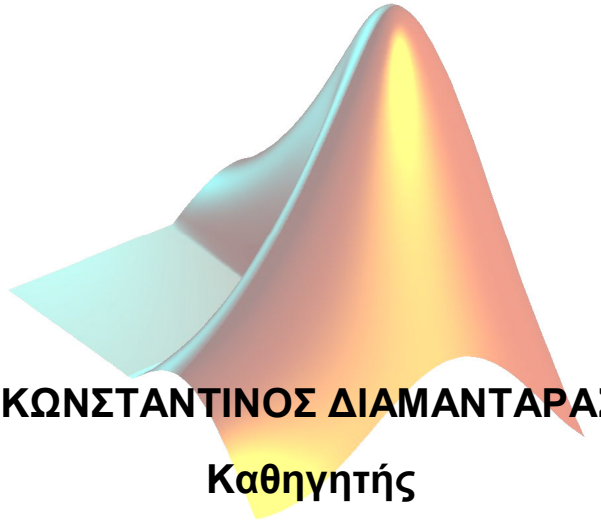
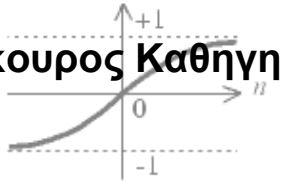
ΕΡΓΑΣΤΗΡΙΑΚΕΣ ΑΣΚΗΣΕΙΣ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ
ΚΑΙ
ΣΥΝΤΟΜΟ ΕΓΧΕΙΡΙΔΙΟ MATLAB



ΚΩΝΣΤΑΝΤΙΝΟΣ ΔΙΑΜΑΝΤΑΡΑΣ

Καθηγητής

ΚΩΝΣΤΑΝΤΙΝΟΣ ΓΟΥΛΙΑΝΑΣ



$a = tansig(n)$

Επίκουρος Καθηγητής

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ

Περιεχόμενα

ΜΕΡΟΣ Α: Εργαστηριακές Ασκήσεις Νευρωνικών Δικτύων

ΜΕΡΟΣ Β: ΣΥΝΤΟΜΟ ΕΓΧΕΙΡΙΔΙΟ MATLAB

8. Γλώσσα προγραμματισμού του Matlab
- 8.1. Βασικά χαρακτηριστικά
- 8.2. Scripts
- 8.3. Matlab Search Path
- 8.4. Μεταβλητές
- 8.5. Διανύσματα, πίνακες, cells
- 8.6. Βασική επεξεργασία πινάκων και διανυσμάτων M-Files
  - 8.6.1. Τελεστής του ανάστροφου
  - 8.6.2. Το εσωτερικό γινόμενο δύο διανυσμάτων
  - 8.6.3. Πράξεις με διανύσματα και πίνακες
- 8.7. Συναρτήσεις rand, randn, ones, zeros, eye
- 8.8. Λογικοί τελεστές και if-then-else
- 8.9. Βρόγχοι
- 8.10. Δομές(structures)
- 8.11. Συναρτήσεις(functions)
- 8.12. Είσοδος από το πληκτρολόγιο
- 8.13. Επεξεργασία αρχείων
  - 8.13.1. Δυαδικά αρχεία
  - 8.13.2. Αρχεία κειμένων
- 8.14. Interpreter και Compiler
- 8.15. Debugger

9. Γραφικές παραστάσεις
- 9.1. 2Δ (plot, quiver, feather)
  - 9.1.1. Plot
  - 9.1.2. Quiver
  - 9.1.3. Feather
- 9.2. 3Δ (mesh, meshgrid, meshc, contour, surf)
  - 9.2.1. Meshgrid
  - 9.2.2. Mesh
  - 9.2.3. Contour
  - 9.2.4. Meshc
  - 9.2.5. Surf
- 9.3. Είδη γραμμών και συμβολών που χρησιμοποιούνται στα plots
- 9.4. Subplot
- 9.5. Plotyy
- 9.6. Hold on/off
- 9.7. Pause

