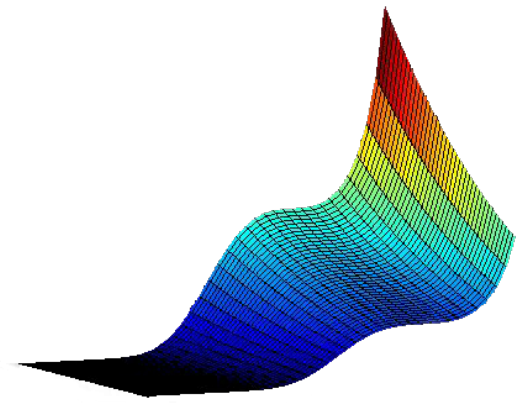
Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
Πρόγραμμα Μεταπτυχιακών Σπουδών
«Πληροφορική»



Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**ΓΡΑΦΙΚΑ ΜΑΘΗΜΑΤΙΚΩΝ ΜΕ ΤΟ MATLAB** **GRAPHICS FOR MATHEMATICS WITH MATLAB**
Ονοματεπώνυμο Φοιτητή	**ΚΥΡΙΑΚΟΠΟΥΛΟΣ ΑΡΓΥΡΙΟΣ**
Πατρώνυμο	**ΑΝΑΣΤΑΣΙΟΣ**
Αριθμός Μητρώου	**ΜΠΠΛ/11025**
Επιβλέπων	**ΠΙΚΡΑΚΗΣ ΑΓΓΕΛΟΣ**, ΕΠΙΚΟΥΡΟΣ ΚΑΘΗΓΗΤΗΣ

Ημερομηνία Παράδοσης : **ΝΟΕΜΒΡΙΟΣ 2016**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
ΠΙΚΡΑΚΗΣ ΑΓΓΕΛΟΣ ΕΠΙΚΟΥΡΟΣ ΚΑΘΗΓΗΤΗΣ	ΧΑΡΑΛΑΜΠΟΣ ΚΩΝΣΤΑΝΤΟΠΟΥΛΟΣ ΕΠΙΚΟΥΡΟΣ ΚΑΘΗΓΗΤΗΣ	ΜΙΧΑΗΛ ΨΑΡΑΚΗΣ ΕΠΙΚΟΥΡΟΣ ΚΑΘΗΓΗΤΗΣ

Ἀφιερώνεται
στοὺς Ἁγίους Ἀναργύρους,
Κοσμὰ καὶ Δαμιανό,
1 Νοεμβρίου.

ΠΕΡΙΛΗΨΗ

Στην παρούσα εργασία παρουσιάζονται οι βασικές εντολές του MatLab(R) με τις οποίες μπορούν να δημιουργηθούν γραφικά που σχετίζονται με τα Μαθηματικά, αλλά και προγραμματιστικές τεχνικές με τις οποίες ο χρήστης μπορεί να δημιουργήσει τις δικές του συναρτήσεις γραφικών καθώς και ολοκληρωμένα περιβάλλοντα διεπαφής (παραθυρικές εφαρμογές - GUI), μέσα από τα οποία η εισαγωγή δεδομένων και παραγωγή εξαγομένων συστηματοποιείται και διευκολύνεται σημαντικά.

Τα γραφικά βρίσκουν εφαρμογή σε πολλές περιοχές των Μαθηματικών, όπως δισδιάστατες και τρισδιάστατες γραφικές παραστάσεις συναρτήσεων, αναπαράσταση μιγαδικών αριθμών, γραφήματα πολικών συντεταγμένων, δημιουργία τομών επιφανειών, χαρτογράφηση ισοϋψών καμπυλών, αναπαράσταση βαθμίδων συναρτήσεων, στατιστικά διαγράμματα με ραβδογράμματα, ιστογράμματα και πίτες, αναπαράσταση μετρήσεων με περιοχή σφάλματος, κ.α.

Στα τρία τελευταία κεφάλαια της εργασίας (10,11,12) παρουσιάζεται η μελέτη μιας περίπτωσης δημιουργίας παραθυρικής εφαρμογής που δέχεται δεδομένα και ρυθμίσεις από τον χρήστη και παράγει τρισδιάστατα γραφήματα συναρτήσεων δύο μεταβλητών με δυνατότητα εισαγωγής της εξίσωσης της συνάρτησης από τον χρήστη, αποθήκευσης των τελικών εξαγομένων σε αρχεία εικόνας, καθώς και καθορισμό της περιοχής της συνάρτησης που αναπαρίσταται, αλλά και του τρόπου και της εμφάνισης αυτής της αναπαράστασης.

Η εργασία αυτή υλοποιήθηκε με την έκδοση 7.0.4 του MatLab(R), καθώς στοχεύει στην ανάπτυξη και παρουσία των βασικών εντολών και εργαλείων του, που αποτελούν υποσύνολο όλων των εκδόσεών του. Η μόνη πηγή γνώσης που αξιοποιήθηκε είναι η παραθυρική βοήθεια του προγράμματος (MatLab Help) και ο διαδικτυακός τόπος της εταιρείας MathWorks, όπου προσφέρονται απαντήσεις και συμβουλές (on-line help): *https://www.mathworks.com/help*

ABSTRACT

In this postgraduate work are demonstrated the basic functions of MatLab(R), with which the MatLab user can create graphics for Mathematics, as well as programming methods with which the user can create his or her own graphics functions and even more, Graphical User Interfaces (GUI), through which entering data and getting visual output is far more easy and organised.

Graphics are used in many mathematical areas, like 2d and 3d function plots, representation of complex numbers, polar axis graphs, surface cuts, isolines, gradiends, charts and pies for statistics, mesurments with error, and more.

The last three chapters of this work (10,11,12) present a case study of a graphical user interface application that gets input from the user and produces 3-dimentional graphs of functions on two independent variables of the form z=f(x,y). The user is allowed to enter the functions type, the region in the x,y axes to be ploted, and many other parameters like the colormap or the output resolution. The final plotting can be saved on the hard drive as an image file.

This assignement was realised in the enviroment of the 7.0.4 version of MatLab(R), as it’s pupose was to be contained in a basic subset of functions that are offered in all MatLab versions of the new millenium. The only information source used was the MatLab’s Help pages and the MathWorks on-line help web page, where plenty advice and answers are offered:

https://www.mathworks.com/help

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΙΣΑΓΩΓΗ

Η ανάπτυξη της τεχνολογίας των ηλεκτρονικών υπολογιστών μετά τον δεύτερο παγκόσμιο πόλεμο δημιούργησε νέες συνθήκες στον τρόπο εφαρμογής των θετικών επιστημών. Υπολογισμοί που στο παρελθόν γινότανε με επίπονες και χρονοβόρες μεθόδους, μπορούσαν πλέον να επιτευχθούν σε ελάχιστα δευτερόλεπτα. Τα μαθηματικά βοήθησαν στη σύλληψη και κατασκευή των ηλεκτρονικών υπολογιστών, από τις πρώιμες αριθμομηχανές των Babbage και Pascal, μέχρι τους ηλεκτρομηχανικούς και ηλεκτρονικούς υπολογιστές της δεκαετίας του 1940. Ακολούθως, όμως, και οι ηλεκτρονικοί υπολογιστές βοήθησαν τα Μαθηματικά, ώστε να εξερευνηθούν νέοι τομείς, να δοκιμαστούν θεωρίες και να αποδειχτούν θεωρήματα.

Το MatLab(R) είναι ένα από τα σύγχρονα προγράμματα που παρέχουν εργαλεία σε τεχνικούς επιστήμονες, ώστε να δημιουργούν τα θαυμαστά τεχνολογικά επιτεύγματα που εξελίσσουν τον κόσμο και τη ζωή των ανθρώπων. Ως παράδειγμα μπορούν να αναφερθούν τα συστήματα ασφαλείας των οχημάτων, οι διαστημικές πτήσεις, οι συσκευές που υποστηρίζουν το νοσηλευτικό έργο και τα έξυπνα φωτοβολταϊκά πάνελ.

Η χρήση του MatLab(R) αφορά την επεξεργασία σημάτων και εικόνων, τα ευφυή συστήματα, την αναγνώριση προτύπων, τις τηλεπικοινωνίες, τις ρομποτικές εφαρμογές, τα ηλεκτρονικά δίκτυα χρηματικών συναλλαγών και πολλά άλλα. Το γραφικό περιβάλλον εργασίας του MatLab(R) σε συνδυασμό με τη μητρωϊκή αναπαράσταση της γλώσσας προγραμματισμού του το καθιστούν την πιο άμεση και φυσιολογική έκφραση των υπολογιστικών μαθηματικών.

Στην παρούσα εργασία γίνεται περιγραφή των βασικών εντολών-συναρτήσεων του MatLab(R) με τις οποίες μπορούμε να δημιουργήσουμε γραφικές παραστάσεις και διαγράμματα που σχετίζονται με τη μαθηματική επιστήμη. Ασφαλώς τα περιθώρια που δίνονται για μια μεταπτυχιακή διατριβή, δεν επαρκούν για να καλυφθεί το σύνολο των εντολών του MatLab(R). Χαρακτηριστικό στοιχείο αποτελεί το γεγονός ότι οι κατά το μέγιστο συμπυκνωμένες σελίδες της Βοήθειας του MatLab(R) ξεπερνούν τις χίλιες σελίδες.

Το τελευταίο κομμάτι της εργασίας, δηλαδή τα κεφάλαια 10, 11 και 12, συναποτελούν μια μελέτη περίπτωσης (case study), δια της οποίας διερευνάται η δημιουργία παραθυρικού προγράμματος με το όνομα “3dPlotter”, το οποίο παρέχει στον χρήστη του τη δυνατότητα να εισαγάγει τον τύπο μιας συνάρτησης δύο μεταβλητών και να λάβει την δισδιάστατη απεικόνιση της τρισδιάστατης επιφάνειας που αντιστοιχεί σε αυτή.

Για την εφαρμογή “3dPlotter” αποφύγαμε τη χρήση των συναρτήσεων του MatLab(R), πέρα από το γενικότερο πλαίσιο, δηλαδή της δημιουργίας του γραφικού περιβάλλοντος διεπαφής, της δυνατότητας αποθήκευσης εικόνων κλπ, ώστε να διερευνήσουμε τον θεωρητικό συλλογισμό, βάση του οποίου μπορεί ένα σύνολο τρισδιάστατων συντεταγμένων να αναπαρασταθεί σε δύο διαστάσεις και με αξονομετρική διάταξη, όπως συμβαίνει με ένα οποιοδήποτε αξονομετρικό σχέδιο.

Ο λόγος που επιλέξαμε το θέμα της αυτής της εργασίας είναι κατά πρώτον η προσωπική επιθυμία της εκμάθησης του MatLab(R), αλλά επίσης και η ταχύτητα και η αποτελεσματικότητά του στις μαθηματικές πράξεις, οι οποίες αποτελούν τον πυρήνα της εφαρμογής “3dPlotter”.

Κλείνοντας την εισαγωγή αυτή, θα θέλαμε να ευχαριστήσουμε τον επίκουρο καθηγητή κ. Άγγελο Πικράκη, που δέχτηκε να αναλάβει την επίβλεψη της εργασίας αυτής, καθώς και για την υπομονή που επέδειξε όλο το περασμένο έτος, κατά το οποίο ασχοληθήκαμε με την συγγραφή της. Επίσης, θα θέλαμε να ευχαριστήσουμε το διοικητικό συμβούλιο του τμήματος για την παραχώρηση χρονικής παράτασης προς αποπεράτωση της μεταπτυχιακής μας διατριβής.

1. ΓΕΝΙΚΑ ΓΙΑ ΤΟ MatLab(R)

MatLab(R) σημαίνει: MATrix LABoratory. Η ονομασία αυτή προέρχεται από το ξεκίνημα του λογισμικού, αφού αρχικά είχε σχεδιαστεί για την πραγματοποίηση υπολογισμών με μητρώα.

Το MatLab(R) στα αρχεία τεκμηρίωσής του ορίζεται ως: μια γλώσσα υψηλών επιδόσεων για τεχνικούς υπολογισμούς.
Είναι λοιπόν μια γλώσσα προγραμματισμού κι όχι ένα πακέτο.

Η διεπαφή συστήματος-χρήστη γίνεται μέσω ενός εύχρηστου γραφικού περιβάλλοντος που περιλαμβάνει γραμμή εντολών και εκτέλεση προγραμμάτων με τη μορφή αρχείων εντολών.
Το γραφικό σύστημα διεπαφής συστήματος-χρήστη ενσωματώνει την δυνατότητα του προγραμματισμού, της πραγματοποίησης υπολογισμών και της παρουσίασης αναπαραστάσεων των αποτελεσμάτων.
Τα προβλήματα και οι λύσεις εκφράζονται με τη βοήθεια συμβολισμών της μαθηματικής γλώσσας.

Το MatLab(R) προσφέρεται για εργασία που σχετίζεται με:

- Μαθηματικά και υπολογιστικές λύσεις προβλημάτων
- Ανάπτυξη αλγορίθμων
- Συλλογή δεδομένων
- Δημιουργία μοντέλων, προσομοιώσεων και πρωτοτύπων
- Ανάλυση, επεξεργασία και παρουσίαση δεδομένων
- Γραφικές απεικονίσεις
- Δημιουργία εφαρμογών με γραφικό περιβάλλον διασύνδεσης χρήστη

Το MatLab(R) είναι ένα σύστημα που μπορεί να διαιρεθεί σε πέντε τομείς:

- Περιβάλλον ανάπτυξης
- Βιβλιοθήκη μαθηματικών συναρτήσεων
- Προγραμματιστική γλώσσα
- Περιβάλλον εξωτερικής διασύνδεσης του MatLab(R) (API)
- Γραφικά

Το Περιβάλλον ανάπτυξης περιλαμβάνει το παράθυρο εντολών, το παράθυρο εργασίας, μνήμη εντολών που έχουν προηγηθεί, επεξεργαστή κειμένου και αποσφαλματωτή, αλλά και δυνατότητα περιήγησης σε αρχεία τεκμηρίωσης και γενικότερα τους καταλόγους αρχείων του συστήματος.

Η βιβλιοθήκη μαθηματικών συναρτήσεων περιλαμβάνει συναρτήσεις αριθμητικών υπολογισμών, συναρτήσεις για πράξεις με μητρώα, συναρτήσεις Bessel και FFT κ.α.

Η προγραμματιστική γλώσσα του MatLab(R) είναι μια υψηλού επιπέδου γλώσσα που χρησιμοποιεί μητρώα, περιλαμβάνει εντολές ελέγχου ροής, συναρτήσεις, δομές δεδομένων και δυνατότητα αντικειμενοστραφούς προγραμματισμού.

Το API περιλαμβάνει συναρτήσεις που δίνουν την δυνατότητα διασύνδεσης κώδικα σε C ή FORTRAN με το MatLab(R), τη δυνατότητα χρήσης του MatLab(R) ως υπολογιστική μηχανή, καθώς και τη δυνατότητα διαχείρισης αρχείων του MatLab(R).

Τα γραφικά του MatLab(R) παρέχουν εξειδικευμένες δυνατότητες απεικόνισης διανυσμάτων και μητρώων, καθώς και δημιουργίας εκτυπώσεων με υπομνήματα. Προσφέρονται συναρτήσεις για δισδιάστατες και τρισδιάστατες αναπαραστάσεις δεδομένων, επεξεργασίας στατιστικής και

κινούμενης εικόνας, δημιουργίας παρουσιάσεων και, τέλος, δημιουργίας γραφικών περιβαλλόντων διασύνδεσης χρήστη-υπολογιστή (GUI ή αλλιώς: Graphical User Interface).

Τα γραφικά αντικείμενα που δημιουργεί και χρησιμοποιεί το MatLab(R) είναι τα εξής:

- Αντικείμενα διαγράμματος
- Αντικείμενα αξόνων
- Αντικείμενα καμπυλών
- Αντικείμενα επιφανειών
- Αντικείμενα κειμένου
- Αντικείμενα κλειστών γραμμών
- Αντικείμενα εικόνων
- Αντικείμενα GUI

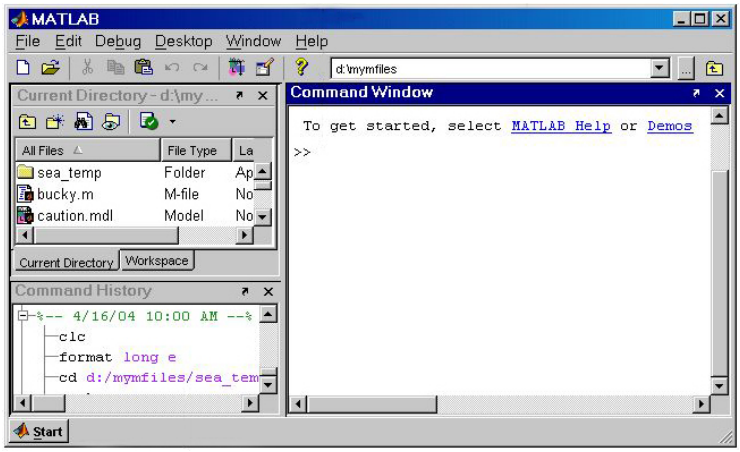
Στα επόμενα κεφάλαια θα ασχοληθούμε με τη δημιουργία και τον έλεγχο των γραφικών αντικειμένων του MatLab(R).

2. ΕΚΚΙΝΗΣΗ ΚΑΙ ΕΠΙΦΑΝΕΙΑ ΕΡΓΑΣΙΑΣ ΤΟΥ MatLab(R)

Το MatLab(R) μπορεί να ξεκινήσει με διπλό κλικ στο αντίστοιχο εικονίδιο ή από το μενού “Start” στο λειτουργικό σύστημα Windows. Σε περιβάλλον Unix μπορεί να εκτελεστεί από τη γραμμή εντολών δίνοντας: matlab.

Το πρώτο παράθυρο που αντιμετωπίζει ο χρήστης είναι η επιφάνεια εργασίας. Περιλαμβάνει εργαλεία διαχείρισης αρχείων, μεταβλητών και εφαρμογών σχετιζόμενων με το MatLab(R).

Στο παράθυρο εντολών (command window) μπορεί ο χρήστης να εισαγάγει εντολές του MatLab(R), ενώ στο παράθυρο τρέχοντος καταλόγου μπορεί να ορίσει τον κατάλογο στον οποίων αναφέρονται οι εντολές όταν αφορούν αρχεία. Υπάρχει επίσης παράθυρο με το ιστορικό των εντολών που έχουν εισαχθεί.
Στο πάνω μέρος βρίσκεται η γραμμή των μενού (File, Edit, Debug, Desktop, Window, Help).



Εικόνα 2.0.1. Η επιφάνεια εργασίας του MatLab(R).

Το MatLab(R) βασίζει τη χρηστική του ευκολία αλλά και αποτελεσματικότητα στη χρήση διανυσμάτων, τα οποία εισάγονται με τη μορφή πινάκων, είτε από τη γραμμή εντολών, είτε μέσω αρχείων κειμένου.

3. ΕΙΣΑΓΩΓΗ ΣΤΗ ΧΡΗΣΗ ΤΟΥ MatLab(R)

3.1 Η γραμμή εντολών

Ο ορισμός μιας μήτρας στο MatLab(R) γίνεται εύκολα από τη γραμμή εντολών ως εξής:

```
A = [ 16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1 ] (RETURN)
```

Η παραπάνω εντολή δημιουργεί έναν πίνακα τεσσάρων γραμμών και τεσσάρων στηλών, ο οποίος εμφανίζεται πριν τη νέα προτροπή της γραμμής εντολών ως εξής:

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

Αν η εντολή δημιουργίας του πίνακα είχε δοθεί με ένα ελληνικό ερωτηματικό στο τέλος ο πίνακας θα καταχωρούνταν στη μνήμη, αλλά η έξοδος θα αποκρυβόταν.

Μπορούμε να ορίσουμε την τιμή ενός στοιχείου του πίνακα με τη βοήθεια δεικτών ως εξής:

A(i,j) = <τιμή> ή A(k) = <τιμή>

π.χ. η εντολή: A(2,3) = 12 , θα μεταβάλει το στοιχείο της δεύτερης γραμμής και τρίτης στήλης από 11 σε 12, ενώ για να πραγματοποιήσουμε την ίδια αλλαγή δίνοντας μία παράμετρο μέσα στην παρένθεση πρέπει να γράψουμε: A(10) = 12 , αφού διατρέχοντας τις στήλες από αριστερά προς τα δεξιά και από πάνω προς τα κάτω το εν λόγω στοιχείο βρίσκεται δέκατο.

Αν εισάγουμε: A(1:3,2) , θα λάβουμε στην έξοδο τα στοιχεία 1 ως 3 της δεύτερης στήλης. Δηλαδή:

```
ans =

          3
          10
          6
```

ενώ δίνοντας A(:,2) , αναφερόμαστε σε όλα τα στοιχεία της δεύτερης στήλης.

Αν γράψουμε A(1:2:3,2) ,θα λάβουμε τα στοιχεία 1 ως 3 της δεύτερης στήλης με βήμα 2. Δηλαδή 1 και 3.

Μπορούμε να αντιγράψουμε έναν πίνακα σε μια άλλη θέση μνήμης δίνοντας μια εντολή όπως:
X=A
όπου X το όνομα του αντιγράφου.
Μπορούμε να δημιουργήσουμε έναν πίνακα γραμμή δίνοντας την εντολή:

```
X=-10:0.1:10;
```

Με την εντολή αυτή ο πίνακας X θα αποτελείται από μία γραμμή με τα στοιχεία από το -10 ως το 10 ανά 0.1. Δηλαδή, -10, -9.9, -9.8, ... -0.1, 0, 0.1, ... 9.8, 9.9, 10.

Μπορούμε να δημιουργήσουμε μια μήτρα πολύ εύκολα χρησιμοποιώντας ειδικές εντολές, όπως:

```
z=zeros(2,4);
o=ones(3,3);
```

Η zeros δημιουργεί έναν πίνακα με τις διαστάσεις της παρένθεσης και αρχικοποιείται με όλα τα στοιχεία της να είναι 0, ενώ η ones κάνει το ίδιο δίνοντας τους την τιμή 1.

Ένας άλλος τρόπος δημιουργίας μιας μήτρας είναι η χρήση της rand:

```
rand(2,3);
```

Η παραπάνω εντολή δημιουργεί έναν 2x3 πίνακα με τα στοιχεία του να λαμβάνουν έναν τυχαίο αριθμό μεταξύ 0 και 0.9999.
Μπορούμε να προσθέσουμε κάποιον αριθμό σε όλα τα στοιχεία ενός πίνακα ως εξής:

```
B=A-2;
```

Αν όμως θέλουμε να πολλαπλασιάσουμε πίνακες, με τον συμβολισμό:

```
A*B;
```

γίνεται ο πολλαπλασιασμός πινάκων, γραμμή επί στήλη, ενώ με τον συμβολισμό:

```
A.*B;
```

γίνεται πολλαπλασιασμός των αντίστοιχων στοιχείων.

Μπορούμε να δημιουργήσουμε έναν πίνακα με συνένωση άλλων πινάκων ως εξής:

```
B=[A A+14; A+48 A*2];
```

Για να σβήσουμε όλα τα στοιχεία της δεύτερης γραμμής του πίνακα Χ δίνουμε:

```
B(:,2)=[];
```

Αν σβήσουμε λιγότερα στοιχεία από μία γραμμή ή στήλη τα υπόλοιπα στοιχεία δίνουν ως αποτέλεσμα έναν πίνακα γραμμή.

3.2 Μεταβλητές

Οι μαθηματικές εκφράσεις που αναφέρονται σε μήτρες περιλαμβάνουν μεταβλητές, αριθμούς, τελεστές και συναρτήσεις.

Οι μεταβλητές δηλώνονται με ένα όνομα που αποτελείται από γράμματα, αριθμούς και κάτω παύλες, με την πρώτη θέση να καταλαμβάνεται από γράμμα και τους πρώτους 31 χαρακτήρες να είναι σημαντικοί για τη διάκριση των ονομάτων μεταξύ τους.
Η μεταβλητή εξισώνεται με μια τιμή, που ουσιαστικά είναι ένας πίνακας μιας μοναδικής τιμής.

3.3 Αριθμοί και τελεστές

Οι αριθμοί δίνονται στις εκφράσεις με μορφή ακεραίων (π.χ. 4, -77), δεκαδικών (π.χ. 0.001, -7.8202, 0.356873e-22), μιγαδικών(1i, -3.14159j, 2e5i).
Ένας μιγαδικός μπορεί να οριστεί ως εξής: z = 1 + 5i
Οι αριθμοί κινητής υποδιαστολής (floating point) ακολουθούν το πρότυπο της επιτροπής IEEE: τα σημαντικά ψηφία είναι 16 και το εύρος των τιμών από 10e-308 ως 10e+308.

Οι τελεστές που υποστηρίζονται για τις μαθηματικές εκφράσεις είναι οι κοινές πράξεις(+,-,*,/), ο τελεστής αριστερής διαίρεσης(\), η δύναμη(^), ο τελεστής του ανάστροφου και/ή συζυγούς μιγαδικού(') και οι παρενθέσεις που ορίζουν προτεραιότητα((,)).

3.4 Συναρτήσεις

Οι συναρτήσεις μπορούν να κληθούν μέσα από μαθηματικές εκφράσεις, από τη γραμμή εντολών ή μέσα από αρχεία.

Διακρίνονται σε τρεις μεγάλες κατηγορίες:

- Στοιχειώδεις μαθηματικές συναρτήσεις (elementary math functions): τριγωνομετρικές, εκθετικές, μιγαδικές, συναρτήσεις στρογγυλοποίησης και υπολοίπου
- Ειδικές μαθηματικές συναρτήσεις: ειδικές μαθηματικές συναρτήσεις, θεωρητικές αριθμητικές συναρτήσεις, συναρτήσεις μετασχηματισμού συντεταγμένων
- Στοιχειώδεις συναρτήσεις μητρών και πράξεων με μήτρες: στοιχειώδεις συναρτήσεις μητρών, συναρτήσεις περιγραφής πινάκων, συναρτήσεις χειρισμού μητρών, συναρτήσεις πολυδιάστατων πινάκων, χρηστικές συναρτήσεις πινάκων, συναρτήσεις ειδικών μεταβλητών και σταθερών, συναρτήσεις ειδικών μητρών

Τα ονόματα των συναρτήσεων δεν είναι δεσμευμένα.
Μπορούν να χρησιμοποιηθούν ως ονόματα μεταβλητών και στη συνέχεια να αποδεσμευτούν με τη χρήση της εντολής clear, για παράδειγμα:

```
cos=4;
x=cos+1;
clear cos;
cos(x);
```

3.5 Φόρτωση δεδομένων

Μπορούμε να φορτώσουμε δεδομένα από ένα αρχείο στη μνήμη του υπολογιστή χρησιμοποιώντας την εντολή load.

Τα δεδομένα συμπεριλαμβάνονται σε ένα αρχείο κειμένου με ονομασία το όνομα της μεταβλητής που θα τα δεχτεί και επέκταση dat, ενώ τα κενά διαστήματα ορίζουν τις στήλες μέσα σε κάθε γραμμή.

```
load m.dat
```

Η παραπάνω εντολή δημιουργεί έναν πίνακα m με στοιχεία τα περιεχόμενα του αρχείου m.dat.

Δύο άλλοι τρόποι να φορτώσουμε δεδομένα από αρχεία κειμένου και δυαδικά αρχεία είναι οι συναρτήσεις fread και fscanf.

Οι συναρτήσεις αυτές είναι καταλληλότερες όταν τα δεδομένα περιλαμβάνουν τίτλους πάνω από τις στήλες ή όταν οι γραμμές αποτελούνται από διαφορετικό αριθμό στηλών.

4. ΒΑΣΙΚΑ ΕΡΓΑΛΕΙΑ ΓΡΑΦΙΚΩΝ ΠΑΡΑΣΤΑΣΕΩΝ

4.1 Δισδιάστατες Γραφικές Παραστάσεις

4.1.1 Η εντολή plot();

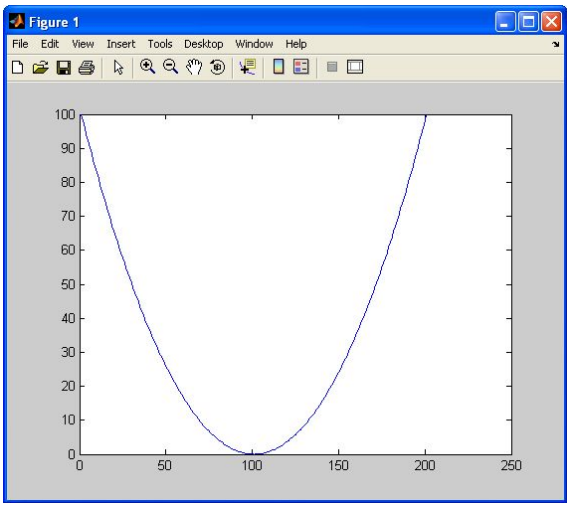
Η εντολή plot() δέχεται μέσα σε παρένθεση έναν αριθμό ορισμάτων, χωρισμένων με κόμμα, που αποτελούν διαστάσεις της γραφικής παράστασης.

Για παράδειγμα, αν χρησιμοποιήσουμε τον πίνακα X που δημιουργήσαμε στην παράγραφο 3, και δημιουργήσουμε έναν νέο πίνακα με τα τετράγωνα των τιμών του X, τον Y, μπορούμε στη συνέχεια να τον εισάγουμε στην εντολή plot και να λάβουμε το γράφημα των τιμών του.

```
X=-10:0.1:10;
Y=(X).^2;
plot(Y);
```

Αν δίναμε στη γραμμή εντολών Y=(X)^2; τότε το MatLab θα προσπαθούσε να υψώσει τον πίνακα X στο τετράγωνο, οπότε, αν δεν είναι τετραγωνικός, όπως συμβαίνει στην περίπτωσή μας, εμφανίζει μήνυμα λάθους.
Με τον τελεστή .^ γίνεται ύψωση των περιεχομένων στοιχείων κι όχι του πίνακα. Δηλαδή, ο Y αποτελείται από τα στοιχεία του X υψωμένα στο τετράγωνο.

Παρακάτω, φαίνεται το γράφημα που προκύπτει αυτόματα από το MatLab, μετά την εκτέλεση των τριών παραπάνω εντολών.

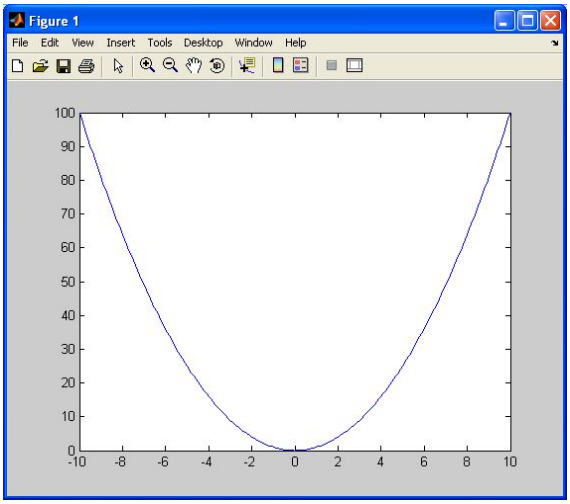


Εικόνα 4.1.1.1. Το γράφημα του Y=(X).^2; με plot(Y);

Αυτή είναι μια απεικόνιση των τιμών του Y. Αν θέλουμε να απεικονιστεί και ο πίνακας X, στον οριζόντιο άξονα, πρέπει να τον συμπεριλάβουμε στην εντολή plot().

```
plot(X,Y);
```

Η παραπάνω εντολή θα μας δείξει το ίδιο γράφημα με τις τιμές του πίνακα X στον οριζόντιο άξονα, όπως φαίνεται παρακάτω.



Εικόνα 4.1.1.2. Το γράφημα του Y=(X).^2; με plot(X,Y);

Η plot() δέχεται και πίνακες που δεν είναι της μορφής ενός πίνακα-γραμμή ή αλλιώς διανύσματος. Δηλαδή, και οι δύο του διαστάσεις είναι μεγαλύτερες από το ένα.

Στην περίπτωση αυτή η συνάρτηση plot() θα χρησιμοποιήσει τις στήλες του πίνακα ως τιμές y, σχηματίζοντας τόσα γραφήματα, όσες είναι και οι στήλες, και τον αριθμό γραμμής στην οποία βρίσκεται το κάθε στοιχείο ως τιμές του άξονα των x.

Στο παρακάτω παράδειγμα, δημιουργείται ένας πίνακας X με τρεις γραμμές που περιλαμβάνουν τον ίδιο αριθμό στοιχείων.
Σε διαφορετική περίπτωση το MatLab θα στείλει μήνυμα λάθους.
Στη συνέχεια, με τον τελεστή της αποστρόφου, ο πίνακας αντιστρέφεται, ώστε οι γραμμές να γίνουν στήλες.

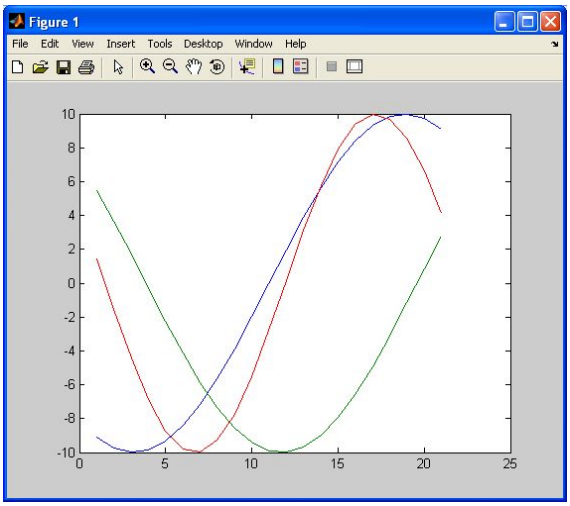
```
X=[-2:0.2:2; -10:0.2:-6; 3:0.3:9];
X=X';
Y=sin(X) * 10;
plot(Y);
```

Ο Y σχηματίζεται από τα ημίτονα των στοιχείων του X πολλαπλασιασμένα με την παράμετρο 10.

Τέλος, η plot() θα χρησιμοποιήσει κάθε μια από τις τρεις στήλες για να σχηματίσει και μία καμπύλη, δηλαδή τρεις καμπύλες στο ίδιο γράφημα.

Η τετμημένη που αντιστοιχεί σε κάθε τιμή λαμβάνεται ως αύξων αριθμός ξεκινώντας από το 1. Δηλαδή, το πρώτο στοιχείο κάθε στήλης του Y θα έχει τετμημένη 1, το δεύτερο τετμημένη 2, το τρίτο 3 κ.ο.κ.

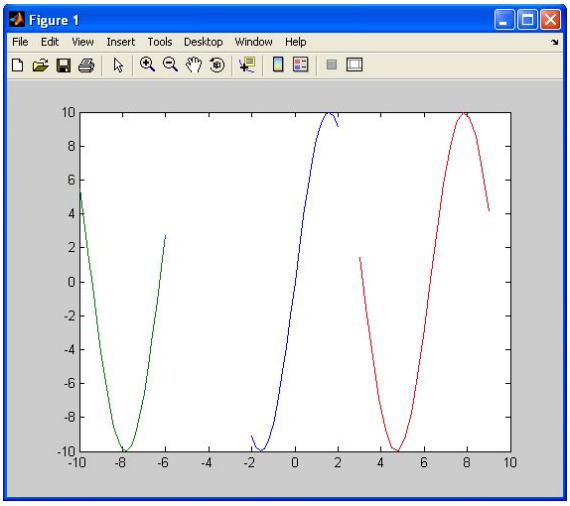
Το αποτέλεσμα φαίνεται στο παρακάτω σχήμα.



Εικόνα 4.1.1.3. Το γράφημα των στηλών του Y με plot(Y);

Αν αντί για plot(Y); δώσουμε plot(X,Y); τότε θα λάβουμε το επόμενο διάγραμμα, όπου τα τρία γραφήματα αντιστοιχούν στα στοιχεία του άξονα x, που λαμβάνονται από τον πίνακα της μεταβλητής X.
Η πρώτη τιμή από την πρώτη στήλη του πίνακα Y έχει ως τετμημένη το πρώτο στοιχείο της πρώτης στήλης του X κ.ο.κ.

Παρατηρούμε πως και οι δύο πίνακες μέσα στην plot() διαβάζονται κατά στήλη.



Εικόνα 4.1.1.4. Το γράφημα των στηλών του Y με plot(X,Y);

Η εντολή plot(X,Y); εξετάζει αυτόματα τα μεγέθη των στηλών και των γραμμών των δύο πινάκων.

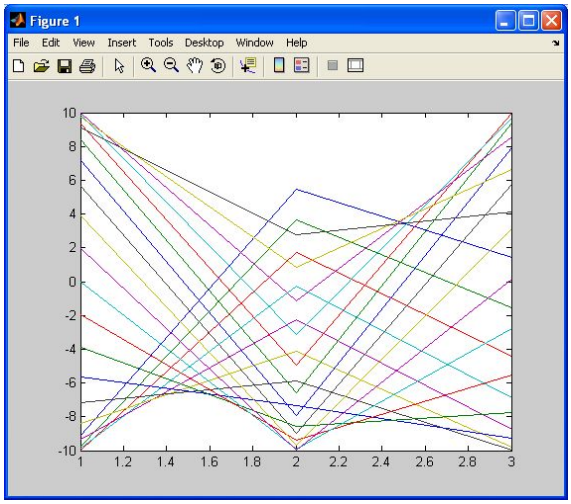
Αν ο πίνακας X είναι διάνυσμα (πίνακας με μία γραμμή) τότε οι στήλες του Y αναπαρίστανται ως προς τα στοιχεία του X, εκτός της περιπτώσεως που οι στήλες του Y είναι άνισες σε αριθμό στοιχείων με το διάνυσμα X, ενώ οι γραμμές περιέχουν τον ίδιο αριθμό στοιχείων με τον X.

Αν όμως ο πίνακας X έχει πολλές γραμμές και πολλές στήλες, όπως και ο Y, τότε η plot(X,Y); θα αναπαραστήσει τις στήλες του Y προς τις στήλες του X,εφόσον όμως οι πίνακες είναι των ίδιων διαστάσεων. Διαφορετικά, το MatLab θα εμφανίσει μήνυμα λάθους.

Μπορούμε να αντιστρέψουμε την αναπαράσταση των στηλών και να τις αντικαταστήσουμε με γραμμές δίνοντας τον ανάστροφο του Y στην plot():

```
plot(Y');
```

Το αποτέλεσμα φαίνεται στην εικόνα 4.1.1.5.



Εικόνα 4.1.1.5. Το γράφημα των γραμμών του Y με plot(Y');

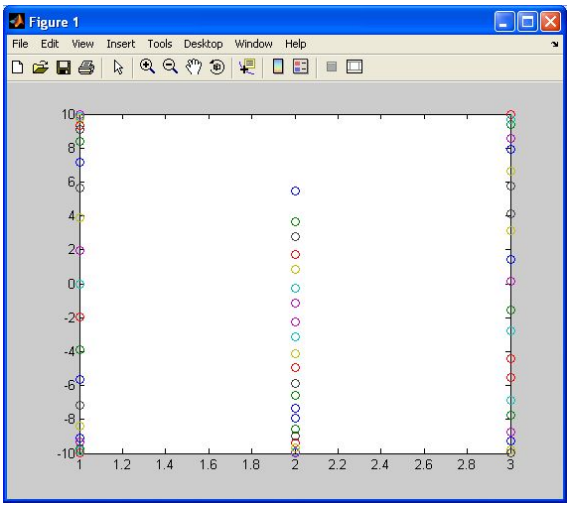
Αυτή η γραφική αναπαράσταση μας δείχνει την ημιτονοειδή τοποθέτηση των σημείων καθ' ύψος.

Παρατηρούμε ότι οι γραμμές που ενώνουν τα σημεία οριζοντίως συγχέουν το αποτέλεσμα. Μπορούμε να βελτιώσουμε την εικόνα παραλείποντας τις γραμμές αυτές χρησιμοποιώντας την ένδειξη 'o' μέσα στην plot().

```
plot(Y','o');
```

Αυτή η ένδειξη πληροφορεί το MatLab να αναπαρασταθούν τα σημεία με μικρά κυκλάκια.

Το αποτέλεσμα φαίνεται στην εικόνα 4.1.1.6.



Εικόνα 4.1.1.6. Το γράφημα των γραμμών του Y με plot(Y','o');

Υπάρχουν πολλά σύμβολα που βοηθούν στην επιθυμητή διαμόρφωση των γραφικών παραστάσεων. Μπορούν να ρυθμιστούν τα χρώματα, το σύμβολο αναπαράστασης των σημείων, ο τύπος των γραμμών.

Πιο αναλυτικά, τα χαρακτηριστικά γραμμής (ή LineSpecs) ανήκουν στις παρακάτω ομάδες:

Σύμβολα χρωμάτων:

'r','red'	Για το κίτρινο
'g','green'	Για το πράσινο
'b','blue'	Για το γαλάζιο
'c','cyan'	Για το κυανό
'm','magenta'	Για τη ματζέντα
'y','yellow'	Για το κίτρινο
'k','black'	Για το μαύρο
'w','white'	Για το λευκό

Σύμβολα χαρακτήρα γραμμής:

-	Για συνεχή γραμμή
--	Για διακεκομμένη γραμμή
:	Για γραμμή από τελείες
.-	Για γραμμή με εναλλαγή παύλας-τελείας

Σύμβολα αναπαράστασης σημείων:

'+'	Για το σύμβολο συν
'o'	Για κυκλάκι
'*'	Για αστεράκι
'.'	Για τελεία
'x'	Για το σύμβολο x
's','square'	Για τετραγωνάκι
'd','diamond'	Για ρόμβο
'^'	Για τρίγωνο με κατεύθυνση πάνω
'v'	Για τρίγωνο με κατεύθυνση κάτω
'>'	Για τρίγωνο με κατεύθυνση δεξιά
'<'	Για τρίγωνο με κατεύθυνση αριστερά
'p','pentagram'	Για πεντάκτινο αστέρι
'h','hexagram'	Για εξάκτινο αστέρι

Τα παραπάνω σύμβολα εισάγονται εντός των παρενθέσεων της εντολής plot() με οποιαδήποτε σειρά.
Για παράδειγμα, με την παρακάτω εντολή η γραμμή θα είναι κόκκινη, διακεκομμένη, με τα σημεία να αναπαρίστανται από αστερίσκους.

```
plot(X,Y,'--r*');
```

Αν ένας πίνακας περιλαμβάνει μιγαδικούς αριθμούς και τον θέσουμε στην, π.χ. plot(X), τότε θα παρασταθούν τα πραγματικά μέρη προς τα φανταστικά.
Αν όμως μαζί με τον πίνακα των μιγαδικών δοθεί και δεύτερος πίνακας, είτε πραγματικών, είτε μιγαδικών, π.χ. plot(X,Y), τα φανταστικά μέρη αγνοούνται και παριστάνονται μόνο τα πραγματικά.

Μπορούν να δοθούν πολλά ζεύγη Xn,Yn με ή χωρίς χαρακτηριστικά γραμμής. Π.χ. :

```
plot(x1,y1,x2,y2,'k-.',x3,y3);
```

Είναι δυνατόν, όπως τα χαρακτηριστικά γραμμής, να περιληφθεί μια ιδιότητα μέσα σε αποστρόφους, ακολουθούμενη από την τιμή της ιδιότητας αυτής.

plot(... , 'ιδιότητα', τιμή_ιδιότητας, ...);
Οι ιδιότητες του γραφήματος ανανεώνονται ως προς την τιμή τους με κάθε νέα κλήση της plot().
Για να καθορίσουμε μόνιμα την τιμή μιας ιδιότητας χρησιμοποιούμε την εντολή set, δίνοντας ως παραμέτρους τον χειριστή (handle) που επιστρέφεται από την plot(), τις ιδιότητες και τις αντίστοιχες τιμές. Π.χ. :

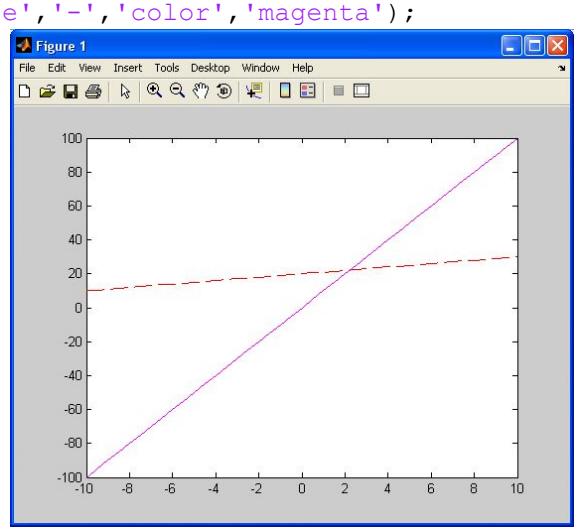
```
h=plot(X,Y);
set(h,'linestyle','--','color','green');
```

Αν δώσουμε: hold on, διατηρούμε ενεργό το ίδιο σχήμα και με μια νέα plot(), αυτή θα έχει τα

προκαθορισμένα χαρακτηριστικά.
Δίνοντας την εντολή: hold all, διατηρούμε όλα τα χαρακτηριστικά του γραφήματος και για την επόμενη γραφική παράσταση, η οποία τοποθετείται στο ίδιο σχήμα (figure).
Επίσης, με hold off, αποδεσμευόμαστε από αυτό.
Αν εισάγουμε μόνο: hold, τότε εναλλάσσεται η κατάσταση από on σε off ή αντίστροφα.

Αν η plot() περιλαμβάνει πολλές γραμμές, τότε επιστρέφει έναν πίνακα στήλη. Διατρέχουμε τους χειριστές των διαφόρων γραμμών με τη βοήθεια ενός δείκτη. Δηλαδή:

```
x=[-10:0.5:10];
h=plot(x,x+20,x,x*10);
set(h(1),'linestyle','--','color','red');
set(h(2),'linestyle','-','color','magenta');
```



Εικόνα 4.1.1.7. Η χρήση χειριστών και ιδιοτήτων με την εντολή set.

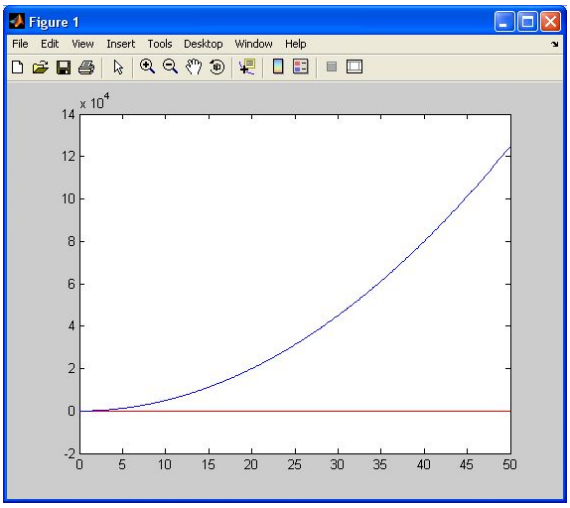
4.1.2 Η εντολή plotyy();

Όταν δύο καμπύλες εξετάζονται στο ίδιο πεδίο ορισμού, δηλαδή στο ίδιο διάστημα του άξονα των x, αλλά οι τιμές τους διαφέρουν κατά πολύ, τότε με την εντολή plot() δεν λαμβάνουμε σαφή εικόνα για τις καμπύλες αυτές, αφού η μία από αυτές, αναγκαστικά, σμικρύνεται πολύ.

Μπορούμε να λύσουμε αυτό το πρόβλημα χρησιμοποιώντας την εντολή plotyy().
Η εντολή αυτή δημιουργεί δύο κατακόρυφους άξονες, έναν στο αριστερό μέρος του διαγράμματος και έναν στο δεξί μέρος, με διαφορετική διαβάθμιση, ώστε οι δύο καμπύλες απεικονίζονται στο ίδιο διάγραμμα, χωρίς η μία να εξαφανίζει την άλλη.

