συνάρτηση αυτή μπορεί στη συνέχεια να τρέξει, πληκτρολογώντας `funcname` στο παράθυρο εντολών, όπως φαίνεται στο ακόλουθο παράδειγμα. Ας υποθέσουμε ότι θέλουμε να μετατρέψουμε το παραπάνω αρχείο script (χωρίς τη δημιουργία γραφημάτων) σε αρχείο function.

```
>> edit freefall.m
```

```
function [u1, h1, t2, u2]=freefall(H,g,t1)
    % freefall Computes the free fall
    % INPUT: H,g,t1
    % OUTPUT: u1, h1, t2, u2
    u1=g*t1;
    h1=H-(1/2)*g*t1^2;
    t2=sqrt(2*H/g);
    u2=g*t2;
end
```

αποθηκεύουμε το αρχείο File → Save και στη συνέχεια στο παράθυρο εντολών γράφουμε:

```
>> clear
>> H=45;
>> g=9.81;
>> t1=2;
>> [a, b, c, d]=freefall(H,g,t1);
```

Οι μεταβλητές που χρησιμοποιεί μια συνάρτηση, εσωτερικά, δηλαδή οι μεταβλητές που δεν είναι στα ορίσματα, ονομάζονται τοπικές μεταβλητές. Οι τοπικές μεταβλητές δεν είναι προσπελάσιμες (accessible) από το παράθυρο εντολών.
Για παράδειγμα αν γράψουμε:

```
>> u1
```

θα εμφανιστεί το μήνυμα σφάλματος:

```
Undefined function or variable 'u1'.
```

Ένα αρχείο μπορεί να περιέχει περισσότερες τις μιας συναρτήσεις. Για παράδειγμα:

```
>> edit freefall2.m
```

```
function [u1, h1, t2, u2]= freefall2(H,g,t1)
    % freefall2 Computes the free fall using localfunc
    % INPUT: H,g,t1
    % OUTPUT: u1, h1, t2, u2
    u1=g*t1;
    h1=H-(1/2)*g*t1^2;
    [t2, u2]=localfunc(H,g);
end
```

```
function [t2, u2]=localfunc(H,g)
    t2=sqrt(2*H/g);
    u2=g*t2;
end
```

αποθηκεύουμε το αρχείο File → Save και στη συνέχεια στο παράθυρο εντολών γράφουμε:

```
>> [a2, b2, c2, d2]=freefall2(H,g,t1);
```

Η συνάρτηση **localfunc** ονομάζεται τοπική συνάρτηση ή υποσυνάρτηση (subfunction) και είναι προσπελάσιμη μόνο από τη κύρια συνάρτηση **freefall2**
Δηλαδή, αν γράψουμε:

```
>> [a3, b3]=localfunc(H,g);
```

θα εμφανιστεί το μήνυμα σφάλματος:

```
Undefined function 'localfunc' for input arguments of type
'double'.
```

3.3 Δομή if

Η σύνταξη της δομής **if** είναι η ακόλουθη:

if *συνθήκη*
  εντολές
end

Οι *εντολές* θα εκτελεστούν μόνο αν η *συνθήκη* είναι αληθής (true)

Τελεστές σύγκρισης (relational operators)

Τελεστής	Επεξήγηση
==	Ίσο
>	Μεγαλύτερο
<	Μικρότερο
>=	Μεγαλύτερο ή ίσο
<=	Μικρότερο ή ίσο
~=	Διάφορο

Παράδειγμα

Η παρακάτω συνάρτηση υπολογίζει και επιστρέφει τον αντίστροφο ενός αριθμού αν είναι διάφορος του μηδενός ενώ επιστρέφει `Error` αν είναι ίσος με το μηδέν.

```
function [y]=inverse(x)
    if x~=0
        y=1/x;
    end
    if x==0
        y='Error'
    end
end
```

```
>> inverse(5)

ans =

    0.2000

>> inverse(-2)

ans =

   -0.5000

>> inverse(0)

ans =

Error
```

Έστω τώρα ότι θέλουμε να χρησιμοποιήσουμε τη συνάρτηση `inverse` μέσα σε αριθμητικές παραστάσεις. Για παράδειγμα:

```
>> 1+inverse(2)

ans =

    1.5000
```

Αν όμως γράψουμε:

```
>> 1+inverse(0)

ans =
```

```
    70   115   115   112   115
```