10. Ανάπτυξη γραφικών εφαρμογών
- 10.1. Παρουσίαση του GUIDE (GUI Layout Editor)
- 10.2. Παρουσίαση και δημιουργία gui αντικειμένων
  - 10.2.1. Menu bar
  - 10.2.2. Default Dialogs
- 10.3. Αποθήκευση και έναρξη εφαρμογής
- 10.4. Fig και m files του figure
- 10.5. Ιδιότητες των gui αντικειμένων
- 10.6. Δέντρο αντικειμένων(handles, findall, `guihandles`)
- 10.7. Οι εντολές `set`, `get`. Διάβασμα και αλλαγή των ιδιοτήτων από το command line
- 10.8. Callbacks
- 10.9. Flexarrays (ActiveX control)

11. Neural Network Toolkit
- 11.1. Παρουσίαση και δυνατότητες

ΜΕΡΟΣ Α

Εργαστηριακές Ασκήσεις Νευρωνικών Δικτύων

ΚΩΝΣΤΑΝΤΙΝΟΣ ΓΟΥΛΙΑΝΑΣ

ΚΕΦΑΛΑΙΟ 1

Ο Τεχνητός Νευρώνας

1.1 Εισαγωγή

Σύμφωνα με το μοντέλο των McCulloch και Pitts η **έξοδος** του νευρώνα δίνεται από τη σχέση

$$y = f(u-\theta)$$

όπου

θ το ***κατώφλι*** (***threshold***)

$u = \sum_{i=1}^{n} w_i x_i = w^T \cdot x$ η **δικτυακή διέγερση** του νευρώνα

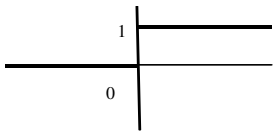
με

$\mathbf{w} = [w_1, ..., w_n]^T$ το διάνυσμα των **συναπτικών** βαρών
$\mathbf{x} = [x_1, ..., x_n]^T$ το διάνυσμα των **εισόδων**

Η **συνάρτηση ενεργοποίησης** (***neuron activation function***) f είναι μια συνάρτηση μιας εισόδου και μιας εξόδου και μπορεί να είναι μια από τις παρακάτω:

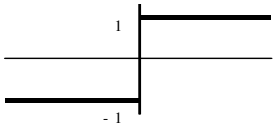
Βηματική 0/1
(step function 0/1)

$$f(u) = \begin{cases} 0, & \text{αν } u \le 0 \\ 1, & \text{αν } u > 0 \end{cases}$$



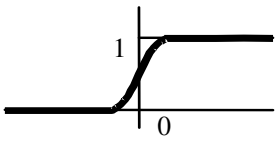
Βηματική –1/1
(step function –1/1)

$$f(u) = \begin{cases} -1, & \text{αν } u \le 0 \\ 1, & \text{αν } u > 0 \end{cases}$$



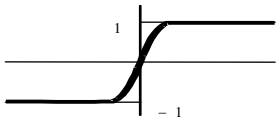
Σιγμοειδής
(sigmoid)

$$f(u) = \frac{1}{1+e^{-u}}$$



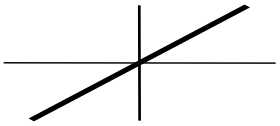
Υπερβολική εφαπτομένη
(hyperbolic tangent):

$$f(u) = tanh(u) = \frac{1-e^{-u}}{1+e^{-u}}$$

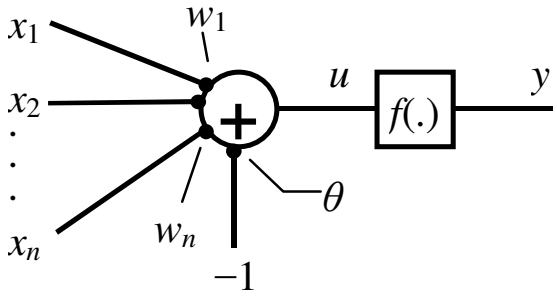


Γραμμική
(linear)

$$f(u) = u$$



Σχηματικά το παραπάνω μαθηματικό μοντέλο παριστάνεται από ένα αθροιστή ακολουθούμενο από ένα μη-γραμμικό μετασχηματιστή f όπως φαίνεται στο επόμενο σχήμα :