Για παράδειγμα, με τις παρακάτω εντολές δημιουργείται το διάγραμμα 4.1.2.1:

```
x=0:0.1:50;
y1=sin(x);
y2=50*x.^2;

plot(x,y1,'r',x,y2,'b');
```

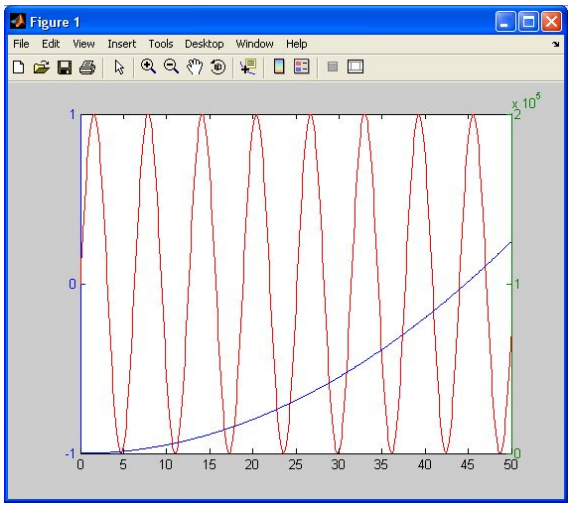


Εικόνα 4.1.2.1. Το διάγραμμα των y1=x.*sin(x) και y2=50*x.^2 με plot(x,y1,'r',x,y2,'b');

Παρατηρούμε στο παραπάνω διάγραμμα πως λόγω της μεγάλης περιοχής του άξονα των y που απεικονίζεται, αφού η συνάρτηση y2 (γαλάζιο χρώμα) φτάνει τιμές ως και 2500, η συνάρτηση y1 (κόκκινο χρώμα) δεν αναδεικνύεται.

Χρησιμοποιώντας την plotyy() όπως φαίνεται πιο κάτω, λαμβάνοντας τρεις πίνακες, τον a, για τα δύο αξονικά συστήματα a(1) και a(2), και τους h1 και h2 για τις καμπύλες (x,y1) και (x,y2), λαμβάνουμε το διάγραμμα που απεικονίζεται στην εικόνα 4.1.2.2.

```
x=0:0.1:50;
y1=sin(x);
y2=50*x.^2;
[a,h1,h2]=plotyy(x,y1,x,y2);
set(h1,'color','r')
set(h2,'color','b');
```



Εικόνα 4.1.2.2. Το διάγραμμα των y1=x.*sin(x) και y2=50*x.^2 με plotyy(x,y1,x,y2);

Στο δεύτερο διάγραμμα (6.1.2.2) βλέπουμε τον κατακόρυφο άξονα αριστερά να βαθμονομείται από το -60 ως το 60, ενώ εκείνον δεξιά, για τις τιμές της y2, από 0 ως 4000.

4.1.3 Η εντολή fplot();

Το MatLab(R) παρέχει την ευκολία στον χρήστη να αφίσει την επιλογή τιμών του πεδίου ορισμού μιας συνάρτησης σε μια αυτόματη διαδικασία μέσω της συνάρτησης fplot();

Η fplot(); συντάσσεται ως εξής:

```
fplot(fun,limits)
fplot(fun,limits,LineSpec)
fplot(fun,limits,tol)
fplot(fun,limits,tol,LineSpec)
fplot(fun,limits,n)
fplot(axes_handle,...)
[X,Y] = fplot(fun,limits,...)
```

όπου fun είναι ένα αλφαριθμητικό που περιλαμβάνει την προς παράσταση συνάρτηση και limits ένα διάνυσμα με τα όρια εντός των οποίων ο χρήστης επιθυμεί να περιοριστεί η γραφική παράσταση, ([xmin xmax]) ή ([xmin xmax ymin ymax]).
LineSpec είναι ένα αλφαριθμητικό χαρακτηριστικών γραμμής.
Η σύνταξη που δέχεται ως τρίτη παράμετρο το n επιτρέπει τον καθορισμό του ελάχιστου αριθμού σημείων για την παράσταση της συνάρτησης, που είναι n+1, ενώ ο μέγιστος αριθμός είναι: (1/n)*(xmax-xmin).
Ο χειρισμός με τον χειριστή αντικειμένου γίνεται κατά τα γνωστά.
Η σύνταξη με την απόδοση σε δύο μεταβλητές X και Y μεταφέρει στους πίνακες αυτούς τις τετμημένες και τις τεταγμένες της συνάρτησης που περιγράφεται από το αλφαριθμητικό fun μεταξύ των δεδομένων ορίων. Στη συνέχεια μπορούμε να εισάγουμε τους πίνακες X και Y στην εντολή plot();

Η μορφή που μπορεί να έχει το αλφαριθμητικό της παραμέτρου fun είναι:

- όνομα συνάρτησης αρχείου m-file
- αλφαριθμητικό συνάρτησης με ανεξάρτητη μεταβλητή x
- έναν χειριστή αντικειμένου συνάρτησης

Η παράμετρος tol στη σύνταξη fplot(fun,limits,tol); εισάγει μια σχετική ανοχή σφάλματος (η προκαθορισμένη τιμή είναι $2*10^{-3}$). Η τιμή tol αποτελεί το όριο της διαφοράς μεταξύ της τιμής της συνάρτησης και της τιμής που προκύπτει από τη γραμμική παρεμβολή μεταξύ των τιμών δύο διαδοχικών σημείων.

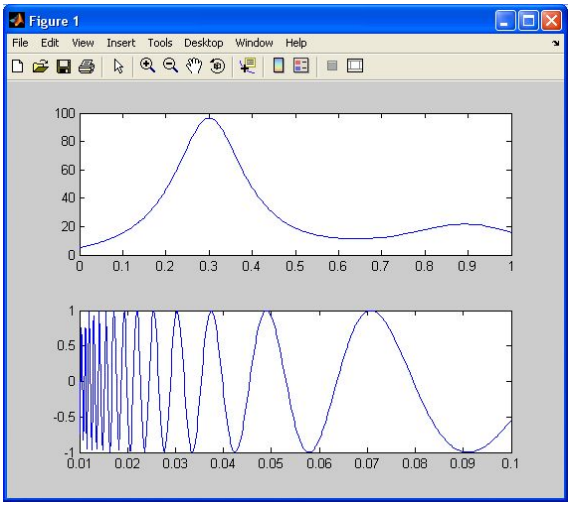
Ακολουθεί ένα παράδειγμα.

Η συνάρτηση humps είναι καταχωρημένη στο MatLab(R) για αυτόματη χρήση.

Με την σύνταξη: @(x) sin(1./x) αντικαθίσταται η έκφραση 'sin(1./x)', μόνο που, επιπλέον, ορίζεται η ανεξάρτητη μεταβλητή.

```
y=1./((x-.3).^2 +0.01)+1./((x-.9).^2+0.04)-6; % εναλλακτικά: y=@humps;
subplot(2,1,1);
fplot(y,[0 1])
sn = @(x) sin(1./x);
subplot(2,1,2);
```

```
fplot(sn,[.01 .1])
```



Εικόνα 4.1.3.1. Η συναρτήσεις humps (πάνω) και sin(1./x) (κάτω).

4.2 Τρισδιάστατες γραφικές παραστάσεις

4.2.1 Η εντολή plot3();

Η εντολή plot3() δέχεται τρεις μεταβλητές και ανάλογα με τις διαστάσεις των πινάκων δίνει διάφορα αποτελέσματα.

Αν οι τρεις μεταβλητές είναι διανύσματα ίδιου μεγέθους, τότε οι τιμές τους με ίδιο δείκτη σχηματίζουν τριάδες (τριπλέτες) που παριστάνονται στο τριαξονικό σύστημα.

Αν οι τρεις μεταβλητές περιέχουν μήτρες ίσων διαστάσεων, τότε παριστάνονται πολλές γραμμές, μία για κάθε τριάδα αντίστοιχων στηλών.

Αν κάποιες από τις μεταβλητές είναι διανύσματα, τότε αυτά θα χρησιμοποιηθούν πολλαπλώς, ώστε να παρασταθούν όλες οι στήλες ή οι γραμμές των μητρών, εφόσον το ή τα διανύσματα ταιριάζουν με αυτές, ή με τις στήλες εφόσον στήλες και γραμμές είναι ίσες.

Αν υπάρχει μεταβλητή με περιεχόμενο διαφορετικής διάστασης από τις άλλες μεταβλητές θα προκύψει μήνυμα λάθους.

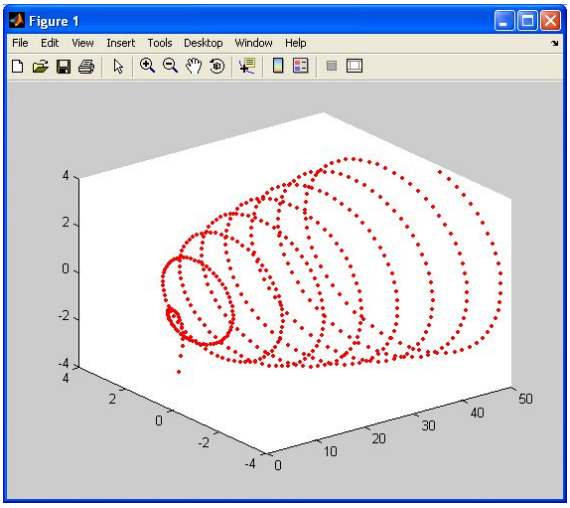
Η σύνταξη της είναι:

```
plot(x,y,z,s);
```
όπου x,y,z είναι πίνακες και s είναι αλφαριθμητικό ιδιοτήτων της γραμμής.

Επίσης, πολλές γραμμές στην ίδια γραφική παράσταση μπορούν να απεικονιστούν με διαδοχικές κλήσεις της plot3(), αφού έχει δοθεί hold on.
Ακόμη, μπορεί να δοθεί μια μοναδική εντολή εισάγοντας διαδοχικές τετράδες στην παρένθεση.
```
plot3(x1,y1,z1,s1,x2,y2,z2,s2,...);
```

Το παρακάτω παράδειγμα δημιουργεί την τρισδιάστατη γραφική παράσταση της εικόνας 4.2.1.1.

```
x=[0:0.1:50];
y=log(x).*sin(x);
z=y.*cot(x);
plot3(x,y,z,'.r');
```



Εικόνα 4.2.1.1. Η συνάρτηση z=y.*cot(x), με y=log(x).*sin(x).

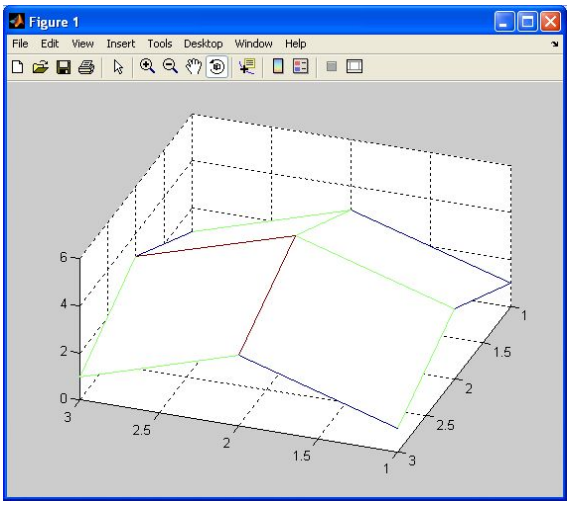
4.2.2 Η εντολή mesh();

Η εντολή mesh() δημιουργεί ένα πλέγμα, μία επιφάνεια με τη μορφή σχάρας η οποία αντιστοιχεί στα δεδομένα που εισάγονται με τρεις δισδιάστατες μήτρες X,Y,Z.
Κάθε σημείο της τρισδιάστατης γραφικής παράστασης έχει συντεταγμένες x_k,y_k,z_k που αντιστοιχούν στις τιμές των τριών πινάκων X,Y,Z για την γραμμή i και στήλη j που αντιστοιχούν στη θέση του σημείου.

Για παράδειγμα, έστω οι τρεις παρακάτω πίνακες:

```
X=[ 1 2 3;
    1 2 3;
    1 2 3 ];
Y=[ 1 1 1;
    2 2 2;
    3 3 3 ];

Z=[ 1 3 1;
    3 5 3;
    1 3 1 ];
```

Η εντολή mesh(X,Y,Z); θα δημιουργήσει μια πολύ απλή επιφάνεια, όπως φαίνεται στην επόμενη εικόνα.



Εικόνα 4.2.2.1. Μια απλή επιφάνεια από την εντολή mesh(X,Y,Z);

Όπως φαίνεται και στην εικόνα, το κεντρικό σημείο έχει συντεταγμένες (2,2,5), δηλαδή τις τιμές των τριών πινάκων (X,Y,Z) για i=2 και j=2: X_{22}, Y_{22}, Z_{22}.

Για να κατασκευάσουμε επιφάνειες πιο σύνθετης μορφής, χρειαζόμαστε πολύ περισσότερα σημεία και είναι δύσκολο να υπολογίσουμε τις τιμές τους μία προς μία ή να τις εισάγουμε από το πληκτρολόγιο.

Το πρόβλημα αυτό λύνεται με τη βοήθεια της εντολής [X,Y]=meshgrid(x,y).

Η εντολή meshgrid δέχεται δύο διανύσματα και επιστρέφει δύο δισδιάστατους πίνακες, επαναλαμβάνοντας την πρώτη παράμετρο-διάνυσμα σε όλες τις γραμμές και την δεύτερη παράμετρο-διάνυσμα, μετά από αναστροφή, σε όλες τις στήλες.

Για παράδειγμα, με την εντολή:

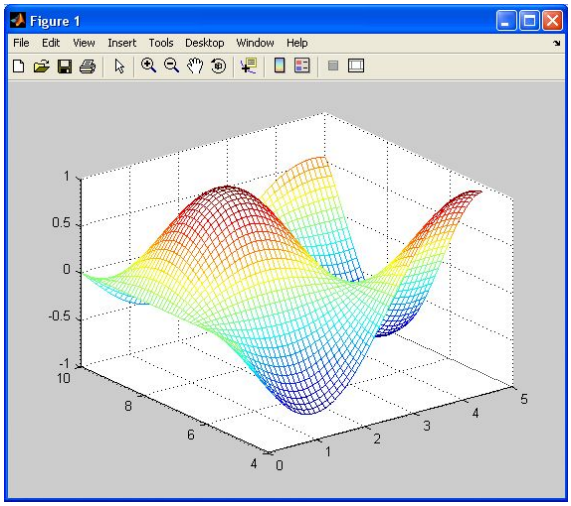
```
[x,y]=meshgrid([1:3],[1:3]);
```

θα δημιουργηθούν οι πίνακες:

```
x =

     1     2     3
     1     2     3
     1     2     3

y =

     1     1     1
     2     2     2
     3     3     3
```

Ως εφαρμογή, μπορούμε να δούμε τη δημιουργία του παρακάτω γραφήματος.

```
X=[0:0.1:5];
Y=[5:0.1:10];
[x,y]=meshgrid(X,Y);
z=sin(x).*sin(y);
mesh(x,y,z);
```



Εικόνα 4.2.2.2. Η τρισδιάστατη γραφική παράσταση της συνάρτησης z=sin(x).*sin(y);

Στο προηγούμενο παράδειγμα χρησιμοποιήσαμε την ημιτονοειδή συνάρτηση.
Είναι δύσκολο ωστόσο, χρησιμοποιώντας ακέραιες τιμές για τα άκρα και το βήμα των διανυσμάτων, να συμπέσουμε με πλήρεις κύκλους, δηλαδή με αριθμό ακτινίων πολλαπλάσιο του π.
Για τον λόγο αυτό, το MatLab διαθέτει την εντολή linspace. Η linspace δέχεται δύο ή τρεις παραμέτρους.

```
linspace(a,b);
linspace(a,b,n);
```

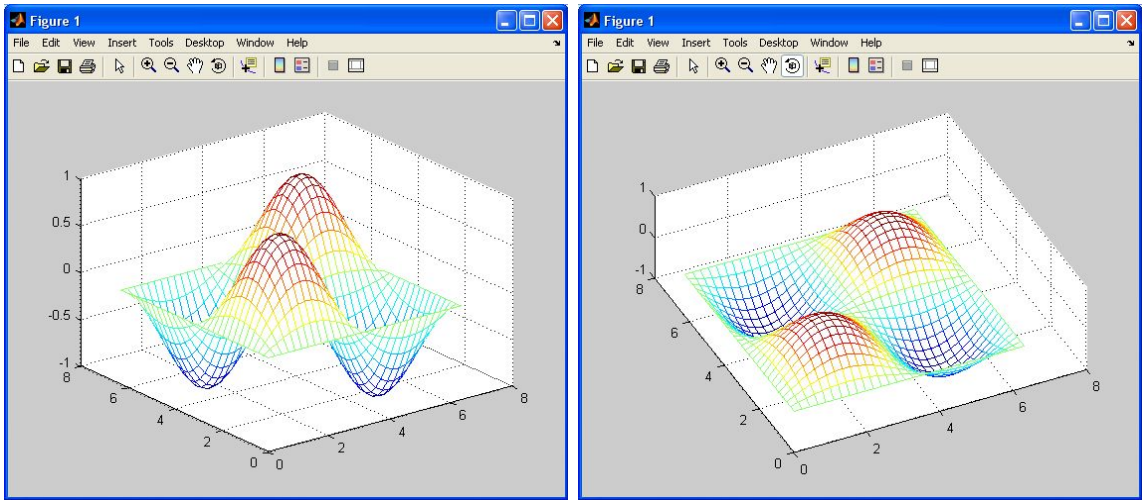
Οι παράμετροι a,b, αποτελούν τα άκρα του διαστήματος και ο n τον αριθμό των υποδιαιρέσεων. Αν ο n παραλειφθεί τότε χρησιμοποιείται η προκαθορισμένη τιμή 100.

Για παράδειγμα, η επόμενη εντολή δημιουργεί ένα διάνυσμα με πρώτη τιμή 0 και τελευταία 2π, και 28 ενδιάμεσες τιμές (συνολικά 30 στήλες μιας μοναδικής γραμμή), που ισαπέχουν η μία από την άλλη (γραμμική υποδιαίρεση σε ίσα διαστήματα):

```
linspace(0, 2*pi, 30);
```

Ας δούμε την προηγούμενη συνάρτηση στο διάστημα μιας περιόδου: 0 ως 2π.

```
[x,y]=meshgrid(linspace(0,2*pi,30),linspace(0,2*pi,30));
z=sin(x).*sin(y);
mesh(x,y,z);
```



Εικόνα 4.2.2.3. Η γραφική παράσταση της συνάρτησης z=sin(x).*sin(y); από 0 ως 2π.

Η εντολή mesh(x,y,z); δημιουργεί ένα πλέγμα που ανταποκρίνεται στη συνάρτηση z=f(x,y).

Το χρώμα του πλέγματος καθορίζεται από την μήτρα z.
Τα δύο ακραία χρώματα, ερυθρό και ιώδες, του φάσματος: ιώδες - κυανό - πράσινο - κίτρινο - πορτοκαλί - ερυθρό, αποδίδονται στη μέγιστη και την ελάχιστη τιμή του πίνακα z αντίστοιχα. Όλες οι άλλες τιμές λαμβάνουν ένα ενδιάμεσο χρώμα αναλόγως της τιμής τους, από την οποία εξαρτάται η θέση τους στο διάστημα $[z_{min},z_{max}]$.

Αν εισάγουμε απλώς mesh(z); ή mesh(z,c); οι άξονες x και y θα βαθμονομηθούν βάσει του αριθμού των τιμών που περιλαμβάνουν. Ειδικότερα, θα έχουμε x = 1:n και y = 1:m, όπου τα m και n είναι οι διαστάσεις της μήτρας z, δηλαδή: [m,n] = size(z).

Άλλες δυνατότητες της εντολής mesh(); είναι οι εξής:

```
mesh(...,'PropertyName',PropertyValue,...);
```

Με την πιο πάνω σύνταξη ρυθμίζονται οι διάφορες ιδιότητες που αφορούν το πλέγμα, ενώ με αυτή που ακολουθεί, επιλέγουμε σε ποιο αξονικό σύστημα θέλουμε να δημιουργήσουμε τη γραφική παράσταση, χρησιμοποιώντας την τιμή της μεταβλητής axes_handle.

```
axes_handles = mesh();
mesh(axes_handles,...);
```

Με την σύνταξη αυτή επιστρέφεται ένας χειριστής (handle) με τον οποίο μπορούμε να αναφερόμαστε στο συγκεκριμένο αξονικό σύστημα του διαγράμματος επιφάνειας.

4.2.3 Οι εντολές meshc(); και meshz();

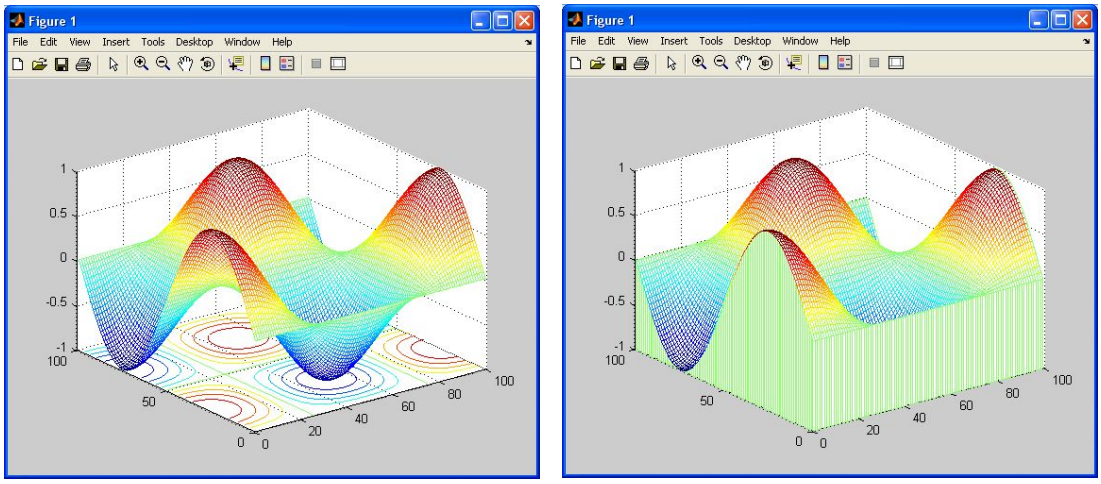
Με την εντολή meshc(); που δέχεται τις ίδιες παραμέτρους όπως παραπάνω, δημιουργείται στο επίπεδο xy ένα διάγραμμα προβολής των περιμέτρων των εγκάρσιων τομών του πλέγματος (το c σημαίνει contour), ενώ με την εντολή meshz(); τα σημεία του πλέγματος ενώνονται προς τις προβολές τους στο επίπεδο xy με κατακόρυφες γραμμές.

Μπορούμε να δούμε αυτά τα γραφήματα εισάγοντας τις παρακάτω γραμμές:

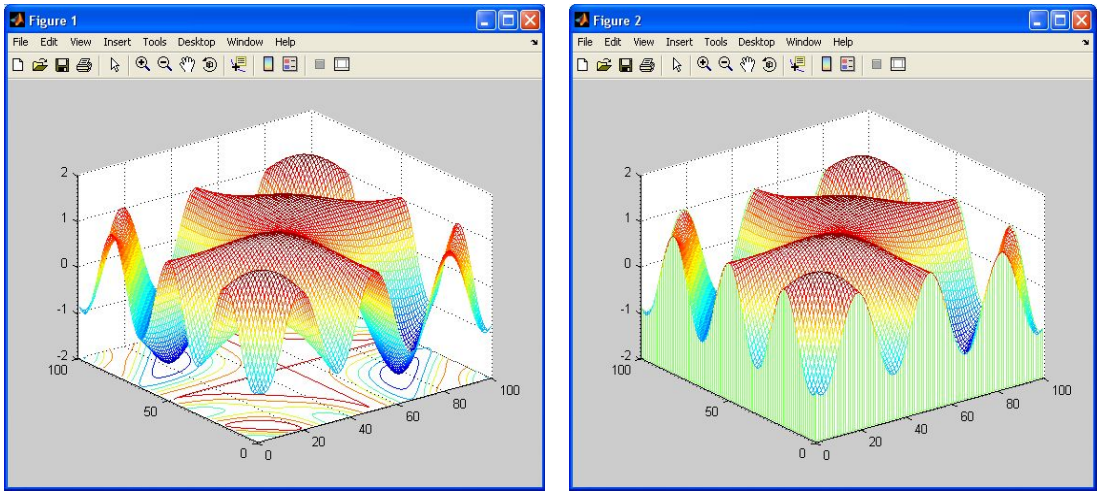
```
[x,y]=meshgrid(linspace(-pi,pi),linspace(-pi,pi));
z=sin(y).*cos(x);
meshc(z);
figure;
meshz(z);
```

Η εντολή figure; εκκινεί τη δημιουργία καινούριου σχήματος, ώστε να μη γράψουμε πάνω στο προηγούμενο.

Τα αποτελέσματα φαίνονται στις παρακάτω εικόνες.



Εικόνα 4.2.3.1. Η συνάρτηση z=sin(y).*cos(x); με meshc(); αριστερά και meshz(); δεξιά.



Εικόνα 4.2.3.2. Η συνάρτηση z=sin(y).*sin(x)+cos(x.*y); με meshc(); και meshz();

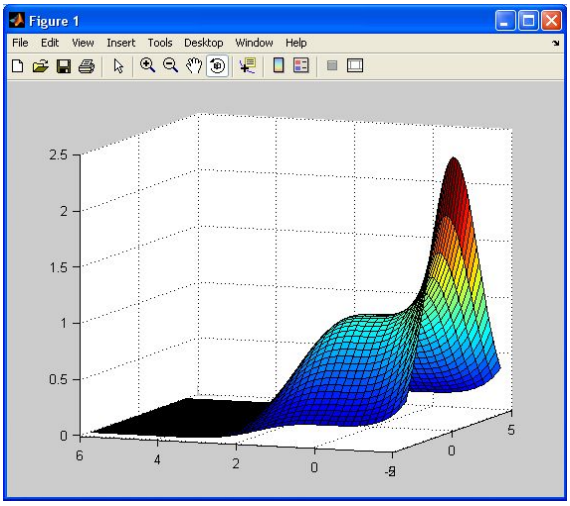
4.2.4 Διαγράμματα σκιαγραφημένων επιφανειών

Εκτός από από διαγράμματα που παριστάνονται με μορφή πλέγματος μπορούμε να δημιουργήσουμε και διαγράμματα επιφάνειας με την εντολή surf();
Τα διαγράμματα επιφάνειας είναι συναρτήσεις δύο ανεξάρτητων μεταβλητών: f(x,y).

Στην πραγματικότητα είναι δυνατόν να μεταβάλλουμε τις ιδιότητες του αντικειμένου και να μεταβούμε από τη μορφή πλέγματος στη μορφή επιφάνειας και αντίστροφα.

Με τις παρακάτω γραμμές δημιουργείται ένα διάγραμμα επιφάνειας.

```
[x,y]=meshgrid(-4:.2:4,-2:.2:6);
z=exp(-(x.^2+y.^3)/9);
surf(x,y,z);
```



Εικόνα 4.2.4.1. Η συνάρτηση z=exp(-(x.^2+y.^3)/9); με surf(x,y,z);

Η εντολή surf(); μπορεί, όπως και η mesh(); να συνταχτεί με διάφορους τρόπους:

```
surf(Z);
```

Η εντολή αυτή βαθμονομεί τους άξονες x και y σύμφωνα με τον αριθμό των γραμμών ή των στηλών του πίνακα Z. Δηλαδή, x = 1:n και y = 1:m, όπου [m,n] = size(Z).
Το χρώμα με το οποίο παριστάνονται οι τιμές της συνάρτησης αντιστοιχεί στο ύψος τους, με το χρωματικό φάσμα, ιώδες - κυανό - πράσινο - κίτρινο - πορτοκαλί - ερυθρό, να εκτείνεται από τις κατώτερες τιμές (ιώδες) μέχρι τις ανώτερες (ερυθρό).

Αν θέλουμε να σχεδιαστεί η επιφάνεια ως προς συγκεκριμένα x και y χρησιμοποιούμε την παρακάτω σύνταξη:

```
surf(X,Y,Z);
```

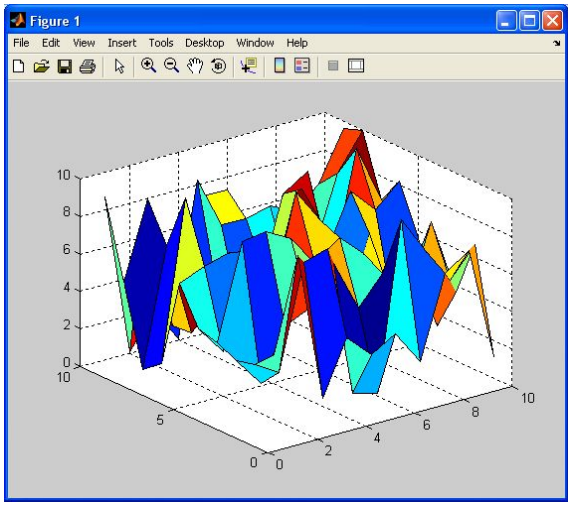
Οι πίνακες X και Y μπορεί να είναι ίδιων διαστάσεων με τον πίνακα Z ή ο X να είναι ένα διάνυσμα μεγέθους ίσου με τις γραμμές του Z και ο Y ένα διάνυσμα μεγέθους ίσου με τις στήλες του Z. Δεν έχει σημασία αν τα διανύσματα αυτά είναι γραμμές ή στήλες.

Για παράδειγμα, οι γραμμές:

```
X=(1:10);
Y=(1:10);
Z=rand(10,10)*10;
surf(X,Y,Z);
```

παράγουν το ίδιο διάγραμμα τυχαίων τιμών, όπως φαίνεται στην εικόνα 6.2.3.2, με τις γραμμές:

```
x=(1:10);
y=(1:10);
[X,Y]=meshgrid(x,y);
% Z=rand(10,10)*10; δεν επανεκτελούμε τη rand, για να μη μεταβληθούν οι τιμές.
surf(X,Y,Z);
```



Εικόνα 4.2.4.2. Οι τυχαίες τιμών Z=rand(10,10)*10; επί μιας περιοχής ακεραίων 10x10.

Αν θέλουμε να καθορίσουμε εμείς τον χρωματισμό του διαγράμματος πρέπει να εισάγουμε άλλον έναν πίνακα, δηλαδή:

```
surf(X,Y,Z,C);
```

Στην περίπτωση αυτή το MatLab πραγματοποιεί έναν γραμμικό μετασχηματισμό, ώστε να αντιστοιχίσει τις τιμές του πίνακα C στον τρέχοντα χρωματικό χάρτη (colormap).

Αν θέλουμε να δημιουργήσουμε έναν διάγραμμα επιλέγοντας ιδιότητες διαφορετικές από τις καθορισμένες, τότε, μετά τους πίνακες με τα δεδομένα, παραθέτουμε σε εισαγωγικά το όνομα της ιδιότητας κι αμέσως μετά την τιμή της, ως εξής:

```
surf(...,'PropertyName',PropertyValue);
```

Αποδίδοντας μια εντολή surf(); σε μια μεταβλητή, λαμβάνουμε έναν χειριστή (handle), τον οποίο μπορούμε να χρησιμοποιούμε στη συνέχεια προκειμένου να αναφερόμαστε στο αξονικό σύστημα που συνδέεται με τον συγκεκριμένο χειριστή. Για παράδειγμα:

```
h = surf(...);    surf(h,...);
```

4.2.5. Η εντολή surfc();

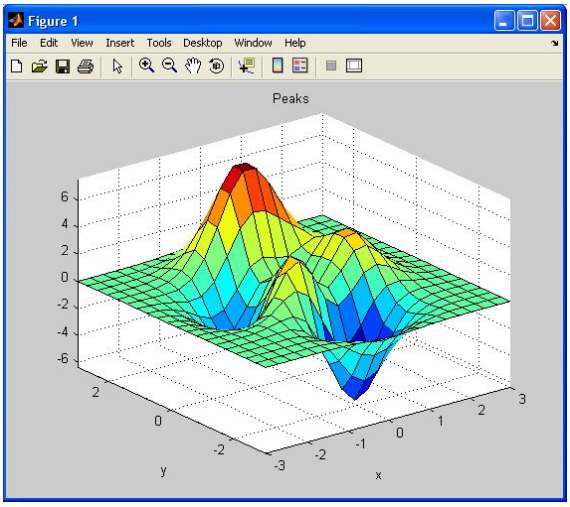
Η εντολή surfc(); δημιουργεί ένα διάγραμμα σκιαγραφημένης επιφάνειας, όπως και η surf(); με τη διαφορά, πως στο επίπεδο xy απεικονίζονται προβολικά ισοϋψείς τομές της τρισδιάστατης επιφάνειας.

Οι παράμετροι που εισάγονται στην εντολή surfc(); είναι ίδιες με αυτές που αναπτύχθηκαν στην προηγούμενη παράγραφο (6.2.4).
Τυπικά, οι μορφές που λαμβάνει η surfc(); έχουν ως εξής:

```
surfc(Z);
surfc(X,Y,Z);
surfc(X,Y,Z,C);
surfc(...,'PropertyName',PropertyValue);
h = surfc(...);
surfc(h,...);
```

Για να παρουσιάσουμε τα αποτελέσματα της εντολής surfc(); θα χρησιμοποιήσουμε τη συνάρτηση peaks(); η οποία παρέχεται από το MatLab για λόγους επίδειξης, για διάφορες περιπτώσεις όπου απαιτείται μια σκιαγραφημένη επιφάνεια.

Η εντολή peaks(20); δημιουργεί αυτόματα μια σκιαγραφημένη επιφάνεια, πάνω σε μια xy περιοχή 20x20, όπως απεικονίζεται παρακάτω.



Εικόνα 4.2.5.1. Η εντολή peaks(20);

Αν συνταχθεί η peaks(); ως εξής:

```
[x,y,z]=peaks(20)
```

το MatLab δεν δημιουργεί το γράφημα, αλλά καταχωρεί αυτόματα στους τρεις πίνακες, x, y, z, τις τιμές που τους αναλογούν.

Για παράδειγμα, μπορούμε, μετά την πιο πάνω εντολή, να δοκιμάσουμε το εξής:

```
length(x)
length(y)
```

Και τα δύο αποτελέσματα θα είναι 20.

Αν τυπώσουμε τα περιεχόμενα των πινάκων x και y, βλέπουμε πως οι τιμές τους εκτείνονται από -3 ως 3.

Τους πίνακες x, y, z, μπορούμε στη συνέχεια να τους εισάγουμε στην εντολή surfc();

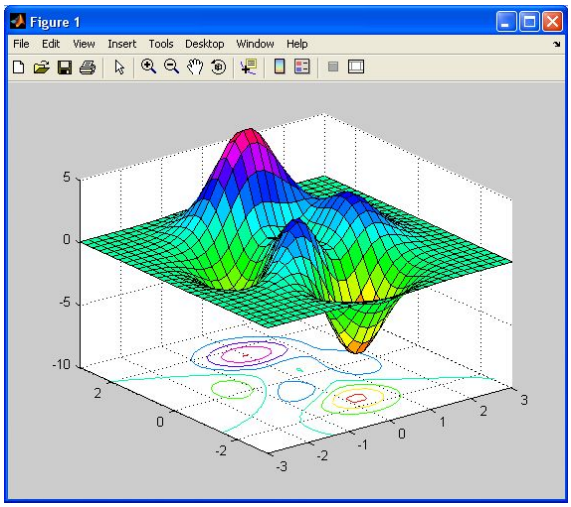
Η εξίσωση που χρησιμοποιεί η peaks(); είναι η εξής:

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

Είναι δυνατόν να παράξουμε το ίδιο αποτέλεσμα, χωρίς την peaks(); χρησιμοποιώντας την παραπάνω εξίσωση, αφού δημιουργήσουμε τους πίνακες x και y με την εντολή: meshgrid(linspace(-3,3,20));

Εισάγοντας τις παρακάτω γραμμές, λαμβάνουμε το γραφικό αποτέλεσμα που απεικονίζεται στη συνέχεια:

```
[X,Y,Z] = peaks(30);
surfc(X,Y,Z);
colormap hsv;
axis([-3 3 -3 3 -10 5]);
```



Εικόνα 4.2.5.2. Η συνάρτηση peaks(); με surfc();

Η εντολή colormap στο παραπάνω παράδειγμα ρυθμίζει το χρωματικό φάσμα που χρησιμοποιείται για το γράφημα.
Αναπτύσσεται στην επόμενη ενότητα (4.3) περί διαμορφώσεως των γραφημάτων.

5. ΒΑΣΙΚΕΣ ΕΝΤΟΛΕΣ ΔΙΑΜΟΡΦΩΣΗΣ ΓΡΑΦΗΜΑΤΩΝ

5.1 Η εντολή axis

Εκτός από τις καμπύλες του γραφήματος, μπορούμε να καθορίσουμε και την εμφάνιση και τον τύπο των αξόνων.

Η εντολή axis ακολουθείται από παρένθεση και μέσα σε αυτή ένα όρισμα κλεισμένο σε αποστρόφους. Π.χ. axis('on'), axis('off').

Μπορούμε να επιλέξουμε την αρχή των αξόνων κάτω αριστερά με θετικό τεταρτημόριο πάνω δεξιά, axis('xy'), ή πάνω αριστερά με θετικό τεταρτημόριο κάτω δεξιά, axis('ij').
Μπορούμε να επιλέξουμε τετραγωνική εμφάνιση για το γράφημα με axis('square').
Για να επανέλθουμε στις προκαθορισμένες ρυθμίσεις δίνουμε: axis('normal').
Για να περιορίσουμε τις διαστάσεις των αξόνων στο σύνολο των δεδομένων που παριστάνονται χρησιμοποιούμε: axis('tight').
Για να έχουν οι δύο άξονες μονάδες ίδιων διαστάσεων: axis('equal').
Για να περιορίσουμε το γράφημα εντός προεπιλεγμένων ορίων, δίνουμε:

```
axis([xmin xmax ymin ymax]);
```

Για να επαναφέρουμε τη γραφική παράσταση στην αρχική της μορφή: `axis('auto')`.

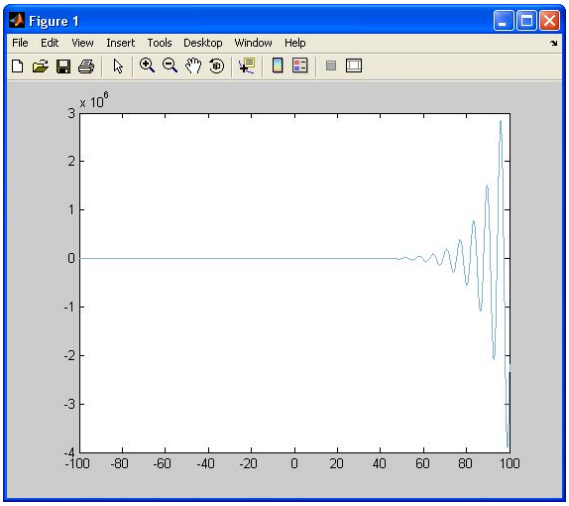
Αν θέλουμε να χρησιμοποιήσουμε την axis με περισσότερα ορίσματα, τα δίνουμε εκτός αποστρόφων και χωρίς παρενθέσεις. Π.χ. axis square equal.

Ας δούμε μια εφαρμογή.

Έστω, έχουμε τα εξής δεδομένα:

```
x=-100:.1:100;
y=log(x).^(x.^0.5).*sin(x);
plot(x,y);
```

Αυτές οι εντολές οδηγούν στο διάγραμμα του παρακάτω σχήματος.



Εικόνα 5.1.1. Η γραφική παράσταση της συνάρτησης y=log(x).^(x.^0.5).*sin(x);

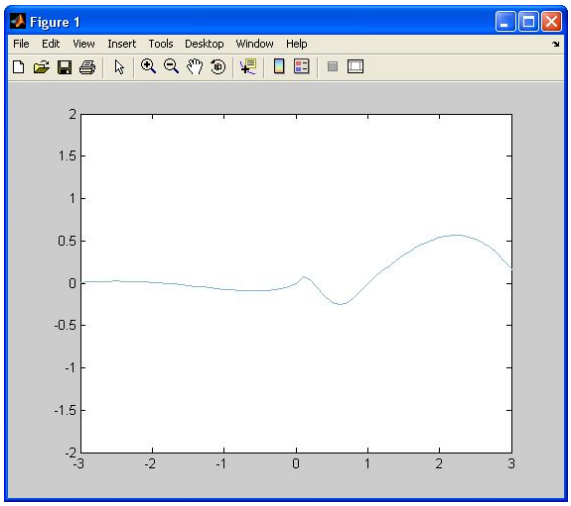
Το παραπάνω γράφημα μας πληροφορεί πως για τιμές του άξονα των x μικρότερες του 40 η καμπύλη είναι μια ευθεία γραμμή.
Αν όμως χρησιμοποιήσουμε την εντολή axis μπορούμε να προσαρμόσουμε τα διαστήματα των δύο αξόνων που εμφανίζονται στο σχήμα.
Δίνοντας την εντολή:

```
axis([-3 3 -2 2]);
```

το τελευταίο διάγραμμα, που είναι και το ενεργό, μεταβάλλεται και ουσιαστικά βλέπουμε μια μικρότερή του περιοχή, σε μεγέθυνση.

Στη νέα μορφή του γραφήματος διαπιστώνουμε πως η συνάρτηση δεν είναι ευθεία κοντά στην αρχή των αξόνων, αλλά παρουσιάζει μικρές διακυμάνσεις, που προς μεγαλύτερα x γίνονται ολοένα εντονότερες.

Το αποτέλεσμα της εντολής axis φαίνεται στο παρακάτω σχήμα.



Εικόνα 5.1.2. Η προσαρμογή του γραφήματος της συνάρτησης με την εντολή axis.

Τα όρια των αξόνων ενός δισδιάστατου διαγράμματος μπορούν να αποθηκευτούν σε ένα διάνυσμα μήκους 4, με ανάθεση της εντολής axis σε κάποια μεταβλητή. Π.χ. :

```
h=axis;
```

Τα όρια αυτά μπορούμε να τα χρησιμοποιήσουμε για να ορίσουμε εκ νέου την εμφανιζόμενη περιοχή των αξόνων, διατηρώντας για παράδειγμα τις ελάχιστες τιμές (h(1) και h(3) στο πιο πάνω παράδειγμα).

Αν μεταβάλουμε τα όρια των αξόνων, τότε οι διαδοχικές εντολές plot(), εφόσον χρησιμοποιούμε hold on, δεν μεταβάλουν τους άξονες. Έτσι, αν θέλουμε να προκαλέσουμε μόνοι μας τη σταθεροποίηση αυτή των αξόνων, μπορούμε να εισάγουμε: axis(axis) ή axis manual. Η axis μέσα στις παρενθέσεις θα προκαλέσει την επιστροφή ενός πίνακα με τις τρέχουσες διαστάσεις των αξόνων, που στη συνέχεια θα αποτελέσουν τις παραμέτρους της νέας κλήσης της axis.

Ακολουθεί μια πλήρης περιγραφή των παραμέτρων που δέχεται η εντολή axis:

```
axis([xmin xmax ymin ymax])
axis([xmin xmax ymin ymax zmin zmax cmin cmax])
```

Στην δεύτερη περίπτωση πιο πάνω, μετά τα ελάχιστα και μέγιστα των αξόνων x και y, ακολουθούν παράμετροι για τον άξονα z καθώς και τα όρια της χρωματικής διαβάθμισης, cmin και cmax.

```
v = axis
```

Απόδοση των ορίων στον πίνακα v, ο οποίος λαμβάνει είτε 4 είτε 6 τιμές. Οι τιμές cmin και cmax λαμβάνονται με την εντολή caxis, η οποία εξηγείται σε επόμενη παράγραφο (5.3).

```
axis auto
axis manual
axis tight
axis fill
```

Με auto γίνεται αυτόματη προσαρμογή των ορίων στο γράφημα, με manual τα όρια ορίζονται από τον χρήστη, με tight τα όρια περιορίζονται στο υπάρχον γράφημα και με fill οι άξονες γεμίζουν το παραλληλόγραμμο του γραφήματος, κατόπιν αυτόματης ρύθμισης των ορίων και της ιδιότητας PlotBoxAspectRatio(Αναλογία διαστάσεων παραλληλογράμμου γραφήματος).

```
sphere;
set(gca,'DataAspectRatio',[1 1 1],'PlotBoxAspectRatio', ...
                                          [1 1 1],'ZLim',[-0.6 0.6]);
axis ij
axis xy
```

Με axis ij η αρχή των αξόνων τοποθετείται στο πάνω-αριστερό σημείο με θετικές τιμές προς τα δεξιά και κάτω. Με axis xy η αρχή των αξόνων επανέρχεται κάτω-αριστερά με θετικά προς τα πάνω και δεξιά.

```
axis equal
axis image
axis square
```

Με axis equal οι μονάδες στους δύο άξονες γίνονται ίσες, με axis image ισχύει το ίδιο με τη διαφορά ότι το παραλληλόγραμμο του γραφήματος προσαρμόζεται γύρω από τις υπάρχουσες καμπύλες, ενώ με axis square το γράφημα, δισδιάστατο ή τρισδιάστατο, γίνεται τετράγωνο ή κυβικό αντίστοιχα, με αναπροσαρμογή της κλίμακας στους άξονες.

```
axis vis3d
axis normal
```

Με axis vis3d απενεργοποιείται τη δυνατότητα μεταβολής των ιδιοτήτων της αναλογίας των διαστάσεων της άποψης (aspect ratio properties), ώστε να καταστεί δυνατή η περιστροφή των τρισδιάστατων αντικειμένων, καταργώντας τον τρόπο απεικόνισης κατά τον οποίο το γράφημα εκτείνεται ώστε να γεμίσει όλο το παράθυρο (axis normal).

```
axis off
axis on
```

Με τις εντολές in και off ενεργοποιούνται και απενεργοποιούνται όλες οι γραμμές και ετικέτες των αξόνων.

```
axis(axes_handles,...)
```

Στο παρακάτω παράδειγμα φαίνεται η χρήση χειριστών αντικειμένου με τους οποίους μπορούμε να κατευθύνουμε την axis ώστε να επιδράσει σε συγκεκριμένους άξονες.

```
h1 = subplot(221);
h2 = subplot(222);
axis([h1 h2],'square');
```

Μπορούμε να αποσπάσουμε τρεις αλφαριθμητικές τιμές για τις ιδιότητες των αξόνων χρησιμοποιώντας την παρακάτω σύνταξη:

```
[mode,visibility,direction] = axis('state');
```

όπου η μεταβλητή mode λαμβάνει τις τιμές: “auto” ή “manual”, η visibility τις τιμές “on” ή “off”, και η direction τις τιμές: “xy” ή “ij”.

5.2 Η εντολή colormap

Η εντολή colormap χρησιμοποιείται για να οριστεί ο τρέχων χρωματικός χάρτης. Δηλαδή, το χρωματικό φάσμα με το οποίο απεικονίζονται οι τιμές μιας εξίσωσης σε ένα γράφημα. Το ένα άκρο του χρωματικού φάσματος προορίζεται για τις κατώτερες τιμές και το άλλο για τις υψηλότερες, ενώ οι ενδιάμεσες τιμές παριστάνονται με τα ενδιάμεσα χρώματα.