Στην περίπτωση αυτή λέμε στο MATLAB να κάνει την πράξη `1 + 'Error'`. Για να γίνει η πράξη αυτή το MATLAB θα μετατρέψει τους χαρακτήρες `E, r, r, o, r` σε αριθμούς, για να δούμε την αντιστοιχία αυτή, γράφουμε:

```
>> double('Error')

ans =

    69   114   114   111   114
```

Για να αποφύγουμε το παραπάνω λογικό λάθος η συνάρτηση μπορεί να τροποποιηθεί ως εξής:

```
function [y]=inverse(x)
    if x~=0
        y=1/x;
    end
    if x==0
        y=inf;
        disp('Warning: Division by zero')
    end
end
```

```
>> 1+inverse(0)
Warnning: Division by zero

ans =

   Inf
```

Στην περίπτωση αυτή, στη μεταβλητή εξόδου `y` εκχωρούμε την προκαθορισμένη μεταβλητή inf και τυπώνουμε στη συνέχεια μια προειδοποίηση.

Αν θέλουμε να διακόψουμε την εκτέλεση της συνάρτησης `inverse` όταν `x=0` πρέπει να χρησιμοποιήσουμε την συνάρτηση **error**

```
function [y]=inverse(x)
    if x~=0
        y=1/x;
    end
    if x==0
       error('Error: Division by zero')
```

```
    end
end
```

```
>> 1+inverse(0)
Error using inverse (line 6)
Error: Division by zero
```

Στην περίπτωση αυτή, δεν εκχωρούμε τιμή στην μεταβλητή εξόδου `y` γιατί η εκτέλεση της συνάρτησης `inverse` θα τερματιστεί στο σημείο που έχουμε βάλει την συνάρτηση `error`

Λογικοί τελεστές (logical operators)

Τελεστής	Επεξήγηση
&&	AND
||	OR
~	NOT

Προτεραιότητα λογικών τελεστών

Τελεστής	Προτεραιότητα
~	Υψηλή
&&	Μεσαία
||	Χαμηλή

Μια *συνθήκη* μπορεί να περιέχει ένα ή περισσότερους αριθμητικούς, λογικούς τελεστές και τελεστές σύγκρισης. Η προτεραιότητα τους φαίνεται στους παρακάτω πίνακες:

Προτεραιότητα τελεστών

Τελεστής	Προτεραιότητα
Αριθμητικός (^ * / + -)	Υψηλή ↓ Χαμηλή
Λογικό NOT (~)	
Σύγκρισης (== > < <= >= ~=)	
Λογικός (&& ||)	

Μεταξύ τελεστών ίσης προτεραιότητας οι πράξεις γίνονται από αριστερά προς τα δεξιά και με τις παρενθέσεις αλλάζουμε την προτεραιότητα των πράξεων.

3.4 Παραλλαγές της δομής `if`

```
if συνθήκη
  εντολές
else
  εντολές
end
```

```
if συνθήκη
  εντολές
elseif συνθήκη
  εντολές
else
  εντολές
end
```

3.5 Επανάληψη `for`

Η σύνταξη της επανάληψης **for** είναι η ακόλουθη:

```
for var=start:step:stop
  εντολές
end
```

Παράδειγμα

```
function forloop()
    for i=5:2:14
        disp(i)
    end
end
```

```
>> forloop
     5

     7

     9
```

```
    11

    13
```

3.5 Επανάληψη υπό συνθήκη `while`

Η σύνταξη της επανάληψης υπό συνθήκη **`while`** είναι η ακόλουθη:

```
while συνθήκη
  εντολές
end
```

Οι *εντολές* θα εκτελούνται συνέχεια όσο η συνθήκη είναι αληθής (true).

Παράδειγμα

Έστω ότι θέλουμε να τυπώσουμε τους αριθμούς από το 1 μέχρι το 4. Αν γράψουμε την παρακάτω συνάρτηση:

```
function whileloop()
    x=0;
    while(x<5)
        x=x+1;
        disp(x)
    end
end
```

τότε θα πάρουμε ως αποτέλεσμα:

```
>> whileloop
     1

     2

     3

     4

     5
```

Παρατηρούμε ότι παρόλο που έχουμε `x<5,` στη συνθήκη, τυπώνεται και ο αριθμός 5 γιατί όταν η μεταβλητή `x` έχει την τιμή 4 (4<5) στη συνέχεια γίνεται η πράξη `x=x+1` και μετά τερματίζεται η επανάληψη (η συνθήκη ελέγχεται στο τέλος κάθε επανάληψης). Η σωστή συνθήκη για να τυπώσουμε τους αριθμούς από το 1 μέχρι το 4 είναι `x<4`