Παρατήρηση

- Το κατώφλι θ είναι ένας πραγματικός αριθμός (θετικός ή αρνητικός) όπως επίσης και τα συναπτικά βάρη w_1, ..., w_n. Επομένως, το **κατώφλι θ** μπορεί να θεωρηθεί σαν ένα **επί πλέον συναπτικό βάρος** w_{n+1} (το οποίο αποκαλείται **πόλωση**) συνδεδεμένο με μια **σταθερή είσοδο x_{n+1}** η οποία έχει πάντα την τιμή **-1**. Έτσι θα μπορούσαμε να γράψουμε

$$u = \sum_{i=1}^{n} w_i x_i - \theta = \sum_{i=1}^{n+1} w_i x_i$$

όπου $w_{n+1} = \theta$ και $x_{n+1} = -1$, οπότε θα έχουμε :

$\boldsymbol{w} = [w_1, w_2, ..., w_{n+1}]^T$ το διάνυσμα των **συναπτικών** βαρών με την πόλωση
$\boldsymbol{x} = [x_1, x_2, ..., x_n, -1]^T$ το διάνυσμα των **εισόδων** με τη σταθερή είσοδο -1

1.2 Προσομοίωση Τεχνητού Νευρώνα με κατώφλι

(Άσκηση 1α)

Να γίνει **πρόγραμμα** - ***script*** στο ***Matlab*** που να προσομοιώνει την παραπάνω διαδικασία. Πιο αναλυτικά το ***script ask1a.m*** θα κάνει τα παρακάτω :

1. **Διαβάζει** τον **αριθμό** των Εισόδων n

2. **Διαβάζει** τον κατώφλι *theta*

3. **Δημιουργεί** με τη συνάρτηση *randn* τυχαίες τιμές στις συνάψεις $\mathbf{w} = [w_1, ..., w_n]^T$

4. **Δημιουργεί** με τη συνάρτηση *rand* τυχαίες τιμές στις εισόδους $\mathbf{x} = [x_1, ..., x_n]^T$

5. **Υπολογίζει** τη διέγερση $u = \sum_{i=1}^{n} w_i x_i = w^T \cdot x$

6. **Υπολογίζει** το νέο $u = u - \theta = \sum_{i=1}^{n} w_i x_i - \theta = w^T \cdot x - \theta$ αφαιρώντας το κατώφλι θ

7. **Εμφανίζει** το παρακάτω ***menu*** επιλογών

```
1. step 0/1
2. step -1/1
3. sigmoid
4. hyperbolic tangent
5. linear
0. Telos

Give choice (0..5) :
```

8. Ο χρήστης θα **δίνει** μια επιλογή choice

9. Αν η **επιλογή** είναι **< 0 ή > 5**, θα εμφανίζει μήνυμα **λάθους**

10. Αν η **επιλογή** είναι **0**, θα τερματίζει το ***menu*** επιλογών

11. Αν η **επιλογή** είναι **μεταξύ 1 και 5**, θα χρησιμοποιεί την αντίστοιχη συνάρτηση υπολογισμού του ***y =f(u)***

12. Θα εμφανίζει την τιμή του ***y =f(u),***αν η επιλογή είναι **μεταξύ 1 και 5**.

Παρατηρήσεις

- Για τα Βήματα 1 και 2 θα χρησιμοποιηθεί η εντολή input
- Για το Βήμα 5 θα πρέπει να υπολογισθεί το Εσωτερικό Γινόμενο $u = w^T \cdot x$

- Αν τα a, b είναι δύο διανύσματα-στήλες (δηλαδή έχουν δηλωθεί σαν a(n,1) και b(n,1)) τότε το εσωτερικό τους γινόμενο ab θα είναι :

```
ab = a'*b
```

- Αν και τα δύο είναι διανύσματα-γραμμές, (δηλαδή έχουν δηλωθεί σαν a(1,n) και b(1,n)) τότε το a πρέπει να παραμείνει ως έχει και το b να αναστραφεί

```
ab = a*b'
```