Ο χρωματικός χάρτης στην πραγματικότητα είναι ένας mx3 πίνακας, όπου οι τρεις στήλες αντιστοιχούν στα χρώματα κόκκινο-πράσινο-γαλάζιο του συστήματος RGB και λαμβάνουν τιμές από 0.0 ως 1.0, ενώ κάθε μια από τις m γραμμές αποτελεί μια τριπλέτα η οποία ορίζει ένα χρώμα.
Η πρώτη τριπλέτα ορίζει το χρώμα που θα έχουν οι κατώτατες τιμές στον άξονα z, ενώ η τελευταία τριπλέτα ορίζει το χρώμα των ανώτατων τιμών του άξονα z, προς τον θετικό ημιάξονα.

Ο θετικός ημιάξονας σύμφωνα με τις προκαθορισμένες ρυθμίσεις απεικονίζεται προς τα πάνω. Αυτές όμως οι ρυθμίσεις, όπως θα φανεί σε παράγραφο που ακολουθεί, μπορούν να τροποποιηθούν.

Δίνοντας: colormap(mp);

τίθεται ως χρωματικός χάρτης το περιεχόμενο του πίνακα με το όνομα mp.

Αν ο πίνακας mp περιλαμβάνει τιμές εκτός των αποδεκτών ορίων [0.0-1.0], τότε παράγεται μήνυμα λάθους : “Colormap must have values in [0,1]”.

Με colormap('default'); ο χρωματικός χάρτης λαμβάνει προκαθορισμένες τιμές.

Μπορούμε να αποδώσουμε το περιεχόμενο του χρωματικού χάρτη σε μια μεταβλητή :

```
cmap = colormap;
```

Αν ισχύουν οι προκαθορισμένες τιμές, τότε η cmap θα αποκτήσει το εξής περιεχόμενο:

0.0000	0.0000	0.5625
0.0000	0.0000	0.6250
0.0000	0.0000	0.6875
0.0000	0.0000	0.7500
0.0000	0.0000	0.8125
0.0000	0.0000	0.8750
0.0000	0.0000	0.9375
0.0000	0.0000	1.0000
0.0000	0.0625	1.0000
0.0000	0.1250	1.0000
0.0000	0.1875	1.0000
0.0000	0.2500	1.0000
0.0000	0.3125	1.0000
0.0000	0.3750	1.0000
0.0000	0.4375	1.0000
0.0000	0.5000	1.0000
0.0000	0.5625	1.0000
0.0000	0.6250	1.0000
0.0000	0.6875	1.0000
0.0000	0.7500	1.0000
0.0000	0.8125	1.0000
0.0000	0.8750	1.0000
0.0000	0.9375	1.0000
0.0000	1.0000	1.0000
0.0625	1.0000	0.9375
0.1250	1.0000	0.8750
0.1875	1.0000	0.8125
0.2500	1.0000	0.7500
0.3125	1.0000	0.6875
0.3750	1.0000	0.6250
0.4375	1.0000	0.5625
0.5000	1.0000	0.5000
0.5625	1.0000	0.4375
0.6250	1.0000	0.3750
0.6875	1.0000	0.3125
0.7500	1.0000	0.2500

0.8125	1.0000	0.1875
0.8750	1.0000	0.1250
0.9375	1.0000	0.0625
1.0000	1.0000	0.0000
1.0000	0.9375	0.0000
1.0000	0.8750	0.0000
1.0000	0.8125	0.0000
1.0000	0.7500	0.0000
1.0000	0.6875	0.0000
1.0000	0.6250	0.0000
1.0000	0.5625	0.0000
1.0000	0.5000	0.0000
1.0000	0.4375	0.0000
1.0000	0.3750	0.0000
1.0000	0.3125	0.0000
1.0000	0.2500	0.0000
1.0000	0.1875	0.0000
1.0000	0.1250	0.0000
1.0000	0.0625	0.0000
1.0000	0.0000	0.0000
0.9375	0.0000	0.0000
0.8750	0.0000	0.0000
0.8125	0.0000	0.0000
0.7500	0.0000	0.0000
0.6875	0.0000	0.0000
0.6250	0.0000	0.0000
0.5625	0.0000	0.0000
0.5000	0.0000	0.0000

Όπως παρατηρούμε οι τιμές ξεκινούν μηδενικές για τις δύο πρώτες στήλες (κόκκινο-πράσινο) και καταλήγουν μηδενικές για τις δύο τελευταίες (πράσινο-γαλάζιο). Έτσι σχηματίζεται το φάσμα από το γαλάζιο προς το ερυθρό.

Ο κατάλογος color του MatLab(R) περιλαμβάνει αρχεία τύπου M-file που περιέχουν χρωματικούς χάρτες.

Ακολουθεί ένα δειγματολόγιο με τις ονομασίες και τον τύπο των διαθέσιμων χρωματικών χαρτών.

Εικόνα 5.2.1. Τύποι και ονομασίες των χρωματικών χαρτών του MatLab(R).

Μπορούμε να καθορίσουμε τον χρωματικό χάρτη χρησιμοποιώντας αυτά τα αρχεία με το όνομά τους και το πλήθος των χρωμάτων ως παράμετρο. Για παράδειγμα:

```
colormap(hsv(128));
```

Η εντολή αυτή θα διαμορφώσει τον τρέχοντα χρωματικό χάρτη με τη διάταξη χρωμάτων HSV (βλ. εικόνα 5.2.1) και 128 διαβαθμίσεις.
Αν δεν δοθεί παράμετρος πλήθους διαβαθμίσεων, ισχύει ο τρέχων αριθμός.
Μπορεί να γίνει επιλογή χρωματικού χάρτη από το παράθυρο του διαγράμματος, από το μενού View>Property Editor ...

Ως παράδειγμα, μπορούμε να δούμε τις παρακάτω εντολές, οι οποίες δημιουργούν μια σφαίρα. Αρχικά, ισχύει ο προκαθορισμένος χρωματικός χάρτης, ενώ στη συνέχεια μεταβάλλεται σε τύπου “Summer”.

```
colormap ('default');
sphere;
axis equal;
```

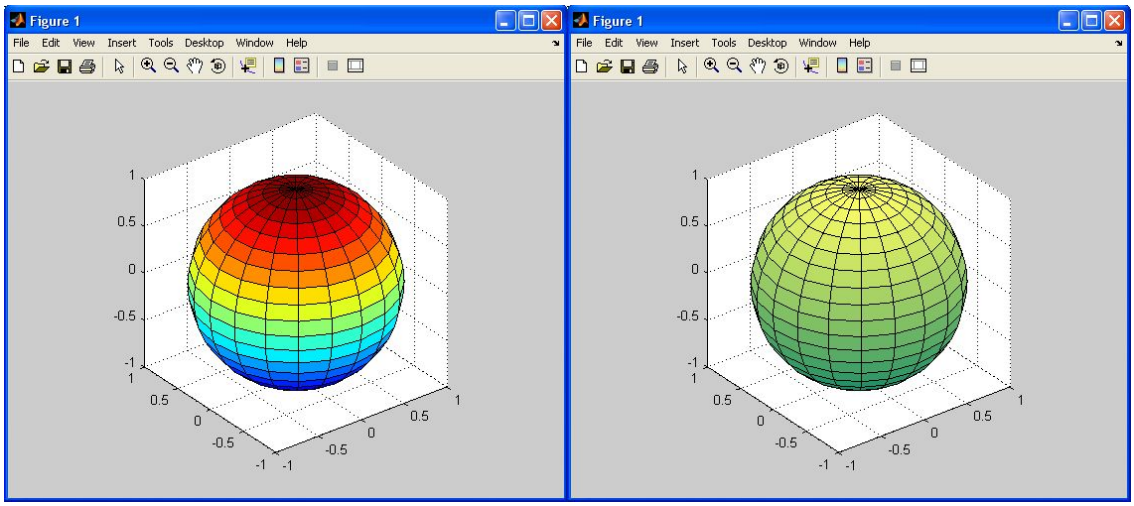
Αυτές οι εντολές θα δημιουργήσουν μια σφαίρα χρωματισμένη με τα χρώματα του προκαθορισμένου χρωματικού χάρτη.

Ενώ, δίνοντας:

```
colormap (summer);
```

Ο χρωματικός χάρτης θα μεταβληθεί σε τύπου “Summer”.

Τα παραπάνω φαίνονται στην εικόνα 5.2.2 που ακολουθεί.



Εικόνα 5.2.2. Μια σφαίρα με χρωματικό χάρτη default, αριστερά, και summer, δεξιά.

Η εντολή sphere(); αναπτύσσεται σε επόμενο τμήμα (6.9).

5.3 Η εντολή caxis

Η εντολή caxis αντιστοιχίζει τις τιμές των δεδομένων στον χρωματικό χάρτη.

Με την επόμενη εντολή ορίζεται ένα διάστημα τιμών, το οποίο αντιστοιχίζεται γραμμικά στον χρωματικό χάρτη.
Μικρότερες ή μεγαλύτερες τιμές αντιστοιχίζονται στα άκρα του διαστήματος.

```
caxis([cmin cmax]);
```

Αν θέλουμε το διάστημα να εκταθεί αυτομάτως σε όλο το εύρος των τιμών δίνουμε:

```
caxis auto;
```

Με την παράμετρο auto τιμές Inf και -Inf αντιστοιχίζονται στα άκρα του χρωματικού χάρτη, ενώ οι τιμές NaN δεν απεικονίζονται.

Με την παράμετρο manual παγιώνεται ο χρωματικός χάρτης και δεν μεταβάλλεται με κάθε επόμενη γραφική παράσταση στο ίδιο σύστημα αξόνων (hold on).

```
caxis manual;
```

Το ίδιο μπορούμε να το πετύχουμε και περνώντας τον τρέχοντα χρωματικό χάρτη, που επιστρέφει η caxis, ως παράμετρο στην ίδια την caxis.

```
caxis(caxis);
```

Για να αποδώσουμε σε μια μεταβλητή τα τρέχοντα όρια του χρωματικού χάρτη, δίνουμε:

```
v = caxis;
```

Προκειμένου να ελέγξουμε τον χρωματικό χάρτη ενός συγκεκριμένου συστήματος αξόνων χρησιμοποιούμε την παρακάτω σύνταξη με τον χειριστή-αριθμό των αξόνων του.

```
caxis(axes_handle,...);
```

Η εντολή caxis μεταβάλλει τις ιδιότητες CLim και CLimMode των γραφικών αντικειμένων.

Η εντολή caxis επηρεάζει αντικείμενα που έχουν δημιουργηθεί με μίας από τις εντολές:

- surface
- patch
- image

Οι εντολές αυτές εξετάζονται σε επόμενα κεφάλαια.

5.4 Η εντολή shading

Με την εντολή shading ορίζεται το είδος σκιαγράφησης της επιφάνειας ενός αντικειμένου που δημιουργήθηκε με τις εντολές surface και patch.

Υπάρχουν τρεις μορφές εισαγωγής της εντολής shading:

```
shading flat;
```

Η παράμετρος flat ορίζει μονοχρωματική απεικόνιση για τα τμήματα του πλέγματος mesh, βάση του ενός άκρου τους, και επίσης μονοχρωματική απεικόνιση για τα τετράγωνα που δημιουργεί το πλέγμα, βάση ενός γωνιακού σημείου τους, αυτού με τον μικρότερο δείκτη.

Τα σημεία του πλέγματος λαμβάνουν το χρώμα τους σύμφωνα με έναν χρωματικό δείκτη που υπολογίζεται γραμμικά με τη βοήθεια του ακόλουθου τύπου:

```
index = fix((C-cmin)/(cmax-cmin)*m)+1
```

όπου C είναι μια τιμή του πίνακα που παριστάνεται ως επιφάνεια, cmin και cmax τα όρια του χρωματικού φάσματος, και m το μέγεθος του χρωματικού φάσματος.

```
shading faceted;
```

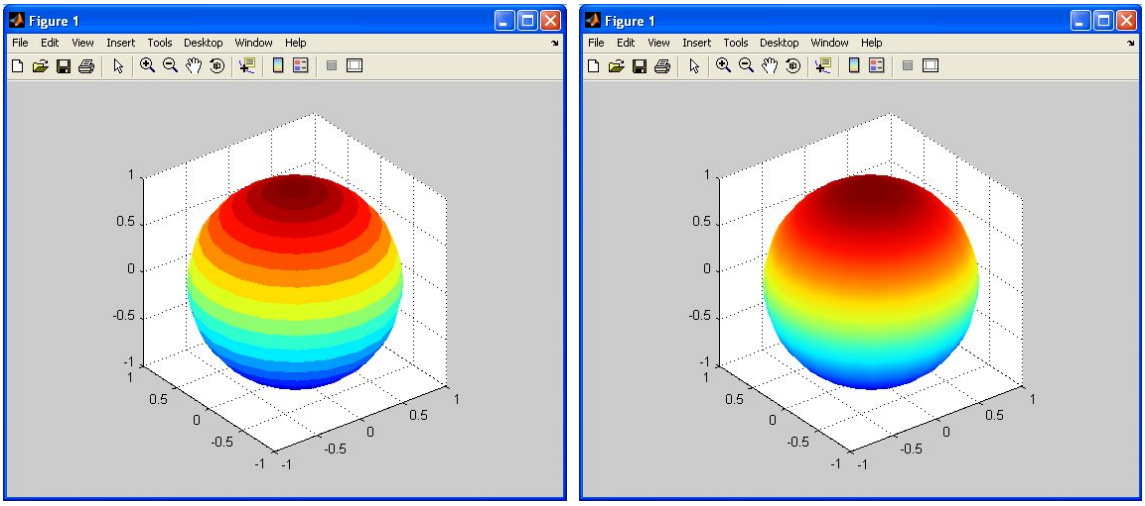
Αυτός είναι ο προκαθορισμένος τύπος (default) της εντολής shading. Είναι όπως με την παράμετρο flat, μόνο που τα τετράγωνα καλύπτονται από ένα πλέγμα μαύρου χρώματος.

shading interp;

Με την παράμετρο interp οι πλευρές του πλέγματος γύρω από κάθε τετράγωνο, καθώς και το ίδιο το τετράγωνο, χρωματίζονται μέχρι τη μέση με ένα χρώμα και κατά το υπόλοιπο με άλλο χρώμα.

Στην εικόνα 5.2.2 είδαμε δύο σφαίρες με shading faceted; και χρωματικό χάρτη default, αριστερά, και summer, δεξιά.

Στην εικόνα που ακολουθεί απεικονίζονται οι μορφές σκιαγράφησης επιφάνειας shading flat και shading interp με χρωματικό χάρτη default.



Εικόνα 5.4.1. Μια σφαίρα shading flat, αριστερά, και μια με shading interp, δεξιά.

5.5 Τοποθέτηση ετικετών και τίτλων

Στα γραφήματα μπορούμε να προσθέσουμε τίτλους, τις ονομασίες των αξόνων, σχόλια, όπως και να επισημάνουμε επιλεγμένα σημεία, κατά τον τρόπο που αυτό γίνεται σε χειρόγραφες γραφικές παραστάσεις.

Για να το πετύχουμε αυτό χρησιμοποιούμε εξειδικευμένες εντολές.

5.5.1 Η εντολή title();

Για να δώσουμε έναν τίτλο στο γράφημα εισάγουμε:

```
title('title of the graph here');
```

Κάθε γράφημα έχει μόνο έναν τίτλο, ο οποίος τίθεται στο κέντρο του πάνω μέρους του.
Αντί για αλφαριθμητικό μέσα σε εισαγωγικά, μπορούμε να περάσουμε στην εντολή title το όνομα μιας συνάρτησης, η οποία καλείται και επιστρέφει ένα αλφαριθμητικό.

```
title(function_name);
```

Είναι δυνατόν να αλλάξουμε τη μορφή εμφάνισης του τίτλου, ελέγχοντας τις ιδιότητές του, δίνοντας το όνομά τους ακολουθούμενο από την επιθυμητή τιμή, ως εξής:

```
title(...,'Property_Name', Property_Value,...)
```

Για να αναφερθούμε σε συγκεκριμένο σύστημα αξόνων, χρησιμοποιούμε την ακόλουθη σύνταξη:

```
title(axes_handle,...);
```

όπου στη μεταβλητή axes_handle έχει αποδοθεί ο χειριστής του.

Το αντικείμενο που δημιουργείται ως τίτλος μπορεί με απόδοση σε μια μεταβλητή να αποκτήσει δικό του αριθμό-χειριστή:

```
h = title(...);
```

Μπορούμε να προσθέσουμε μεταβλητές και κλήσεις συναρτήσεων στον τίτλο:

```
f = 70;
c = (f--32)/1.8;
title(['Temperature is ',num2str(c),'C','on ',date]);
```

όπου date η τρέχουσα ημερομηνία του συστήματος και num2str(); η συνάρτηση που μετατρέπει έναν αριθμό σε αλφαριθμητικό.

Η συνάρτηση int2str(); μετατρέπει έναν ακέραιο σε αλφαριθμητικό, ενώ η ιδιότητα 'Color' ακολουθούμενη από το σύμβολο ενός χρώματος μας δίνει τη δυνατότητα να επιλέξουμε το χρώμα του τίτλου:

```
n = 3;
title(['Case number #',int2str(n)],'Color','y')
```

Μπορούμε να συμπεριλάβουμε ελληνικούς χαρακτήρες στους τίτλους, οι οποίοι είναι χρήσιμοι για τις διάφορες μαθηματικές εκφράσεις:

```
title('\ite^{i\omega\tau} = cos(\omega\tau) + isin(\omega\tau)');
```

Σε αυτή την έκφραση βλέπουμε να ορίζονται πλάγιοι χαρακτήρες: \it, εκθέτες: ^{ }, καθώς και ελληνικοί χαρακτήρες: \omega\tau.

Για να δημιουργήσουμε δείκτες χρησιμοποιούμε την κάτω παύλα (underscore). Για παράδειγμα, για να εμφανιστεί ο συμβολισμός “X_1” δίνουμε:

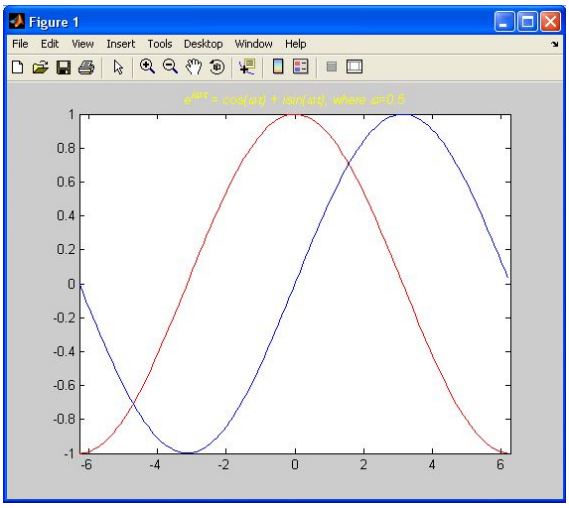
```
title('X_1');
```

Μπορούμε να επεκτείνουμε τον τίτλο σε περισσότερες γραμμές, χρησιμοποιώντας τον χαρακτήρα column (;), ως εξής:

```
title({'First line';'Second line'});
```

Ακολουθεί ένα παράδειγμα.

```
v=0.5
t=(-2*pi:0.1:2*pi)
f=exp(i*v*t)
plot(t,real(exp(i*v*t)),'r')

hold on
plot(t,imag(exp(i*v*t)),'b')
axis([-2*pi 2*pi -1 1])

title(['\ite^{i\omega\tau} = cos(\omega\tau)+isin(\omega\tau), ...
                            where \omega=',num2str(v)],'Color','y');
```



Εικόνα 5.5.1.1. Η γραφική παράσταση της συνάρτησης $e^{i\omega t}$.

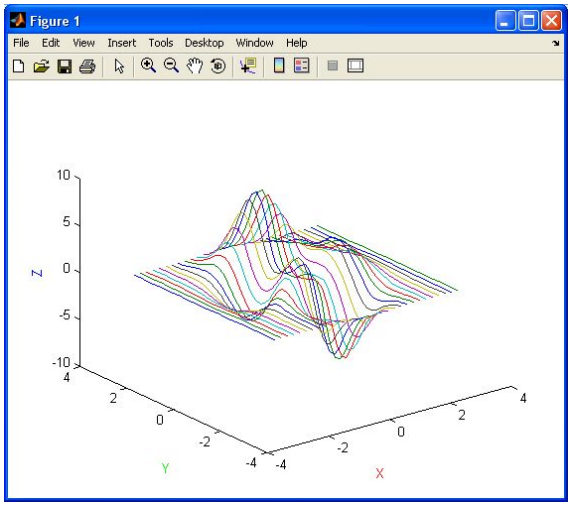
5.5.2 Οι εντολές xlabel(); ylabel(); zlabel();

Για να ονομάσουμε κάποιον από τους άξονες, χρησιμοποιούμε τις παρακάτω εντολές:

```
xlabel('x');     ylabel('y');     zlabel
```

Ασφαλώς, η ονομασία τους μπορεί να διαφέρει, δεν είναι απαραίτητο να χρησιμοποιηθούν τα x,y,z. Επίσης, μπορούν να προστεθούν και ιδιότητες με τις τιμές τους. Στη συνέχεια παρατίθεται ένα παράδειγμα, και πάλι με την συνάρτηση peaks();

```
[x,y,z] = peaks(30);    plot3(x,y,z);
axis([-3 3 -3 3 -10 5]);
xlabel('X','Color','r');      ylabel('Y','Color','g');
zlabel('Z','Color','b');
```



Εικόνα 5.5.2.1. Ονοματοδότηση των αξόνων με τις εντολές xlabel(); ylabel(); zlabel();

Οι εντολές title(); xlabel(); ylabel(); zlabel(); τοποθετούν αλφαριθμητικά στον περίγυρο του διαγράμματος, με αποτέλεσμα η οποιαδήποτε μεταβολή του να μην επηρεάζει τη θέση τους. Υπάρχουν κι άλλες εντολές τοποθέτησης αλφαριθμητικού στο διάγραμμα, που πρέπει όμως να χρησιμοποιηθούν αφού έχει ολοκληρωθεί η τελική του διαμόρφωση, όπως η text(); και η gtext();

5.5.3 Η εντολή text();

Οι συντακτικές μορφές της εντολής text(); είναι οι εξής:

```
text(x,y,'string');
text(x,y,z,'string');
```

Τα x,y,z, είναι συντεταγμένες στο διαξονικό (x,y) ή τριαξονικό (x,y,z) σύστημα. Μέσα σε απλά εισαγωγικά περικλείεται το αλφαριθμητικό.

```
text(...'PropertyName',PropertyValue...);
```

Μπορούμε να μεταβάλλουμε τις ιδιότητες του αντικειμένου αλφαριθμητικού εισάγοντας το όνομα της ιδιότητας σε απλά εισαγωγικά και μετά από κόμμα την τιμή της.

```
h = text(...);
```

Μπορούμε να αποδώσουμε τη δημιουργία του αλφαριθμητικού αντικειμένου σε μια μεταβλητή, δημιουργώντας ένα διάνυσμα-στήλη που περιλαμβάνει τους χειριστές αντικειμένου όλων των αλφαριθμητικών και στη συνέχεια να αναφερόμαστε σε κάθε αντικείμενο χρησιμοποιώντας τον αντίστοιχο χειριστή μέσω της μεταβλητής.

Για πραγματοποιηθεί αυτό, πρέπει να εισαχθούν πολλά σημεία με τη μορφή διανυσμάτων, όπου στο πρώτο διάνυσμα τίθενται όλα τα x, στο δεύτερο όλα τα y και στο τρίτο όλα τα z. Το αλφαριθμητικό θα τυπωθεί τόσες φορές, όσα είναι και τα σημεία. Για παράδειγμα:

```
h=text([0 1],[2 3],'abc');
```

η εντολή αυτή θα τυπώσει στο διάγραμμα, δύο φορές το αλφαριθμητικό “abc”, μία στη θέση (0,2) και μία στη θέση (1,3), ενώ στην οθόνη θα εκτυπωθεί το διάνυσμα με τους χειριστές των αντικειμένων:

```
h =

  153.0238
  154.0238
```

Με την επόμενη εντολή χρωματίζεται πρασινωπό το φόντο του αλφαριθμητικού που τυπώθηκε στη θέση (0,2):

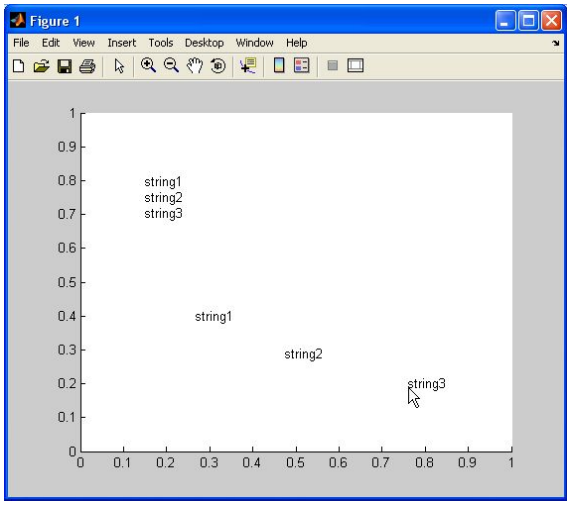
```
set(h(1),'BackgroundColor',[.7 .9 .7]);
```

5.5.4 Η εντολή gtext();

Η εντολή gtext(); δίνει τη δυνατότητα να τοποθετήσει ο χρήστης ένα αλφαριθμητικό πάνω στο διάγραμμα υποδεικνύοντας το σημείο με τον δείκτη του ποντικιού και συντάσσεται με τους παρακάτω τρόπους:

```
gtext('string');
gtext({'string1','string2','string3',...});
gtext({'string1';'string2';'string3';...});
h = gtext(...);
```

Αν τα αλφαριθμητικά είναι χωρισμένα με κόμμα, ο χρήστης κάνει κλικ μία φορά κι αυτά τοποθετούνται το ένα κάτω από το άλλο. Αν είναι χωρισμένα με ελληνικό ερωτηματικό, τότε ο χρήστης υποδεικνύει τρία σημεία, ένα για κάθε αλφαριθμητικό.



Εικόνα 5.5.4.1. Η εφαρμογή της εντολής gtext();

Με απόδοση σε μεταβλητή, λαμβάνουμε κατά τα γνωστά έναν χειριστή αντικειμένου.

5.5.5 Η εντολή legend();

Με την εντολή legend(); δημιουργείται ένα υπόμνημα που παρέχει πληροφορίες για κάθε καμπύλη ή επιφάνεια που εμφανίζεται στο διάγραμμα.

```
legend('string1','string2',...)
legend(h,'string1','string2',...)
```

Η σειρά των αλφαριθμητικών που εισάγονται στην εντολή αντιστοιχίζεται στη σειρά με την οποία έχουν δημιουργηθεί οι καμπύλες του διαγράμματος.
Ο χειριστής αντικειμένου h χρησιμοποιείται για αναφορά σε συγκεκριμένο διάγραμμα.

```
legend(string_matrix)
legend(h,string_matrix)
```

Η παραπάνω σύνταξη επιτρέπει τη χρήση ενός πίνακα αλφαριθμητικών. Η πρώτη τιμή του πίνακα αντιστοιχίζεται με την καμπύλη που έχει δημιουργηθεί πρώτη κ.ο.κ.

Για χρήση χειριστού αντικειμένου αξόνων, η σύνταξη είναι ως εξής:

```
legend(axes_handle,...)
```

Για να αφαιρέσουμε το υπόμνημα δίνουμε:

```
legend('off'), legend(axes_handle,'off')
```

Για να μεταβάλλουμε την κατάσταση του υπομνήματος από ανενεργό σε ενεργό και αντίστροφα (αν δεν υπάρχει δημιουργείται):

```
legend('toggle'), legend(axes_handle,'toggle')
```

Για να κρύψουμε και να εμφανίσουμε το υπόμνημα:

```
legend('hide'), legend(axes_handle,'hide')
legend('show'), legend(axes_handle,'show')
```

Για να εξαφανίσουμε και να εμφανίσουμε το περίγραμμα του υπομνήματος:

```
legend('boxoff'), legend(axes_handle,'boxoff')
legend('boxon'), legend(axes_handle,'boxon')
```

Για να αποδώσουμε τον χειριστή αντικειμένου υπομνήματος σε μεταβλητή και να αναφερθούμε στο αντικείμενο αυτό:

```
legend_handle = legend(...)
legend(legend_handle,...)
```

Για να αποδώσουμε χειριστές αντικειμένων του υπομνήματος, των αντικειμένων του υπομνήματος, των αντικειμένων του διαγράμματος και των αλφαριθμητικών κειμένου του υπομνήματος:

```
[legend_h,object_h,plot_h,text_strings] = legend(...)
```

Για να επηρεάσουμε την τοποθέτηση και τον προσανατολισμό του υπομνήματος:

```
legend(...,'Location',location)
legend(...,'Orientation',orientation)
```

όπου location είναι διάνυσμα 1x4 ([αριστερό_όριο κάτω_όριο πλάτος ύψος]) ή ένα από τα εξής αλφαριθμητικά:
North, South, East, West, NorthEast (default), NorthWest, SouthEast, SouthWest, NorthOutside, SouthOutside, EastOutside, WestOutside, NorthEastOutside, NorthWestOutside, SouthEastOutside, SouthWestOutside, Best, BestOutside
και orientation: vertical ή horizontal.

Μπορούμε επίσης να αποδώσουμε αλφαριθμητικά σε αντικείμενα πληροφοριών υπομνήματος είτε με παράθεση είτε με τη βοήθεια μήτρας αλφαριθμητικών:

```
legend(li_object,string1,string2,string3)
legend(li_object,M)
```

Για να ανανεώσουμε τα υπομνήματα του γραφήματος δίνουμε:

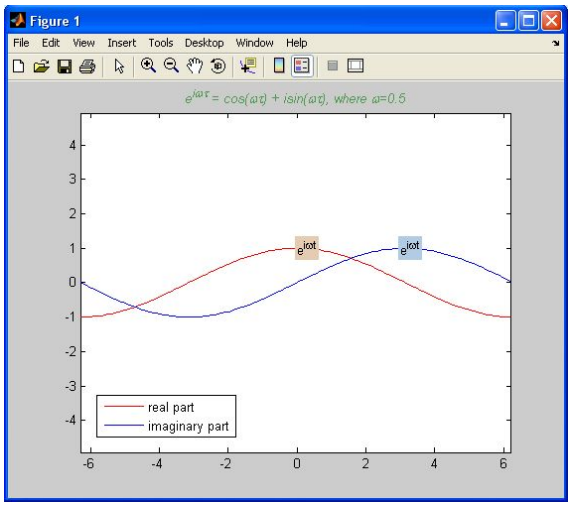
```
legend;
```

Ως παράδειγμα, οι παρακάτω γραμμές θα αναπαραγάγουν το διάγραμμα της της συνάρτησης $e^{i\omega t}$ (εικόνα 5.5.1).

Εδώ θα δημιουργηθούν δύο αλφαριθμητικά " $e^{i\omega t}$ " με χρωματιστό φόντο, ένα για το πραγματικό κι ένα για το φανταστικό μέρος, και μία λεζάντα που να υποδεικνύει τη γραμμή του πραγματικού και τη γραμμή του φανταστικού μέρους.

```
t=(-2*pi:0.1:2*pi);
f=exp(i*v*t);
plot(t,real(exp(i*v*t)),'r');
hold on;

plot(t,imag(exp(i*v*t)),'b');
axis equal;

h=text([0 3],[1 1],'e^{i\omegat}')
set(h(1),'BackgroundColor',[.9 .8 .7]);
set(h(2),'BackgroundColor',[.7 .8 .9]);
title(['\ite^{i\omega\tau} = cos(\omega\tau)+isin(\omega\tau), ...
                  where \omega=',num2str(v)],'Color',[.3 .6 .3]);
```

Τα σημεία που δίνονται μέσα στις αγκύλες της εντολής text είναι τα (0,1) και (3,1). Δηλαδή, διαβάζονται τα πρώτα στοιχεία ως ζευγάρι και μετά τα δεύτερα.



Εικόνα 5.5.5.1. Η συνάρτηση $e^{i\omega t}$ με τίτλο, πεδία κειμένου και λεζάντα.

5.5.6 Η διαμόρφωση των ιδιοτήτων

Μπορούμε να μεταβάλλουμε τις ιδιότητες ενός γραφήματος χρησιμοποιώντας το γραφικό πάνελ του διαχειριστή ιδιοτήτων (property editor) που ενεργοποιείται από τη γραμμή των μενού του παραθύρου ενός διαγράμματος:

Edit > Figure properties ... ή

View > Property editor

ή από τη γραμμή εντολών, δίνοντας:

```
propertyeditor;
```
ή

```
propertyeditor('on');
propertyeditor('off');
propertyeditor('toggle');
propertyeditor(figure_handle,...);
```

Οι τιμές των διαφόρων ιδιοτήτων μπορούν να ελεγχθούν με τις εντολές get(); και set();

5.5.6.1 Η διαμόρφωση των ιδιοτήτων κειμένου

Οι ιδιότητες κειμένου περιλαμβάνουν μια πληθώρα χαρακτηριστικών που διαμορφώνουν το χρώμα, το μέγεθος και το είδος των γραμμάτων, και πολλά άλλα.

Όλες αυτές οι ιδιότητες παρουσιάζονται στον επόμενο πίνακα αναλυτικά.
Όπου στον πίνακα αναφέρεται η τιμή: ColorSpec, υπονοείται μία τιμή σαν αυτές που παρατίθενται στο μεθεπόμενο πινακάκι (5.5.6.1.2).

Ιδιότητα	**Δυνατές τιμές**	**Σχόλια**
BackgroundColor	ColorSpec | {none}	
BeingDeleted	on | {off}	read only
BusyAction	cancel | {queue}	
ButtonDownFcn	string | function handle	
Children	matrix	read only
Clipping	on | {off}	
Color	ColorSpec	
CreateFcn	string | function handle	
DeleteFcn	string | function handle	
EdgeColor	ColorSpec | {none}	
Editing	on | {off}	
EraseMode	{normal} | none | xor | background	
Extent	position rectangle	read only
FontAngle	{normal} | italic | oblique	
FontName	A name, such as Courier, or the string FixedWidth	
FontSize	size in FontUnits	
FontWeight	light | {normal} | demi | bold	
FontUnits	{points} | normalized | inches | centimeters | pixels	
HandleVisibility	{on} | callback | off	
HitTest	{on} | off	
HorizontalAlignment	{left} | center | right	
Interpreter	latex | {tex} | none	
Interruptible	{on} | off	

LineStyle	{-} | -- | : | -. | none	
LineWidth	scalar (points)	
Margin	scalar (pixels)	
Parent	handle of axes, hggroup, or hgtransform	
Position	[x,y,[z]]	
Rotation	scalar (default = 0)	
Selected	on | {off}	
SelectionHighlight	{on} | off	
String	string	
Tag	string	
Type	string (read only)	
Units	pixels | normalized | inches | centimeters | points | {data}	
UserData	matrix	
UIContextMenu	handle of a uicontextmenu object	
VerticalAlignment	top | cap | {middle} | baseline | bottom	
Visible	{on} | off	

Πίνακας 5.5.6.1.1. Οι ιδιότητες κειμένου και οι δυνατές τιμές τους.

Ακολουθούν παραδείγματα τιμών ColorSpec.

RGB Value	Short Name	Long Name
[1 1 0]	y	yellow
[1 0 1]	m	magenta
[0 1 1]	c	cyan
[1 0 0]	r	red
[0 1 0]	g	green
[0 0 1]	b	blue
[1 1 1]	w	white
[0 0 0]	k	black

Πίνακας 5.5.6.1.2. Παραδείγματα τιμών ColorSpec.

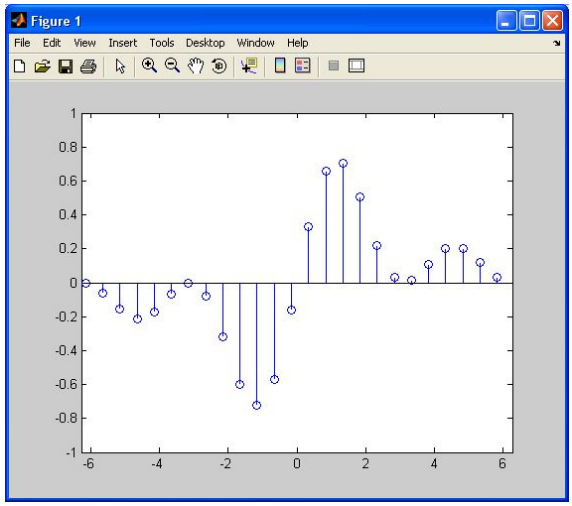
Μια τιμή ColorSpec δεν είναι απαραίτητο να εμπίπτει στις παραπάνω τυποποιημένες τιμές. Μπορεί για παράδειγμα να είναι: [.4 .7 .3]

6. ΕΠΙΠΛΕΟΝ ΕΡΓΑΛΕΙΑ ΓΡΑΦΙΚΩΝ ΠΑΡΑΣΤΑΣΕΩΝ

6.1 Γραμμωτά διαγράμματα (stem)

Τα γραμμωτά διαγράμματα δημιουργούνται με την εντολή stem(); και συνήθως χρησιμοποιούνται για την αναπαράσταση σημάτων.

```
x=(-100*pi:.5:100*pi);
stem(x,(sin(x).^2)./x);
axis ([-2*pi 2*pi -1 1 ]);
```



Εικόνα 6.1.1. Η συνάρτηση f = sin(x).^2)./x από την εντολή stem();

Αν δοθεί μόνο το διάνυσμα y στη stem(); τότε η απεικόνιση των σημείων θα γίνει ως προς τον δείκτη που έχουν εντός του διανύσματος y.

Μπορούμε να δώσουμε και άλλες παραμέτρους, όπως είδος γραμμής, είδος συμβόλου επισήμανσης σημείων και χρωματισμένο ή μη. Για παράδειγμα, από τις επόμενες εντολές, η πρώτη χρωματίζει τα κυκλάκια και η δεύτερη τα αντικαθιστά με αστερίσκους και την συνεχή με διακεκομμένη γραμμή:

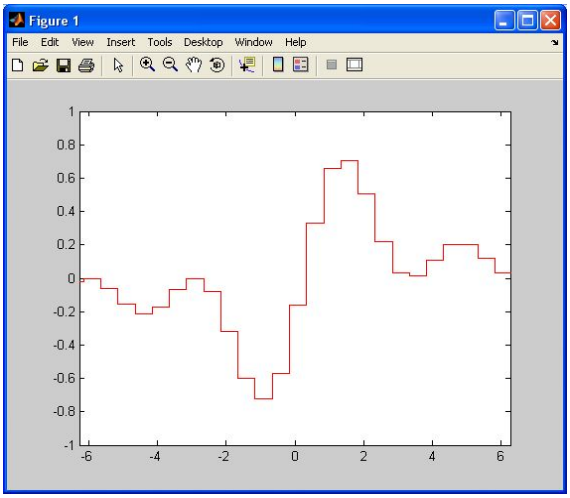
```
x=(-100*pi:.5:100*pi);
stem(x,(sin(x).^2)./x,'filled');
stem(x,(sin(x).^2)./x,'--*');
```

6.2 Κλιμακωτά διαγράμματα (stairs)

Τα κλιμακωτά διαγράμματα έχουν το πλεονέκτημα να κάνουν εμφανές το χαρακτηριστικό της διακριτότητας των στοιχείων ενός συνόλου τιμών. Η διακριτή φύση των δεδομένων αφορά κυρίως διακριτά σήματα ή διακριτές εξισώσεις.

Το γράφημα της προηγούμενης παραγράφου με την εντολή stairs(); έχει ως εξής:

```
x=(-100*pi:.5:100*pi);
stairs(x,(sin(x).^2)./x,'r');
axis ([-2*pi 2*pi -1 1 ]);
```



Εικόνα 6.2.1. Η συνάρτηση f = sin(x).^2)./x από την εντολή stairs();

6.3 Διαγράμματα με περιοχή σφάλματος (errobar)

Τα διαγράμματα αυτά τυπώνουν την καμπύλη των τιμών, ενώ συγχρόνως παρουσιάζουν και το σφάλμα γύρω από κάθε τιμή.
Τα σφάλματα καταχωρούνται σε ένα διάνυσμα ίσου μεγέθους με το διάνυσμα τιμών της καμπύλης.
Μπορούμε να προσομοιώσουμε μια τέτοια περίπτωση κατασκευάζοντας τα δεδομένα μας με τη βοήθεια της συνάρτησης παραγωγής τυχαίων αριθμών του MatLab(R).

Ας χρησιμοποιήσουμε τη συνάρτηση f = (sin(x).^2)./x.

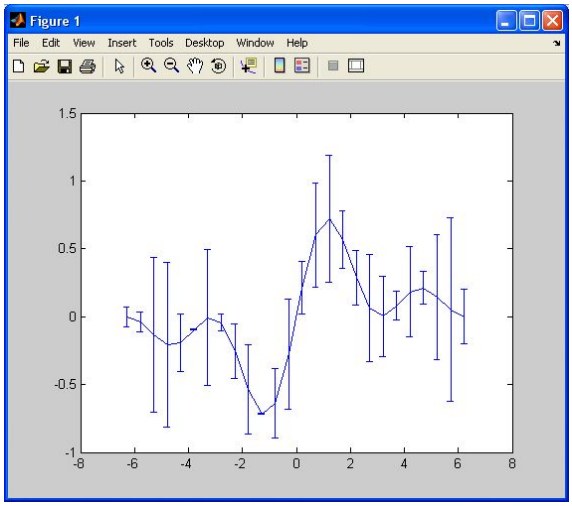
```
x = (-2*pi:.5:2*pi);
y = (sin(x).^2)./x;
```

Θα δημιουργήσουμε ένα διάνυσμα σφαλμάτων βάση της τυπικής απόκλισης των τιμών του y και τυχαίων διακυμάνσεων από τη συνάρτηση randn();

```
e = std(y)*ones(size(x));
e= e + randn(1,length(y)) ./ 5;
```

Το διάγραμμα δημιουργείται με την εντολή errorbar(); Ο ίδιος συντελεστής σφάλματος εφαρμόζεται εις διπλούν (μπάρα πάνω - μπάρα κάτω), σε όλες τις τιμές.
Συνολικά:

```
x = (-2*pi:.5:2*pi);
y = (sin(x).^2)./x;
e = std(y)*ones(size(x));
e= e + randn(1,length(y)) ./ 5;
errorbar(x,y,e)
```



Εικόνα 6.3.1. Γραφική παράσταση της συνάρτησης (sin(x).^2)./x με περιοχές σφάλματος.

6.4 Συστήματα πολικών συντεταγμένων

Οι πολικές συντεταγμένες στα μαθηματικά αποτελούν έναν τρόπο παράστασης σημείων στο επίπεδο με τη βοήθεια μια γωνίας και ενός μήκους.

Ξεκινώντας από τον οριζόντιο άξονα και διαγράφοντας γωνία με φορά αντίθετη από αυτή των δεικτών του ρολογιού λαμβάνεται μια γωνία θ. Έπειτα, πάνω στον άξονα αυτόν, που σχηματίζει γωνία θ με τον οριζόντιο άξονα, λαμβάνεται μήκος ρ.

Η παράσταση πολικών συντεταγμένων επιτυγχάνεται με την εντολή polar();
Η σύνταξη της έχει ως εξής:

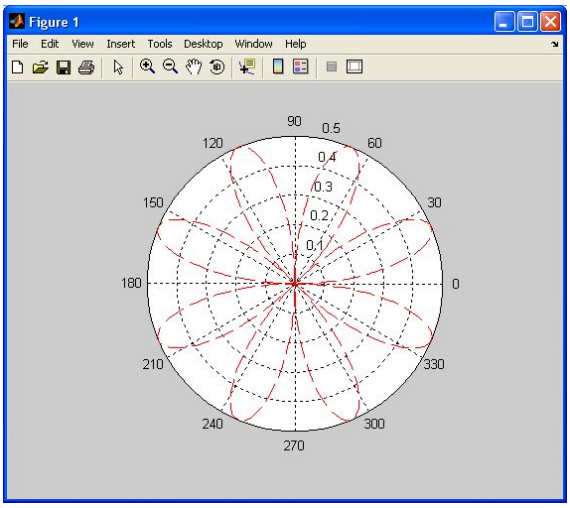
```
polar(theta,rho)
polar(theta,rho,LineSpec)
polar(axes_handle,...)
h = polar(...)
```

όπου theta η γωνία θ σε ακτίνια, rho το μήκος ρ και LineSpec ο τύπος και το χρώμα της γραμμής, καθώς και το σύμβολο παράστασης των σημείων.

Ως παράδειγμα μπορούμε να εισάγουμε τις παρακάτω γραμμές:

```
t = 0:.01:2*pi;
polar(t,sin(2*t).*cos(2*t),'--r');
```

Το αποτέλεσμα φαίνεται στο επόμενο σχήμα.



Εικόνα 6.4.1. Η εντολή polar();

Μια άλλη εντολή χρήσιμη για την παράσταση δεδομένων γωνιών είναι η rose();

Η εντολή rose(); στην πραγματικότητα δημιουργεί ένα γωνιοϊστόγραμμα παριστάνοντας την κατανομή των περιεχόμενων ενός διανύσματος γωνιών σε ένα πολικό σύστημα συντεταγμένων. Ο κυκλικός δίσκος διαμερίζεται σε τομείς και πάνω σε κάθε τομέα προβάλλεται μία τριγωνική μπάρα με μέγεθος ανάλογο του αριθμού γωνιών που αντιπροσωπεύει.

Η σύνταξη της εντολής rose(); έχει ως εξής:

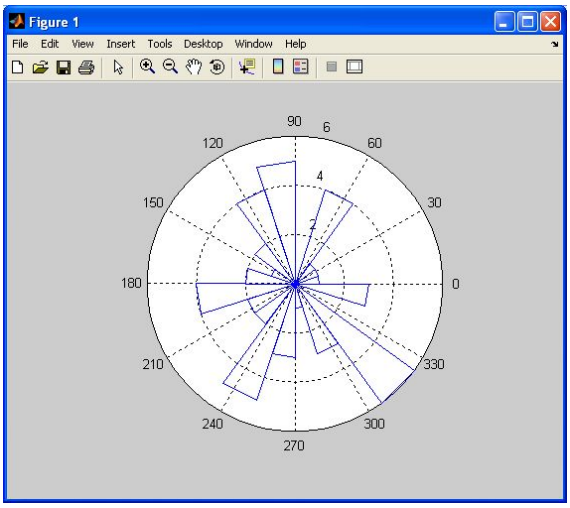
```
rose(theta)
rose(theta,x)
rose(theta,nbins)
rose(axes_handles,...)
h = rose(...)
[tout,rout] = rose(...)
```

όπου theta ένα διάνυσμα που περιέχει γωνίες, x ένα διάνυσμα που περιέχει τις κεντρικές γωνίες των τομέων του κυκλικού δίσκου στους οποίους αυτός θα διαμεριστεί (ο αριθμός τους είναι ίσος με length(x)), και τέλος, nbins μια μεταβλητή που απλώς περιέχει τον αριθμό των τομέων, οι οποίοι θα δημιουργηθούν ίσοι (προκαθορισμένη τιμή 20).

Ως παράδειγμα, παρατίθενται οι πιο κάτω γραμμές:

```
theta = 2*pi*rand(1,50);
rose(theta);
```

Η παράσταση του πίνακα theta με τυχαίες τιμές γωνιών φαίνεται στην επόμενη εικόνα.



Εικόνα 6.4.2. Η εντολή rose();

Αν κατά τη δημιουργία του γραφήματος αποδώσουμε την εντολή rose(); σε δύο μεταβλητές (δεν δημιουργείται γράφημα σε αυτό το σημείο), μπορούμε κατόπιν να τις χρησιμοποιήσουμε με την εντολή plot(); για να σχηματίσουμε το ιστόγραμμα κατά τη στιγμή που το επιθυμούμε (μέσα σε ένα m-file για παράδειγμα).

```
theta = 2*pi*rand(1,50);
[t,r]=rose(theta);
polar(t,r);
```

Οι παραπάνω γραμμές, αν δεν εκτελέσουμε ξανά την πρώτη γραμμή που θα μεταβάλει το διάνυσμα theta, θα δημιουργήσουν ξανά το γράφημα της εικόνας 6.4.2.