Η επανάληψη υπό συνθήκη δεν χρησιμοποιείται στην πράξη με την πιο πάνω μορφή γιατί δεν έχουμε απόλυτο έλεγχο της εξόδου από την επανάληψη, πρέπει δηλαδή να ολοκληρωθούν όλες οι εντολές που βρίσκονται μέσα στην επανάληψη και μετά να τερματιστεί. Για το λόγο αυτό χρησιμοποιείται η παρακάτω μορφή:

```
while true
  εντολές
 if συνθήκη
  break
 end
  εντολές
end
```

ή πιο απλά χρησιμοποιώντας το σύμβολο **,** (δυο ή περισσότερες εντολές μπορούν να γραφτούν στην ίδια γραμμή αρκεί να χωρίζονται με το σύμβολο **;** ή το **,)**

```
while true
  εντολές
 if συνθήκη, break,end
  εντολές
end
```

Παρατηρούμε ότι αντί για μια οποιοδήποτε συνθήκη βάζουμε την προκαθορισμένη μεταβλητή true. Αν δεν υπήρχε η δομή **if** *συνθήκη,* **break,end** τότε η παραπάνω επανάληψη δεν θα τερματιζόταν ποτέ (αέναη επανάληψη).

Η συνάρτηση που φτιάξαμε στο προηγούμενο παράδειγμα μπορεί να τροποποιηθεί ως εξής:

```
function whileloop()
    x=0;
    while true
        x=x+1;
        if x>4,break,end
        disp(x)
    end
end

>> whileloop
     1

     2

     3

     4
```

3.5 Ανώνυμες συναρτήσεις (Anonymous functions)

Οι συναρτήσεις αυτές μας επιτρέπουν τη δημιουργία συναρτήσεων χωρίς να φτιάξουμε m-files και ορίζονται απευθείας στη γραμμή εντολών. Για παράδειγμα αν θέλουμε να φτιάξουμε τη συνάρτηση $f(x)=x^2+5$ χρησιμοποιούμε την ακόλουθη σύνταξη:

```
>> f=@(x) x^2+5

f =

    @(x)x^2+5
```

όπου f το όνομα της συνάρτηση(function handler).
Μπορούμε να υπολογίσουμε την τιμή της συνάρτησης στο x=4 γράφοντας:

```
>> f(4)

ans =

    21
```

3.6 Συναρτήσεις ως ορίσματα συναρτήσεων

Μια ανώνυμη συνάρτηση μπορούμε να την περάσουμε ως όρισμα σε άλλες συναρτήσεις είτε του MATLAB είτε σε συναρτήσεις που έχουμε φτιάξει ως m-files. Για παράδειγμα:

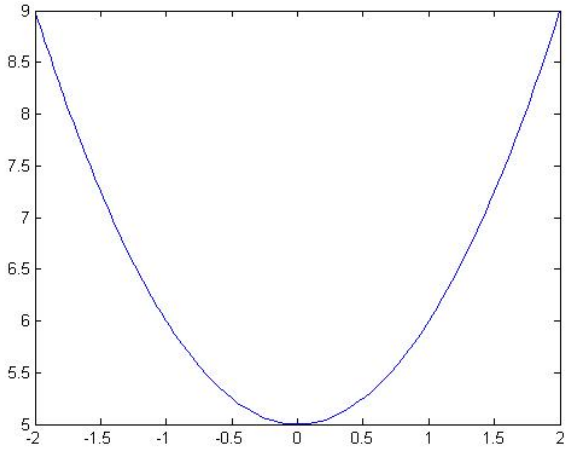
Η συνάρτηση **`fplot`** δημιουργεί διαγράμματα συναρτήσεων. Η σύνταξή της είναι:

```
fplot(func,limits)
```

όπου `func` το όνομα της μεταβλητής που έχουμε ορίσει ως function handler και
`limits` τα όρια της συνάρτησης δοσμένα μέσα σε brackets `[ ]`

Έτσι αν θέλουμε να φτιάξουμε το διάγραμμα της συνάρτησης με function handler f (βλ. προηγούμενο παράδειγμα) στο διάστημα [-2 2] γράφουμε:

```
>> fplot(f,[-2 2])
```



Άσκηση: Πως θα φτιάξετε το παραπάνω διάγραμμα χρησιμοποιώντας της συνάρτηση `plot`;

Αν θέλουμε να φτιάξουμε το διάγραμμά μια έτοιμης συνάρτησης του MATLAB, για παράδειγμα της συνάρτησης `sin` τότε γράφουμε:

```
>> fplot(@sin,[0 2*pi])
```