- Μπορεί να χρησιμοποιηθεί η συνάρτηση dot του MATLAB :

```
ab = dot(a,b)
```

➢ Για το Βήμα 7 θα πρέπει να χρησιμοποιηθεί η εντολή

```
fprintf('1. step01\n 2. step-11\n…');
```

➢ Για τα Βήματα 9, 10, 11 θα πρέπει να χρησιμοποιηθεί η εντολή switch-case, η οποία θα έχει την παρακάτω μορφή :

```
switch (choice)
case 1
   εντολές-step01;
case 2
   εντολές-step-11;
case 3
   εντολές-sigmoid;
case 4
   εντολές-hyperbolic tangent;
case 5
   εντολές-linear;
case 0
   μήνυμα-τέλους;
otherwise
   μήνυμα -λάθους;
end;
```

➢ Για το Βήμα 12 θα πρέπει να χρησιμοποιηθεί ο τελεστής & για την εντολή if με διπλή συνθήκη ($1 \le choice \le 5$).

1.3 Προσομοίωση Τεχνητού Νευρώνα με πόλωση

(Άσκηση 1b)

➢ Να τροποποιηθεί η Άσκηση 1a, ώστε αντί για κατώφλι να χρησιμοποιεί πόλωση, οπότε το u θα είναι :

$$u = \sum_{i=1}^{n+1} w_i x_i,\ w_{n+1} = \theta, x_{n+1} = -1$$

Παρατηρήσεις

- Δεν θα υπάρχουν τα Βήματα 2 και 6.

- Στα Βήματα 3 και 4 θα πρέπει να δημιουργηθούν οι πίνακες $\boldsymbol{w} = [w_1, w_2, ..., w_{n+1}]^T$ και $\boldsymbol{x} = [x_1, x_2, ..., x_n, -1]^T$

1.4 Προσομοίωση Τεχνητού Νευρώνα με πόλωση & χρήση συναρτήσεων - functions

(Άσκηση 1c)

➢ Να τροποποιηθεί η Άσκηση 1b, ώστε να χρησιμοποιεί τις παρακάτω συναρτήσεις - functions :

Αρχείο	*Συνάρτηση MATLAB*	*Μαθηματική Συνάρτηση*
`step01.m`	`step01`	Βηματική 0/1
`step11.m`	`step11`	Βηματική –1/1
`sigmoid.m`	`sigmoid`	Σιγμοειδής

Παρατηρήσεις

- Η συνάρτηση `tanh` δίνεται έτοιμη στο ίδιο το MATLAB ενώ η γραμμική συνάρτηση περιττεύει (δε χρειάζεται να υλοποιηθεί).
- Η συνάρτηση μπορεί να δεχτεί σαν είσοδο αριθμό ή διάνυσμα.
- Το όνομα της συνάρτησης θα πρέπει να είναι το ίδιο με το όνομα του αρχείου. Π.χ. το αρχείο **`step01.m`** θα περιέχει τα παρακάτω :

```
function y = step01(u)
if ( u > 0 )
  y = 1;
else
  y = 0;
end;
```

1.5 Γραφικές Παραστάσεις των Συναρτήσεων Ενεργοποίησης

(Άσκηση 1d)

➢ Να γίνει πρόγραμμα που να εμφανίζει τη γραφική παράσταση των συναρτήσεων `step01`, `step11`, `sigmoid`, `tanh`, και `linear` χρησιμοποιώντας για είσοδο κάθε συνάρτησης το διάνυσμα `x = -5:0.1:5.`

Παρατηρήσεις

- Η συνάρτηση `plot` δέχεται σαν ορίσματα δύο διανύσματα και για το κάθε ζεύγος τιμών απεικονίζει στο επίπεδο το αντίστοιχο σημείο.
- Τα δύο διανύσματα πρέπει να έχουν τον ίδιο αριθμό στοιχείων. Αν π.χ. ορίσουμε το `x` σαν `x = -5:0.1:5,` το διάνυσμα `x` θα αποτελείται από μια γραμμή

με τα στοιχεία −5, −4.9, . . .,4.9,5, τα οποία μπορούμε να μετρήσουμε με την εντολή

```
[lines columns] = size(x);
```

η οποία επιστρέφει τον αριθμό γραμμών (lines) και στηλών (columns) του πίνακα x.