6.5 Καμπύλες με contour

Η εντολή contour(); χρησιμεύει για τη δημιουργία δισδιάστατων και τρισδιάστατων γραφημάτων ισοδιάστατων καμπυλών. Για να ακριβολογήσουμε, είναι δυνατόν να μη ληφθούν τα περιγράμματα των τομών με ισοδιάσταση, αλλά σε επίπεδα που θα καθοριστούν από ένα διάνυσμα.

Η σύνταξη της εντολής contour(); έχει ως εξής:

```
contour(Z)
contour(Z,n)
contour(Z,v)
contour(X,Y,Z)
contour(X,Y,Z,n)
contour(X,Y,Z,v)
contour(...,LineSpec)
[C,h] = contour(...)
[C,h] = contour('v6',...)
```

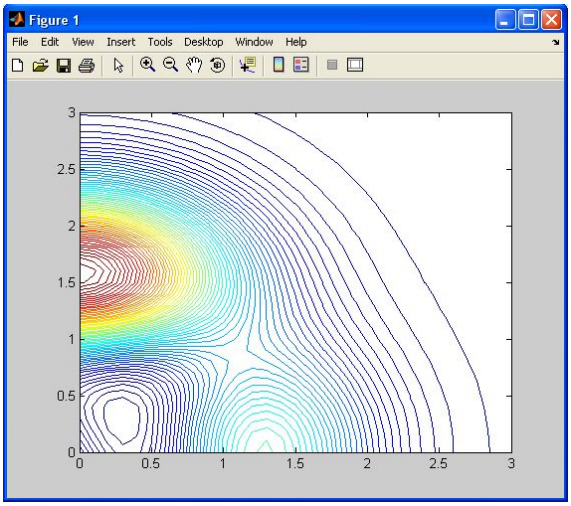
Μπορούμε να εισάγουμε μόνο έναν πίνακα τιμών-υψών (Z) ή τρεις (X,Y,Z).
Κατά τα γνωστά, μόνο με πίνακα υψών θα έχουμε ένα διάγραμμα των τιμών αυτών προς τους δείκτες των γραμμών και των στηλών, ενώ οι τρεις πίνακες λειτουργούν όπως με την εντολή surf(); Πρέπει δηλαδή να είναι ίσων διαστάσεων.

Μετά τους πίνακες των δεδομένων της επιφάνειας μπορούμε να εισάγουμε τον αριθμό των ισοϋψών που επιθυμούμε ή να αφήσουμε το MatLab(R) να το κάνει αυτόματα για μας. Μπορούμε επίσης να δώσουμε έναν πίνακα με τις στάθμες που επιθυμούμε, χωρίς να είναι απαραίτητο να ισαπέχουνε.

Ακόμη, μπορούμε να εισαγάγουμε αλφαριθμητικό χαρακτηριστικών γραμμής ή να αποδώσουμε την εντολή contour(); σε έναν πίνακα και έναν χειριστή αντικειμένου.

Ακολουθεί ένα απλό παράδειγμα.

```
x=(0:0.1:3);
[X,Y]=meshgrid(x);
z=peaks(x);
contour(X,Y,z,60);
```

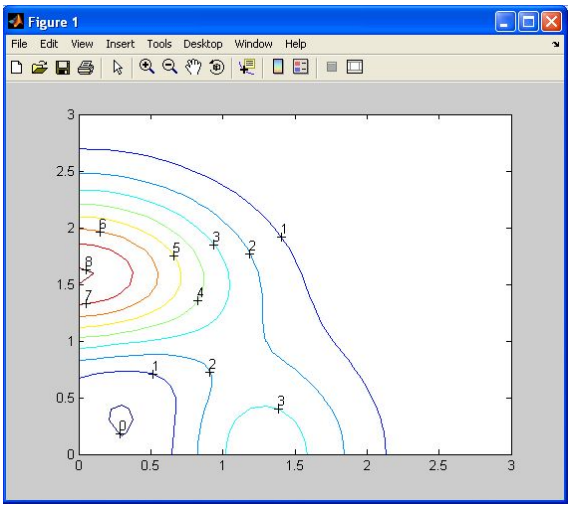


Εικόνα 6.5.1. Η εντολή contour(); για την peaks(); με 60 επίπεδα ισοδιάστασης.

Με την εντολή clabel(); αναγράφουμε την τιμή ύψους για κάθε ισοϋψή καμπύλη.

```
x=(0:0.1:3);
[X,Y]=meshgrid(x);
z=peaks(x);
c=contour(X,Y,z);
clabel(c);
```

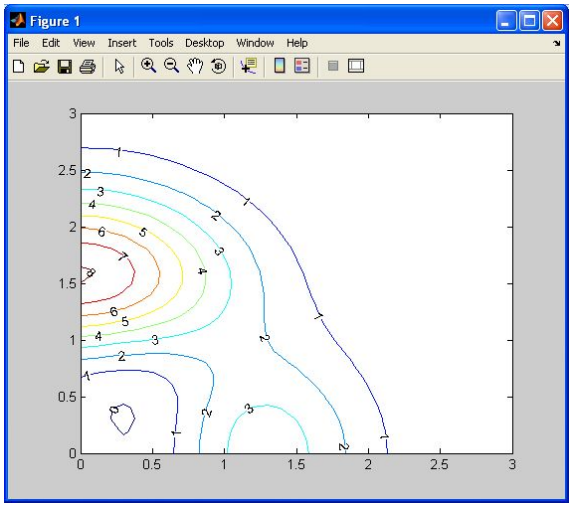
Με τον τρόπο αυτό δημιουργείται το παρακάτω γράφημα.



Εικόνα 6.5.2. Η χρήση της clabel();

Αν χρησιμοποιήσουμε την παρακάτω σύνταξη με λήψη ενός χειριστή αντικειμένου και εισαγωγή του στην clabel(); ενεργοποιείται μια νεότερη μέθοδος που τοποθετεί τα ύψη διακόπτοντας τις ισοϋψείς.

```
x=(0:0.1:3);
[X,Y]=meshgrid(x);
z=peaks(x);
[c,h]=contour(X,Y,z);
clabel(c,h);
```

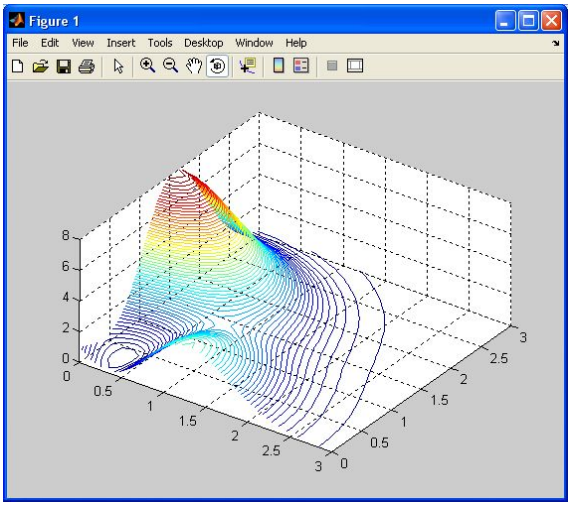


Εικόνα 6.5.3. Τοποθέτηση υψών διακόπτοντας τις ισοϋψείς.

Υπάρχουν και άλλες εντολές σχετικά με διαγράμματα ισοϋψών.

Η contour3(); χρησιμοποιείται για τη δημιουργία τρισδιάστατου διαγράμματος ισοϋψών.

```
x=(0:0.1:3);
[X,Y]=meshgrid(x);
z=peaks(x);
contour3(X,Y,z,60);
```



Εικόνα 6.5.4. Η εντολή contour3();

Μπορούμε να υποδείξουμε εμείς πού θέλουμε να τοποθετηθούν τα ύψη των ισοϋψών δίνοντας την παράμετρο 'manual' στην εντολή clabel();

```
clabel(c,'manual');
clabel(c,h,'manual');
```

Μπορούμε να αποδώσουμε τα χαρακτηριστικά ενός διαγράμματος ισοϋψών με την εντολή contourc();
Η διαφορά της είναι πως λειτουργεί σε χαμηλό επίπεδο. Δεν δημιουργεί διάγραμμα, αλλά έναν πίνακα ο οποίος αποτελείται από στήλες δύο γραμμών με πληροφορίες για το διάγραμμα.
Η σύνταξή της έχει ως εξής:

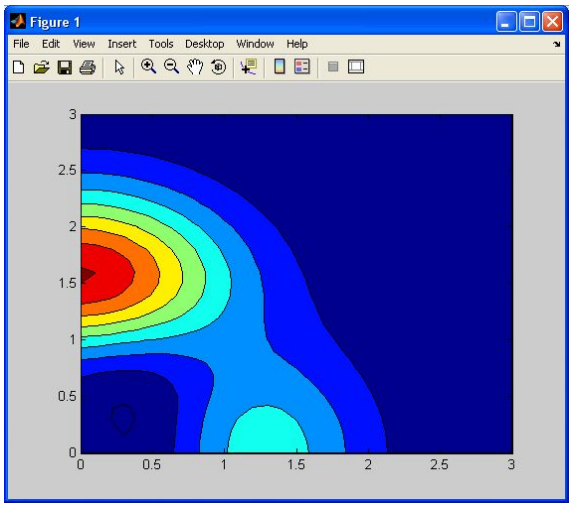
```
C = contourc(Z)
C = contourc(Z,n)
C = contourc(Z,v)
C = contourc(x,y,Z)
C = contourc(x,y,Z,n)
C = contourc(x,y,Z,v)
```

Με την εντολή contourf(); δημιουργούμε διαγράμματα με χρωματισμένες ισοϋψείς περιοχές. Συντάσσεται ως εξής:

```
contourf(Z)
contourf(Z,n)
contourf(Z,v)
contourf(X,Y,Z)
contourf(X,Y,Z,n)
contourf(X,Y,Z,v)
contourf(axes_handle,...)
[C,h,CF] = contourf(...)
```

Κατά τα γνωστά, X, Y, Z, είναι οι πίνακες με τα δεδομένα της επιφάνειας, n ο αριθμός των ισοϋψών, v ένα διάνυσμα με συγκεκριμένες τιμές υψών για τις ισοϋψείς.
Με την σύνταξη: [C,h,CF] = contourf(...); αποδίδεται ένας πινάκας C, όπως αυτός που επιστρέφεται από την contourc(); ένας χειριστής αντικειμένου, και ένας πινάκας CF με πληροφορίες για τις χρωματιστές επιφάνειες.

```
X=(0:0.1:3); z=peaks(x);
contourf(x,x,z);
```



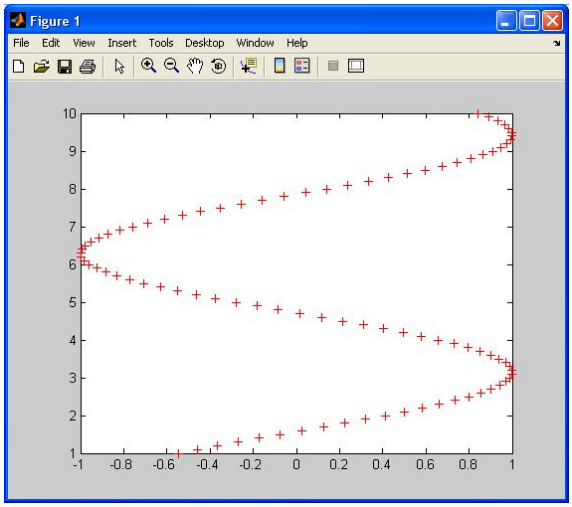
Εικόνα 6.5.5. Η εντολή contourf();

6.6 Αναπαράσταση μιγαδικών αριθμών

Αν στην εντολή plot(); δοθεί ένας πίνακας που περιέχει μιγαδικές τιμές, τότε η καμπύλη που θα δημιουργηθεί θα έχει ως τετμημένες το πραγματικό μέρος των τιμών και ως τεταγμένες το φανταστικό.

Αν δώσουμε δύο διανύσματα, ως plot(x,y); τότε το φανταστικό μέρος των μιγαδικών τιμών του y θα αγνοηθεί και θα τυπωθούν μόνο οι πραγματικές τιμές του y προς τις τιμές του x.

```
x = (1:0.1:10);
y = (sin(x)+[10:-0.1:1]*i);
plot(y,'+r');
```



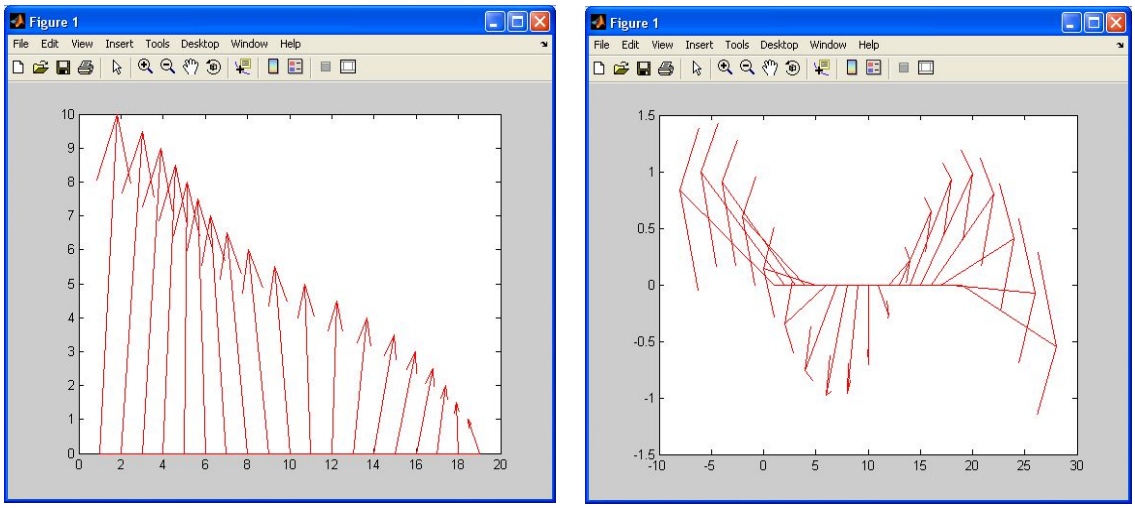
Εικόνα 6.6.1. Παράσταση μιγαδικών αριθμών με την εντολή plot();

Εναλλακτικά, μπορούμε να παραστήσουμε μιγαδικές τιμές με τη βοήθεια της εντολής feather(); Η feather(); δέχεται ένα ή δύο διανύσματα και τυπώνει βέλη που καταδεικνύουν τις τιμές στο μιγαδικό επίπεδο. Το μέγεθος της κεφαλής των βελών είναι ανάλογο του μεγέθους της τιμής που καταδεικνύουν.

```
feather(U,V)
feather(Z)
feather(...,LineSpec)
feather(axes_handle,...)
h = feather(...)
```

Παρακάτω, η ίδια συνάρτηση: y = (sin(x)+[10:-0.1:1]*i); αναπαρίσταται από την feather();

```
x1=(1:0.5:10);
y1=(sin(x1)+[10:-0.5:1]*i);
feather(y1,'r'); feather(x1,y1,'r');
```



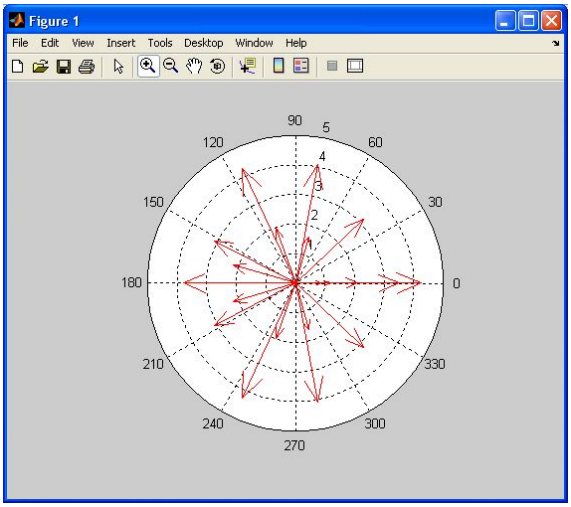
Εικόνα 6.6.2. Παράσταση μιγαδικών αριθμών με feather(y1,'r'); και feather(x1,y1,'r');

Επίσης, η εντολή compass(); παρέχει τη δυνατότητα παράστασης μιγαδικών αριθμών. Χρησιμοποιείται όπως η feather();
Δέχεται έναν ή δύο πίνακες και ένα αλφαριθμητικών ιδιοτήτων γραμμής.

Οι επόμενες γραμμές δημιουργούν ένα διάνυσμα z που περιλαμβάνει τις χαρακτηριστικές ρίζες μιας τυχαίας 20x20 μήτρας.
Στη συνέχεια, οι ρίζες αυτές παριστάνονται από την εντολή compass();

```
z = eig(randn(20,20));
compass(z,'r');
```

Το αποτέλεσμα φαίνεται στην εικόνα 6.6.3.



Εικόνα 6.6.3. Η εντολή compass();

6.7 Διαγράμματα βαθμίδων με την εντολή quiver();

Με την εντολή quiver(); μπορούμε να αναπαραστήσουμε τη βαθμίδα (gradient) μιας επιφάνειας, δηλαδή μιας συνάρτησης δύο ανεξάρτητων μεταβλητών: z=f(x,y).

Η σύνταξη της εντολής quiver(); είναι η ακόλουθη:

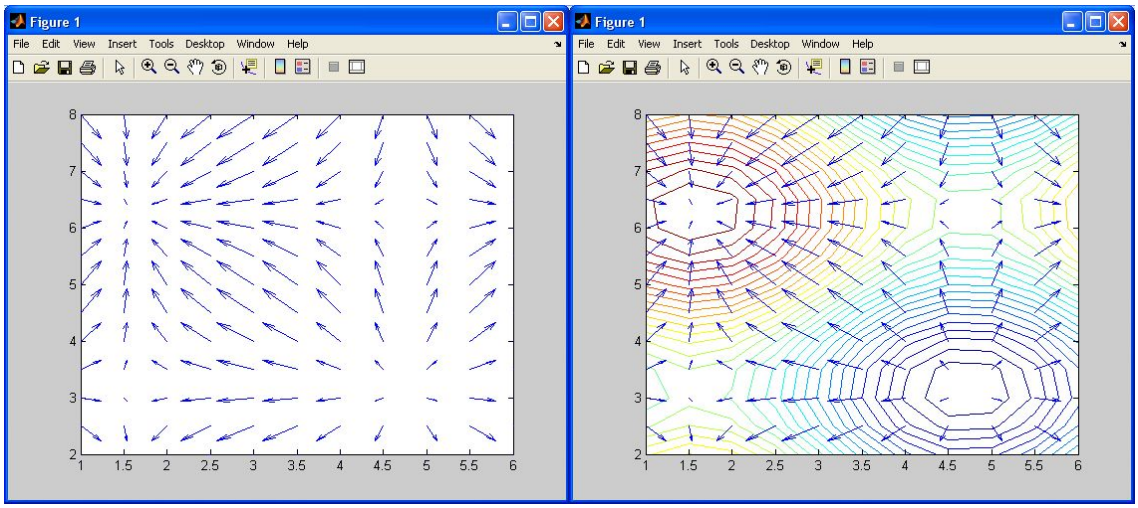
```
quiver(x,y,u,v)
quiver(u,v)
quiver(...,scale)
quiver(...,LineSpec)
quiver(...,LineSpec,'filled')
quiver(axes_handle,...)
h = quiver(...)
hlines = quiver('v6',...)
```

όπου x,y, είναι οι πίνακες με τις θέσεις των διανυσμάτων, u,v, οι πίνακες με την τιμή της βαθμίδας για κάθε διάνυσμα.
Μπορούμε να συνδυάσουμε την απεικόνιση των βαθμίδων με το διάγραμμα ισοϋψών.
Απλώς εμφανίζουμε το διάγραμμα της συνάρτησης με την εντολή contour(); και κατόπιν δίνουμε hold on; και την εντολή quiver();

Οι παρακάτω γραμμές δημιουργούν ένα διάγραμμα βαθμίδων πάνω από μία απεικόνιση ισοϋψών καμπυλών για τη συνάρτηση: z=sin(x)+cos(y);

```
X=(0:0.5:10);
Y=(1:0.5:11);
[x,y]=meshgrid(X,Y);
z=sin(x)+cos(y);
contour(x,y,z,30);
hold on;
dx=diff(y(1:2,1));
```

```
dy=diff(x(1,1:2));
[px,py]=gradient(z,dx,dy);
quiver(x,y,px,py);
axis([1 6 2 8]);
```

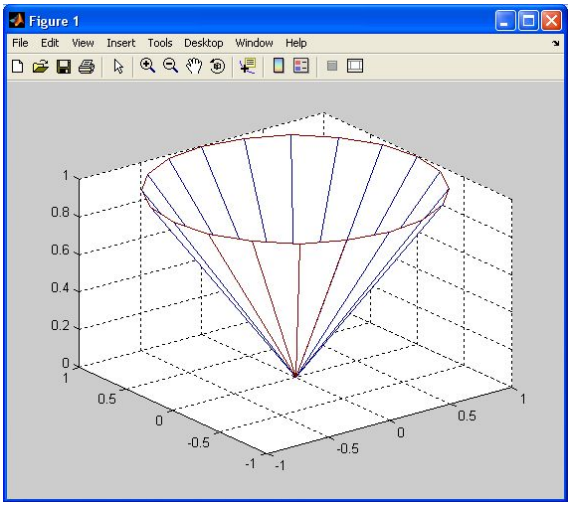


Εικόνα 6.7.1. Η εντολή quiver(); μόνη της αριστερά και μετά από contour(); δεξιά.

6.8 Η εντολή cylinder();

Η εντολη cylinder(); μπορεί να χρησιμοποιηθεί για την κατασκευή ενός στερεού κυλίνδρου, ή ενός κώνου αν δοθεί ως πίνακας των καθ' ύψος ακτινών αντί [1 1], ο [0 1]. Τότε στην κορυφή η ακτίνα είναι μέγιστη (1), ενώ στη βάση είναι 0 (ανάποδος κώνος).
Για να μεταβάλουμε το μέγεθος και την τοποθέτηση του κυλίνδρου, όπως και με την εντολή sphere(); κάνουμε απόδοση των δεδομένων σε τρεις πίνακες και στη συνέχεια τους εισάγουμε σε μια από τις εντολές mesh(); ή surf();

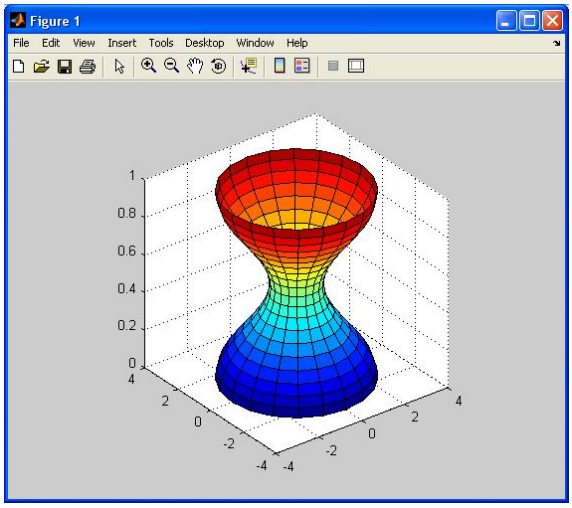
```
[X,Y,Z]=cylinder([0 1]);
surf(X,Y,Z);
```



Εικόνα 6.8.1. Ένας ανάποδος κώνος με την εντολή cylinder();

Εισάγοντας μια συνάρτηση στην εντολή cylinder(); δημιουργείται ένα προφίλ το οποίο με περιστροφή 2π, παράγει μια τρισδιάστατη επιφάνεια περιέλιξης.

```
t = 0:pi/10:2*pi;
[X,Y,Z] = cylinder(2+cos(t));
surf(X,Y,Z);
axis square;
```



Εικόνα 6.8.2. Ένα στερεό από περιστροφή της συνάρτησης 2+cos(x);

6.9 Η εντολή sphere();

Με την εντολή sphere δημιουργείται μια τρισδιάστατη επιφάνεια με τη μορφή σφαίρας.

```
sphere
sphere(n)
[X,Y,Z] = sphere(...)
```

Αν δεν δοθεί παράμετρος η σφαίρα θα εμφανίζει 20 πολύγωνα κατά μήκος και 20 κατά πλάτος, δηλαδή θα είναι 20x20. Αν δοθεί ως sphere(n); τα πολύγωνα της επιφάνειάς της θα είναι 50x50.

Μπορούμε να αποθηκεύσουμε τις συντεταγμένες της επιφάνειας της σφαίρας σε τρεις πίνακες X,Y,Z. στη συνέχεια, για να δημιουργήσουμε τη σφαίρα χρησιμοποιούμε τις εντολές surf και mesh.

Εικόνες με παραδείγματα σφαιρών βρίσκονται στο τέλος του τμήματος 5.2.

6.10 Διαγράμματα κλειστών καμπυλών - Οι εντολές area(); patch();

Με την εντολή area(); μπορούμε να παραστήσουμε τις τιμές ενός πίνακα y ως προς έναν πίνακα x χρωματίζοντας την περιοχή που περικλείεται από την δημιουργούμενη καμπύλη και τους άξονες.

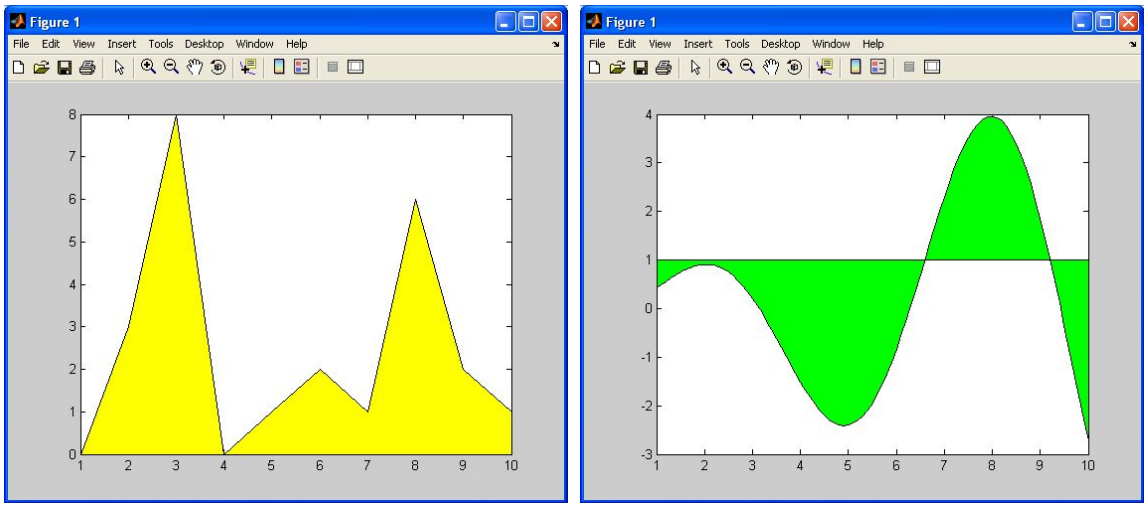
Αν ο y αποτελείται από μία γραμμή τότε το MatLab(R) τον αναστρέφει αυτόματα και οι τιμές της γίνονται μια στήλη. γίνονται Αν ο πίνακας y αποτελείται από πολλές γραμμές, τότε δημιουργούνται πολλές καμπύλες, μία για κάθε στήλη του πίνακα, με τα μεγέθη κάθε γραμμής να μην αλληλεπικαλύπτονται, αλλά να παρατίθενται καθύψος. Η μέθοδος αυτή δημιουργεί προβλήματα, ωστόσο, σε περίπτωση που στον πίνακα y περιλαμβάνονται αρνητικές τιμές.

Οι τρόποι σύνταξης έχουν ως εξής:

```
area(Y)
area(X,Y)
area(...,basevalue)
area(...,'PropertyName',PropertyValue,...)
area(axes_handle,...)
h = area(...)
area('v6',...)
```

Με τις παρακάτω γραμμές δημιουργούμε ένα διάνυσμα x και με τη βοήθεια της συνάρτησης τυχαίων τιμών δύο πίνακες y1 και y2, τους οποίους παριστάνουμε γραφικά με τη βοήθεια της area(); Η area(); δέχεται και επιπλέον παραμέτρους, όπως χρώμα, και επίπεδο χρωματισμού. Το επίπεδο αυτό είναι μια οριζόντια ευθεία που αντικαθιστά τον άξονα x στον σχηματισμό των κλειστών περιοχών.

```
x=(1:10); y1=fix(10*rand(1,10));
area(x,y1,'facecolor','y');
x=(1:.1:10);
y2=sin(x).*x;
area(x,y2,1,'facecolor','g');
```

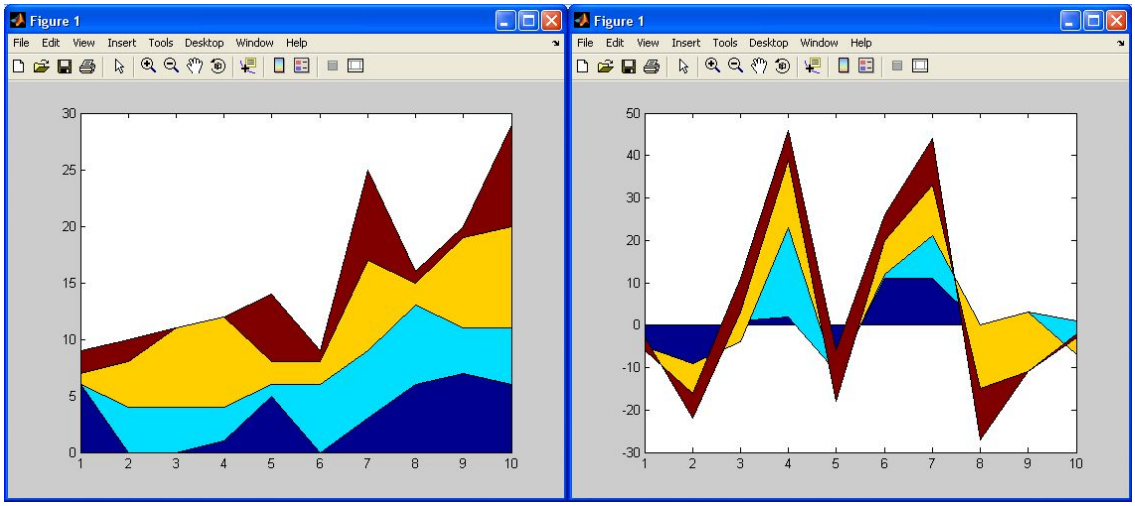


Εικόνα 6.10.1. Η εντολή area(); με πίνακα μιας καμπύλης.

Το παρακάτω παράδειγμα δημιουργεί δύο γραφήματα πολλών καμπυλών με την εντολή area(); Το δεύτερο περιλαμβάνει και αρνητικές τιμές.

```
x=(1:10);
y3=fix(10*rand(10,4));
area(x,y3);
```

```
x=(1:10);
y4=fix(10*randn(10,4));
area(x,y4);
```



Εικόνα 6.10.1. Area(); με πολλές καμπύλες. Δεξιά ο πίνακας περιέχει και αρνητικές τιμές.

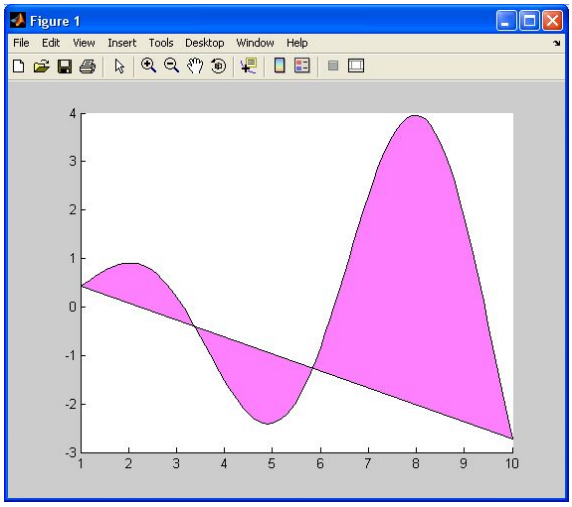
Η εντολή patch(); δέχεται δύο ή τρία διανύσματα των οποίων οι τιμές με ίδιο δείκτη αποτελούν κορυφές ενός κλειστού πολυγώνου. Το πολύγωνο χρωματίζεται με την RGB τριάδα (τιμές από 0.0 ως 1.1) που εισάγεται ως διάνυσμα στην τρίτη θέση.

Οι τρόποι σύνταξης είναι οι εξής:

```
patch(X,Y,C)
patch(X,Y,Z,C)
patch(FV)
patch(...'PropertyName',PropertyValue...)
patch('PropertyName',PropertyValue...) PN/PV pairs only
handle = patch(...)
```

Ακολουθεί ένα παράδειγμα, το γράφημα του οποίου εικονίζεται αμέσως παρακάτω.

```
x=(1:.1:10);
y=sin(x).*x;
patch(x,y,[1 0.5 1]);
```



Εικόνα 6.10.3. Η εξίσωση y=sin(x).*x; με την εντολή patch();

7. ΕΠΙΠΛΕΟΝ ΕΝΤΟΛΕΣ ΔΙΑΜΟΡΦΩΣΗΣ ΓΡΑΦΗΜΑΤΩΝ

7.1 Η μέθοδος subplot

Η μέθοδος απεικόνισης γραφημάτων subplot(); διαιρεί το παράθυρο του γραφήματος σε περισσότερα του ενός τμήματα, σε καθένα από τα οποία μπορεί να απεικονιστεί και ένα διαφορετικό γράφημα.
Η εντολή subplot(); δέχεται τρεις παραμέτρους. Οι δύο πρώτες καθορίζουν τις γραμμές και τις στήλες που θα σχηματίζουν τα επιμέρους τμήματα, ενώ η τρίτη το ενεργό τμήμα, όπου θα εμφανιστούν τα αποτελέσματα μετά από κάποια εντολή απεικόνισης.
Για παράδειγμα, η εντολή:

```
subplot(2,2,3);
```

θα δημιουργήσει ένα γράφημα με τέσσερα τμήματα (υπογραφήματα) και ενεργό θα είναι το τρίτο, δηλαδή αυτό που βρίσκεται στη δεύτερη γραμμή και πρώτη στήλη (κάτω αριστερά).

Αν δοθεί νέα εντολή subplot(); με διαφορετικό αριθμό γραμμών ή στηλών, τότε το προηγούμενο γράφημα αντικαθίσταται από ένα νέο (χάνεται).
Αν θέλουμε να αλλάξουμε το γράφημα, χωρίς να μεταβληθεί ο αριθμός των γραμμών και των στηλών των υπογραφημάτων, δίνουμε:

```
subplot(m,n,p,'replace');
```

Η γενική σύνταξη είναι η εξής:

```
subplot(m,n,p)
subplot(mnp)
```

όπου mnp είναι τρεις αριθμοί οι οποίοι παρατίθενται στη σειρά: subplot(223); Αν δώσουμε subplot(2103); δημιουργείται σύγχυση και πρέπει να χρησιμοποιήσουμε το κόμμα: subplot(2,10,3);

Ευθυγράμμιση των υπογραφημάτων, χωρίς να εμποδίζεται η αλληλεπικάλυψη των πλαισίων κειμένου:

```
subplot(m,n,p,'align');
```

Για να καθοριστεί η τοποθέτηση και το μέγεθος του υπογραφήματος

```
subplot('Position',[left bottom width height]);
```

όπου στην αγκύλη περιλαμβάνονται το αριστερό και κάτω όριο, καθώς και το πλάτος και το ύψος, που είναι κανονικοποιημένα μεγέθη με διάστημα τιμών απλό 0.0 ως 1.0.

Απόδοση χειριστή αντικειμένου υπογραφήματος σε μεταβλητή:

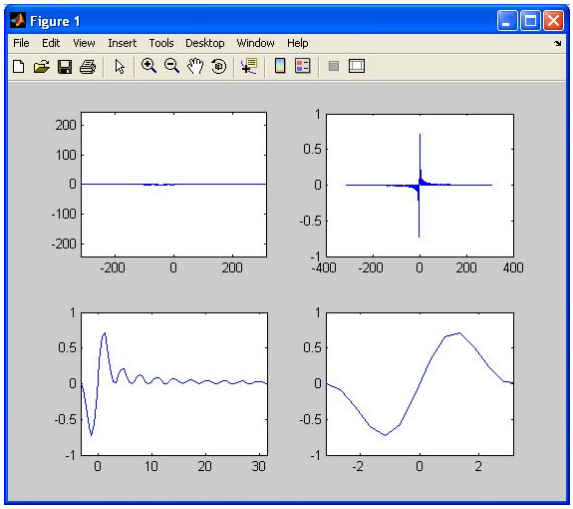
```
h = subplot(...);
```

Για ενεργοποίηση υπογραφήματος με τη βοήθεια χειριστή αντικειμένου:

```
subplot(h);
```

Στο παρακάτω παράδειγμα δημιουργείται ένα τετραπλό γράφημα, με τη συνάρτηση στο πρώτο τμήμα και τρεις λεπτομέρειες στα επόμενα.

```
x=(-100*pi:.5:100*pi);
subplot(2,2,1);
plot(x,(sin(x).^2)./x);
axis equal;
subplot(2,2,2);
plot(x,(sin(x).^2)./x);
subplot(2,2,3);
plot(x,(sin(x).^2)./x);
axis ([-pi 10*pi -1 1]);
subplot(2,2,4);
plot(x,(sin(x).^2)./x);
axis ([-pi pi -1 1])
```



Εικόνα 7.1.1. Η συνάρτηση (sin(x).^2)./x

7.2 Συνδυαστικές μέθοδοι γραφημάτων και η εντολή quiver3

Οι συνδυαστικές μέθοδοι γραφημάτων χρησιμοποιούνται για την αναπαράσταση δεδομένων με ποικίλους τρόπους, ώστε η πληροφορία που το γράφημα μεταδίδει να είναι πληρέστερη.
Ως παράδειγμα συνδυαστικού γραφήματος μπορούμε να αναφέρουμε τη χρήση των εντολών quiver και contour με hold on, όπως φαίνεται στην εικόνα 6.7.1 του προηγούμενου κεφαλαίου.

Επίσης, οι εντολές meshc (παράγραφος 4.2.3) και surfc (παράγραφος 4.2.5) δημιουργούν αυτομάτως μια προβολή ισοϋψών στο χαμηλότερο, κάθετο στον άξονα z, επίπεδο που περιλαμβάνεται στο διάγραμμα.
Η η εντολή quiver3 διατίθεται από την έκδοση 5 του MatLab(R).

```
quiver3(x,y,z,u,v,w)
quiver3(z,u,v,w)
quiver3(...,scale)
quiver3(...,LineSpec)
quiver3(...,LineSpec,'filled')
quiver3(axes_handle,...)
h = quiver3(...)
```

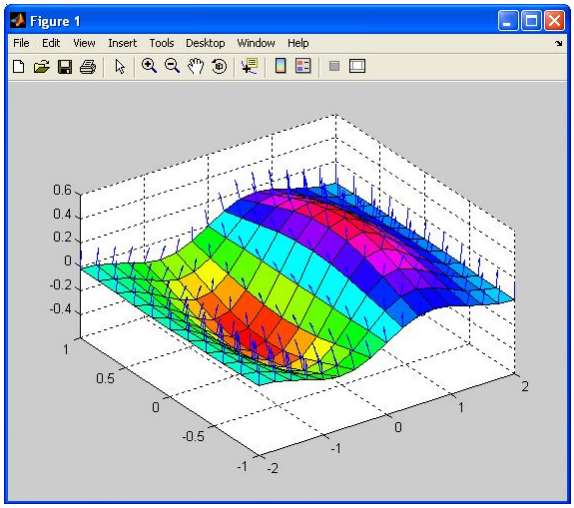
Η quiver3 τυπώνει στο διάγραμμα διανύσματα με συνιστώσες u,v,w, στα σημεία x,y,z.
Αν οι πίνακες x και y παραλειφθούν τυπώνονται διανύσματα στα ύψη που περιέχονται στον πίνακα z και σε ίσες μεταξύ τους αποστάσεις και με κλίμακα που ρυθμίζεται αυτομάτως, ώστε να μην αλληεπικαλύπτονται.

Είναι δυνατόν ο χρήστης να εισάγει την δική του κλίμακα με μια παράμετρο στο τέλος των εισαγομένων (scale), να καθορίσει τον τύπο και το χρώμα των γραμμών (LineSpec ή και “filled”), καθώς και να καθορίσει το σύστημα στο οποίο στοχεύει η εντολή (axes_handle).

Με απόδοση σε μεταβλητή η εντολή επιστρέφει ένα διάνυσμα με χειριστές γραμμής.

Το παρακάτω παράδειγμα δημιουργεί ένα σύνθετο διάγραμμα με μια επιφάνεια χρωματικού χάρτη hsv και αναπαράσταση των κανονικών διανυσμάτων της επιφάνειας.

```
[X,Y] = meshgrid(-2:0.25:2,-1:0.2:1);
Z = X.* exp(-X.^2 - Y.^2);
[U,V,W] = surfnorm(X,Y,Z);
quiver3(X,Y,Z,U,V,W,0.5); hold on;
surf(X,Y,Z); colormap hsv;
view(-35,45); axis ([-2 2 -1 1 -.6 .6]); hold off;
```



Εικόνα 7.2.1. Σύνθετο διάγραμμα της συνάρτησης Z = X.* exp(-X.^2 - Y.^2); με αναπαράσταση των κανονικών διανυσμάτων

Τα κανονικά (normal) διανύσματα δημιουργούνται με την εντολή [U,V,W]=surfnorm(X,Y,Z);

7.3 Έλεγχος της τρισδιάστατης απεικόνισης

Έχουμε ήδη αναφερθεί στην εντολή axis στην παράγραφο 5.1.

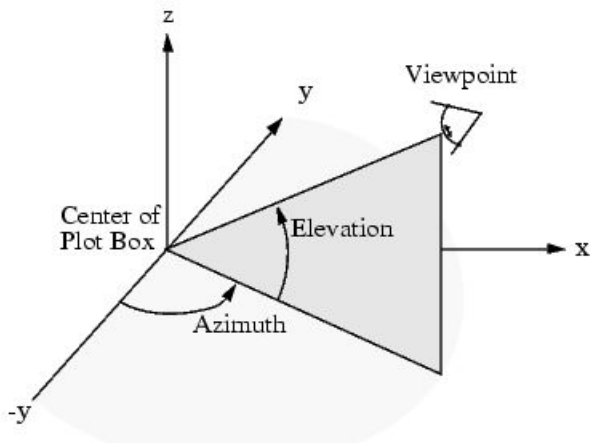
Με την εντολή view μπορούμε να καθορίσουμε την οπτική γωνία από την οποία παρατηρούμε το διάγραμμα.

Η αλλαγή της γωνίας αυτής μπορεί να προσφέρει καλύτερη απεικόνιση, μεταδίδοντας περισσότερη ή και εγκυρότερη πληροφορία.

Η εντολή view δέχεται δύο παραμέτρους σε μοίρες, αζιμούθιο και πλάτος.
Αζιμούθιο είναι η γωνία της οριζόντιας περιστροφής, ενώ πλάτος είναι γωνία ως προς το οριζόντιο επίπεδο. Συντάσσεται ως εξής:

```
view(azimuth,elevation);
view([azimuth elevation]);
view([x,y,z]);
view(2);
view(3);
[az,el] = view;
view(T);
T = view;
```

Η τοποθέτηση του σημείου άποψης του διαγράμματος φαίνεται στο παρακάτω διάγραμμα.



Εικόνα 7.3.1. Το σημείο άποψης όπως ορίζεται από το αζιμούθιο και το πλάτος.

Εναλλακτικά, μπορούν να δοθούν τρεις καρτεσιανές συντεταγμένες, ή με την παράμετρο 2 να δημιουργηθεί μια δισδιάστατη άποψη αντίστοιχη με : az = 0, el = 90, ή με την παράμετρο 3 μια τρισδιάστατη άποψη αντίστοιχη με : az = -37.5, el = 30.
Μπορούμε ακόμη να αναθέσουμε σε μεταβλητές το αζιμούθιο και το πλάτος ενός διαγράμματος. Οι δύο τελευταίες μορφές σύνταξης περιλαμβάνουν μια μήτρα T που παράγεται με την εντολή viewmtx(); και αποτελεί μήτρα μετασχηματισμού. Προφανώς η μήτρα μετασχηματισμού μπορεί να δοθεί ως όρισμα ή να αποσπαστεί από το τρέχον διάγραμμα.

Η εντολή viewmtx(); συντάσσεται ως εξής :

```
T = viewmtx(az,el);
T = viewmtx(az,el,phi);
T = viewmtx(az,el,phi,xc);
```

Η εντολή viewmtx υπολογίζει μια 4x4 μήτρα μετασχηματισμού, ορθογραφική ή προοπτική, η οποία προβάλλει ομογενοποιημένα διανύσματα 4 διαστάσεων σε μια επιφάνεια 2 διαστάσεων, όπως είναι η οθόνη του υπολογιστή. Για παράδειγμα :

```
t=viewmtx(-45,19) [RETURN]

t =

      0.7071      -0.7071      0           0
      0.2302      0.2302       0.9455      0
      -0.6686     -0.6686      0.3256      0
      0           0            0           1.0000
```

Το αποτέλεσμα είναι ανάλογο με την εντολή view, εκτός του ότι δεν μεταβάλλεται η οπτική γωνία του διαγράμματος. Για παράδειγμα η επόμενη εντολή πρώτα περιστρέφει το διάγραμμα και μετά αποδίδει τον πίνακα στην μεταβλητή t.

```
t=view(-45,19) [RETURN]

t =

      0.7071      -0.7071      0           0
      0.2302      0.2302       0.9455      -0.7030
      0.6686      0.6686       -0.3256     8.1545
      0           0            0           1.0000
```

Αν προσθέσουμε μέσα στην παρένθεση την τρίτη παράμετρο phi, δημιουργείται μια προοπτική μήτρα μετασχηματισμού, με την παράμετρο phi να είναι η προοπτική γωνία παραμόρφωσης. Για τις διάφορες τιμές του phi έχουμε :

0 ° Ορθή προβολή
10 ° Τηλεφακός
25 ° Κανονικός φωτογραφικός φακός
60 ° Ευρυγώνιος φωτογραφικός φακός

Μετά την γωνία phi μπορούμε να προσθέσουμε έναν πίνακα τριών συντεταγμένων [x,y,z], ο οποίος να αποτελέσει ένα σημείο εστίασης της άποψης.

Ο πίνακας που δημιουργείται με απόδοση της viewmtx σε μια μεταβλητή μπορεί να δοθεί ως εισαγόμενο στην εντολή view και να αναπαραχθεί η ίδια άποψη: view(T);

Ακολουθεί ένας συνοπτικός κατάλογος εντολών που ελέγχουν την οπτική γωνία.

Εντολές ελέγχου της οπτικής γωνίας της κάμερας:

camdolly	Μετακίνηση της θέσης της κάμερας και της στόχευσής της
camlookat	Επιλογή συγκεκριμένων αντικειμένων προς αναπαράσταση
camorbit	Περιφορά γύρω από το πεδίο στόχευσης
campan	Περιστροφή του πεδίου στόχευσης γύρω από τη θέση της κάμερας
campos	Ορισμός ή απόδοση της θέσης της κάμερας
camproj	Ορισμός ή απόδοση του τύπου προβολής
camroll	Περιστροφή της κάμερας περί του άξονα απεικόνισης
camtarget	Ορισμός ή απόδοση του πεδίου στόχευσης
cameratoolbar	Έλεγχος της εργαλειοθήκης της κάμερας προγραμματιστικώς
camup	Ορισμός ή απόδοση του κατακόρυφου προς τα πάνω διανύσματος
camva	Ορισμός ή απόδοση της οπτικής γωνίας της κάμερας
camzoom	Zoom in και out

view	Ορισμός του σημείου παρατήρησης
viewmtx	Δημιουργία και απόδοση πίνακα μετασχηματισμού άποψης
makehgtform	Δημιουργία πίνακα μετασχηματισμού αντικειμένων

Εντολές καθορισμού της αναλογίας διαστάσεων διαγράμματος και των αξονικών ορίων:

daspect	Ορισμός ή απόδοση δεδομένων αναλογιών εικόνας
pbaspect	Ορισμός ή απόδοση αναλογιών παραλληλογράμμου εκτύπωσης
xlim	Ορισμός ή απόδοση των τρεχόντων ορίων του άξονα x
ylim	Ορισμός ή απόδοση των τρεχόντων ορίων του άξονα y
zlim	Ορισμός ή απόδοση των τρεχόντων ορίων του άξονα z

Εντολές χειρισμού αντικειμένων:

pan	Κύλιση οπτικού παραθύρου ενεργή ή ανενεργή
reset	Επαναφορά αρχικών ρυθμίσεων αξόνων και διαγράμματος
rotate	Περιστροφή αντικειμένων γύρω από δοθέν σημείο και διεύθυνση
rotate3d	Περιστροφή άποψης τρισδιάστατου διαγράμματος αλληλεπιδραστικά
selectmoveresize	Αλληλεπιδραστική επιλογή, κίνηση, ή αλλαγή μεγέθους αντικειμένων
zoom	Μεγέθυνση και σμίκρυνση δισδιάστατου διαγράμματος

Εντολές επιλογής περιοχής ενδιαφέροντος:

dragrect	Μετακίνηση XOR παραλληλογράμμων με το ποντίκι
rbbox	Δημιουργία παραλληλογράμμου με το ποντίκι

7.4 Τρισδιάστατες επιφάνειες από πολύγωνα (trisurf)

Με την εντολή trisurf δημιουργείται μια σειρά από τρίγωνα με κορυφές που επιλέγονται από μια σειρά σημείων ορισμένων στους πίνακες X,Y,Z, με τη βοήθεια του πίνακα Tri.

Ο πίνακας Tri τίθεται πρώτος στη σειρά των ορισμάτων και κάθε γραμμή του περιλαμβάνει τρεις αριθμούς, οι οποίοι υποδεικνύουν ποιες από τις κορυφές ορισμένες στους πίνακες X,Y,Z, θα χρησιμοποιηθούν για κάθε τρίγωνο. Κάθε γραμμή του δηλαδή ορίζει και ένα τρίγωνο.

```
trisurf(Tri,X,Y,Z)
trisurf(Tri,X,Y,Z,C)
trisurf(...'PropertyName',PropertyValue...)
h = trisurf(...)
```

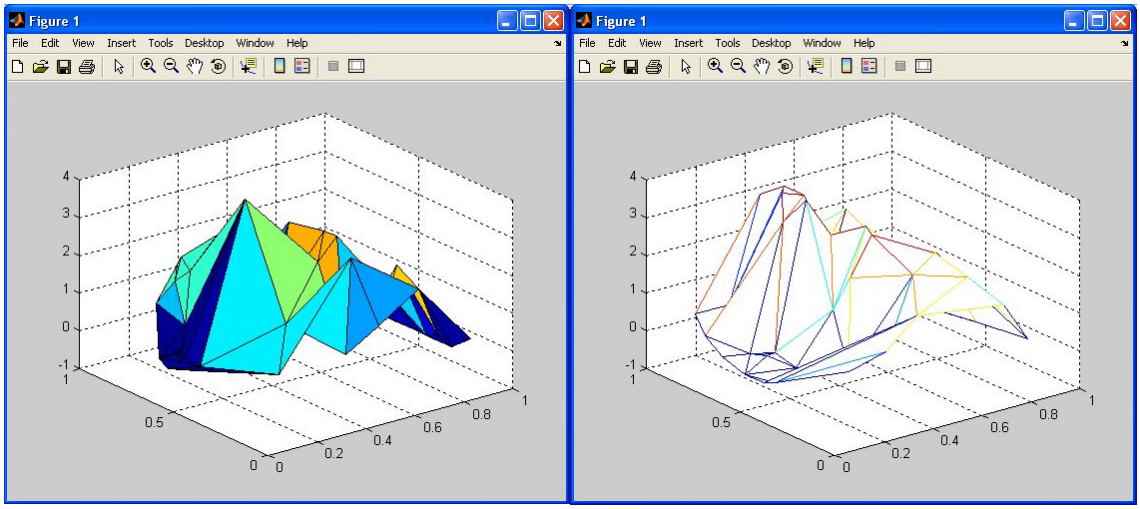
Μετά τους πίνακες X,Y και Z μπορεί να δοθεί ένας πίνακας χρωματικού χάρτη C, κι ακόμη ιδιότητες και τιμές ιδιοτήτων κατά τα γνωστά.

Το παρακάτω παράδειγμα δημιουργεί δύο πίνακες, x, y, με τυχαίες τιμές που χρησιμεύουν για να οριστούν τυχαία σημεία της επιφάνειας peaks.
Στη συνέχεια με τη βοήθεια της συνάρτησης delaunay δημιουργείται τριγωνισμός μεταξύ των σημείων αυτών και αποθηκεύεται στον πίνακα tri. Τέλος, η συνάρτηση trisurf σχηματίζει την επιφάνεια που φαίνεται στο σχήμα 7.4.1 αριστερά.

Αν αντί της trisurf χρησιμοποιήσουμε την εντολή trimesh με τα ίδια ακριβώς εισαγόμενα θα δημιουργηθεί ένα πλέγμα έγχρωμων ευθυγράμμων τμημάτων όπως στο σχήμα 7.4.1 δεξιά.

```
x = rand(1,50);
y = rand(1,50);
z = peaks(6*x-3,6*x-3);
tri = delaunay(x,y);
trisurf(tri,x,y,z);
```

```
x = rand(1,50);
y = rand(1,50);
z = peaks(6*x-3,6*x-3);
tri = delaunay(x,y);
trimesh(tri,x,y,z);
```

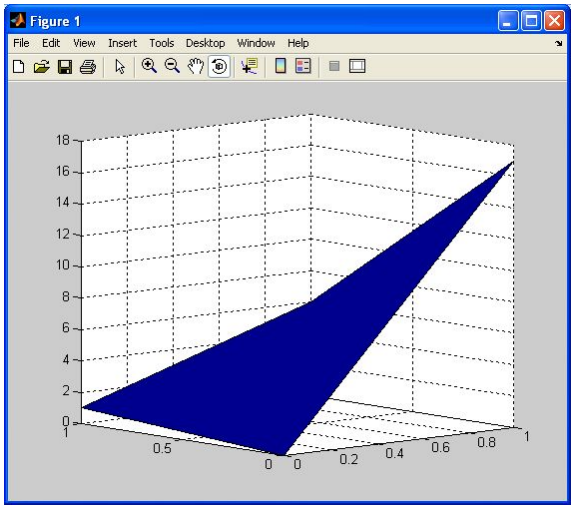


Εικόνα 7.4.1. Η εντολή trisurf(); αριστερά και η trimesh(); δεξιά.

Με την εντολή trisurf μπορούμε να ενώσουμε και περισσότερα των τριών σημεία σχηματίζοντας πολύγωνα.

Οι παρακάτω γραμμές δημιουργούν το τετράπλευρο της εικόνας 7.4.2.

```
x = [0 0 1 1];
y = [0 1 1 0];
z = [0 1 6 17];
trisurf([1 2 3 4 1],x,y,z);
```



Εικόνα 7.4.2. Δημιουργία τετραπλεύρου με την εντολή trisurf();

Με ανάλογο τρόπο μπορούμε να δημιουργήσουμε έγχρωμες τριγωνικές επιφάνειες χρησιμοποιώντας τις εντολές patch ή tetramesh.

Η εντολή delaunay(); δέχεται ως εισαγόμενο τα σημεία που αναπαρίστανται από τους πίνακες x,y, και εξάγει έναν πίνακα (TRI) που περιέχει τριάδες γειτονικών σημείων. Ομαδοποιεί δηλαδή τα εισαγόμενα σημεία σε γειτονικά παρέχοντάς μας τη δυνατότητα να δημιουργήσουμε μια συνεχή επιφάνεια με εντολές που όπως η trisurf.

```
TRI = delaunay(x,y)
TRI = delaunay(x,y,options)
```

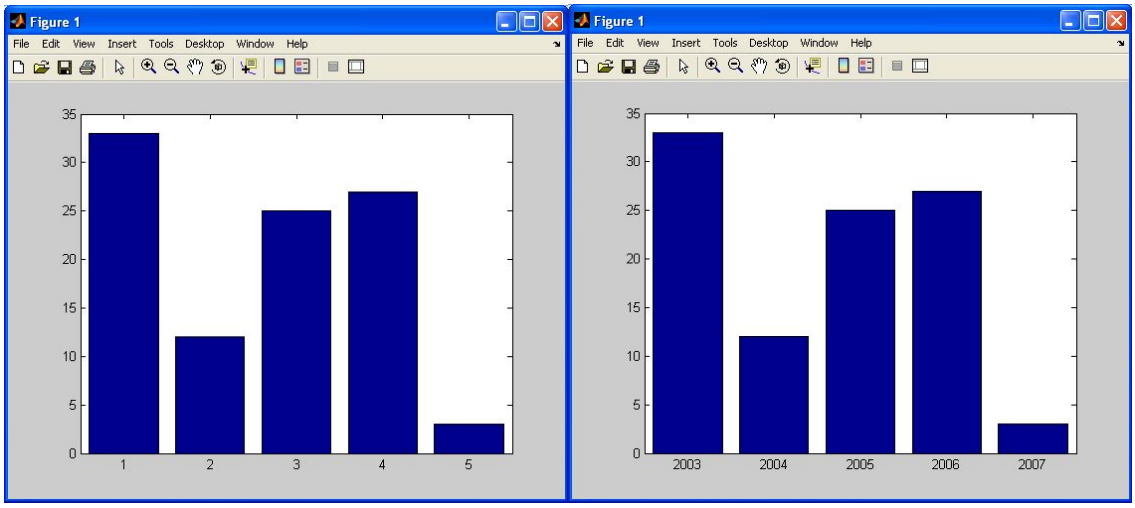
8. ΡΑΒΔΟΓΡΑΜΜΑΤΑ ΚΑΙ ΠΙΤΕΣ

8.1 Η εντολές bar(); barh(); bar3(); και bar3h();

Τα ραβδογράμματα στο MatLab(R) δημιουργούνται με την εντολή bar(); ή barh();
Δέχονται ένα διάνυσμα που περιλαμβάνει τις τιμές των στηλών του ραβδογράμματος.
Η προσαρμογή των ορίων του ραβδογράμματος γίνεται αυτόματα.

Αν επιθυμούμε να δημιουργηθούν οι ράβδοι όχι προς τον δείκτη του μέσα στο διάνυσμα, αλλά προς κάποια άλλα μεγέθη, π.χ. χρονολογίες, πρέπει να εισαγάγουμε στην εντολή bar ένα δεύτερο διάνυσμα ίσου μεγέθους.

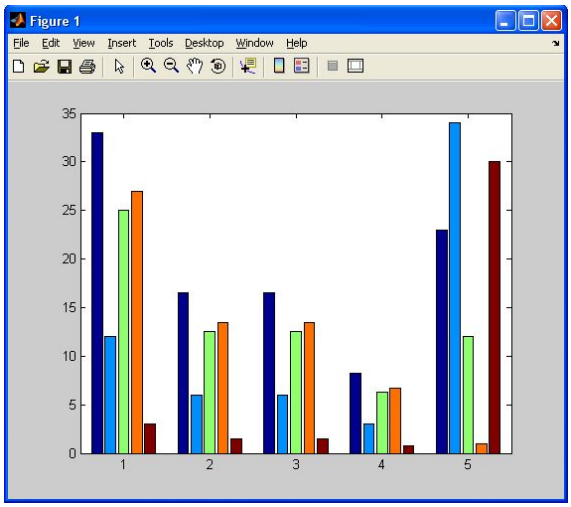
```
x=[2003 2004 2005 2006 2007];
y=[33 12 25 27 3];
bar(y);
bar(x,y);
```



Εικόνα 8.1.1. Τα ραβδογράμματα που δημιουργούν οι εντολές bar(y); και bar(x,y);

Αν ο πίνακας που περιέχει τις τιμές των ράβδων (τα ύψη) είναι πίνακας περισσοτέρων της μίας γραμμών, τότε πρέπει ο αριθμός των γραμμών αυτών να είναι ίσος με το μέγεθος του πίνακα οριζοντίων τιμών, αν χρησιμοποιηθεί.
Κάθε γραμμή του πίνακα τιμών (υψών) των ράβδων δημιουργεί μια ξεχωριστή ομάδα ράβδων, που αντιστοιχίζεται και σε ένα στοιχείο του πίνακα οριζοντίων τιμών.
Για παράδειγμα, οι παρακάτω γραμμές θα δημιουργήσουν το ραβδόγραμμα που ακολουθεί.

```
X = [1 2 3 4 5];
y = [33 12 25 27 3;
      16.5 6 12.5 13.5 1.5;
      16.5 6 12.5 13.5 1.5;
      8.25 3 6.25 6.75 0.75;
      23 34 12 1 3;]

bar(x,y);
```

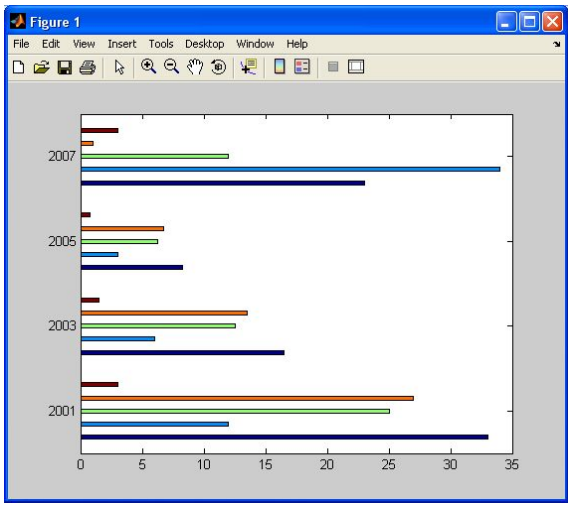


Εικόνα 8.1.2. Ραβδόγραμμα από πίνακα πολλαπλών γραμμών.

Με την εντολή barh(); οι ράβδοι σχηματίζονται οριζόντια.

Εκτός από τους δύο πίνακες, οι εντολές bar(); και barh(); δέχονται και άλλες παραμέτρους.
Δέχονται έναν αριθμό που καθορίζει την απόσταση μεταξύ των ράβδων.
Αν αυτός ο αριθμός είναι 1, τότε οι ράβδοι ακουμπούν μεταξύ τους.
Αν είναι μικρότερος του 1, τότε έχουν μεταξύ τους απόσταση.
Τέλος, αν είναι μεγαλύτερος του 1, τότε οι ράβδοι αλληλεπικαλύπτονται.
Η προκαθορισμένη τιμή είναι 0.8.
Στο παρακάτω οριζόντιο ραβδόγραμμα ο συντελεστής απόστασης των ράβδων είναι 0.3.

```
x = [2001 2003 2005 2007];
y = [33 12 25 27 3; 16.5 6 12.5 13.5 1.5; ...
             8.25 3 6.25 6.75 0.75; 23 34 12 1 3;]

barh(x,y,0.3);
```



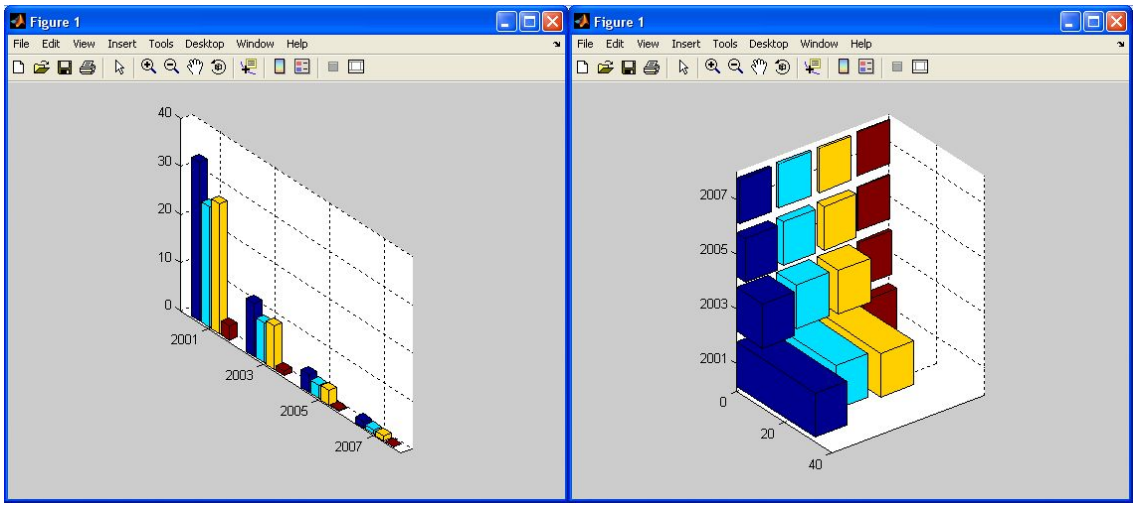
Εικόνα 8.1.3. Οριζόντιο ραβδόγραμμα από πίνακα πολλαπλών γραμμών.

Είναι δυνατόν να δημιουργήσουμε και τρισδιάστατα ραβδογράμματα.
Για να το πετύχουμε αυτό χρησιμοποιούμε τις εντολές bar3(); και bar3h();
Οι εντολές αυτές συντάσσονται ακριβώς όπως και οι αντίστοιχές τους δισδιάστατες.

Ακολουθεί ένα παράδειγμα.

```
x = [2001 2003 2005 2007];
y = [33 25 27 3;
       11 8.3 9 1;
       3.6 2.7 3 0.3;
       1.2 0.9 1 0.1];

bar3(x,y,'grouped');
bar3h(x,y,'detached');
```

Η απεικόνιση της εντολής με την ιδιότητα “grouped” φαίνεται αριστερά, ενώ αυτή με την ιδιότητα “detached”, δεξιά.



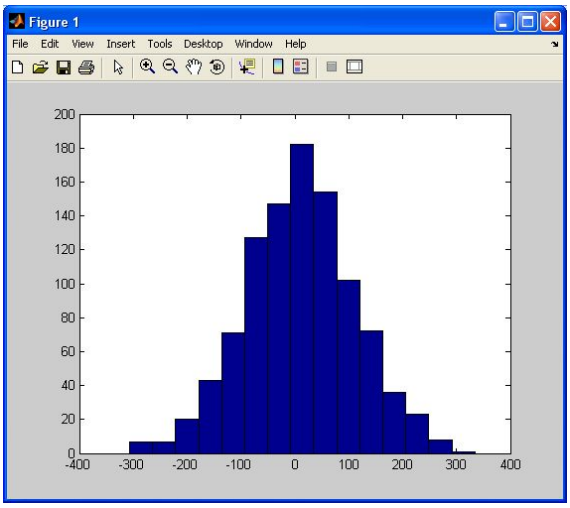
Εικόνα 8.1.4. Η εντολές bar3(); και bar3h();

8.2 Η εντολή hist();

Η εντολή hist(); δημιουργεί ένα ιστόγραμμα που απεικονίζει τη συχνότητα των τιμών μέσα σε δεδομένο δείγμα.
Κατά προκαθορισμένη επιλογή, η hist(); διαμερίζει το δείγμα σε 10 κλάσεις. Αυτό μπορεί να μεταβληθεί.

Για λόγους παρουσίασης της εντολής, θα δημιουργήσουμε πρωτίστως ένα σύνολο τυχαίων τιμών.
Μετά θα περάσουμε το διάνυσμα στην εντολή hist(); και την παράμετρο 15, ώστε να σχηματιστούν 15 κλάσεις.

```
y = fix (100*randn(1,1000));
hist(y,15);
```



Εικόνα 8.2.1. Το ιστόγραμμα κατανομών που δημιουργεί η hist();

Αντί για το πλήθος των κλάσεων μπορούμε να δώσουμε τα κέντρα αυτών των διαστημάτων, χρησιμοποιώντας ένα διάνυσμα που τα περιέχει.
Για παράδειγμα:

```
hist(y,(-1000:100:1000));
```

Το διάνυσμα με τα κέντρα των κλάσεων, για σωστά αποτελέσματα, πρέπει να είναι ισοδιαβαθμισμένο και σε αύξουσα σειρά τιμών, αφού η εντολή hist(); χρησιμοποιεί τις υπηρεσίες της bar();

Αν αποδώσουμε την εντολή hist(); σε δύο μεταβλητές:

```
[h,v] = hist(y);
```

η πρώτη, h, θα περιλάβει ένα διάνυσμα με το πλήθος των στοιχείων που περιέχονται σε κάθε κλάση, και η δεύτερη ένα διάνυσμα με τα κέντρα των κλάσεων.

8.3 Οι εντολές pie(); και pie3();

Οι εντολή pie(); δημιουργεί τη γνωστή από τον κλάδο της στατιστικής “πίτα”.
Δέχεται ως παράμετρο ένα διάνυσμα-γραμμή με τα επί τοις εκατό ποσοστά.
Αν το άθροισμά τους είναι μικρότερο του 100 το υπόλοιπο ισομοιράζεται στα ποσοστά που έχουν δοθεί, ενώ αν είναι μεγαλύτερο αφαιρείται η ίδια ποσότητα από το κάθε ποσοστό, ώστε το άθροισμά τους να γίνει 100.
Μπορούμε να αποδώσουμε χειριστή αντικειμένου διαγράμματος πίτας σε μεταβλητή και να τον χρησιμοποιήσουμε για να αναφερθούμε στο συγκεκριμένο αντικείμενο κατά τα γνωστά.

Η σύνταξη της εντολής έχει ως εξής:

```
pie(X);
pie(X,explode);
```

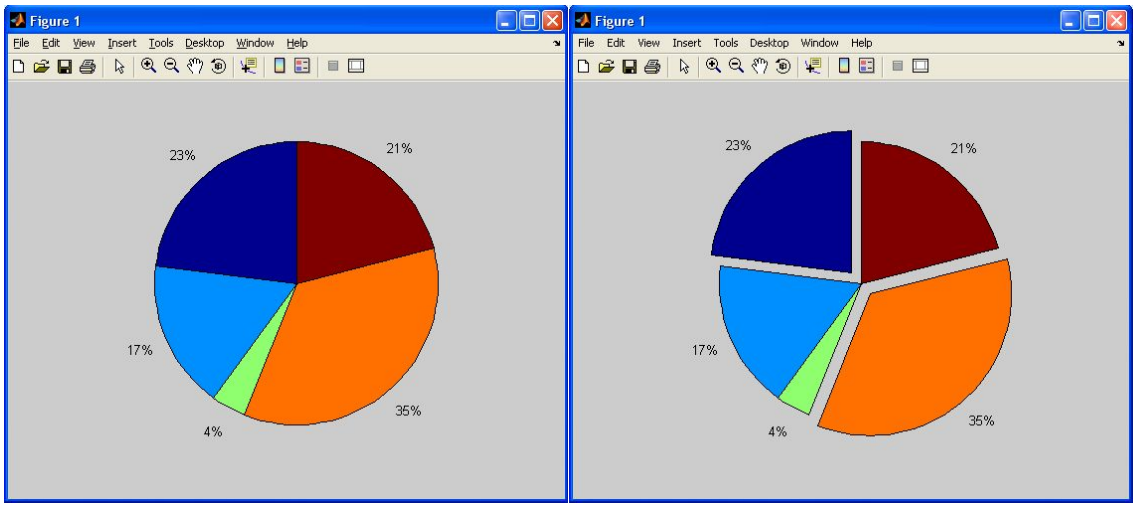
```
pie(...,labels);
pie(axes_handle,...);
h = pie(...);
```

Μια απλή χρήση είναι η ακόλουθη:

```
x = [23 17 4 35 21];
pie(x);
```

Για να δώσουμε έμφαση σε κάποιο κομμάτι της πίτας, μπορούμε να το απομακρύνουμε προς τα έξω. Δημιουργούμε, λοιπόν έναν δεύτερο πίνακα ίσου μεγέθους με τον πίνακα των ποσοστών και όσες τιμές του είναι διάφορες του μηδενός, τα κομμάτια του πίνακα μεριδίων με τον αντίστοιχο δείκτη θα τραβηχτούν πιο έξω από την πίτα.

```
x = [23 17 4 35 21];
e = [1 0 0 1 0];
pie(x,e);
```

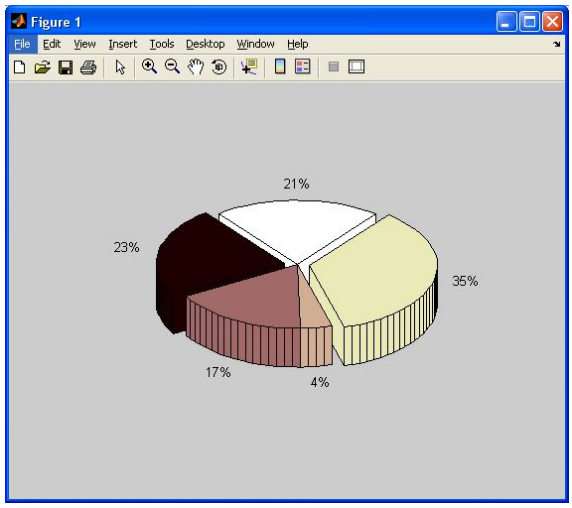


Εικόνα 8.3.1. Διάγραμμα "πίτας" από την pie(x); αριστερά και την pie(x,h); δεξιά.

Χρησιμοποιώντας την εντολή pie3(); ακριβώς με τον ίδιο τρόπο, μπορούμε να δώσουμε μια τρισδιάστατη προοπτική στο σχήμα.
Μπορούμε επίσης, όπως και με την pie(); να επιλέξουμε τον επιθυμητό χρωματικό χάρτη.

```
x = [23 17 4 35 21];
e = [1 0 0 1 0];

pie3(x,e);

colormap pink;
```

Το αποτέλεσμα φαίνεται στην εικόνα 8.3.2.

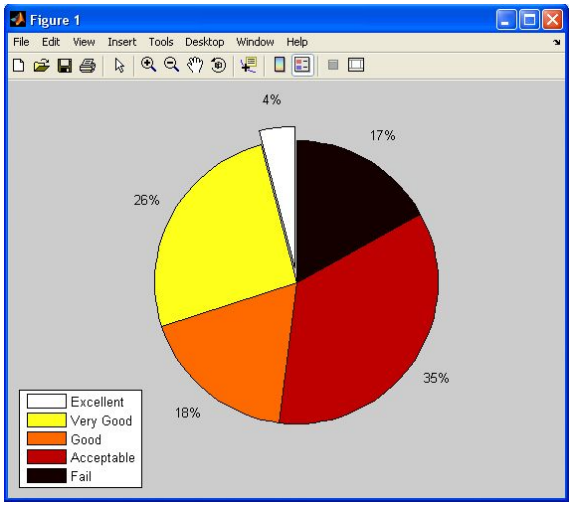


Εικόνα 8.3.2. Διάγραμμα "πίτας" από την pie3(x,h);

Στο επόμενο παράδειγμα φαίνεται η επιλογή χρωματικού χάρτη και η τοποθέτηση υπομνήματος.

```
x = [4 26 18 35 17];
e = [1 0 0 0 0];
h=pie(x,e);
h1=[h(1) h(3) h(5) h(7) h(9)];
hot1=hot(33);
hot2=[hot1(33,:);hot1(25,:);hot1(17,:);hot1(9,:);hot1(1,:)]
colormap(hot2);
legend(h1,'Excellent','Very Good','Good','Acceptable','Fail');
```

Το αποτέλεσμα φαίνεται στην επόμενη εικόνα.



Εικόνα 8.3.2. Η χρήση χρωματικού χάρτη και υπομνήματος με την εντολή pie();

Το δισδιάστατο γράφημα πίτας επιστρέφει δύο χειριστές αντικειμένου για κάθε κομμάτι. Ο πρώτος χρησιμοποιείται για αναφορά σε αυτό.

Με την τέταρτη γραμμή δημιουργούμε ένα διάνυσμα με τους χειριστές αντικειμένου των κομματιών, ώστε να τους εισάγουμε στην legend. Στη συνέχεια τα αλφαριθμητικά με τις ετικέτες αντιστοιχίζονται στους χειριστές αντικειμένου που περιλαμβάνει το διάνυσμα h1.
Με την πέμπτη γραμμή αναπτύσσεται ο χρωματικός χάρτης hot σε 33 διαβαθμίσεις.
Με την έκτη γραμμή αντιστρέφουμε τα χρώματα, ώστε το “Excellent” να συμπέσει με το λευκό και το “Fail” με το μαύρο.

9. ΦΩΤΙΣΜΟΣ ΚΑΙ ΠΡΟΧΩΡΗΜΕΝΑ ΓΡΑΦΙΚΑ

Στο κεφάλαιο αυτό θα δούμε εν συντομία κάποιες από τις συναρτήσεις του MatLab(R) που βοηθάνε τον χρήστη να δημιουργήσει φωτορεαλιστικά εφέ στα διαγράμματά του ή να παίξει με τα χρώματα ή, ακόμη, να δημιουργήσει και να αποθηκεύσει εικόνες.

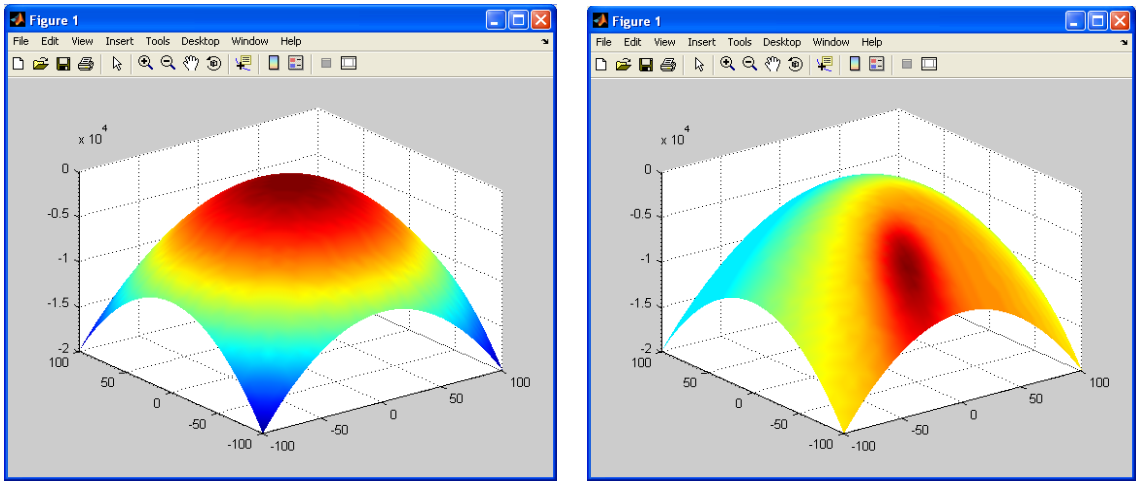
9.1 Φωτισμός επιφανειών τρισδιάστατων γραφικών παραστάσεων

Τα διάφορα τρισδιάστατα αντικείμενα που περιέχονται σε διαγράμματα των συναρτήσεων surf(), shpere(), και άλλων, είναι δυνατόν να φωτιστούν από κάποια πηγή φωτός, τη διεύθυνση της οποίας καθορίζει ο χρήστης, με τη βοήθεια της συνάρτησης surfl();, όπου το L στο τέλος του ονόματός της σημαίνει : light.

Η χρήση του φωτισμού είναι απλή. Απλώς, μέσα στην παρένθεση της εντολής surfl(); και μετά τους τρεις πίνακες x,y,z, παραθέτουμε ένα διάνυσμα-γραμμή, με δύο ή τρία στοιχεία.
Αν το διάνυσμα περιέχει τρία στοιχεία, τότε αυτό αντιπροσωπεύουν τις συντεταγμένες της πηγής φωτισμού στον χώρο.
Αν το διάνυσμα περιλαμβάνει δύο στοιχεία, τότε αυτά αντιπροσωπεύουν αζιμούθιο και πλάτος.

Ακολουθεί ένα παράδειγμα.
Πρώτα κατασκευάζουμε μια επιφάνεια με τη surf(); και κατόπιν με την surfl(); και τη χρήση φωτισμού.

```
[x,y]=meshgrid((-100:5:100),(-100:5:100));
z = -(x.^2+y.^2);
hs=surf(x,y,z);
shading interp;
colormap(Jet);
figure;
[x,y]=meshgrid((-100:5:100),(-100:5:100));
z = -(x.^2+y.^2);
hs=surfl(x,y,z,[0 35]);
shading interp;
colormap(Jet);
```



Εικόνα 9.1.1. Μια επιφάνεια με surf(); (αριστερά) και surfl(); με φωτισμό :[0 35] (δεξιά).

Το MatLab(R) διαθέτει κι άλλες συναρτήσεις που ρυθμίζουν τον φωτισμό, όμως η ανάπτυξή τους, όπως και πολλών άλλων, κείται εκτός εμβέλειας της παρούσης εργασίας.

Στην τεκμηρίωση του MatLab(R) μπορεί ο χρήστης να βρει συναρτήσεις που να δημιουργούν φωτισμό αντανάκλασης (diffuse light), διάχυτο φωτισμό (ambient light), καθώς και φωτισμό προβολέα (specular light).

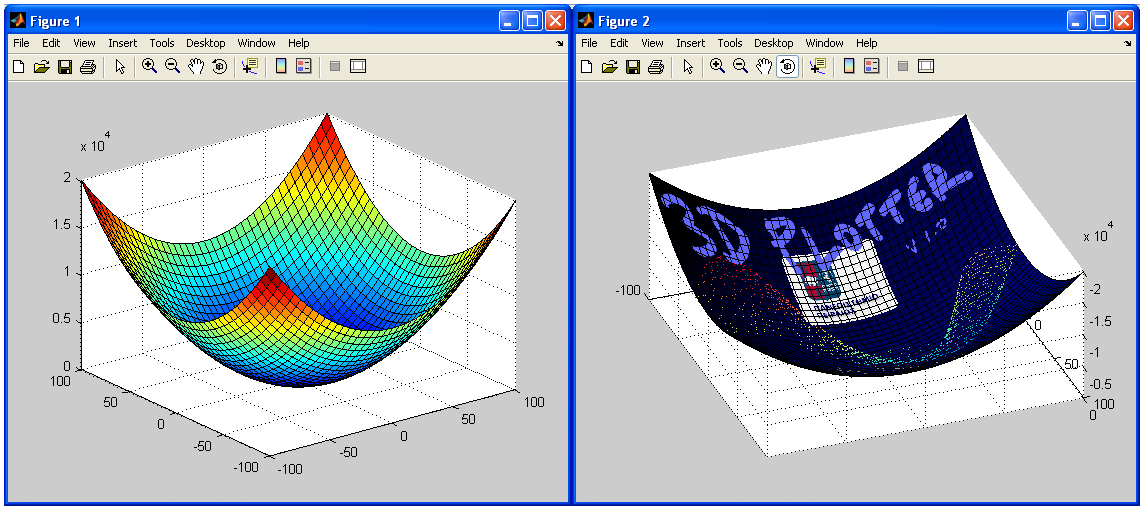
9. 2 Η χρήση φωτογραφιών ως επένδυση επιφανειών

Εκτός από τα colormaps το MatLab(R) επιτρέπει την επένδυση των επιφανειών δια της χρήσεως αρχείων εικόνας.

Η εφαρμογή είναι απλή. Τίθεται η τιμή 'texturemap' στην ιδιότητα 'facecolor' ενός διαγράμματος που σχηματίστηκε με την εντολή surf(); και κατόπιν στην ιδιότητα 'cdata' δίνεται μια μεταβλητή πίνακας που περιέχει τις χρωματικές τιμές των εικονοστοιχείων.
Θα φορτώσουμε τον πίνακα από ένα αρχείο mat το οποίο έχει δημιουργηθεί για την εφαρμογή που αναπτύσσεται στο κεφάλαιο 12. Το αρχείο plogui8.mat περιλαμβάνει μια εικόνα αποθηκευμένη στον πίνακα : a.

```
load plogui8.mat
[x,y]=meshgrid((-100:5:100),(-100:5:100));
h=figure;
hs=surf(x,y,(-(x.^2+y.^2)));
set(hs,'facecolor','texturemap','cdata',a);
```

Το αποτέλεσμα φαίνεται στην εικόνα 9.2.1.

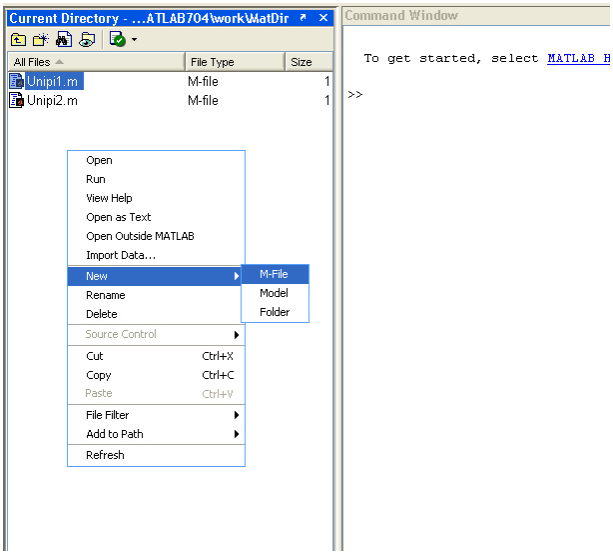


Εικόνα 9.2.1. Η επικόλληση ψηφιακών εικόνων σε τρισδιάστατες επιφάνειες με τη χρήση της ιδιότητας : texturemap.

10. ΔΗΜΙΟΥΡΓΙΑ ΣΥΝΑΡΤΗΣΕΩΝ

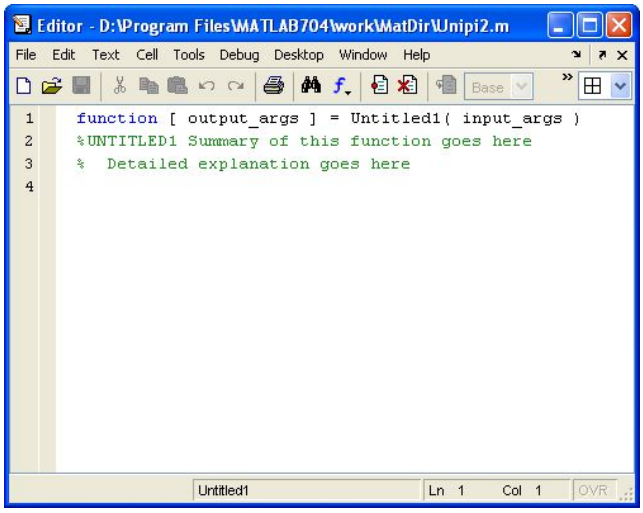
Συνάρτηση είναι μια ρουτίνα που προγραμματίζει ο χρήστης και καταχωρείται σε μορφή αρχείου στον δίσκο του Η/Υ.

Το αρχείο έχει την επέκταση ".m" και ο ευκολότερος τρόπος να το δημιουργήσουμε είναι κάνοντας δεξί κλικ σε κενό χώρο στο παράθυρο "Current Directory", όπως φαίνεται στην επόμενη εικόνα, και επιλέγοντας : New > M-File



Εικόνα 10.0.1. Δημιουργία αρχείου συνάρτησης *.m

Κάνοντας διπλό κλικ στο όνομα του αρχείου στο παράθυρο Current Directory ανοίγει ένα νέο παράθυρο που περιλαμβάνει το βασικό πρότυπο συνάρτησης όπως φαίνεται στην παρακάτω εικόνα :



Εικόνα 10.0.2. Το βασικό πρότυπο μιας συνάρτησης.

Η δήλωση μιας συνάρτησης αρχίζει με την εντολή function, στη συνέχεια δίνονται η μεταβλητή ή οι μεταβλητές που θα παραλάβουν την ή τις τυχών τιμές που επιστρέφει η συνάρτηση, ενώ μετά το ίσον τίθεται το όνομα της συνάρτησης, που πρέπει να συμφωνεί με το όνομα του αρχείου, αν είναι να εκτελείται απευθείας όταν δίνεται στη γραμμή εντολών το όνομα του αρχείου, και, τέλος, παρατίθενται οι μεταβλητές που εισάγονται στην συνάρτηση ως δεδομένα της.

Στη συνέχεια ακολουθούν οι εντολές που επιθυμεί ο χρήστης.
Είναι δυνατόν να οριστούν κι άλλες συναρτήσεις μέσα στο αρχείο, οι οποίες να λειτουργούν ως υποπρογράμματα.

Ως παράδειγμα, παρατίθεται ο παρακάτω κώδικας 10.0.1.

```
% 10.0.1
function [ testout ] = test( inp )
%
    testout = inp;

    func1;
    mx
    func2;
    mx
    if mx==1000
            if inp==123 disp('*******'); end
            inp=inp+1
    else
            inp=inp+2
    end

    inp = mx;

    function func1
        mx = -1000;
    end

    function func2
        mx = 1000;
    end

%
end
```

Όταν καλέσουμε τη συνάρτηση αυτή από τη γραμμή εντολών δίνοντας :

test(123);

θα εκτελεστούν οι εντολές που παρατίθενται παραπάνω, με ανάθεση του αριθμού 123 στη μεταβλητή ins, που περικλείεται από τις παρενθέσεις δίπλα στο όνομα test.

Η πρώτη εντολή μετά την επικεφαλίδα της συνάρτησης αποδίδει το περιεχόμενο της ins, που είναι 123 για το παρόν παράδειγμα, στην μεταβλητή testout.
Επειδή η μεταβλητή testout έχει οριστεί ως μεταβλητή επιστρεφόμενης τιμής, αν δεν αλλάξει περιεχόμενο με το πέρας της εκτέλεσης, η τιμή που έλαβε από την ins θα είναι αυτή που θα επιστρέψει η συνάρτηση.

Στη συνέχεια ακολουθούν λογικοί έλεγχοι με if - else - end και τέλος παρατίθενται δύο άλλες συναρτήσεις, οι func1 και func2, οι οποίες καλούνται κατά τη ροή του προγράμματος, ανάλογα με τα αποτελέσματα των λογικών ελέγχων.

Το παραπάνω παράδειγμα δίνει στη γραμμή εντολών :

```
>> test([123])

mx =

       -1000


mx =

        1000

*******

inp =

   124


ans =

   123
```

11. ΓΡΑΦΙΚΟ ΠΕΡΙΒΑΛΛΟΝ ΔΙΕΠΑΦΗΣ (ΓΠΔ)

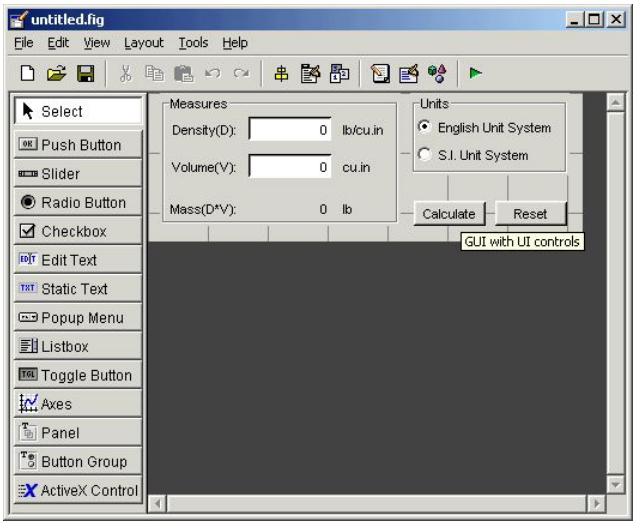
Γραφικό περιβάλλον διεπαφής (graphical user interface - GUI) ονομάζεται μια διεπαφή μεταξύ χρήστη και ηλεκτρονικού υπολογιστή που υλοποιείται με τη βοήθεια γραφικών στοιχείων (user interface controls) και υποπρογραμμάτων που συνδέονται με αυτά και που ανταποκρίνονται σε συγκεκριμένες ενέργειες του χρήστη που εισάγονται μέσω ποντικιού ή αφής ή άλλης συσκευής.

Τα στοιχεία του γραφικού περιβάλλοντος διεπαφής (ΓΠΔ) είναι διαφόρων ειδών :

- Push Button
- Slider
- Radio Button
- Check Box
- Edit Text
- Static Text
- Pop-up Menu
- List Box
- Toggle Button
- Axis
- Panel
- Button Broup
- ActiveX Control

Το MatLab(R) παρέχει στον χρήστη ένα βοηθητικό πρόγραμμα για την εύκολη δημιουργία γραφικών περιβαλλόντων διεπαφής, τον Layout Editor.
Για να εκκινήσει ο χρήστης τον Layout Editor εισάγει GUIDE στη γραμμή εντολών. Στη συνέχεια επιλέγει ένα από το διαθέσιμα πρότυπα ή κενό πρότυπο και οδηγείται στον Layout Editor.

Κοιτάζοντας αριστερά στην επόμενη εικόνα, 11.0.1, μπορούμε να πάρουμε μια πρώτη ιδέα για τη μορφή που έχουν τα στοιχεία ΓΠΔ.



Εικόνα 11.0.1. Ο Layout Editor με τα στοιχεία ΓΠΔ στην στήλη αριστερά.

11.1 Τα στοιχεία του ΓΠΔ

Τα Push Buttons αποτελούν αναπαράσταση ενός πλήκτρου. Όταν ο χρήστης τα πιέσει εκτελείται απευθείας κάποια ενέργεια που έχει προκαθοριστεί.

Τα Sliders είναι ράβδοι μέσα στις οποίες μπορεί να μετακινείται ένας δείκτης καθορίζοντας μια τιμή μέσα σε ορισμένο εύρος τιμών. Η κίνηση του δείκτη μπορεί να γίνει κάνοντας κλικ στα βέλη που υπάρχουν αριστερά και δεξιά (1% μεταβολή) ή μέσα στη ράβδο, ανάμεσα στον δείκτη και το βέλος (10% μεταβολή), ή ακόμη και σύροντας τον δείκτη με το ποντίκι.

Τα Radio Buttons συνδέονται μεταξύ τους σε ομάδες. Όταν ενεργοποιηθεί ένα εξαυτών, τα υπόλοιπα απενεργοποιούνται.

Το στοιχείο Check Box (τετράγωνο επιλογής) ενεργοποιεί ή απενεργοποιεί μια δυνατότητα.

Το Edit Text δημιουργεί ένα πλαίσιο μέσα στο οποίο ο χρήστης μπορεί να εισαγάγει αλφαριθμητικά. Το μέγεθός του μπορεί να μεταβάλλεται, ενώ μπορεί να περιέχει ένα αρχικό αλφαριθμητικό το οποίο ο χρήστης να τροποποιήσει, αν θέλει.

Το Static Text παρέχει τη δυνατότητα εμφάνισης ενός αλφαριθμητικού, χωρίς να επιτρέπεται η μεταβολή του. Χρησιμοποιείται για τη δημιουργία ενδείξεων και μηνυμάτων προς τον χρήστη.

Το Pop-up Menu είναι ένα στοιχείο που παρέχει στο χρήστη τη δυνατότητα να επιλέξει από μια σειρά διαθέσιμων επιλογών που εμφανίζονται όταν κάνει κλικ επάνω του. Για την επιλογή μιας εξαυτών πρέπει να διατηρήσει πιεσμένο το πλήκτρο του ποντικιού, να τοποθετήσει τον δείκτη πάνω της και στη συνέχεια να το απελευθερώσει.

Το List Box παρέχει μια λίστα επιλογών που όμως είναι ορατή διαρκώς. Αν οι καταχωρήσεις είναι περισσότερες, διατίθεται αυτομάτως μία ράβδος κύλισης στο πλάι.