- Η συνάρτηση `plot` δέχεται σαν όρισμα και ένα string με το σύμβολο της απεικόνισης, το είδος και το χρώμα της γραμμής. Π.χ. `plot(x,y,'b-+')`.

- Μπορούμε να βάλουμε στο γράφημα Τίτλο και Ονομασία του Άξονα x και y. Π.χ.

```
xlabel('x axis');
title( 'Grafhma Synarthsewn Energopoihshs','fontsize',14 );
```

- Μπορούμε να χρησιμοποιήσουμε το ίδιο γράφημα και για τις 5 γραφικές παραστάσεις. Αυτό γίνεται με την εντολή `hold on` μετά από κάθε εντολή `plot`.

1.6 Γραφικές Παραστάσεις των Συναρτήσεων Ενεργοποίησης με Μια Εντολή `plot`

(Άσκηση 1e)

➢ Να τροποποιηθεί η άσκηση `1d`, ώστε να εμφανίζει τη γραφική παράσταση των συναρτήσεων `step01`, `step11`, ...`linear` με μια εντολή `plot`.

Παρατηρήσεις

- Για κάθε συνάρτηση θα χρησιμοποιείται διαφορετικό `y ( y1, y2,... )`.
- Όλα τα ζεύγη `(x, y1), (x, y2),...` θα υπάρχουν σε μια εντολή `plot`.

ΚΕΦΑΛΑΙΟ 2

Perceptron

2.1 Εισαγωγή

- Το **perceptron** αποτελείται από ένα νευρώνα τύπου McCulloch και Pitts με `n` **εισόδους** και μία **έξοδο**.
- Χρησιμοποιεί σαν Συνάρτηση Ενεργοποίησης τη `step function.`
- Μπορεί να **διαχωρίζει** πρότυπα 2 κλάσεων οι οποίες είναι **γραμμικά διαχωρίσιμες**.
- Εκτός από τα **πρότυπα** χρειάζονται και **στόχοι** (`0` για την πρώτη κλάση, `1` για τη δεύτερη).
- Είναι ένα δίκτυο που εκπαιδεύεται με **επίβλεψη**.
- Στην εκπαίδευση εισάγονται τα πρότυπα με τη σειρά.
- Η εισαγωγή όλων των προτύπων με τη σειρά αποκαλείται **`εποχή`**.
- Η έξοδος συγκρίνεται με τον αντίστοιχο στόχο και διορθώνονται οι συνάψεις.
- Οι συνάψεις τροποποιούνται σύμφωνα με τον **Κανόνα Δέλτα** (Delta Rule) :

$$\mathbf{w} = \mathbf{w} + \beta\,(\mathbf{d} - \mathbf{y})\,\mathbf{x}$$

όπου :

$\mathbf{w} = [w_1, w_2, ..., w_{n+1}]^T$ = το διάνυσμα των **συναπτικών** βαρών
$\mathbf{x} = [x_1, x_2, ..., x_n, -1]^T$ = το πρότυπο που εισάγεται κάθε φορά
`d` = ο στόχος, η κλάση στην οποία ανήκει το πρότυπο με τιμές `0, 1`.
`y` = η έξοδος του νευρώνα με τιμές `0, 1`.

- Η εκπαίδευση τελειώνει, όταν δεν διορθώνονται πλέον οι συνάψεις.