Το Toggle Button έχει δύο καταστάσεις όπως οι διακόπτες, δηλαδή ανοιχτό ή κλειστό, ενεργό ή ανενεργό. Κάνοντας κλικ επάνω του μεταπηδά από τη μία κατάσταση στην άλλη.

Το Axes επιτρέπει την εμφάνιση γραφημάτων και εικόνων μέσα στο ΓΠΔ.

Το Panel χρησιμεύει στην ομαδοποίηση των στοιχείων ΓΠΔ.

Τα Button Groups μοιάζουν με τα Panels, όμως επιτρέπουν την δημιουργία ιδιαίτερων τρόπων με τους οποίους μπορεί κανείς να κάνει επιλογές πατώντας Radio Buttons ή Toggle Buttons.

Τα στοιχεία ActiveX αφορούν το Λ.Σ. των Windows και δίνουν τη δυνατότητα να συμπεριληφθούν στο ΓΠΔ υποπρογράμματα τύπου plug-in.

Τα στοιχεία ΓΠΔ μπορούν να ομαδοποιηθούν και με χρήση των Frames (πλαισίων).

11.2 Οι ιδιότητες των στοιχείων ΓΠΔ

Τα στοιχεία γραφικού περιβάλλοντος διεπαφής ελέγχονται μέσω ιδιοτήτων που μεταξύ άλλων περιλαμβάνουν το είδος, την εμφάνιση, τοποθέτηση κ.α.

Ακολουθεί συνοπτική παράθεση των ιδιοτήτων στοιχείων ΓΠΔ.

Ιδιότητες είδους στοιχείου ΓΠΔ και εμφάνισής του

Όνομα ιδιότητας	Περιγραφή	Εύρος τιμών	Προεπιλογή
BackgroundColor	Το χρώμα φόντου του στοιχείου	ColorSpec	Εξαρτάται από το Λ.Σ.
CData	Εμφάνιση εικόνας (Truecolor) πάνω στο στοιχείο	matrix	
ForegroundColor	Χρώμα γραμματοσειράς	ColorSpec	[0 0 0]
SelectionHighlight	Τονισμός στοιχείου κατά την επιλογή του	on, off	on
String	Ετικέτα στοιχείου ΓΠΔ, καταχωρήσεις στοιχείων list box και pop-up menu	string	
Visible	Ορατό ή μη ορατό στοιχείο	on, off	on

Πίνακας 11.2.1

Γενικές πληροφορίες για το στοιχείο ΓΠΔ

Όνομα ιδιότητας	Περιγραφή	Εύρος τιμών	Προεπιλογή
Children	Τα στοιχεία να μην έχουν παιδιά		
Enable	Ενεργοποίηση στοιχείου	on, inactive, off	on
Parent	Πατρικό στοιχείο	figure, uipanel, or uibuttongroup handle	
Selected	Επιλεγμένο ή μη επιλεγμένο	on, off	off
SliderStep	Το βήμα ενός στοιχείου Slider	two-element vector	[0.01 0.1]
Style	Τύπος στοιχείου ΓΠΔ	pushbutton, togglebutton, radiobutton, checkbox, edit, text, slider, listbox, popupmenu	pushbutton
Tag	Identifier οριζόμενο από τον χρήστη	string	
TooltipString	Content of object's tooltip	string	
Type	Τάξη αντικειμένου γραφικών	string (read-only)	uicontrol
UserData	Δεδομένα παρεχόμενα από τον χρήστη	matrix	

Πίνακας 11.2.2

Εμφάνιση γραμματοσειράς και ετικέτας

Όνομα ιδιότητας	Περιγραφή	Εύρος τιμών	Προεπιλογή
FontAngle	Κλίση γραμματοσειράς	normal, italic, oblique	normal
FontName	Οικογένεια γραμματοσειράς	string	Εξαρτάται από το Λ.Σ.
FontSize	Μέγεθος γραμματοσειράς	size in FontUnits	Εξαρτάται από το Λ.Σ.
FontUnits	Μονάδες μεγέθους γραμματοσειράς	points, normalized, inches, centimeters, pixels	points
FontWeight	Πάχος γραμματοσειράς	light, normal, demi, bold	normal
HorizontalAlignment	Ευθυγράμμιση αλφαριθμητικού ετικέτας	left, center, right	Εξαρτάται από το στοιχείο ΓΠΔ
String	Ετικέτα στοιχείου ΓΠΔ, καταχωρήσεις στοιχείων list box και pop-up menu	string	

Πίνακας 11.2.4

Έλεγχος εκτέλεσης ρουτίνας CallBack

Όνομα ιδιότητας	Περιγραφή	Εύρος τιμών	Προεπιλογή
BusyAction	Διακοπή ρουτίνας Callback	cancel, queue	queue
ButtonDownFcn	Callback ρουτίνα καλούμενη μέσω πίεσης πλήκτρου ποντικιού	string or function handle	
Callback	Εκτελούμενη ενέργεια	string or function handle	
CreateFcn	Ρουτίνα Callback εκτελούμενη κατά τη διάρκεια δημιουργίας αντικειμένου	string or function handle	
DeleteFcn	Ρουτίνα Callback εκτελούμενη κατά τη διάρκεια διαγραφής αντικειμένου	string or function handle	
Interruptible	Δυνατότητα διακοπής της ρουτίνας Callback	on, off	on
KeyPressFcn	Ρουτίνα callback πίεσης πλήκτρου	string or function handle	
UIContextMenu	Αντικείμενο Uicontextmenu συσχετισμένο με το στοιχείο ΓΠΔ	handle	

Πίνακας 11.2.5

Πληροφορίες για την τρέχουσα κατάσταση

Όνομα ιδιότητας	Περιγραφή	Εύρος τιμών	Προεπιλογή
ListboxTop	Δείκτης καταχωρήσεως της λίστας που πρέπει να εμφανιστεί πρώτη στο list box	scalar	[1]
Max	Μέγιστη τιμή (Εξαρτάται από το στοιχείο ΓΠΔ)	scalar	Εξαρτάται από το στοιχείο ΓΠΔ
Min	Ελάχιστη τιμή (Εξαρτάται από το στοιχείο ΓΠΔ)	scalar	Εξαρτάται από το στοιχείο ΓΠΔ
Value	Τρέχουσα τιμή του στοιχείου ΓΠΔ	scalar or vector	Εξαρτάται από το στοιχείο ΓΠΔ

Πίνακας 11.2.6

Έλεγχος πρόσβασης στα στοιχεία ΓΠΔ

Όνομα ιδιότητας	Περιγραφή	Εύρος τιμών	Προεπιλογή
HandleVisibility	Δυνατότητα πρόσβασης στον χειριστή αντικειμένου από τη γραμμή εντολών και τα διάφορα ΓΠΔ	on, callback, off	on
HitTest	Δυνατότητα αυξομείωσης μεγέθους διαμέσου κλικ του ποντικιού	on, off	on

Πίνακας 11.2.7

Τοποθέτηση του στοιχείου ΓΠΔ

Όνομα ιδιότητας	Περιγραφή	Εύρος τιμών	Προεπιλογή
Position	Μέγεθος και τοποθέτηση του στοιχείου ΓΠΔ	position rectangle	[20 20 60 20]
Units	Μονάδες διανύσματος τοποθέτησης	pixels, normalized, inches, centimeters, points, characters	pixels

Πίνακας 11.2.3

Στη συνέχεια περιγράφονται τα βασικότερα εκ των στοιχείων ΓΠΔ.

Η ιδιότητα CallBack περιλαμβάνει μια έκφραση η οποία είτε καλείται είτε υπολογίζεται σαν να είχε δοθεί στη γραμμή εντολών ως όρισμα στην eval :

```
eval(callbackstring);
```

Δηλαδή, αποτελεί την ανταπόκριση του στοιχείου, μόλις ο χρήστης το ενεργοποιήσει με το πάτημα ενός πλήκτρου του ποντικιού ή, αν πρόκειται για στοιχείο Edit Text, με μια ενέργεια που σηματοδοτεί την ολοκλήρωση της εισαγωγής δεδομένων όπως κλικ έξω από το πλαίσιο εισαγωγής κειμένου ή πίεση του πλήκτρου TAB.

Η ιδιότητα CallBack ορισμένων στοιχείων όπως το Static Text δεν υπολογίζεται ποτέ.

Η ιδιότητα SliderStep λαμβάνει ως τιμή ένα διάνυσμα δύο αριθμών. Ο πρώτος φανερώνει το ποσοστό μετακίνησης του κουμπιού ενός στοιχείου Slider όταν ο χρήστης κάνει κλικ σε ένα από τα βελάκια (προκαθορισμένη τιμή 0.01 = 1%), ενώ ο δεύτερος το ποσοστό μετακίνησης του κουμπιού όταν ο χρήστης κάνει κλικ στο αυλάκι, ανάμεσα στο κουμπί και ένα από τα βελάκια (προκαθορισμένη τιμή 0.1 =10%).

Οι ιδιότητες Min, Max και Value είναι σημαντικές για τον καθορισμό της συμπεριφοράς των στοιχείων ΓΠΔ. Ανάλογα με την περίπτωση μπορεί να λαμβάνουν διαφορετικό νόημα.
Κατά την ενεργοποίηση του στοιχείου οι ιδιότητες Min και Max λαμβάνουν την τιμή 0 και 1 αντίστοιχα, ενώ η ιδιότητα Value λαμβάνει μια ακέραια τιμή, έστω κι αν στη συνέχεια είναι δυνατό να της αποδοθεί μια διανυσματική τιμή.
Τα στοιχεία Edit Text δέχονται εισαγωγή μίας γραμμής αν ισχύει : (Max - Min) =< 1 και εισαγωγή πολλών γραμμών αν ισχύει : (Max - Min) > 1.
Τα Push Buttons μεταφέρουν την τιμή Min στην ιδιότητα Value, εκτός από την περίοδο κατά την οποία η CallBack υπολογίζεται, οπότε η ιδιότητα Value λαμβάνει την τιμή Max.
Τα στοιχεία Check Box και Radio Button κατά την απενεργοποίησή τους αποδίδουν την τιμή Min στην ιδιότητα Value, ενώ κατά την ενεργοποίησή τους της αποδίδουν την τιμή Max.
Η ιδιότητα Value φανερώνει ποιά από τις καταχωρήσεις ενός στοιχείου Pop-up Menu είναι επιλεγμένη κάθε στιγμή.
Τα στοιχεία Sliders αποδίδουν την τιμή Min όταν το κουμπί τους βρίσκεται στο αριστερό ή το κάτω άκρο και την τιμή Max όταν βρίσκεται στο δεξί ή πάνω άκρο.
Ωστόσο, από την έκδοση 5 του MatLab(R) επιτρέπεται η ιδιότητα Min να λάβει μεγαλύτερη τιμή από την ιδιότητα Max, οπότε είναι δυνατή ή εναλλαγή του ελαχίστου με το μέγιστο άκρο, αν ο χρήστης το επιθυμεί.
Οι ιδιότητες Min και Max δεν επηρεάζουν τα στοιχεία Static Text και Pop-up Menu.

Η ιδιότητα String χρησιμεύει για τη δημιουργία ετικετών και καταχωρίσεων.
Για τα στοιχεία Push Button, Check Box, Radio Button και Static Text μόνο η πρώτη γραμμή του πίνακα που αποδίδεται στην ιδιότητα έχει σημασία.
Τα στοιχεία Edit Text και Pop-up Menu αξιοποιούν και τις υπόλοιπες γραμμές.

Πρωτίστως δημιουργείται ο πίνακας :

```
string_content = { 'Line1';'Line2';'Line3';'Line4'}; ή

string_content = ['Line1|Line2|Line3|Line4'];
```

και στη συνέχεια αποδίδεται στην ιδιότητα :

```
set(uic_h,'string',string_content);
```

Η ιδιότητα Interruptible μπορεί να ενεργοποιηθεί ή απενεργοποιηθεί. Σε κατάσταση on σημαίνει πως όταν η συνάρτηση CallBack κληθεί, είναι δυνατόν να διακοπεί από άλλη ενέργεια του χρήστη και να συνεχίσει μετά το πέρας της ενέργειας που τη διέκοψε.

Οι υπόλοιπες ιδιότητες αφορούν γενικά την εμφάνιση των στοιχείων ΓΠΔ, της γραμματοσειράς της ετικέτας, την τοποθέτηση και το μέγεθος του στοιχείου στο παράθυρο, τις πληροφορίες που το συνοδεύουν, καθώς και τον έλεγχο επί του στοιχείου, για παράδειγμα αν θα είναι ενεργό ή εμφανές ή αδρανές.

11.3 Δημιουργία βασικών στοιχείων ΓΠΔ

Η σύνταξη που χρησιμοποιούμε έχει ως εξής :

```
handle = uicontrol( handle_of_figure, ...
                                'PropertyName',PropertyValue, ... );
```

Αν παραλείψουμε τον χειριστή αντικειμένου του διαγράμματος handle_of_figure το στοιχείο ΓΠΔ θα προστεθεί στο τρέχον διάγραμμα ή, αν δεν υπάρχει, σε ένα νέο που θα δημιουργηθεί.

Κάθε στοιχείο ΓΠΔ που δημιουργούμε τοποθετείται πάνω από τα προηγούμενα. Άρα, αυτά που βρίσκονται στην αρχή ενός αρχείου συνάρτησης κρύβονται πίσω από τα επόμενα.

Μπορούμε να μεταβάλλουμε τη σειρά τους αλλάζοντας τη διάταξη των χειριστών αντικειμένου των στοιχείων ΓΠΔ στην ιδιότητα Children του διαγράμματος.

Κάθε στοιχείο ΓΠΔ ορίζεται μέσω της ιδιότητας Style που ανάλογα με τον τύπο του στοιχείου παίρνει τις παρακάτω τιμές :

pushbutton, togglebutton, radiobutton, checkbox, popupmenu, slider, edit για το στοιχείο Edit Text, και text, για το στοιχείο Static Text.

Με την παρακάτω γραμμή δημιουργείται ένα Radio Button στο κάτω-αριστερό άκρο ενός νέου διαγράμματος :

h = uicontrol('Style','radio','string','My Radio Button','Position', [10 10 100 20]);

Οι αριθμοί στο διάνυσμα-τιμή της ιδιότητας Position είναι τετμημένη και τεταγμένη με αρχή των αξόνων κάτω-αριστερά, και κατόπιν πλάτος και ύψος : [x y width height].

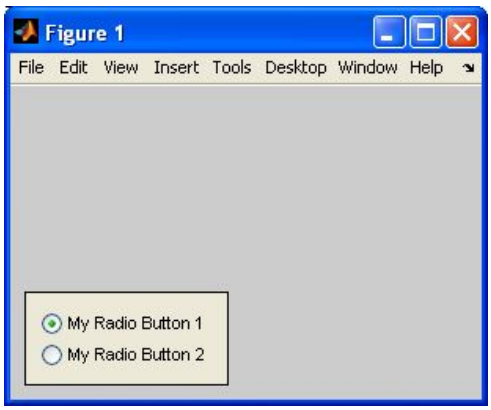
Συνήθως τα Radio Buttons χρησιμοποιούνται για να επιλέγεται ένα μόνο από αυτά κάθε φορά, και για το λόγο αυτόν έχουν ονομαστεί “radio buttons”.
Έτσι είναι προτιμότερο να ομαδοποιούνται οπτικά και λειτουργικά.

Για την οπτική τους ομαδοποίηση χρησιμοποιούμε είτε τα παλιότερα πλαίσια (frames) είτε τα νεότερα πάνελ (panels).

Οι παρακάτω γραμμές δημιουργούν δύο Radio Buttons μέσα σε ένα πάνελ στο κάτω αριστερό άκρο του διαγράμματος.

Το αποτέλεσμα φαίνεται στην εικόνα που ακολουθεί.

```
h0 = figure('Position',[400 400 300 200]);

h1 = uicontrol('Style','frame','Position', [10 10 130 60]);

h2 = uicontrol('Style','radio','string','My Radio Button 1', ...
                                'Position', [20 40 110 20],'Value',1);

h3 = uicontrol('Style','radio','string','My Radio Button 2', ...
                                'Position', [20 20 110 20]);
```



Εικόνα 11.3.1. Οπτικώς ομαδοποιημένα Radio Buttons πάνω σε frame.

Για να ομαδοποιήσουμε τα Radio Buttons με τρόπο που να είναι ενεργό αποκλειστικά ένα κάθε φορά, χρειάζεται να γράψουμε περισσότερο κώδικα. Πρέπει να γίνεται μια αρχικοποίηση της διάταξης των κουμπιών και στη συνέχεια, κάθε φορά που ο χρήστης θα πιέζει ένα κουμπί, η συνάρτηση να καλεί τον εαυτό της και να μεταβάλει το κουμπί που πατήθηκε από ανενεργό σε ενεργό, ενώ όλα τα υπόλοιπα να τίθενται ανενεργά.

Εικόνα 11.3.2. Radio Buttons με αποκλειστική ενεργοποίηση.

Ο κώδικας 11.3.1 που χρειάζεται για να παραχθεί το αποτέλεσμα της εικόνας 11.3.2 έχει ως εξής:

```
% 11.3.1

function RadioGroup(str,bch)

%1
```

```
if nargin < 1
    str = 'initialize';
end

if ~strcmp(str,'initialize')
    handles = get(gcf,'userdata');
    h_radio = handles(1:3);
end

%2

if strcmp(str,'initialize')

    h_fig = figure('position',[200 200 260 230],...
       'resize','off','numbertitle','off','MenuBar','none');
    % Create Frame that covers entire
    h_frame = uicontrol(h_fig,'style','frame',...
                        'position',[50 50 160 130]);
    % Create overall label
    h_stext = uicontrol(h_fig,'style','text',...
                        'string','Exclusive Radio Buttons',...
                        'position',[80 125 100 40]);
    % Create set of 3 Radio buttons
    h_radio(1) = uicontrol(h_fig,'style','radio',...
                       'callback','RadioGroup(''newRB'',1);',...
                        'string','Radio Button 1',...
                        'position',[80 105 100 20],...
                        'value',1);
    h_radio(2) = uicontrol(h_fig,'style','radio',...
                       'callback','RadioGroup(''newRB'',2);',...
                        'string','Radio Button 2',...
                        'position',[80 80 100 20]);
    h_radio(3) = uicontrol(h_fig,'style','radio',...
                       'callback','RadioGroup(''newRB'',3);',...
                        'string','Radio Button 3',...
                        'position',[80 55 100 20]);

    handles = [h_radio];
    set(h_fig,'userdata',handles);

%3

elseif strcmp(str,'newRB')
    str
    num_buttons = length(h_radio);
    button = bch;
    if get(h_radio(button),'value') == 1
      set(h_radio([1:(button-1), (button+1):num_buttons]),'value',0);
    else
      set(h_radio(button),'value',1);
    end

%4

end % END command_str comparison check
```

Στις πρώτες γραμμές (%1-%2) πραγματοποιείται ένας έλεγχος. Αν η συνάρτηση καλείται από τη γραμμή εντολών, καλείται χωρίς παράμετρο. Στην περίπτωση το αλφαριθμητικό λαμβάνει το περιεχόμενο 'initialize' και κατόπιν εκτελείται η διακλάδωση που δημιουργεί για πρώτη φορά τα κουμπιά (%2-%3). Αφού δημιουργηθούν τα κουμπιά, η συνάρτηση ξανακαλείται μόνο εφόσον έχει πατηθεί κάποιο από αυτά. Στην περίπτωση αυτή η συνάρτηση καλείται με δύο παραμέτρους, το αλφαριθμητικό 'newRB' και τον αριθμό 1,2 ή 3, ανάλογα με το κουμπί που πατήθηκε. Κατόπιν εκτελείται το τμήμα (%3-%4), το οποίο ενεργοποιεί το κουμπί στο οποίο ο χρήστης έκανε κλικ και απενεργοποιεί τα υπόλοιπα.

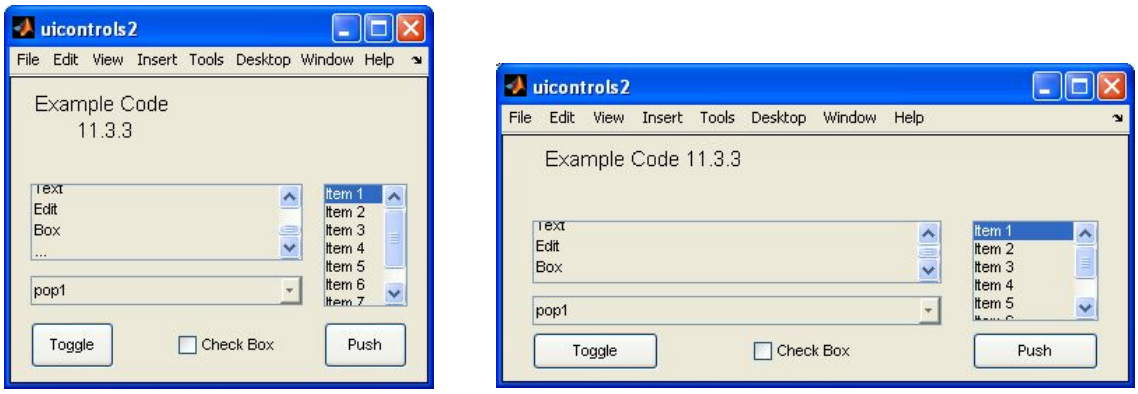
Κατά τη δημιουργία του παραθύρου διαγράμματος, εκτός από 'numbertitle','off' και 'menubar',none', δόθηκε στην ιδιότητα 'resize' η τιμή 'off'. Κατ' αυτόν τον τρόπο ο χρήστης, εκτός από την απουσία τίτλου παραθύρου και βασικού μενού, δεν έχει τη δυνατότητα να μεταβάλλει το μέγεθος του παραθύρου χρησιμοποιώντας το ποντίκι.

Αν θέσουμε αυτή την ιδιότητα στο 'on' θα μπορεί ο χρήστης να μεγαλώσει ή να μικρύνει το μέγεθος του παραθύρου. Τότε όμως τα κουμπιά η άλλα στοιχεία του ΓΠΔ θα εξαφανιστούν. Για να αντιμετωπίσουμε αυτό το φαινόμενο, δίνουμε στην ιδιότητα 'units' των διαφόρων στοιχείων την τιμή 'nomalized'. Στη συνέχεια η τοποθέτηση και το μέγεθός τους ορίζεται ως ποσοστό επί τοις εκατό, εκφρασμένο με έναν δεκαδικό αριθμό από το 0.0 ως το 1.0.

Με τον παρακάτω κώδικα 11.3.2 θα δημιουργηθεί ένα παράθυρο με κάποια στοιχεία ΓΠΔ, το οποίο μπορεί να μεταβληθεί ως προς το μέγεθος με τη χρήση του ποντικιού.

```
% 11.3.2
function out = Resizable ( ins )
figure('position',[150 100 300 220], ...
                        'numbertitle','off','name','uicontrols2');

h_frame = uicontrol('style','frame','units','normalized', ...
    'position',[0 0 1 1]);

h_popum = uicontrol('style','popup','units','normalized', ...
    'string','pop1|pop2|pop3','position',[0.05 0.1 0.65 0.25]);

h_stxt = uicontrol('style','text','units','normalized', ...
    'string','Example Code 11.3.3','position', ...
      [0.05 0.7 0.35 0.25],'fontsize',11);

h_medt =uicontrol('style','edit','units','normalized','string',...
      {'This';'Is';'A';'Multiple';'Line';'Text';'Edit';'Box'}, ...
      'HorizontalAlignment','left', ...
      'position',[0.05 0.4 0.65 0.25],'max',3);

h_tglbtn = uicontrol('style','togglebutton', ...
      'units','normalized','string','Toggle', ...
      'position',[0.05 0.05 0.2 0.15]);

h_chkbx = uicontrol('style','checkbox','units','normalized', ...
    'string','Check Box','position',[0.4 0.05 0.25 0.15]);

h_lstbx = uicontrol('style','listbox','units','normalized', ...
      'string',{'Item 1';'Item 2';'Item 3';'Item 4';'Item 5'; ...
      'Item 6';'Item 7';'Item 8';},'position',[0.75 0.25 0.20 0.4]);

h_pshbtn =uicontrol('style','pushbutton','units','normalized', ...
    'string','Push','position',[0.75 0.05 0.20 0.15]);
```

Ο πιο πάνω κώδικας 11.3.2 δημιουργεί το αποτέλεσμα που φαίνεται στην εικόνα 11.3.3.



Εικόνα 11.3.3. Ένα ΓΠΔ που περιλαμβάνει τα normalized στοιχεία : Static Text, Edit Box, List Box, Pop-up Menu, Toggle Button, Check Box και Push Button, πριν (αριστερά) και μετά (δεξιά) από μεταβολή μεγέθους του παραθύρου από το χρήστη.

Αν ο χρήστης μικρύνει πολύ το παράθυρο, κάποιες από τις ετικέτες των στοιχείων αλληλεπικαλύπτονται ή εξαφανίζονται κατά ένα μέρος τους. Παρόλα αυτά, παραμένει κάποιο τμήμα εμφανές, ώστε να καταλαβαίνει πως αν μεγεθύνει το παράθυρο θα παρουσιαστεί περισσότερη πληροφορία.

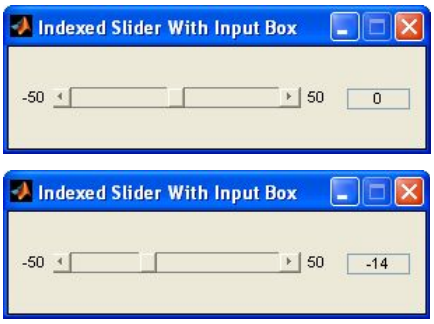
Τα Sliders είναι το τελευταίο στοιχείο ΓΠΔ που θα εξετάσουμε σ' αυτή την ενότητα. Πρόκειται, όπως έχει ήδη αναφερθεί για ένα αυλάκι που περιέχει ένα κουμπί. Το κουμπί αυτό μπορεί και κινείται δεξιά - αριστερά ή πάνω - κάτω, σύμφωνα με τη διεύθυνση του άξονα του Slider. Η τιμή (value) του στοιχείου μεταβάλλεται ανάλογα με τη θέση του κουμπιού μέσα στο αυλάκι.
Αν θέλουμε να λαμβάνουμε αριθμητική ένδειξη για την τιμή του Slider κάθε φορά που μετατοπίζουμε το κουμπί της, πρέπει να γράψουμε επιπλέον κώδικα, ώστε να συνδέσουμε το Slider με ένα στοιχείο Static Text.

Με τον κώδικα 11.3.3 που ακολουθεί δημιουργείται ένα Slider με συνοδεία αριθμητικών ενδείξεων.

```
% 11.3.3

function sldr(str)

%1

if nargin < 1
    str = 'initialize';
end

if ~strcmp(str,'initialize')
    handles = get(gcf,'userdata');
    h_sldr = handles(1);
    h_val = handles(2);
end
```

```
%2

if strcmp(str,'initialize')

    h_fig = figure('position',[100 200 300 75],...
        'resize','off','numbertitle','off',...
        'name','Indexed Slider With Input Box','MenuBar','none');

    h_frame = uicontrol(h_fig,'style','frame',...
        'units','normalized','position',[0 0 1 1]);

    h_sldr = uicontrol(h_fig,...
        'callback','sldr(''SldrBtnM'');','style','slider', ...
        'min',-50,'max',50,'units','normalized', ...
        'position',[0.10 0.4 0.6 0.2]);

    h_min = uicontrol(h_fig,'style','text',...
        'string',num2str(get(h_sldr,'min')),...
        'units','normalized','position',[0.01 0.39 0.1 0.2]);

    h_max = uicontrol(h_fig,'style','text',...
        'string',num2str(get(h_sldr,'max')),...
        'units','normalized','position',[0.71 0.39 0.05 0.2]);

    h_val = uicontrol(h_fig,...
        'callback','sldr(''VCh'');','style','edit',...
        'string',num2str(get(h_sldr,'value')),...
        'units','normalized','position',[0.81 0.39 0.15 0.2]);

    handles = [h_sldr h_val];
    set(h_fig,'userdata',handles);

%3

elseif strcmp(str,'VCh')
    UsrVal = str2num(get(h_val,'string'));
    if ~length(UsrVal)
     UsrVal = (get(h_sldr,'max')+get(h_sldr,'min'))/2;
    end
    UsrVal = min([UsrVal get(h_sldr,'max')]);
    UsrVal = max([UsrVal get(h_sldr,'min')]);
    set(h_sldr,'value',UsrVal);
    set(h_val,'string',num2str(get(h_sldr,'value')));

%4

elseif strcmp(str,'SldrBtnM')
    set(h_val,'string',num2str(round(get(h_sldr,'value'))));

end

%5
```

Το αποτέλεσμα του παραπάνω κώδικα 11.3.3 φαίνεται στην εικόνα 11.3.4 που ακολουθεί.



Εικόνα 11.3.4. Sliders με ένδειξη μεγίστου και ελαχίστου, καθώς και επιλεγμένης τιμής. Ο χρήστης μπορεί να εισάγει την νέα τιμή στο τετράγωνο δεξιά.

Ο παραπάνω κώδικας ξεκινά (`%1-%2`) ελέγχοντας αν είναι η πρώτη φορά που εκτελείται η συνάρτηση. Σε αυτή την περίπτωση εκτελείται το τμήμα (`%2-%3`) που αναλαμβάνει την κατασκευή των στοιχείων ΓΠΔ.

Αν μετακινηθεί το κουμπί του Slider τότε η συνάρτηση καλεί τον εαυτό της με παράμετρο το αλφαριθμητικό 'SldrBtnM'. Αυτή η παράμετρος αυτή τη φορά γίνεται αιτία να εκτελεστεί το τμήμα (`%4-%5`), το οποίο αλλάζει την τιμή του στοιχείου Edit Text.

Η αλλαγή της τιμής του Edit Text προκαλεί για άλλη μια φορά την κλήση της συνάρτησης με παράμετρο 'VCh'. Έτσι, εκτελείται το τμήμα (`%3-%4`) το οποίο ελέγχει αν η τιμή είναι επιτρεπτή, οπότε τίθεται στο τετράγωνο Edit Text, ή δεν είναι επιτρεπτή, οπότε και επιλέγεται το κοντινότερο άκρο το διαστήματος τιμών.

Ο χρήστης μπορεί να αλλάξει απευθείας την τιμή του τετραγώνου Edit Text, οπότε παράγεται κλήση με παράμετρο 'VCh'. Κατά την εκτέλεση όμως του τμήματος (`%4-%5`), εκτός από τον έλεγχο και την επανεγγραφή της ένδειξης του Edit Text, τίθεται και στο Slider η νέα τιμή.

12. ΜΕΛΕΤΗ ΠΕΡΙΠΤΩΣΗΣ (CASE STUDY) - 3D PLOTING

Ως μελέτη περίπτωσης στα πλαίσια της μεταπτυχιακής αυτής εργασίας, θα παρουσιάσουμε την κατασκευή ενός γραφικού περιβάλλοντος διεπαφής που συνδεδεμένο με ορισμένες συναρτήσεις θα μπορεί να δημιουργήσει μια τρισδιάστατη γραφική παράσταση της δοσμένης μαθηματικής συνάρτησης δύο αγνώστων.

Το πρόγραμμα αυτό δεν θα κάνει χρήση των έτοιμων συναρτήσεων του MatLab(R) για γραφική απεικόνιση. Στόχος είναι η διερεύνηση της θεμελιώδους λογικής που αξιοποιείται προκειμένου να παραχθεί μια τρισδιάστατη γραφική παράσταση.

12.1 Το πρόγραμμα Plotter

Θέλουμε να δημιουργήσουμε έναν υπολογιστικό πυρήνα που θα δέχεται τα δεδομένα του χρήστη, όπως για παράδειγμα μια εξίσωση της μορφής f(x,y), και θα παράγει μια εικόνα δύο διαστάσεων, η οποία θα απεικονίζει το τρισδιάστατο γράφημα της συνάρτησης.

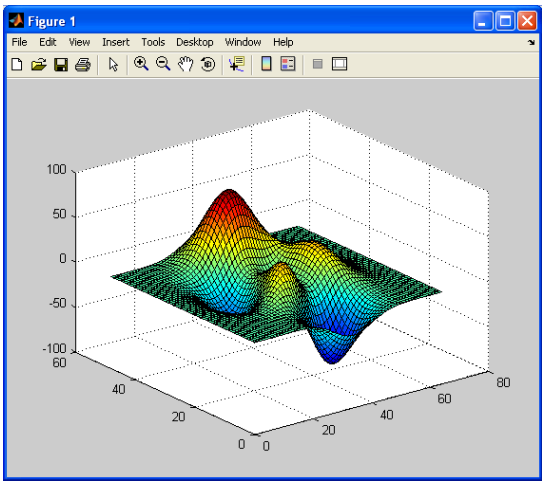
Θα ξεκινήσουμε διαμορφώνοντας τον υπολογιστικό πυρήνα ο οποίος θα δημιουργεί τα δεδομένα της προς εκτύπωση εικόνας.

Θα χρησιμοποιήσουμε την παρακάτω συνάρτηση, που είναι η γνωστή μας peaks(x,y) :

```
function [z] = zf (x,y)
      z =3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) ...
            - 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) ...
          - 1/3*exp(-(x+1).^2 - y.^2);
end
```

Μια αυτοματοποιημένη γραφική παράσταση της συνάρτησης αυτής ΄μέσω των εντολών του MatLab(R), meshgrid και surf(), φαίνεται στην επόμενη εικόνα :

```
[x,y]=meshgrid(-3.2:0.1:3.2,-2.4:0.1:2.4);
z =10 .* (  3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) ...
      - 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) ...
      - 1/3*exp(-(x+1).^2 - y.^2) );
surf(z);
```



Εικόνα 12.1.1. Η συνάρτηση peaks(x,y).

Για ευκολία και ταχύτητα υπολογισμών θα επιλέξουμε να δουλέψουμε με ακέραιες τιμές.
Το πεδίο τιμών λοιπόν της f(x,y) θα είναι για τα x: -320 .. 320, και για τα y: -240 .. 240.

Για κάθε ζεύγος (x,y) πρέπει να υπολογίζεται το z=f(x,y) και για κάθε τριάδα (x,y,z) πρέπει να τυπώνεται ένα σημείο, έστω με τη συνάρτηση: drawpointf(x,y,z);

Σύμφωνα με τα προηγούμενα, δημιουργούμε το παρακάτω βρόχο:

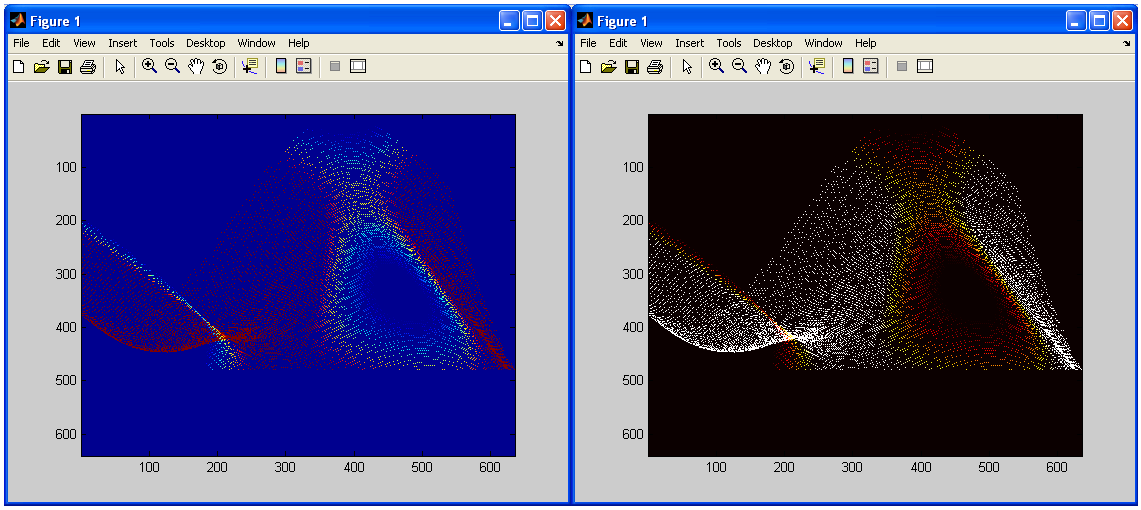
```
for y = -240:s:240
      for x = -320:s:320
            z = fix (100 * getzf(x/350,y/350));
            drawpointf(x,y,z);
     end
end
```

όπου getzf() είναι η συνάρτηση zf() ή peaks() του MatLab(R) που παρουσιάστηκε στην προηγούμενη σελίδα.

Μια πρώτη εκδοχή της συνάρτησης drawpointf(x,y,z) είναι η παρακάτω :

```
function drawpointf(dpx,dpy,dpz)
     a=20;

     yi=fix(cos(a)*dpy); yj=fix(sin(a)*dpy);
     fx=320+yi+dpx;
      fy=240+yj+dpz;

     if (fx>0 & fy>0 & fx<640 & fy<480)
      img(fy,fx)= dpz;
      end
end
```

η οποία παράγει την παρακάτω εικόνα :



Εικόνα 12.1.2. Η συνάρτηση drawpointf(x,y,z) με colormap jet αριστερά και hot δεξιά.

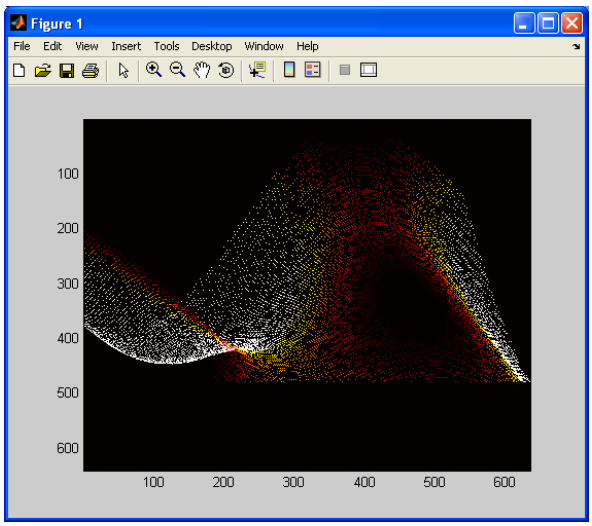
Η συνάρτηση αυτή δέχεται τρεις συντεταγμένες (dpx,dpy,dpz) και αφού πραγματοποιήσει έναν απλό τριγωνομετρικό υπολογισμό, ώστε να δημιουργηθεί αξονομετρική προοπτική (yi=...yj=...), απεικονίζει τα τρισδιάστατα σημεία σε δισδιάστατη μορφή μέσα στον πίνακα img, με τιμή που αντιστοιχεί στο z=f(x,y) του κάθε σημείου (img(fy,fx)=dpz;).
Ο έλεγχος if ... end γίνεται ώστε να μη επιχειρηθεί εγγραφή σε δείκτες εκτός του πίνακα, καθότι το MatLab(R) θα παράξει μήνυμα λάθους και θα διακοπεί το πρόγραμμα.
Η αξονομετρική γωνία ορίζεται ως a=20, ώστε να μπορεί αργότερα να παραμετροποιηθεί.

Όπως φαίνεται και στην εικόνα, δεν υπάρχει μεγάλη ευκρίνεια. Δηλαδή, δεν μπορούμε να δούμε καθαρά ποια σημεία είναι μπροστά και ποια σημεία είναι πίσω. Για την ακρίβεια, τα σημεία που βρίσκονται πίσω από άλλα πρέπει να μην απεικονιστούν, αλλά να παραλειφθούν.
Για να το κάνουμε αυτό δημιουργούμε και αρχικοποιούμε δύο πίνακες. Έναν για να σώζουμε τα σημεία που πήρανε μέγιστο z (mx) και έναν για να σώζουμε τα σημεία που πήρανε ελάχιστο z (mn). Έτσι θα μπορούμε να επιλέγουμε να τυπώσουμε μόνο όσα σημεία είναι ορατά.

Βελτιώνουμε λίγο τον κώδικα της συνάρτησης drawpointf(x,y,z) και έχουμε τα εξής :

```
mx=(0:639);
mx(:)=-10000;
mn=(0:639);
mn(:)=10000;
...
function drawpointf(dpx,dpy,dpz)
      a=20;
      yi=fix(cos(a)*dpy); yj=fix(sin(a)*dpy);
      fx=320+yi+dpx; fy=240+yj+dpz;

      if (fx>0 & fy>0 & fx<640 & fy<480)
            if (fy >= mx(fx)) | (fy <= mn(fx))
            img(fy,fx)= dpz;
          end
     end
end
```

Ο κώδικας αυτός παράγει την εικόνα 12.1.3 που φαίνεται αμέσως πιο κάτω.



Εικόνα 12.1.3. Ελαφρά βελτίωση με απόκρυψη των πίσω σημείων.

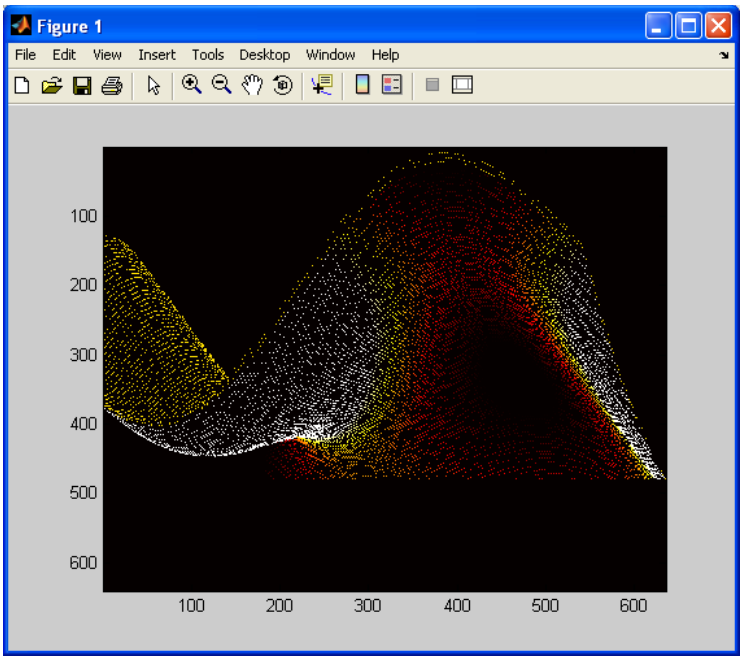
Για να αναβαθμίσουμε το αποτέλεσμα εισάγουμε μια επιπλέον τεχνική : τη δημιουργία σκιάς. Αυτό σημαίνει πως τα σημεία που βρίσκονται στην κάτω επιφάνεια της συνάρτησης θα μετατοπιστούν χρωματικά προς κατώτερες τιμές του χρωματικού χάρτη, κατά ένα σταθερό διάστημα.
Μετά από δοκιμές επιλέχθηκε το διάστημα να είναι 90 μονάδες.

Ο κώδικας λοιπόν λαμβάνει την εξής μορφή :

```
function drawpointf(dpx,dpy,dpz)
       sh = 0; a=20;
      yi=fix(cos(a)*dpy); yj=fix(sin(a)*dpy);
      fx=320+yi+dpx; fy=240+yj+dpz;

      if (fx>0 & fy>0 & fx<640 & fy<480)
       if (fy > mx(fx)) mx(fx) = fy; sh=0; end
            if (fy < mn(fx)) mn(fx) = fy; sh=dpz-90; end

            if (fy >= mx(fx)) | (fy <= mn(fx))
              img(fy,fx)= dpz - sh;
            end
      end
end
```

Με τους δύο επιπλέον ελέγχους if ... end ορίζεται η χρωματική πτώση του σημείου με τη βοήθεια της μεταβλητής sh και στη συνέχεια αφαιρείται από την τιμή dpz η οποία καταγράφεται στον πίνακα της οθόνης img.

Στην παρακάτω εικόνα φαίνεται το αποτέλεσμα της σκίασης της κάτω επιφάνειας του γραφήματος.



Εικόνα 12.1.4. Βελτίωση της απεικόνισης με χρήση σκίασης στην κάτω επιφάνεια.

Για να βελτιώσουμε την οθόνη θα δημιουργήσουμε μια χωριστή συνάρτηση οθόνης, την showimg, την οποία θα καλούμε από το κύριο πρόγραμμα που θα ονομάσουμε plotter.

Έστω ότι η οθόνη μας έχει διαστάσεις 640x480. Θα αναπαραστήσουμε την οθόνη αυτή με έναν πίνακα διαστάσεων 640x480.

```
img=zeros(480,640);
```

Με την εντολή αυτή δημιουργείται ο πίνακας των 640x480 στοιχείων, τα οποία αρχικοποιούνται στην τιμή 0.

Στη συνέχεια δημιουργούμε ένα παράθυρο διαγράμματος και εκτείνουμε την επιφάνεια εκτύπωσης ώστε να καταλάβει όλη την περιοχή του παραθύρου.
Η δημιουργία του παραθύρου γίνεται αυτόματα, με την εντολή αυτή :

```
set(gca, 'Position', [0 0 1 1]);
```

Για να τοποθετήσουμε το παράθυρο στην οθόνη του υπολογιστή χρησιμοποιούμε τις εντολές :

```
[m,n,p] = size(img);
set(gcf,'Units', 'pixels', 'Position', [40 40 n m]);
```

Λαμβάνουμε τις διαστάσεις της οθόνης img αντί να τις παράσχουμε με σταθερές, ώστε αργότερα να μπορούμε να χειριστούμε οθόνες διαφόρων μεγεθών, αν το θελήσουμε.

Θέτουμε τους άξονες του διαγράμματος να είναι equal και τους αποκρύπτουμε ως εξής :

```
axis equal;
axis off;
```

Για να τυπώσουμε την πρώτη μας οθόνη χρησιμοποιούμε την εντολή image που εξετάσαμε στο κεφάλαιο 9.

```
h=image(img);
```

Για να ελέγξουμε τα χρώματα του διαγράμματος χρησιμοποιούμε εντολές χειρισμού του χρωματικού χάρτη.

```
cm=colormap(bone(128));
```

Ορίζουμε τα χαρακτηριστικά του παραθύρου, για παράδειγμα εξαφανίζουμε το μενού, επιλέγουμε τη μορφή του δείκτη του ποντικιού, καθορίζουμε τον χρωματικό χάρτη κ.τ.λ.

```
cm=colormap(bone(128));

set(gcf,'BackingStore','on','ButtonDownFcn','mbimg', ...
                        'DockControls','off','DoubleBuffer','on');

set(gcf,'MenuBar','none','Pointer','crosshair', ...
                                  'colormap',cm,'color',[.4 .4 .4]);
```

Τέλος, τα τοποθετούμε όλα σε ένα m-file που ονομάζουμε showimg και χρησιμοποιούμε δύο παραμέτρους ως εισαγόμενα : Τον πίνακα img και έναν πίνακα χρωματικού χάρτη, ώστε να μπορούμε να επιλέξουμε εμείς τα χρώματα του διαγράμματος.

Ολόκληρος ο κώδικας 12.1.1 της συνάρτησης showimg έχει ως εξής :

```
% 12.1.1
function [h] = showimg (img,mp)
%
cm=colormap(bone(128));

set(gcf,'BackingStore','on','ButtonDownFcn','mbimg', ...
                        'DockControls','off','DoubleBuffer','on');

set(gcf,'MenuBar','none','Pointer','crosshair', ...
                                    'colormap',cm,'color',[.4 .4 .4]);

hold on;
[m,n,p] = size(img);
set(gca, 'Position', [0 0 1 1]);
set(gcf,'Units', 'pixels', 'Position', [40 40 n m]);

axis equal;
axis off;

h=image(img);
mp=colormap(mp);
colormap(mp);

%
end
```

Αυτό που χρειαζόμαστε τώρα είναι να συνδέσουμε την showimg με το κύριο πρόγραμμα που παράγει την προς εκτύπωση οθόνη img. Ο πίνακας img περιλαμβάνει το διάγραμμα και πρέπει να μεταβιβάζεται στην συνάρτηση showimg μαζί με τον χρωματικό χάρτη που επιλέγουμε κάθε φορά.

Προσθέτουμε την παρακάτω λίστα μετά την απόδοση των τιμών στον πίνακα z και την οθόνη img.

```
mnc=min(img(:));
for c=1:length(img)
    if (img(c)>=1) img(c)=img(c)- mnc; end
end
mxc=max(img(:));
img=fix(img./(mxc/180));

showimg(img,'hot');
```

Ο βρόχος for ... end τοποθετεί τον χρωματικό χάρτη καλύτερα, ώστε να βλέπουμε όλο του το φάσμα.
Το 180 είναι ένας συντελεστής που ρυθμίζει το μέγιστο χρώμα. Αν θέταμε 130 δέν θα βλέπαμε λευκά σημεία στις κορυφές για χρωματικό χάρτη 'hot' αλλά μόνο κίτρινα.

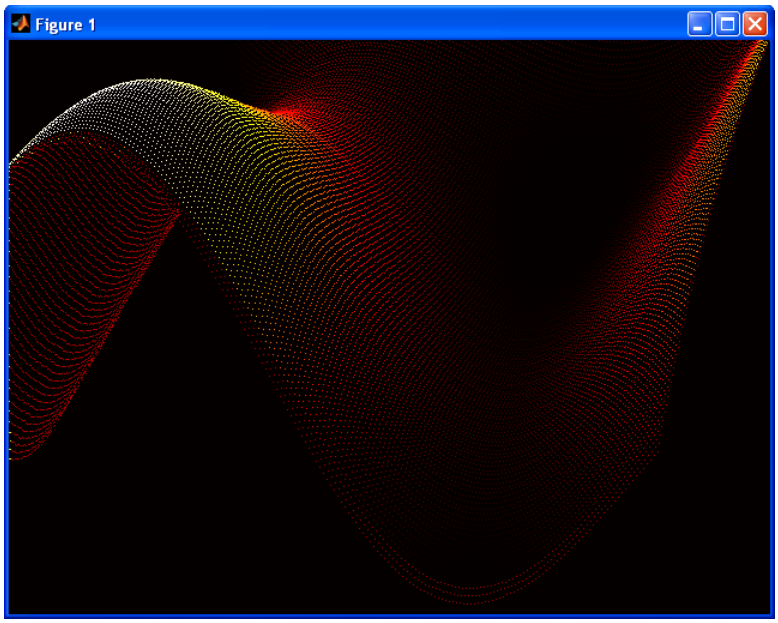
Ακολουθεί ο κώδικας 12.1.2 της συνάρτησης plotter.

```
% 12.1.2
function [ pltout ] = plotter( pltinp )
%
s=3;

%%%%
      mx=(0:639);
      mx(:)=-10000;
      mn=(0:639);
     mn(:)=10000;
%%%%
      for y = -240:s:240
      for x = -320:s:320
            z = fix(100 * getzf(x/350,y/350));
            drawpointf(x,y,z);
            end
      end

mnc=min(img(:));
for c=1:length(img)
    if (img(c)>=1) img(c)=img(c)-mnc; end
end
mxc=max(img(:));
img=fix(img./(mxc/180));

showimg(img,'hot');

%%%%%

    function [getz] = getzf (gzx,gzy)
        getz =3*(1-gzx).^2.*exp(-(gzx.^2) - (gzy+1).^2) ...
           - 10*(gzx/5 - gzx.^3 - gzy.^5).*exp(-gzx.^2-gzy.^2) ...
           - 1/3*exp(-(gzx+1).^2 - gzy.^2);
    end

    function drawpointf(dpx,dpy,dpz)
        cl=255; sh = 0; a=20;
        yi=fix(cos(a)*dpy); yj=fix(sin(a)*dpy);
        fx=320+yi+dpx; fy=240+yj+dpz;

        if (fx>0 & fy>0 & fx<640 & fy<480)
            if (fy > mx(fx)) mx(fx) = fy; sh=0; end
            if (fy < mn(fx)) mn(fx) = fy; sh=dpz-90; end
            if (fy >= mx(fx)) | (fy <= mn(fx))
                img(fy,fx)= dpz - sh;
            end
        end
    end

function drawlinef(x1,y1,x2,y2)
    lwg=1; y=0;
    y1=480-y1; y2=480-y2;
    for x = x1:x2
      px = x;
          py = y;
          y = round((y1 + (x - x1) * (y2 - y1))/(x2 - x1));
```

```
    if (x>0 & x<640 & px>0 & px<640 & y>0 & y<400 & py>0 & py<400)
                    if (lwg==1)
                        [bx,by]=bresenham1(px,py,x,y);
                        for a=1:length(bx)
                            img(by(a)+1,bx(a)+1)=127;
                        end
                    end
                    lwg = 1;
            end
        end
    end
%
end
```

Η μεταβλητή s=3; στις πρώτες γραμμές δημιουργεί ένα πεδίο ορισμού ανά τρία σημεία, ώστε να βελτιωθεί η ταχύτητα εκτύπωσης.

Το αποτέλεσμα φαίνεται στην εικόνα 12.1.5.



Εικόνα 12.1.5. Το πρόγραμμα plotter εκτυπώνοντας δια του υποπρογράμματος showimg.

12.2 Δημιουργία ΓΠΔ για το πρόγραμμα plotter

Το πρώτο πράγμα που θέλουμε να κάνουμε είναι να κατασκευάσουμε ένα παράθυρο που να μπορεί να δεχτεί έναν τύπο συνάρτησης από τον χρήστη και στη συνέχεια, με το πάτημα ενός κουμπιού να υπολογίζει τις τιμές της συνάρτησης z=f(x,y).
Χρησιμοποιούμε ένα πλαίσιο εισαγωγής κειμένου Edit Text, το οποίο περιλαμβάνει εξαρχής ένα τύπο συνάρτησης, τη συνάρτηση peaks(); του MatLab(R).

Χρησιμοποιώντας τα όσα παρατέθηκαν στο κεφάλαιο 11, έχουμε τον εξής κώδικα 12.2.1 :

```
% 12.2.1
function [ plgout ] = plogui1 ( plginp )
%1
if nargin < 1
    plginp = 'initialize';
end
if strcmp(plginp,'initialize')

      h_fig = figure('position',[750 700 300 220], ...
         'numbertitle','off','menubar','none','name','Plotter GUI');

      h_frame = uicontrol('style','frame','units','normalized', ...
                           'position',[0 0 1 1]);

      h_funced = uicontrol('style','edit', ...
          'string','10 .* ( 3*(1-x).^2.*exp(-(x.^2) - ...
            (y+1).^2) - 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2)- ...
            1/3*exp(-(x+1).^2 - y.^2) )', ...
            'HorizontalAlignment','center','position', ...
            [10 150 280 20],'callback','plogui1(''ev'');');

      h_pshbtn = uicontrol('style','pushbutton', ...
            'string','Eval','position',[10 10 40 20], ...
            'callback','plogui1(''ev'')');
%2
      handles = guihandles(h_fig);
      handles.h_fed = h_funced;
      guidata(h_fig,handles);
%3
elseif strcmp(plginp,'ev')
      mgf=350;
      [x,y]=meshgrid((-320:5:320)*(1/mgf),(-240:5:240)*(1/mgf));
     handles = guidata(gcbo);
     funcstr=get(handles.h_fed,'string');
     z=fix(100.*eval(funcstr));
%4
      figure('Name','Plotter Output','Numbertitle','Off');
      surf(z);
end

%5
end
```

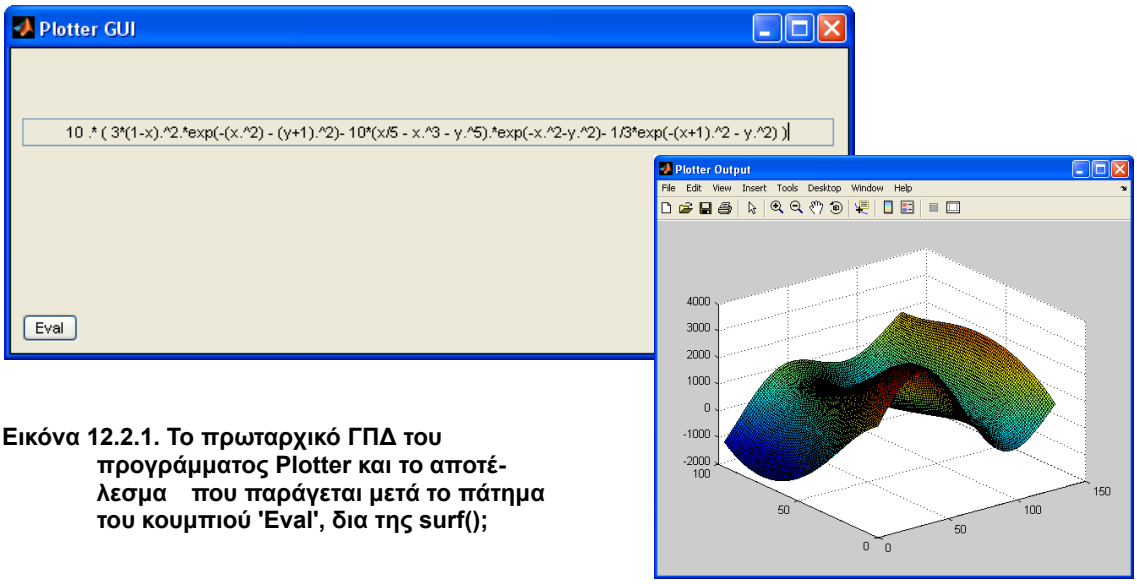
Κατά τα γνωστά, στο τμήμα (%1-%2) ορίζονται τα στοιχεία ΓΠΔ, δηλαδή το παράθυρο διαγράμματος, ένα frame που εκτείνεται σε όλη την επιφάνεια του παραθύρου, ένα πλαίσιο κειμένου Edit Text που περιλαμβάνει τον τύπο της συνάρτησης peaks() και ένα Pushbutton με την ετικέτα 'Eval'.
Όταν ο χρήστης πιέσει το κουμπί 'Eval' το πρόγραμμα πραγματοποιεί υπολογισμό του πίνακα z με ανεξάρτητες τιμές από τους πίνακες x και y που δημιουργούνται στην αρχή του μέρους (%3), μετά την εντολή : elseif.

Ο τύπος της συνάρτησης πρέπει να δοθεί με τη σύνταξη που αναγνωρίζεται από τη γραμμή εντολών του MatLab(R) σαν να είχε δοθεί η εντολή :

```
eval(func_str);
```

με func_str = '10.*(3*(1-x).^2 ...'

Στο μέρος (%4) χρησιμοποιούμε την έτοιμη συνάρτηση του MatLab(R), surrf(), για πάρουμε μια πρώτη ιδέα για το αποτέλεσμα που παράγεται όταν πιέσουμε το κουμπί 'Eval', όπως φαίνεται στις παρακάτω εικόνες.



Εικόνα 12.2.1. Το πρωταρχικό ΓΠΔ του προγράμματος Plotter και το αποτέλεσμα που παράγεται μετά το πάτημα του κουμπιού 'Eval', δια της surf();

Το αποτέλεσμα της εικόνας 12.2.1 παράγεται επειδή στον ορισμό του κουμπιού 'Eval' έχουμε θέσει ως συνάρτηση callback το ίδιο το πρόγραμμα, το plogui1. Ξανακαλείται δηλαδή η γενική συνάρτηση όλου του προγράμματος με εισαγόμενο το αλφαριθμητικό “ev”.

```
h_pshbtn = uicontrol('style','pushbutton', ...
    'string','Eval','position',[10 10 40 20], ...
    'callback','plogui2(''ev'')');
```

Επειδή αυτή η τιμή διαφέρει από την “initialize” το MatLab(R) προσπερνά όλο το τμήμα αρχικοποίησης και εκτελεί τον κώδικα που εισάγεται με τη γραμμή :

```
%3
elseif strcmp(plginp,'ev')
```

όπου δημιουργούνται οι πίνακες x,y και z, και καλείται η συνάρτηση surf().
Ωστόσο, κάθε φορά που εκτελείται το πρόγραμμα, το περιβάλλον του στη μνήμη διαγράφεται και αναδημιουργείται, ώστε τελικά, οι μεταβλητές να πρέπει να σωθούν και να μεταφερθούν στη νέα εκδοχή του προγράμματος Plotter1.

Αν δεν θέλουμε να εργαστούμε στο βασικό χώρο του MatLab(R), (base workspace), αυτό μπορεί να επιτευχθεί με δύο τρόπους :

- Να σώζουμε τα δεδομένα στο userspace του διαγράμματος, ένα κομμάτι μνήμης που συνοδεύει κάθε διάγραμμα, ή

- Να χρησιμοποιούμε τις διάφορες συναρτήσεις του MatLab(R) για να αποθηκεύουμε και να ανακαλούμε τα δεδομένα κάθε φορά που δημιουργείται μια νέα εκδοχή του προγράμματος.

Ο δεύτερος τρόπος είναι πιο πολύπλοκος και απαιτεί περισσότερες γραμμές κώδικα. Ωστόσο, προσφέρει το πλεονέκτημα, ότι μπορούμε να αποθηκεύουμε και δομημένες μεταβλητές, εκτός από μια απλή σειρά τιμών.
Αυτό μας είναι πολύ χρήσιμο, αφού θα πρέπει να αποθηκεύσουμε τετραγωνικούς πίνακες, για τις μεταβλητές x,y και z.
Αν χρησιμοποιήσουμε την πρώτη μέθοδο η διαμόρφωση του πίνακα θα χαθεί και θα πάρουμε ένα διάνυσμα-γραμμή με όλες τις τιμές σε σειρά.

Η συνάρτηση peaks(); του MatLab(R) που χρησιμοποιείται για τη δημιουργία του γραφήματος αποθηκεύεται στο αλφαριθμητικό στοιχείου ΓΠΔ Edit Text ως εξής :

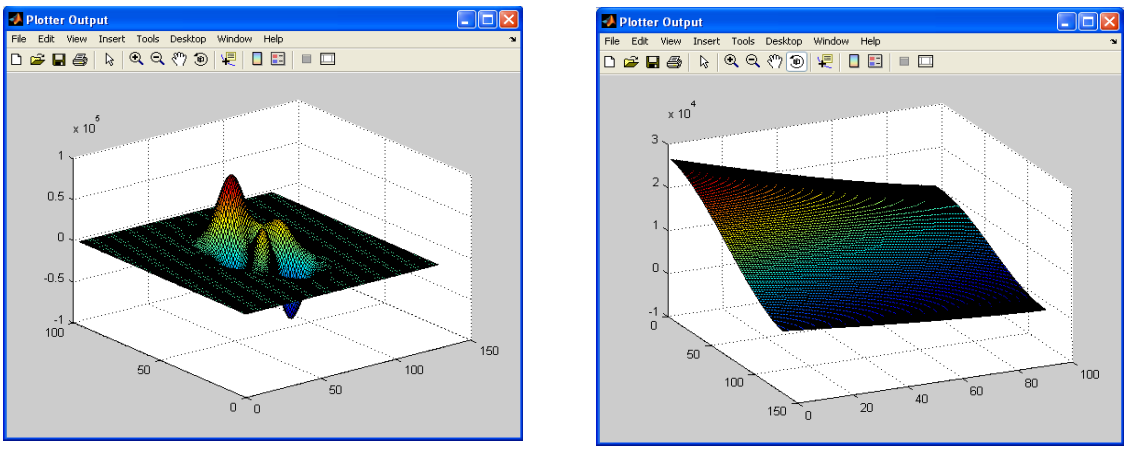
```
h_funced = uicontrol('style','edit','value',0, ...
      'string','10 .* ( 3*(1-x).^2.*exp(-(x.^2) - ...
      (y+1).^2) - 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2)- ...
      1/3*exp(-(x+1).^2 - y.^2) )', ...
      'HorizontalAlignment','center','position', ...
      [10 150 280 20],'callback','plogui1(''ev'');');
```

Θα μπορούσαμε να αποδώσουμε τον τύπο της συνάρτησης σε μια μεταβλητή, peaks για παράδειγμα, και να κάνουμε τον κώδικα πιο ευανάγνωστο.
Και στο στοιχείο αυτό, όπως και στο κουμπί 'Eval' ορίζεται ως callback η συνάρτηση Plogui1 με εισαγόμενο “ev”, ώστε να λαμβάνουμε το αποτέλεσμα με ένα δεύτερο τρόπο, φεύγοντας από το πεδίο εισαγωγής Edit Text.

Η μεταβλητή mgf που ορίζεται στη δεύτερη γραμμή του τμήματος (`%3`) αποτελεί έναν παράγοντα μεγέθυνσης.
Όσο μεγαλύτερη είναι αυτή η τιμή τόσο μεγαλύτερη περιοχή του γραφήματος θα είναι ορατή στο τελικό διάγραμμα, κι όσο μικρότερη είναι τόσο πιο στοιχειώδης επιφάνεια του γραφήματος θα προκύψει.

Για τη συγκεκριμένη συνάρτηση peaks(); μια τιμή mgf = 50; θα δημιουργήσει την απεικόνιση ενός μικρού, σχεδόν επίπεδου τμήματος.
Αντίθετα, για mgf = 1050; μια ευρύτερη περιοχή θα απεικονιστεί, που θα περιλαμβάνει ολόκληρο το ενδιαφέρον τμήμα της συνάρτησης, αφού πέρα από κάποια όρια, από την αρχή των αξόνων, η συνάρτηση γίνεται επίπεδη.

Για να το δείξουμε αυτό παραθέτουμε τα δύο γραφήματα της παρακάτω εικόνας 12.2.2.

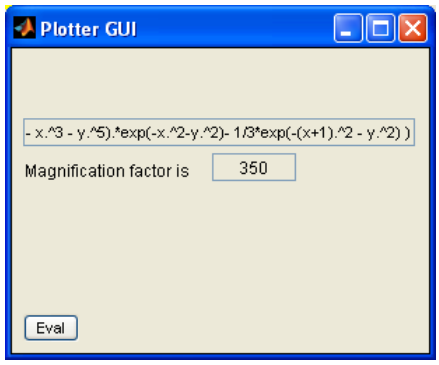


Εικόνα 12.2.2, Το παράθυρο Plotter Output για mgf=1050 (αριστερά) και 50 (δεξιά).

Αν δημιουργήσουμε ένα πεδίο Edit Text, μπορούμε να δώσουμε στον χρήστη τη δυνατότητα να καθορίζει εκείνος τον παράγοντα μεγέθυνσης.

Συμπληρώνουμε λοιπόν στο τμήμα (%1-%2) τις παρακάτω γραμμές για να δημιουργηθούν τα απαραίτητα στοιχεία ΓΠΔ, ένα πεδίο στατικού κειμένου και ένα πεδίο εισαγωγής δεδομένων (box).

```
h_mgf = uicontrol('style','text','FontSize',9, ...
    'string','Magnification factor is', ...
    'HorizontalAlignment','left','position',[10 320 120 20]);

h_edmgf = uicontrol('style','edit',
    'string','350','fontsize',9, ...
    'HorizontalAlignment','center','position',[145 325 60 20]);
```



Εικόνα 12.2.3. Προσθήκη πεδίου παράγοντα μεγέθυνσης.

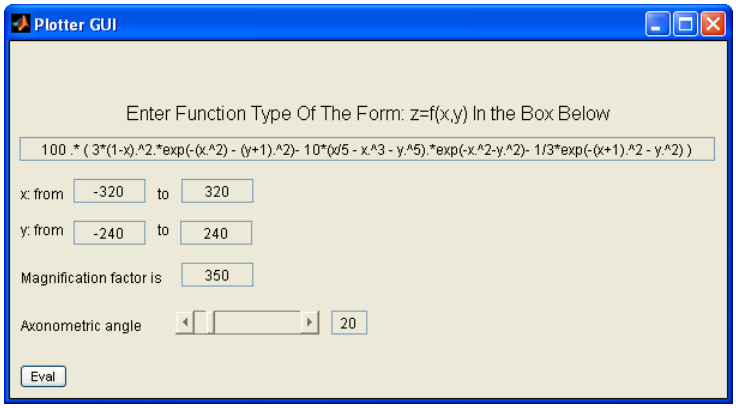
Συνεχίζουμε να εργαζόμαστε με τον ίδιο τρόπο και προσθέτουμε κι άλλα πεδία Static και Edit Text, καθώς και ένα κυλιόμενο κουμπί, συνδεδεμένο με Edit Text Box, που να παρέχει αριθμητική ένδειξη, όπως είδαμε στον κώδικα 11.3.3, κουτιά εισαγωγής του Π.Ο. της συνάρτησης κατά x και κατά y, κι έναν τίτλο στην αρχή, με στοιχείο Static Text.

Έτσι, έχουμε τον εξής κώδικα 12.2.2 :

```
%12.2.2
function [ plgout ] = plogui2 ( plginp )

%1
if nargin < 1
    plginp = 'initialize';
end

%2
if strcmp(plginp,'initialize')

h_fig=figure('position',[750 300 600 300],'numbertitle','off', ...
    'menubar','none','name','Plotter GUI');

h_frame = uicontrol('style','frame','units','normalized', ...
    'position',[0 0 1 1]);
```

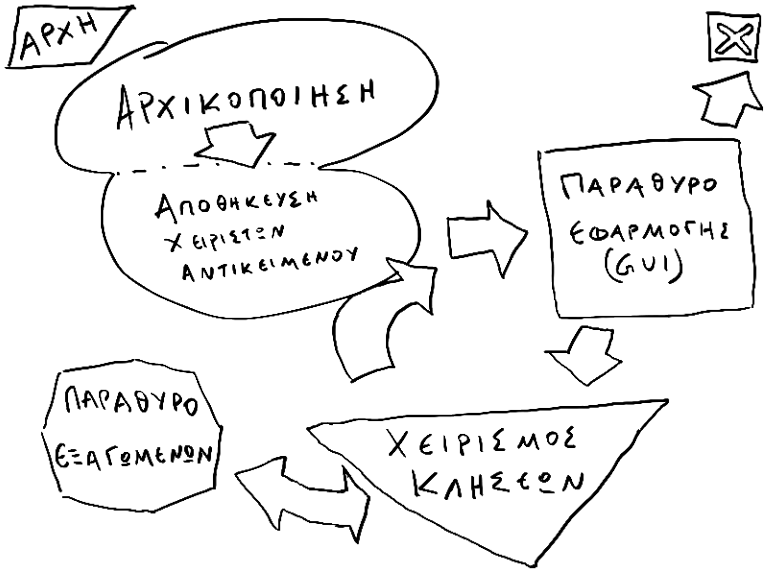
```
 h_sttitl = uicontrol('style','text','value',0,'string', ...
 'Enter Function Type Of The Form: z=f(x,y) In the Box Below', ...
 'fontsize',12,'HorizontalAlignment','center', ...
 'position',[10 230 580 20]);


h_funced = uicontrol('style','edit','value',0,'fontsize',9, ...
    'string','100 .* ( 3*(1-x).^2.*exp(-(x.^2) - (y+1).^2)- ...
      10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2)- ...
      1/3*exp(-(x+1).^2 - y.^2) )', ...
     'position',[10 200 580 20],'HorizontalAlignment','center');


h_stxint = uicontrol('style','text','value',0,'fontsize',9, ...
    'string','x: from                          to', ...
    'HorizontalAlignment','left','position',[10 130 200 20]);

    h_edxmn = uicontrol('style','edit','value',0,'fontsize',9, ...
        'string','-320','HorizontalAlignment','center',  ...
        'position',[55 165) 60 20]);
    h_edxmx = uicontrol('style','edit','value',0,'fontsize',9, ...
        'string','320','HorizontalAlignment','center',  ...
        'position',[145 165 60 20]);


h_styint = uicontrol('style','text','value',0,'fontsize',9, ...
    'string','y: from                          to', ...
    'HorizontalAlignment','left','position',[10 130 200 20]);

    h_edymn = uicontrol('style','edit','value',0,'fontsize',9, ...
        'string','-240','HorizontalAlignment','center',  ...
        'position',[55 130 60 20]);

    h_edymx = uicontrol('style','edit','value',0,'fontsize',9, ...
        'string','240','HorizontalAlignment','center',  ...
        'position',[144 130 60 20]);


h_stmgf = uicontrol('style','text','value',0,'fontsize',9, ...
    'string','Magnification factor is', ...
    'HorizontalAlignment','left','position',[10 120 120 20]);

    h_edmgf = uicontrol('style','edit','value',0, ...
        'string','350','fontsize',9, ...
        'HorizontalAlignment','center','position',[145 95 60 20]);



h_strtx = uicontrol('style','text','value',0,'fontsize',9, ...
    'string','Axonometric angle', ...
    'HorizontalAlignment','left','position',[10 50 120 20]);

    h_slrtx =  uicontrol('style','slider','value',20, ...
            'min',0,'max',180,'callback','plogui2(''slxCH'');', ...
            'fontsize',9,'position',[140 55 120 20]);
```

```
    h_edrtx = uicontrol('style','edit','value',0,'fontsize',9, ...
        'string',num2str(get(h_slrtx,'value')), ...
        'callback','plogui2(''edrxCH'');', ...
        'HorizontalAlignment','center','position',[270 55 30 20]);


h_pshbtn = uicontrol('style','pushbutton', ...
    'string','Eval','position',[10 10 40 20], ...
    'callback','plogui2(''ev'')');

      handles = guihandles(h_fig);
      handles.h_fed = h_funced;
      handles.edrtx = h_edrtx;
      handles.slrtx = h_slrtx;
      guidata(h_fig,handles);

%3
elseif strcmp(plginp,'slxCH')
    handles = guidata(gcbo);
    slxVal = str2num(get(handles.edrtx,'string'));

  if ~length(slxVal)
   slxVal = (get(handles.slrtx,'max')+get(handles.slrtx,'min'))/2;
  end
    slxVal = min([slxVal get(handles.slrtx,'max')]);
    slxVal = max([slxVal get(handles.slrtx,'min')]);

  set(handles.edrtx,'string',num2str(get(handles.slrtx,'value')));

%4
elseif strcmp(plginp,'edrxCH')
    handles = guidata(gcbo);
    slxMn = get(handles.slrtx,'min');
    slxMx = get(handles.slrtx,'max');
    edrxVal = str2num(get(handles.edrtx,'string'));
    if (~length(edrxVal) | (edrxVal<slxMn) | (edrxVal>slxMx) )
            edrxVal = get(handles.slrtx,'value');
            set(handles.edrtx,'string', ...
                  num2str(get(handles.slrtx,'value')));
    end
    set(handles.slrtx,'value',edrxVal);

%5
elseif strcmp(plginp,'ev')
     mgf=350;
     [x,y]=meshgrid((-320:5:320)*(1/mgf),(-240:5:240)*(1/mgf));
     handles = guidata(gcbo);
     funcstr=get(handles.h_fed,'string');
     z=fix(100.*eval(funcstr));
      figure('Name','Plotter Output','Numbertitle','Off');
      surf(z);
end

%
end
```



Εικόνα 12.2.4. Το ΓΠΔ του προγράμματος Plotter2 με περισσότερα στοιχεία.

Η λειτουργία του έως τώρα προγράμματος μπορεί να συνοψιστεί στο παρακάτω σχήμα :



Εικόνα 12.2.5. Η διάρθρωση του προγράμματος Plotter2.

Γίνεται, δηλαδή, μια αρχικοποίηση και αποθηκεύονται οι μεταβλητές. Κατόπιν τρέχει η κεντρική εφαρμογή κι ανάλογα με τις ενέργειες του χρήστη καλείται το τμήμα χειρισμού κλήσεων, που επεξεργάζεται τα εισαγόμενα και είτε καλεί πάλι το παράθυρο της εφαρμογής είτε δημιουργεί ένα παράθυρο εξαγομένων (Output).

Η αποθήκευση των μεταβλητών γίνεται με τις παρακάτω τελευταίες γραμμές του δευτέρου τμήματος `(%2-%3)` :

```
handles = guihandles(h_fig);
handles.h_fed = h_funced;
```

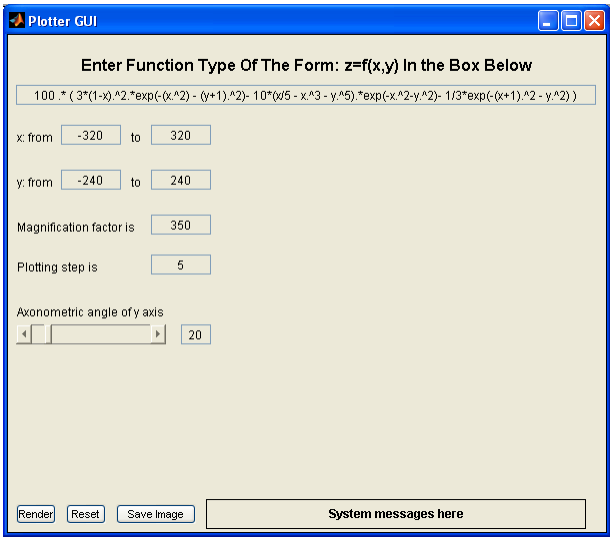
```
handles.edrtx = h_edrtx;
handles.slrtx = h_slrtx;
guidata(h_fig,handles);
```

Αρχικά, η μεταβλητή handles, με τη βοήθεια της συνάρτησης guihandles φορτώνεται με τους χειριστές αντικειμένων του διαγράμματος h_fig.
Στη συνέχεια τα διάφορα πεδία της δομής handles λαμβάνουν τις αντίστοιχες τιμές των χειριστών που τους αποδίδονται και οι αλλαγές καταχωρούνται στη handles με τη βοήθεια της συνάρτησης guidata.

Στο τμήμα `%4` γίνονται οι απαραίτητοι χειρισμοί προκειμένου να συντονιστεί η μεταβολή του πεδίου Edit Text που βρίσκεται δίπλα στο κυλιόμενο κουμπί (slider), με την τιμή που περιλαμβάνει αυτό το slider.
Αρχικά διαβάζονται η ελάχιστη και η μέγιστη τιμή που έχουν καταχωρηθεί στο στοιχείο slider κατά την αρχικοποίηση, καθώς και η τιμή του κουτιού Edit Text.
Ελέγχεται αν η τιμή που έθεσε ο χρήστης στο κουτί είναι ανύπαρκτη ή ξεφεύγει από τα όρια του slider, οπότε του αποδίδεται η τιμή του slider και στη συνέχεια η τιμή αυτή γράφεται ως τιμή στο στοιχείο slider.
Δηλαδή, το slider αλλάξει τιμή, μόνο αν ο αριθμός που εισήχθηκε στο κουτί είναι έγκυρος, διαφορετικά το κουτί ξαναπαίρνει αυτόματα την παλιά του τιμή, που βρισκότανε αποθηκευμένη στα δεδομένα του στοιχείου slider.

Στη συνέχεια προσθέτουμε με τον ίδιο τρόπο επιπλέον στοιχεία : ένα κουμπί “Reset” για να επαναφέρουμε τις τιμές των πεδίων στην αρχικοποιημένη τους κατάσταση, ένα κουμπί “Save Image” για να μπορούμε να αποθηκεύσουμε το διάγραμμα ως ψηφιακή εικόνα στον δίσκο του υπολογιστή και ένα πεδία στατικού κειμένου, μέσα από το οποίο η εφαρμογή να μπορεί να επικοινωνεί με τον χρήστη και τον ενημερώνει για την εκάστοτε κατάστασή της.

Το αποτέλεσμα του παραθύρου ΓΠΔ φαίνεται στην επόμενη εικόνα.



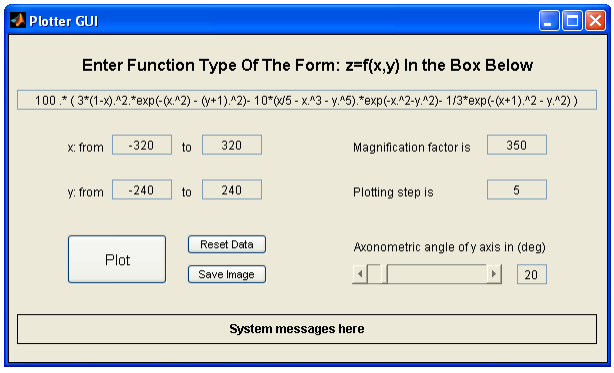
Εικόνα 12.2.6. Προσθήκη κουμπιών “Reset”, “Save Image” και πεδίου μηνυμάτων.

Η αξιοποίηση των καινούριων πεδίων γίνεται με την προσθήκη του παρακάτω κώδικα που αναβαθμίζει το τμήμα εκτύπωσης στο παράθυρο εξόδου (Plotting).

```
% plotting
elseif strcmp(plginp,'rend')
     handles = guidata(gcbo);
     mgf = str2num(get(handles.edmgf,'string'));
     xmn = str2num(get(handles.edxmn,'string'));
     xmx = str2num(get(handles.edxmx,'string'));
     ymn = str2num(get(handles.edymn,'string'));
     ymx = str2num(get(handles.edymx,'string'));
     [x,y]=meshgrid((xmn:5:xmx)*(1/mgf),(ymn:5:ymx)*(1/mgf));
     funcstr=get(handles.h_fed,'string');
     z=fix(100.*eval(funcstr));

      h_curfig = figure('Name','Plotter Output', ...
            'Numbertitle','Off');
      %
      handles.curfig = h_curfig;
      handles.figin = handles.figin + 1;
      guidata(handles.fig,handles);
      %
      surf(z);
        set(handles.stsmsg,'string','Function has been plotted');
```

Βλέπουμε τη δημιουργία δύο τετραγωνικών πινάκων x και y με τη βοήθεια της εντολής meshgrid και τη χρήση των τιμών που λαμβάνονται από τα πεδία εισαγωγής δεδομένων : mgf, xmn, xmx, ymn, ymx.
Στη μεταβλητή funcstr αποθηκεύεται το αλφαριθμητικό που περιλαμβάνει τον τύπο της συνάρτησης και τέλος υπολογίζεται ο πίνακας z.
Ακολουθεί η δημιουργία του διαγράμματος, αφού αποθηκευτεί ο χειριστής αντικειμένου του, καθώς και ένας μετρητής (handles.figin), ώστε η εφαρμογή να γνωρίζει πόσα διαγράμματα έχουν παραχθεί από τη συνάρτηση.

Τα παραπάνω στοιχεία ΓΠΔ μπορούμε να τα τοποθετήσουμε λίγο καλύτερα.



Εικόνα 12.2.7. Το αναμορφωμένο παράθυρο ΓΠΔ για την εφαρμογή Plotter.

Μέχρι το σημείο αυτό χρησιμοποιήσαμε τις έτοιμες συναρτήσεις του MatLab(R) όπως η surf(); για να δημιουργήσουμε το διάγραμμα της συνάρτησης peaks();

Στην επόμενη παράγραφο θα επιχειρήσουμε να συνδέσουμε το ΓΠΔ με το πρόγραμμα Plotter που έχουμε ήδη κατασκευάσει στην ενότητα 12.1.

12.3 Σύνδεση του ΓΠΔ με το πρόγραμμα Plotter

Η ιδέα είναι να αφήσουμε χωριστά τα δύο προγράμματα και να μετατρέψουμε το ΓΠΔ, ώστε να αποστέλλει τα απαραίτητα δεδομένα που έχει συλλέξει από τον χρήστη στο πρόγραμμα Plotter καλώντας το με το όνομα της συνάρτησής του, και αποστέλλοντας τα δεδομένα ως μεταβλητές της συνάρτησης.

Χρειαζόμαστε λοιπόν έναν ορισμό συνάρτησης που να μοιάζει ως εξής :

```
function [ out ] = plotter( mgf,pst,ang,xmn,xmx,ymn,ymx,x,y,z )
```

όπου στην παρένθεση έχουμε : παράγοντας μεγέθυνσης mgf, βήμα εκτύπωσης pst, αξονομετρική γωνία ang, ελάχιστες και μέγιστες τιμές για x και y, και τρείς πίνακες x,y, και z για τις τρεις μεταβλητές.

Μιας και για την γραφική παράσταση της συνάρτησης θα χρησιμοποιήσουμε τον κώδικα που αναπτύξαμε στο κεφάλαιο 11, δηλαδή έναν πίνακα-οθόνη, τον img, πρέπει να φέρουμε τις τιμές των πινάκων x,y και στο θετικό τεταρτημόριο, γιατί το MatLab(R) χρησιμοποιεί θετικές (>0) τιμές για τους δείκτες ενός πίνακα.

Δηλαδή, αν αναφερθούμε στην τιμή img(-4,3) το MatLab(R) θα διακόψει την εκτέλεση του προγράμματος παράγοντας μήνυμα λάθους.

```
y=y+abs(min(y(:)))+1;
x=x+abs(min(x(:)))+1;
z=z+abs(min(z(:)))+1;
```

Ο υπολογισμός των ζευγών σ(x,y) στα οποία τα τρισδιάστατα σημεία Σ(x,y,z) αντιστοιχίζονται μπορεί να γίνει ως εξής :

```
for i=1:(size(x(:)))
        yi=cos(ang)*y(i); yj=sin(ang)*y(i);
        gx(i)=yi+x(i); gy(i)=yj+z(i);
end
```

Έτσι σχηματίζονται δύο πίνακες διανύσματα, ο gx και ο gy, με οριζόντια διάσταση πολλών χιλιάδων στοιχείων.

Επειδή προκύπτουν τιμές εκτός του θετικού τεταρτημορίου αναγκαζόμαστε να μετατοπίσουμε πάλι τις τιμές, οπότε ακυρώνουμε τις πρώτες δύο γραμμές (`y=y+abs...;` και `x=x+abs...;`) και δίνουμε στο σημείο αυτό, μετά το βρόχο `for ... end` :

```
    gy=gy+abs(min(gy));
    gx=gx+abs(min(gx));
```

Στη συνέχεια σμικρύνουμε το εύρος των τιμών ώστε να χωρέσει στα όρια της διαθέσιμης οθόνης, που ας πούμε είναι 640x400 pixels :

```
    Y=(gy./((max(gx)-min(gx))/(480-1)))+1;
    X=(gx./((max(gy)-min(gy))/(640-1)))+1;
```

Έπειτα, δημιουργούμε τον πίνακα img και τοποθετούμε τα ζεύγη τιμών (X,Y) που έχουμε ήδη υπολογίσει στα αντίστοιχα στοιχεία του.

```
    img=zeros(480,640);
```

```
for i=1:size(z(:))
    img(fix(Y(i)) , fix(X(i)) ) = z(i);
end
```

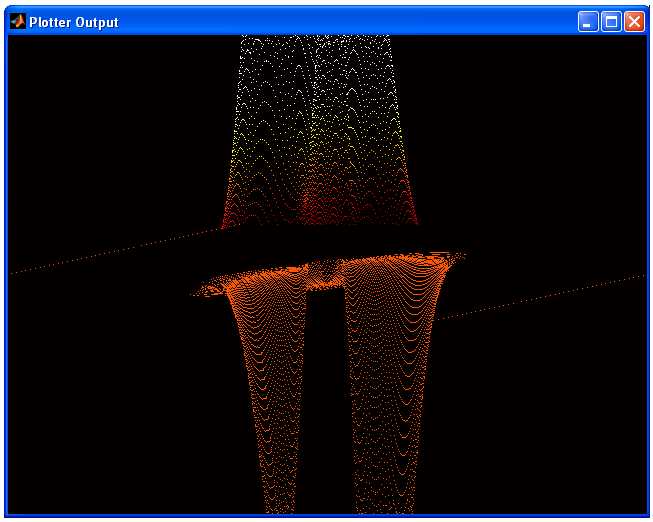
Τέλος, αναπροσαρμόζουμε τις τιμές του πίνακα z, ώστε να περιοριστούν στα όρια ενός χρωματικού χάρτη βάθους 128 (τα σχετικά έχουν αναλυθεί στα πρώτα κεφάλαια) και δημιουργούμε το διάγραμμα με την εντολή image.

```
mxc=max(img(:));
img=fix(img./(mxc/118));

imgage(img);
colormap('Hot');
```

Στο πρόγραμμα plotter-gui, που έχουμε ήδη αναπτύξει, αντικαθιστούμε την κλήση surf(z); με μια κλήση προς το πρόγραμμα plotter :

```
plotter4(mgf,pst,ang,xmn,xmx,ymn,ymx,x,y,z);
```

Το αποτέλεσμα, με συγκεκριμένες ρυθμίσεις, mgf και ang, φαίνεται στην παρακάτω εικόνα.



Εικόνα 12.3.1. Το αποτέλεσμα της σύνδεσης του GUI με το πρόγραμμα Plotter.

Παρατηρούμε πως η κλιμάκωση των τιμών της συνάρτησης με πολλαπλασιασμό επί τον αριθμό 100 που χρησιμοποιήθηκε στο τμήμα (`% plotting`) του Plotter-gui (κεφ. 12.2), δημιουργεί πολύ υψηλά όρη στη συνάρτηση peaks();

```
z=fix(100.*eval(funcstr));
```

Εξάλλου ο χρήστης μπορεί να μη θέλει καμιά παραμόρφωση στην εκτύπωση των τιμών της συνάρτησης.

Για τον λόγο αυτό εισάγουμε μια νέα δυνατότητα, τον βαθμό κλιμάκωσης των τιμών της συνάρτησης zsc.

Αρχικά δημιουργούμε τα στοιχεία ΓΠΔ δια των οποίων ο χρήστης θα μπορεί να εισαγάγει τον βαθμό κλιμάκωσης τιμών που επιθυμεί : ένα κουτί τύπου Edit Text, συνοδευόμενο από ένα πλαίσιο στατικού κειμένου Static Text.

```
h_stzsc = uicontrol('style','text','value',0,'fontsize',9, ...
     'string','Scaling to z axis is','HorizontalAlignment', ...
      'left','position',[345 (pin1-110) 120 20]);

    h_edzsc = uicontrol('style','edit','value',0, ...
        'string','100','fontsize',9,'HorizontalAlignment', ...
        'center','position',[480 (pin1-105) 60 20]);
```

Στη συνέχεια, σώζουμε τον χειριστή αντικειμένου :

```
    handles.edzsc = h_edzsc;
```

Στο τμήμα (`% plotting`), δημιουργούμε μια μεταβλητή που λαμβάνει το περιεχόμενο του στοιχείου, από τη δομή handles, και το εισάγει στον υπολογισμό των τιμών του πίνακα z.

```
    zsc = str2num(get(handles.edzsc,'string'));

    z=fix(zsc.*eval(funcstr));
```

Τέλος, δεν ξεχνάμε να προσθέσυμε μια γραμμή για επαναφορά της αρχικής τιμής του στοιχείου ΓΠΔ, στο τμήμα (`% reset`).

```
    set(handles.edzsc,'string','10');
```

Για να κάνουμε την εφαρμογή να δίνει στο χρήστη επιπλέον δυνατότητες, όπως την επιλογή της ανάλυσης ή του χρωματικού χάρτη του διαγράμματος με τη βοήθεια μενού που αναδιπλώνονται προσθέτουμε επιπλέον κώδικα, σύμφωνα με όσα αναλύσαμε πιο πάνω :

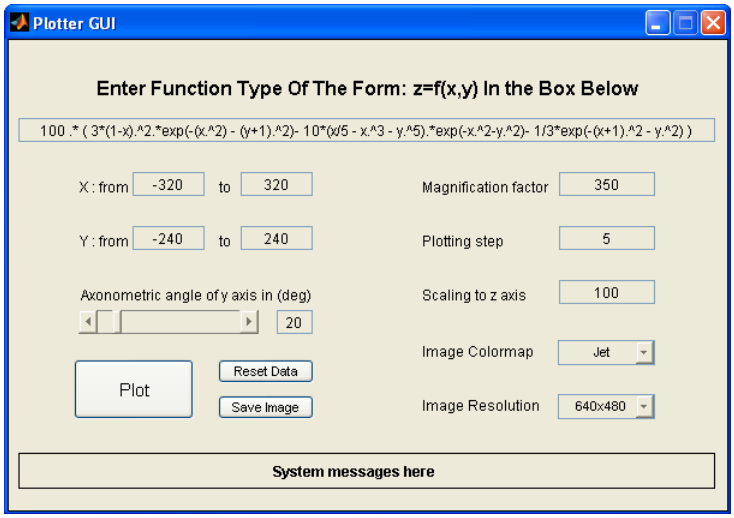
```
h_stpmcm = uicontrol('style','text','value',0,'fontsize',9, ...
    'string','Colormap to render', ...
    'HorizontalAlignment','left','position',[60 175 195 20]);


h_pumcm = uicontrol('style','popup','string', ...
    'Jet|HSV|Hot|Cool|Spring|Summer|Autumn|Winter|Gray|Bone|
                                        Copper|Pink|Lines', ...
    'position',[194 175 61 20]);


h_stres = uicontrol('style','text','value',0,'fontsize',9, ...
    'string','Image Resolution', ...
    'HorizontalAlignment','left','position',[345 168 195 20]);

h_pumres = uicontrol('style','popup','string', ...
    '640x480|800x600|1024x768|1280x800|1400x1050|1600x1200|
                                                1920x1080',
...
    'HorizontalAlignment','center','position',[459 170 81 20]);
```

Το σύμβολο "|" που χρησιμοποιείται ως διαχωριστικό, όταν εισάγεται σε τιμές της ιδιότητας `'string'`, προκαλεί την αλλαγή γραμμής από τμήμα σε τμήμα του αλφαριθμητικού που ορίζει. Λειτουργεί όμως μόνο για τα στοιχεία ListBox και PopupMenu.

Όπως και πριν, συμπληρώνουμε τον κώδικα με εισαγωγή γραμμών για τη δημιουργία της δομής handles, καθώς και στα τμήματα (`% reset`) και (`% plotting`).

Το πρόγραμμα Plotting τώρα πρέπει να κληθεί με επιπλέον μεταβλητές : cm για το Colormap και resx, resy, για την ανάλυση της οθόνης του διαγράμματος.

Το παράθυρο ΓΠΔ της εφαρμογής λαμβάνει τη μορφή που φαίνεται στην εικόνα 12.3.2.

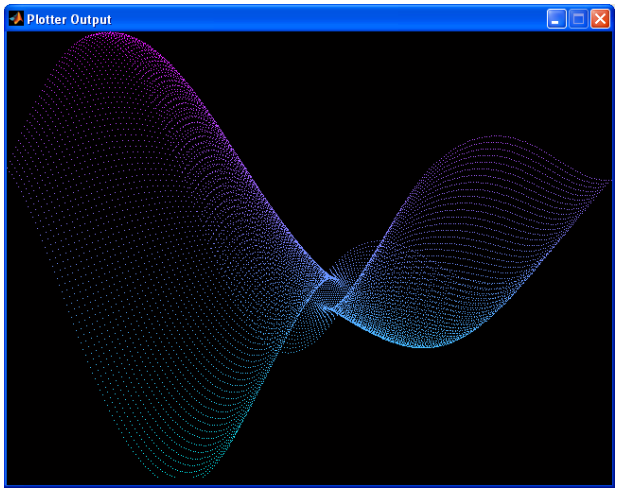


Εικόνα 12.3.2. Το παράθυρο ΓΠΔ με PopupMenus για επιλογή ανάλυσης και χρωματικού χάρτη της οθόνης του διαγράμματος.

Στο τμήμα (`% plotting`) πρέπει να επεξεργαστούμε την τιμή (value) των PopupMenus με δύο ενότητες switch, αφού οι επιλογές που κάνει ο χρήστης με τον δείκτη του ποντικιού μεταφράζονται σε έναν αύξοντα αριθμό. Εμείς πρέπει να αντιστοιχίσουμε τον αριθμό αυτό σε ένα αλφαριθμητικό :

```
pumcm = get(handles.pumcm,'value');

switch pumcm
    case 1
        pmc_str='Jet';
    case 2
        pmc_str='HSV';
    case 3
        pmc_str='Hot';
    case 4
        pmc_str='Cool';
    case 5
        pmc_str='Spring';
    case 6
        pmc_str='Summer';
```

```
        case 7
            pmc_str='Autumn';
        case 8
            pmc_str='Winter';
        case 9
            pmc_str='Gray';
        case 10
            pmc_str='Bone';
        case 11
            pmc_str='Copper';
        case 12
            pmc_str='Pink';
        case 13
            pmc_str='Lines';
    end
    cm=pmc_str;

    pumres = get(handles.pumres,'value');
    switch pumres
        case 1
            resx=640;
            resy=480;
        case 2
            resx=800;
            resy=600;
        case 3
            resx=1024;
            resy=768;
        case 4
            resx=1280;
            resy=800;
        case 5
            resx=1400;
            resy=1050;
        case 6
            resx=1600;
            resy=1200;
        case 7
            resx=1920;
            resy=1080;
    end
```



Εικόνα 12.3.3. Η οθόνη διαγράμματος της συνάρτησης peaks, με χρωματικό χάρτη “cool”.

Μπορούμε, αντί να κάνουμε την εκτύπωση από το αρχείο Plotter, να μεταφέρουμε τις σχετικές εντολές σε ένα χωριστό αρχείο με ονομασία για παράδειγμα : showimg.m και κάθε φορά που απαιτείται δημιουργία διαγράμματος να καλούμε τη συνάρτηση showimg μεταβιβάζοντας την οθόνη img και τον χρωματικό χάρτη cm, ως εξής :

```
showimg(img,cm);
```

Η συνάρτηση showimg θα πρέπει να είναι όπως πιο κάτω.

```
function [h] = showimg (img,mp)
%

image(img);
mp=colormap(mp);
mp(1,:)=0;
colormap(mp);

%
end
```

Μηδενίζουμε την πρώτη γραμμή του πίνακα χρωματικού χάρτη, γιατί οι μηδενικές τιμές της οθόνης img αντιστοιχούν στο φόντο.
Με τον τρόπο αυτό ορίζουμε το φόντο να είναι μαύρο.

Για να διαμορφώσουμε καλύτερα το οπτικό αποτέλεσμα χρησιμοποιούμε επιπλέον εντολές στην αρχή της συνάρτησης. Αφού ρυθμιστούν οι ιδιότητες του παραθύρου, λαμβάνονται οι διαστάσεις του πίνακα οθόνης, και δημιουργείται ένα παράθυρο με το αντίστοιχο μέγεθος.

```
set(gcf,'BackingStore','on','ButtonDownFcn','mbimg','DockControls','of
f','DoubleBuffer','on');
set(gcf,'MenuBar','none','Pointer','crosshair');

hold on;
[m,n,p] = size(img);
set(gca, 'Position', [0 0 1 1]);
set(gcf,'Units', 'pixels', 'Position', [40 40 n m]);
```

Η τοποθέτηση ζευγών τιμών στον πίνακα img και στο αρχείο Plotter μπορεί να βελτιωθεί, με λίγο επιπλέον κώδικα.
Μπορούμε, για παράδειγμα, να δημιουργήσουμε απόκρυψη των γραμμών που κανονικά δέν φαίνονται από τη θέση παρατήρησης του χρήστη, χρησιμοποιώντας τον παρακάτω κώδικα :

```
    for i=1:size(z(:))
        if (Y(i) > mx(X(i))) mx(X(i)) = Y(i); end
        if (Y(i) < mn(X(i))) mn(X(i)) = Y(i); end

        if (hlr==0)
            img( Y(i),X(i) ) = z(i);
        elseif ((Y(i) >= mx(X(i))) | (Y(i) <= mn(X(i))))
            img( Y(i),X(i) ) = z(i);
        end
    end
```

Αν το διάγραμμα δημιουργείται όχι με απόκρυψη των γραμμών που δεν είναι ορατές η μεταβλητή hlr έχει την τιμή 0, (hlr==0).

Τότε εκτελείται η εντολή που έχουμε δει :

```
img( Y(i),X(i) ) = z(i);
```

Παρατηρούμε πως η συνάρτηση fix() έχει αφαιρεθεί από τα Y(i) και X(i), προκειμένου να προστεθεί στην προετοιμασία των τιμών που έχει προηγηθεί.

Αν (hlr==1) θα εκτελεστεί η γραμμή μετά το elseif, η οποία ελέγχει αν το ζεύγος που πρόκειται να εκτυπωθεί βρίσκεται ψηλότερα ή χαμηλότερα, από όσα έχουν τυπωθεί ήδη, συγκρίνοντας την τιμή με το ως τώρα μέγιστο που έχει καταχωρηθεί στους πίνακες mn,mx.
Οι πίνακες αυτοί δημιουργούνται ως εξής :

```
mx=(1:resx);
mx(:)=min(Y(:));
mn=(1:resx);
mn(:)=max(Y(:));
```

Ο πίνακας των μεγίστων υψών αρχικοποιείται με την ελάχιστη τιμή, ενώ ο πίνακας των ελαχίστων με τη μέγιστη. Έτσι δεν υπάρχει περίπτωση να απολεσθεί κάποια τιμή.
Η προετοιμασία των τιμών γίνεται ως εξής :

```
gxmn = min(gx);
gxmx = max(gx);
gymn = min(gy);
gymx = max(gy);
lx = (gxmx-gxmn);
ly = (gymx-gymn);

gy=gy+abs(gymn);
gx=gx+abs(gxmn);
Y=fix((gy./(ly/(resy-1)))+1);
X=fix((gx./(lx/(resx-1)))+1);
```

Υπολογίζονται τα μέγιστα και τα ελάχιστα των δύο μονοδιάστατων πινάκων που περιέχουν τις τιμές των ζευγών, καθώς και τα μέγιστα ανοίγματα τιμών κατά x και κατά y.
Στη συνέχεια, με απλές πράξεις προσαρμόζουμε τα δεδομένα ώστε να ταυτίζονται κατά x και y με τις διαστάσεις της οθόνης διαγράμματος που έχουν εισαχθεί στη συνάρτηση δια των μεταβλητών resx και resy.

Παρατηρούμε εδώ το σημείο όπου μεταφέρθηκαν οι συναρτήσεις fix(), όπως αναφέρθηκε προηγουμένως.
Οι νέοι πίνακες X και Y, με τα αναπροσαρμοσμένα στοιχεία εισάγονται στο βρόχο for...end που παρουσιάστηκε αμέσως πριν.

Εκτός από την απόκρυψη των μη ορατών γραμμών, μπορούμε να δημιουργήσουμε ενός είδους σκίαση στο κάτω μέρος της σχεδιαζόμενης επιφάνειας.
Αυτό πολλές φορές βοηθά να διακρίνουμε καλύτερα ποιο μέρος της επιφάνειας βρίσκεται πλησιέστερα στον παρατηρητή και ποιο βρίσκεται μακρύτερα, αυξάνοντας την ευκρίνεια της εικόνας.

```
for i=1:size(z(:))
   if (Y(i) > mx(X(i))) mx(X(i)) = Y(i); sh=0; end
   if (Y(i) < mn(X(i))) mn(X(i)) = Y(i); sh=z(i)*(shdi/100); end

   if (hlr==0)
```

```
      img( Y(i),X(i) ) = z(i)-sh;
    elseif ((Y(i) >= mx(X(i))) | (Y(i) <= mn(X(i))))
         img( Y(i),X(i) ) = z(i)-sh;
    end
  end
```

Εισάγουμε λοιπόν μια μεταβλητή sh στην οποία καταχωρείται ο βαθμός σκίασης.
Για τα πάνω σημεία είναι μηδέν (sh=0;), ενώ για τα κάτω είναι ένα ποσοστό του κανονικού χρώματος (shdi/100), το οποίο πρόκειται να καθορίζεται από τον χρήστη.
Έτσι κατά την εγγραφή στον πίνακα img καταχωρείται η τιμή z(i)-sh, όπου sh είναι η υποβάθμιση του χρώματος.

Τέλος, μπορούμε να προσθέσουμε τέσσερις γραμμές και να κάνουμε το πρόγραμμα να τυπώνει την συνάρτηση γραμμή προς γραμμή, αντί να μας δείχνει έτοιμο το αποτέλεσμα :

```
      [r,c]=ind2sub(size(z),i);

      if (rtr==1)&(r==1)
            image(flipud(img)); pause(0.0005);
      end
```

Πρώτα λαμβάνουμε στη μεταβλητή r τον αριθμό της στήλης του εμβαδού που αποτελεί το Π.Ο. της συνάρτησης και αν η στήλη αυτή ισούται με ένα (r==1) και ο χρήστης έχει θέσει τύπωμα “γραμμή προς γραμμή”, δηλαδή (rtr == 1), τότε δημιουργείται μια εικόνα, με όσες γραμμές έχουν καταχωρηθεί μέχρι την συγκεκριμένη στιγμή.

Παρατηρούμε πως ενώ η εντολή ind2sub() αποδίδει στη μεταβλητή r τη γραμμή του z, εμείς την ονομάζουμε στήλη, γιατί οι πίνακες x,y και z έχουν ήδη αναδιαταχθεί με τις παρακάτω εντολές :

```
x=x';y=y';z=z';
x=fliplr(x);
y=fliplr(y);
z=fliplr(z);
```

Αυτό συμβαίνει γιατί το MatLab(R) διαβάζει τους πίνακες κατά στήλες, αν χρησιμοποιήσουμε μόνο έναν δείκτη. Δηλαδή, από πάνω προς τα κάτω και από αριστερά προς τα δεξιά.

Στην περίπτωση όμως που έχουμε άμεσο τύπωμα, γραμμή προς γραμμή, πρέπει να προετοιμάσουμε το παράθυρο του διαγράμματος, αφού θα ήταν πολύ δύσκολο να καλούμε σε κάθε εκτέλεση του βρόχου την εξωτερική συνάρτηση showimg.
Συμπληρώνουμε, λοιπόν, πριν το βρόχο τις εξής γραμμές :

```
if (rtr==1)
    h_tmp=gcf;
    close(h_tmp);
    img=zeros(resy,resx);
    h_out=figure;
    set(h_out,'resize','off','Name','Plotter Output', ...
                                          'Numbertitle','Off');

    set(gcf,'BackingStore','on','DockControls','off', ...
                                        'DoubleBuffer','on');

    set(gca,'Position',[0 0 1 1]);
```

```
    set(gcf,'MenuBar','none','Units','pixels', ...
                                    'Position',[100 100 resx resy]);

    axis equal;
    axis off;
    z=fix(z./(max(z(:))/64));
    mp=colormap(cm);
    mp(1,:)=0;
    colormap(mp);
end
```

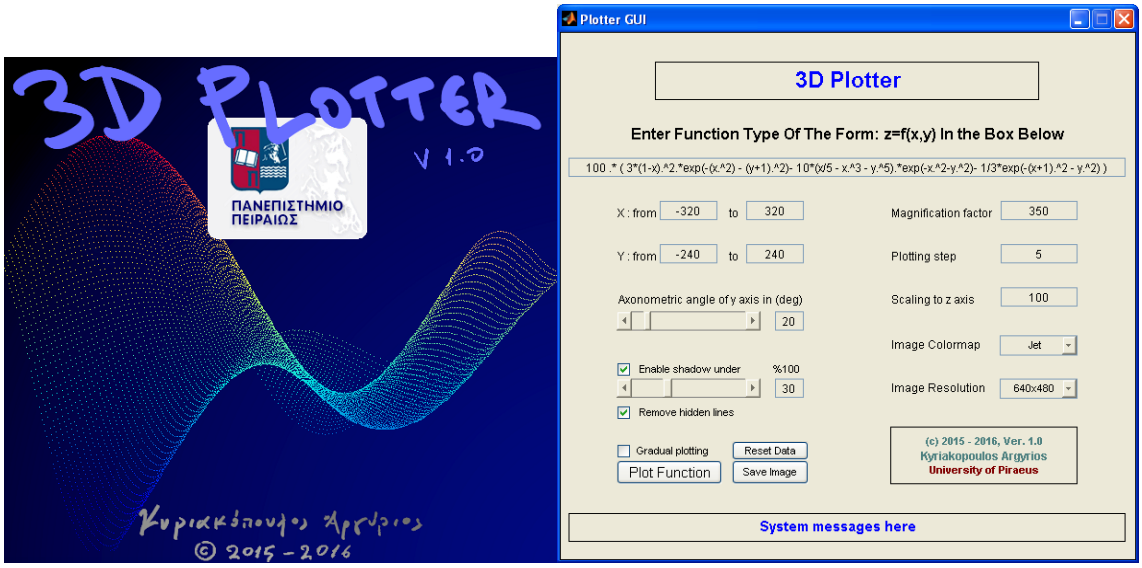
Αν ο χρήστης επιλέξει τύπωμα “γραμμή προς γραμμή” (rtr==1), τότε εκτελούνται οι παραπάνω γραμμές, που κάνουν όσα αναφέραμε και για την εξωτερική συνάρτηση showimg.

Για να λειτουργήσουν αυτές οι τρείς επιπλέον δυνατότητες κατασκευάζουμε τα κουμπιά και τα κουτιά τσεκαρίσματος στο πάνελ και τα συνδέουμε μεταξύ τους και καταχωρούμε handles και εντολές για επαναφορά τιμών όπως έχουμε περιγράψει ήδη για τα υπόλοιπα στοιχεία ΓΠΔ. Δεν ξεχνάμε βεβαίως να τροποποιήσουμε την κλήση και τον ορισμό της συνάρτησης Plotter, ώστε να περιλάβουμε τις νέες μεταβλητές στα ορίσματά της.

Επίσης, στο τελικό πάνελ προσθέτουμε έναν τίτλο : “3D Plotter”, καθώς και μία επιγραφή με την κατοχύρωση των πνευματικών δικαιωμάτων, το όνομα του δημιουργού και του Πανεπιστημίου Πειραιώς.

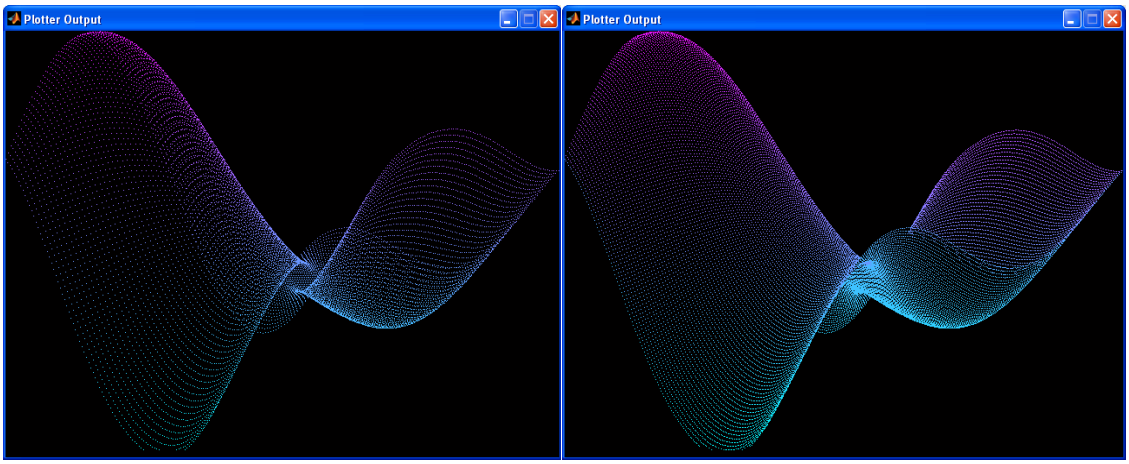
Για να ολοκληρωθεί η εφαρμογή προσθέτουμε μια κάρτα εκκίνησης με το λογότυπο της εφαρμογής που εμφανίζεται πριν το πάνελ και βρίσκεται αποθηκευμένη στον ίδιο κατάλογο του δίσκου, σε μορφή *.mat.

Η εφαρμογή “3D Plotter” διαμορφώνεται κατά την εικόνα 12.3.4.

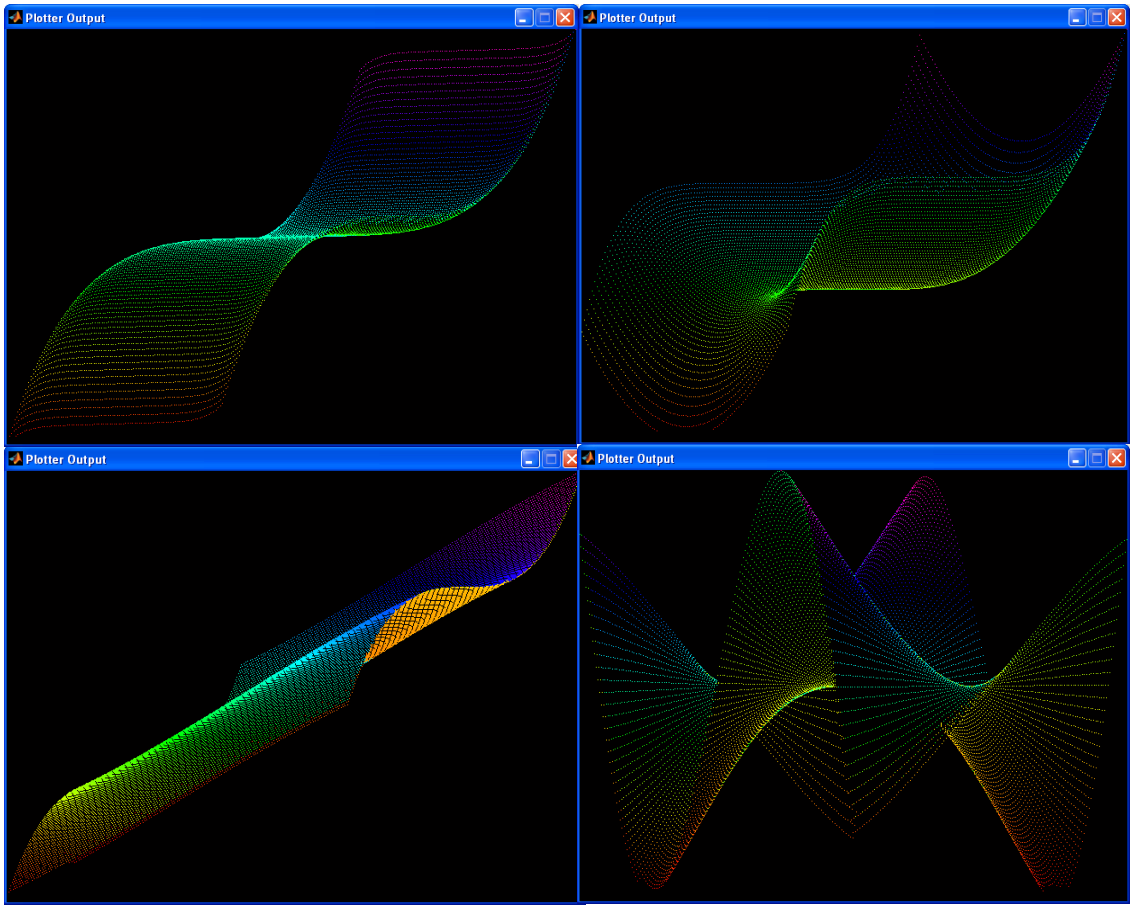


Εικόνα 12.3.4. η τελική διαμόρφωση της εφαρμογής “3D Plotter”.

Στις επόμενες εικόνες παρουσιάζεται μια ενδεικτική χρήση της εφαρμογής και διάφορα αποτελέσματα που προκύπτουν με τροποποίηση των ρυθμίσεων του κεντρικού πάνελ.



Εικόνα 12.3.5. Η συνάρτηση peaks(); με βήμα 5 και χωρίς απόκρυψη γραμμών αριστερά, και με βήμα 3, απόκρυψη γραμμών και σκίαση δεξιά.



Εικόνα 12.3.6. Οι συναρτήσεις : x.^3+y.^7 (πάνω-αριστερά), x.^3+y.^7 (πάνω-δεξιά) x.^3+sin(y) (κάτω-αριστερά) και sin(2.*x).*sin(y).*sqrt(x.^2+y.^2) (κάτω-δεξιά).

12.4 Κώδικας προγραμμάτων της εφαρμογής “3D Plotter”

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Kyriakopoulos Argyrios mppl11025                                           %
%  Dissertation on Msc Informatics                                            %
%  University of Piraeus                                                      %
%  (c) 2015-2016                                                              %
%                                                                             %
%  3D Plotter                                                                 %
%                                                                             %
%  Files: Plogui8.m, Plotter8.m, Picture.m, Showimg.m,Plogui8.mat             %
%                                                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%

function [ plgout ] = plogui8 ( plginp )
%
if nargin < 1
    plginp = 'initialize';
end
%
if strcmp(plginp,'initialize')

h_fig = figure('position',[300 400 600 550],'numbertitle','off', ...
    'resize','off','menubar','none','name','Plotter GUI');

    picture; pause(5);

h_frame = uicontrol('style','frame','units','normalized', ...
    'position',[0 0 1 1]);

h_frttl = uicontrol('style','frame','position',[100 480 400 40]);

    h_stttl = uicontrol('style','text', ...
      'string','3D Plotter', ...
      'fontsize',17,'FontWeight','bold','foregroundcolor',[0,0,1], ...
      'HorizontalAlignment','center','position',[105 485 390 30]);


h_frmsg = uicontrol('style','frame','position',[10 20 580 30]);

    h_stsmsg = uicontrol('style','text', ...
      'string','System messages here', ...
      'fontsize',11,'FontWeight','bold','foregroundcolor',[0,0,1], ...
      'HorizontalAlignment','center','position',[110 25 360 20]);

h_frcpr = uicontrol('style','frame','position',[345 80 195 60]);

    h_stcpr1 = uicontrol('style','text', ...
      'string','(c) 2015 - 2016, Ver. 1.0','fontsize',8, ...
      'FontWeight','bold','foregroundcolor',[0.3,0.5,0.5], ...
      'HorizontalAlignment','center','position',[350 111 185 20]);

    h_stcpr2 = uicontrol('style','text', ...
      'string','Kyriakopoulos Argyrios','fontsize',9, ...
```

```
     'FontWeight','bold','foregroundcolor',[0.3,0.5,0.5], ...
     'HorizontalAlignment','center','position',[350 97 185 20]);
   h_stcpr3 = uicontrol('style','text', ...
     'string','University of Piraeus', ...
     'fontsize',8,'FontWeight','bold','foregroundcolor',[0.5,0,0], ...
     'HorizontalAlignment','center','position',[350 81 185 20]);


h_sttitl = uicontrol('style','text','value',0,'string', ...
    'Enter Function Type Of The Form: z=f(x,y) In the Box Below', ...
    'fontsize',12,'FontWeight','bold', ...
    'HorizontalAlignment','center','position',[10 435 580 20]);

h_funced = uicontrol('style','edit','value',0,'fontsize',9, ...
    'string','100 .* ( 3*(1-x).^2.*exp(-(x.^2) - (y+1).^2)- 10*(x/5 -
    x.^3 - y.^5).*exp(-x.^2-y.^2)- 1/3*exp(-(x+1).^2 - y.^2) )', ...
    'HorizontalAlignment','center','position',[10 400 580 20]);


h_stxint = uicontrol('style','text','value',0,'fontsize',9, ...
    'string','X : from                          to', ...
    'HorizontalAlignment','left','position',[60 350 200 20]);

    h_edxmn = uicontrol('style','edit','value',0,'fontsize',9, ...
        'string','-320','HorizontalAlignment','center',  ...
        'position',[105 355 60 20]);
    h_edxmx = uicontrol('style','edit','value',0,'fontsize',9, ...
        'string','320','HorizontalAlignment','center',  ...
        'position',[195 355 60 20]);


h_styint = uicontrol('style','text','value',0,'fontsize',9, ...
    'string','Y : from                          to', ...
    'HorizontalAlignment','left','position',[60 305 200 20]);

    h_edymn = uicontrol('style','edit','value',0,'fontsize',9, ...
        'string','-240','HorizontalAlignment','center',  ...
        'position',[105 310 60 20]);

    h_edymx = uicontrol('style','edit','value',0,'fontsize',9, ...
        'string','240','HorizontalAlignment','center',  ...
        'position',[195 310 60 20]);


h_strtx = uicontrol('style','text','value',0,'fontsize',9, ...
    'string','Axonometric angle of y axis in (deg)', ...
    'HorizontalAlignment','center','position',[60 260 195 20]);

    h_slrtx =
      uicontrol('style','slider','value',20,'min',0,'max',180,...
        'callback','plogui8(''slxCH'');','fontsize',9, ...
        'sliderstep',[1/180 1/18], ...
        'position',[60 240 150 20]);

    h_edrtx = uicontrol('style','edit','value',0,'fontsize',9, ...
        'string',num2str(get(h_slrtx,'value')), ...
```

```
        'callback','plogui8(''edrxCH'');', ...
        'HorizontalAlignment','center', ...
        'position',[225 240 30 20]);

h_chbshd = uicontrol('style','checkbox','value',1, ...
     'string','  Enable shadow under          %100', ...
     'position',[60 190 225 20], ...
     'fontsize',8,'callback','plogui8(''chbshdCH'')');

    h_slshd = uicontrol('style','slider','value',30, ...
        'min',0,'max',100, ...
        'callback','plogui8(''slshdCH'');','fontsize',9, ...
        'sliderstep',[1/100 1/10], ...
        'position',[60 170 150 20]);

    h_edshd = uicontrol('style','edit','fontsize',9, ...
        'string',num2str(get(h_slshd,'value')), ...
        'callback','plogui8(''edshdCH'');', ...
        'HorizontalAlignment','center', ...
        'position',[225 170 30 20]);


h_chbhlr = uicontrol('style','checkbox','value',1, ...
    'string','  Remove hidden lines','position',[60 145 225 20], ...
    'fontsize',8);


h_stmgf = uicontrol('style','text','value',0,'fontsize',9, ...
    'string','Magnification factor', ...
    'HorizontalAlignment','left','position',[345 350 120 20]);

    h_edmgf = uicontrol('style','edit','value',0, ...
        'string','350','fontsize',9, ...
        'HorizontalAlignment','center','position',[460 355 80 20]);


h_stpst = uicontrol('style','text','value',0,'fontsize',9, ...
    'string','Plotting step', ...
    'HorizontalAlignment','left','position',[345 305 120 20]);

    h_edpst = uicontrol('style','edit','value',0, ...
        'string','5','fontsize',9, ...
        'HorizontalAlignment','center','position',[460 310 80 20]);


h_stzsc = uicontrol('style','text','value',0,'fontsize',9, ...
    'string','Scaling to z axis', ...
    'HorizontalAlignment','left','position',[345 260 120 20]);

    h_edzsc = uicontrol('style','edit','value',0, ...
        'string','100','fontsize',9, ...
        'HorizontalAlignment','center','position',[460 265 80 20]);

h_stpmcm = uicontrol('style','text','value',0,'fontsize',9, ...
    'string','Image Colormap', ...
    'HorizontalAlignment','left','position',[345 213 195 20]);
```

```
    h_pumcm = uicontrol('style','popup','string', ...
      '         Jet|        HSV|         Hot|        Cool|      Spring
       |      Summer|      Autumn|      Winter|        Gray
       |        Bone|      Copper|        Pink|       Lines', ...
      'HorizontalAlignment','center','position',[459 215 81 20]);


h_stres = uicontrol('style','text','value',0,'fontsize',9, ...
    'string','Image Resolution', ...
    'HorizontalAlignment','left','position',[345 168 195 20]);

    h_pumres = uicontrol('style','popup','string', ...
        '    640x480|    800x600|  1024x768|  1280x800| 1400x1050
        | 1600x1200| 1920x1080', ...
        'HorizontalAlignment','center','position',[459 170 81 20]);
    set(h_pumres,'value',1);


h_pbrend = uicontrol('style','pushbutton', ...
    'string','Plot Function','position',[60 80 110 25], ...
    'fontsize',11,'callback','plogui8(''rend'')');

h_chbrtr = uicontrol('style','checkbox','value',0, ...
    'string',' Gradual plotting','position',[60 105 120 20], ...
    'fontsize',8);

h_pbres = uicontrol('style','pushbutton', ...
    'string','Reset Data','position',[180 105 80 20], ...
    'fontsize',8,'callback','plogui8(''res'')');

h_pbsvim = uicontrol('style','pushbutton', ...
    'string','Save Image','position',[180 80 80 25], ...
    'fontsize',8,'callback','plogui8(''sfg'')');

handles = guihandles(h_fig);
handles.fig = h_fig;
handles.funced = h_funced;
handles.edxmn = h_edxmn;
handles.edxmx = h_edxmx;
handles.edymn = h_edymn;
handles.edymx = h_edymx;
handles.edmgf = h_edmgf;
handles.edpst = h_edpst;
handles.pumcm = h_pumcm;
handles.pumres = h_pumres;
handles.chbrtr = h_chbrtr;
handles.edzsc = h_edzsc;
handles.edrtx = h_edrtx;
handles.slrtx = h_slrtx;
handles.edshd = h_edshd;
handles.chbshd = h_chbshd;
handles.slshd = h_slshd;
handles.chbhlr = h_chbhlr;
handles.stsmsg = h_stsmsg;
handles.pbsvim = h_pbsvim;
```

```
handles.curfig = -1;
handles.figin = 0;
guidata(h_fig,handles);

% shadow slider change
elseif strcmp(plginp,'chbshdCH')
    handles = guidata(gcbo);
    if (get(handles.chbshd,'value')==0)
        set(handles.slshd,'enable','off');
        set(handles.edshd,'enable','off');
    else
        set(handles.slshd,'enable','on');
        set(handles.edshd,'enable','on');
    end
% slider change
elseif strcmp(plginp,'slshdCH')
    handles = guidata(gcbo);
    slshdVal = str2num(get(handles.edshd,'string'));
    if ~length(slshdVal)
        slshdVal = (get(handles.slshd,'max')
+get(handles.slshd,'min'))/2;
    end
    slshdVal = min([slshdVal get(handles.slshd,'max')]);
    slshdVal = max([slshdVal get(handles.slshd,'min')]);

set(handles.edshd,'string',num2str( fix(get(handles.slshd,'value'))) )
;
% next to slider box change
elseif strcmp(plginp,'edshdCH')
handles = guidata(gcbo);
    slshdMn = get(handles.slshd,'min');
    slshdMx = get(handles.slshd,'max');
    edshdVal = str2num(get(handles.edshd,'string'));
    if (~length(edshdVal) | (edshdVal<slshdMn) | (edshdVal>slshdMx) )
        edshdVal = get(handles.slshd,'value');

set(handles.edshd,'string',num2str(fix(get(handles.slshd,'value'))));
    end
    set(handles.slshd,'value',edshdVal);

%axonometric angle change
elseif strcmp(plginp,'slxCH')
    handles = guidata(gcbo);
    slxVal = str2num(get(handles.edrtx,'string'));
    if ~length(slxVal)
        slxVal = (get(handles.slrtx,'max')
+get(handles.slrtx,'min'))/2;
    end
    slxVal = min([slxVal get(handles.slrtx,'max')]);
    slxVal = max([slxVal get(handles.slrtx,'min')]);

set(handles.edrtx,'string',num2str( fix(get(handles.slrtx,'value'))) )
;
% next to slider box change
elseif strcmp(plginp,'edrxCH')
```

```
    handles = guidata(gcbo);
    slxMn = get(handles.slrtx,'min');
    slxMx = get(handles.slrtx,'max');
    edrxVal = str2num(get(handles.edrtx,'string'));
    if (~length(edrxVal) | (edrxVal<slxMn) | (edrxVal>slxMx) )
        edrxVal = get(handles.slrtx,'value');
set(handles.edrtx,'string',num2str(fix(get(handles.slrtx,'value'))));
    end
    set(handles.slrtx,'value',edrxVal);

% reset
elseif strcmp(plginp,'res')
    handles = guidata(gcbo);
    set(handles.funced,'string','100 .* ( 3*(1-x).^2.*exp(-(x.^2) -
       (y+1).^2)- 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2)-
            1/3*exp(-(x+1).^2 - y.^2) )');
    set(handles.edxmn,'string','-320');
    set(handles.edxmx,'string','320');
    set(handles.edymn,'string','-240');
    set(handles.edymx,'string','240');
    set(handles.edmgf,'string','350');
    set(handles.edzsc,'string','100');
    set(handles.pumcm,'value',1);
    set(handles.pumres,'value',1);
    set(handles.chbrtr,'value',0);
    set(handles.edrtx,'string','20');
    set(handles.slrtx,'value',20);
    set(handles.chbshd,'value',1);
    set(handles.slshd,'enable','on');
    set(handles.edshd,'enable','on');
    set(handles.edshd,'string','30');
    set(handles.slshd,'value',30);
    set(handles.chbhlr,'value',1);
    set(handles.edpst,'string',5);
    set(handles.stsmsg,'string','Values have been reset');

% save image
elseif strcmp(plginp,'sfg')
    handles = guidata(gcbo);

    if ishandle(handles.curfig)
        set(handles.stsmsg,'string','Saving image to specified
folder');
        [filnam,pthnam]=uiputfile('*.png','Accepted filetype is:
PNG');
        if filnam~=0
            saveas(handles.curfig,filnam,'png')
            set(handles.stsmsg,'string','Image has been saved');
        else
            set(handles.stsmsg,'string','No filename was specified');
        end
    else
            set(handles.stsmsg,'string', ...
                  'Problem emerged while saving image');
            msgbox('No figure exists to create image file', ...
                        'Problem: saving image','error');
```

```
    end

% render
elseif strcmp(plginp,'rend')
        handles = guidata(gcbo);
        set(handles.stsmsg,'string', ...
            'Please, wait while calculating function values');
        pause(0.01);
        mgf = str2num(get(handles.edmgf,'string'));
        ang = str2num(get(handles.edrtx,'string'));
        rtr = get(handles.chbrtr,'value');
        pumcm = get(handles.pumcm,'value');

        switch pumcm
            case 1
                pmc_str='Jet';
            case 2
                pmc_str='HSV';
            case 3
                pmc_str='Hot';
            case 4
                pmc_str='Cool';
            case 5
                pmc_str='Spring';
            case 6
                pmc_str='Summer';
            case 7
                pmc_str='Autumn';
            case 8
                pmc_str='Winter';
            case 9
                pmc_str='Gray';
            case 10
                pmc_str='Bone';
            case 11
                pmc_str='Copper';
            case 12
                pmc_str='Pink';
            case 13
                pmc_str='Lines';
        end
        cm=pmc_str;

        pumres = get(handles.pumres,'value');
        switch pumres
            case 1
                resx=640;
                resy=480;
            case 2
                resx=800;
                resy=600;
            case 3
                resx=1024;
                resy=768;
            case 4
                resx=1280;
```

```
                resy=800;
            case 5
                resx=1400;
                resy=1050;
            case 6
                resx=1600;
                resy=1200;
            case 7
                resx=1920;
                resy=1080;
        end

        if (get(handles.chbshd,'value')==1)
            shdi=str2num(get(handles.edshd,'string'));
        else
            shdi=0;
        end

        hlr = get(handles.chbhlr,'value');
        xmn = str2num(get(handles.edxmn,'string'));
        xmx = str2num(get(handles.edxmx,'string'));
        ymn = str2num(get(handles.edymn,'string'));
        ymx = str2num(get(handles.edymx,'string'));
        pst = str2num(get(handles.edpst,'string'));
        zsc = str2num(get(handles.edzsc,'string'));
        [x,y]=meshgrid((xmn:pst:xmx)*(1/mgf),(ymn:pst:ymx)*(1/mgf));
        funcstr=get(handles.funced,'string');
        z=(zsc.*eval(funcstr));

    if ishandle(handles.curfig)
        close(handles.curfig);
    end

    h_curfig = figure('Name','Plotter Output','Numbertitle','Off');
    handles.curfig = h_curfig;
    handles.figin = handles.figin + 1;
    guidata(handles.fig,handles);

    hold on;
    plotter8cm,mgf,pst,zsc,ang,rtr,resx,resy,shdi,hlr, ...

      xmn,xmx,ymn,ymx,x,y,z);
    set(handles.stsmsg,'string','Function has been plotted');

else
    disp(' ');
    disp('Please run the application with NO input');
    disp('    or give "initialize" inside simple quotes.');
    disp('Thanks');
    disp(' ');
end

%
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Kyriakopoulos Argyrios mppl11025                                           %
%                                                                             %
%  Dissertation on Msc Informatics                                            %
%  University of Piraeus                                                      %
%  (c) 2015-2016                                                              %
%                                                                             %
%  3D Plotter                                                                 %
%                                                                             %
%  Files: Plogui8.m, Plotter8.m, Picture.m, Showimg.m, Plogui8.mat            %
%                                                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function [ img1 ] = plotter(
      cm,mgf,pst,zsc,ang,rtr,resx,resy,shdi,hlr,xmn,xmx,ymn,ymx,x,y,z)
%
x=x';y=y';z=z';
x=fliplr(x);
y=fliplr(y);
z=fliplr(z);

z=z+abs(min(z(:)))+1;

for i=1:(size(x(:)))
        yi=2*cos(ang)*y(i); yj=2*sin(ang)*y(i);
        gx(i)=yi+x(i); gy(i)=yj+z(i);
end

    gxmn = min(gx); gxmx = max(gx);
    gymn = min(gy); gymx = max(gy);
    lx = (gxmx-gxmn);
    ly = (gymx-gymn);
    gy=gy+abs(gymn);
    gx=gx+abs(gxmn);
    Y=fix((gy./(ly/(resy-1)))+1);
    X=fix((gx./(lx/(resx-1)))+1);

if (rtr==1)
    h_tmp=gcf;
    close(h_tmp);
    img=zeros(resy,resx);
    h_out=figure;
    set(h_out,'resize','off','Name','Plotter Output', ...
                                          'Numbertitle','Off');
    set(gcf,'BackingStore','on','DockControls','off', ...
                                          'DoubleBuffer','on');
    set(gca,'Position',[0 0 1 1]);
    set(gcf,'MenuBar','none','Units','pixels', ...
                                  'Position',[100 100 resx resy]);

    axis equal;
    axis off;
    z=fix(z./(max(z(:))/64));
    mp=colormap(cm);
    mp(1,:)=0;
    colormap(mp);
```

```
end
        mx=(1:resx);
        mx(:)=min(Y(:));
        mn=(1:resx);
        mn(:)=max(Y(:));

    for i=1:size(z(:))
        [r,c]=ind2sub(size(z),i);

        if (Y(i) > mx(X(i))) mx(X(i)) = Y(i); sh=0; end
        if (Y(i) < mn(X(i))) mn(X(i)) = Y(i); sh=z(i)*(shdi/100); end

        if (hlr==0)
            img( Y(i),X(i) ) = z(i)-sh;
        elseif ((Y(i) >= mx(X(i))) | (Y(i) <= mn(X(i))))
            img( Y(i),X(i) ) = z(i)-sh;
        end

        if (rtr==1)&(r==1)
            image(flipud(img)); pause(0.0005);
        end
    end


if (rtr==0)
        mxc=max(img(:));
        img=fix(img./(mxc/118));

        showimg(img,cm);
else
        msgbox('          Step by step plotting is finished
', ...
            'Output Status','modal');
end

%
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Kyriakopoulos Argyrios mppl11025                                           %
%  Dissertation on Msc Informatics                                            %
%  University of Piraeus                                                      %
%  (c) 2015-2016                                                              %
%                                                                             %
%  3D Plotter                                                                 %
%                                                                             %
%  Files: Plogui8.m, Plotter8.m, Picture.m, Showimg.m, Plogui8.mat            %
%                                                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%

function [h] = showimg (img,mp)
%
cm=colormap(bone(128));

set(gcf,'BackingStore','on','DockControls','off','DoubleBuffer','on');
set(gcf,'resize','off','MenuBar','none','Pointer','crosshair');

hold on;
[m,n] = size(img);
set(gca, 'Position', [0 0 1 1]);
set(gcf,'Units', 'pixels', 'Position', [100 100 n m]);

axis equal;
axis off;

image(img);
mp=colormap(mp);
mp(1,:)=0;
colormap(mp);

%
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Kyriakopoulos Argyrios mppl11025                                           %
%  Dissertation on Msc Informatics                                            %
%  University of Piraeus                                                      %
%  (c) 2015-2016                                                              %
%                                                                             %
%  3D Plotter                                                                 %
%                                                                             %
%  Files: Plogui8.m, Plotter8.m, Picture.m, Showimg.m, Plogui8.mat            %
%                                                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function [h] = picture (a,b)
%
load('Plogui8.mat');
image(a);
colormap(b);

hold on;
[m,n,p] = size(a);
set(gca, 'Position', [0 0 1 1]);
set(gcf,'Units', 'pixels', 'Position', [200 350 n m]);

axis equal;
axis off;

%
end
```

ΣΥΜΠΕΡΑΣΜΑΤΑ

Το MatLab(R) αποτελεί ένα πολύ ισχυρό εργαλείο για εργασία στον τομέα των μαθηματικών και μάλιστα στον τομέα των εφαρμοσμένων μαθηματικών.

Η υπολογιστική ταχύτητα του MatLab(R) είναι ίσως η καλύτερη από οποιοδήποτε λογισμικό του χώρου, ανοικτό ή εμπορικό.

Στην πραγματικότητα δεν πρόκειται για ένα πακέτο γραφικών με υπολογιστικές δυνατότητες, αλλά για μια γλώσσα προγραμματισμού, η οποία εκφράζει με τον πιο φυσικό τρόπο τα υπολογιστικά προβλήματα που απαντώνται στις εφαρμοσμένες επιστήμες.

Το MatLab(R) παρέχει συναρτήσεις υψηλού επιπέδου, αλλά δίνει επίσης τη δυνατότητα στον χρήστη να προγραμματίσει μόνος του τις ρουτίνες του, με τον τρόπο που εκείνος επιθυμεί.

Το παραθυρικό περιβάλλον προσφέρεται με μεγάλη ευκολία, ώστε τελικά ο χρήστης μπορεί να επικεντρωθεί στα υπολογιστικά προβλήματα που τον απασχολούν, χωρίς να περισπάται η προσοχή του σε δευτερεύοντα ζητήματα, που δεν σχετίζονται με τον κορμό της επίλυσης που επιχειρεί, αλλά απλώς υποστηρίζουν τις διάφορες λειτουργίες του ηλεκτρονικού υπολογιστή.

Η εφαρμογή που με ευκολία αναπτύχθηκε στο τελευταίο μέρος του βιβλίου προσφέρει μια αδρή εικόνα των δυνατοτήτων του MatLab(R). Με ένα πρόγραμμα συνολικά περίπου 500 γραμμών δίνεται στον χρήστη η δυνατότητα να αναπαραστήσει τρισδιάστατα τις δικές του γραφικές παραστάσεις.

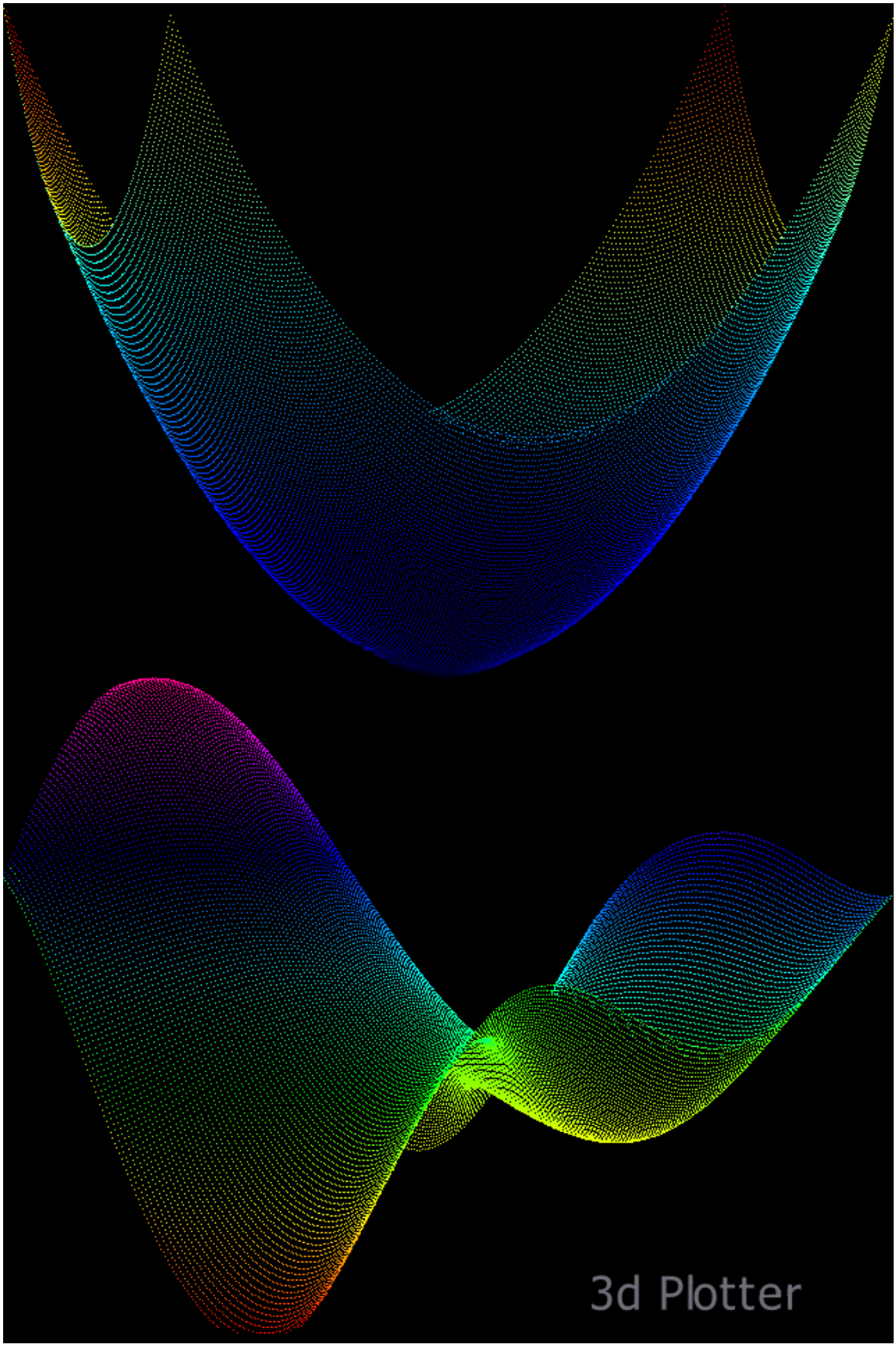
Το πρόγραμμα “3d Plotter” μπορεί να επεκταθεί περαιτέρω. Μπορεί να αποκτήσει λειτουργίες όπως αποθήκευση του περιβάλλοντος εργασίας, δηλαδή καταγραφή των τρεχουσών ρυθμίσεων σε αρχείο, τρισδιάστατη περιστροφή του τριαξονίου και συνεπώς και της γραφικής παράστασης, αναπαράσταση του τριαξονίου και της βαθμονόμησής τους μαζί με την συνάρτηση.

Ασφαλώς, οι δυνατότητες του MatLab(R) δεν σταματούν σχεδόν πουθενά. Σε αυτή την περιορισμένη εργασία, των περίπου 150 σελίδων, αναγκαστήκαμε να αναφερθούμε μόνο στα βασικά περί των γραφικών των μαθηματικών που προσφέρει το πακέτο MatLab(R). Φροντίσαμε όμως να καλύψουμε όλο το υλικό που χρειάζεται για να παρακολουθήσει κανείς τη δημιουργία της εφαρμογής “3d Plotter”.

Τέλος, θα θέλαμε να αναφέρουμε πως η εφαρμογή “3d Plotter” αναπτύχθηκε με σκοπό αποκλειστικά την απόκτηση γνώσης και σε καμία περίπτωση δεν διεκδικεί αναγνώριση εμπορικής εφαρμογής.

Ευχαριστούμε για την ανάγνωση.

Νοέμβριος 2016.


3d Plotter