2.2 Διαχωρισμός Γραμμικά Διαχωρίσιμων Κλάσεων με Perceptron

(Άσκηση 2α)

Να γίνει **πρόγραμμα** - ***script*** στο ***Matlab*** που να διαχωρίζει στο επίπεδο τα πρότυπα 2 (δύο) **Γραμμικά Διαχωρίσιμων** Κλάσεων με ένα **Perceptron** 2 εισόδων (3 με την πόλωση). Πιο αναλυτικά το ***script ask2a.m*** θα κάνει τα παρακάτω :

1. **Διαβάζει** τον **αριθμό** των Προτύπων `n ( άρτιος αριθμός )`
2. **Διαβάζει** το Συντελεστή Εκπαίδευσης (Learning Rate) `beta`
3. **Διαβάζει** το Μέγιστο Αριθμό Επαναλήψεων `max_num_of_epochs`
4. **Δημιουργεί** με τη συνάρτηση `randn` τυχαίες τιμές στις συνάψεις $\mathbf{w} = [w_1, w_2, ..., w_{n+1}]^T$
5. **Δημιουργεί** με τη συνάρτηση `rand` τυχαίες τιμές για τα πρότυπα **p** (Το κάθε πρότυπο αποτελείται από 2 τιμές `x, y`: Για τα πρότυπα της $1^{ης}$ κλάσης `1,2,…,n/2`, θα πρέπει να ισχύει $1 \le$ `x, y` ≤ 3, ενώ για τα πρότυπα της $2^{ης}$ κλάσης `n/2+1, n/2+2,…,n`, θα πρέπει να ισχύει $7 \le$ `x, y` ≤ 9, έτσι ώστε να είναι γραμμικά διαχωρίσιμα**.**
6. **Δίνει** τις τιμές `0, 1` για τους στόχους $\mathbf{d} = [d_1, ..., d_n]^T$
7. **Εμφανίζει** το γράφημα των προτύπων των 2 κλάσεων
8. **Δίνει** αρχική τιμή στις εποχές `0`.
9. **Για όσο** (Γίνονται αλλαγές στις συνάψεις) **και** (εποχή < `max_num_of_epochs`)

   A. **Για κάθε πρότυπο `i = 1:n`**

      i. **Υπολογίζει** την έξοδο `y(i)`

      ii. **Αν** `( y(i) ≠ d(i) )`

      Διόρθωση των συνάψεων με τον Κανόνα του Δέλτα

      Τέλος Αν

   Τέλος Για

   B. **Εμφανίζει** το γράφημα των προτύπων των 2 κλάσεων, ανάλογα με την κλάση στην οποία κατατάσσεται

10. **Δημιουργεί** δύο τυχαία πρότυπα, ένα απ' την κάθε κλάση

11. **Για κάθε πρότυπο `i = 1:2`**

    i. **Υπολογίζει** την έξοδο `y(i)`

    ii. **Εμφανίζει** το πρότυπο και την κλάση στην οποία κατατάσσεται

Παρατηρήσεις

- Για τα Βήματα `1, 2` και `3` θα χρησιμοποιηθεί η εντολή `input`
- Για το Βήμα `5` θα πρέπει να χρησιμοποιηθεί η συνάρτηση `rand` και να προβληθούν οι τιμές που δημιουργούνται στο διάστημα `[0, 1]` στο διάστημα `[1, 3]` και `[ 7, 9]`. Αν $x = rand(n/2,2) \in [0,1]$ και $y \in [1,3]$, αν $y = ax + b$, τότε το `0` απεικονίζεται στο `1`

και το 1 στο 3.Παίρνουμε τις εξισώσεις $1 = a \cdot 0 + b$ και $3 = a \cdot 1 + b$ οπότε λύνοντας το σύστημα θα έχουμε $b = 1$ και $a = 2$, οπότε το y γίνεται $y = 2 * rand(n/2,2) + 1$ για τα πρότυπα της κλάσης 0.

- Για το Βήμα 6 θα πρέπει να χρησιμοποιηθούν οι συναρτήσεις zeros και ones.
- Για το Βήμα 9 μπορεί να χρησιμοποιηθεί μια μεταβλητή flag που θα ξεκινάει με την τιμή 0 και θα γίνεται 1, όταν γίνονται αλλαγές στις συνάψεις, ή μια μεταβλητή oldw που θα κρατάει τις προηγούμενες τιμές των συνάψεων και στο while θα συγκρίνεται το w με το oldw.
- Για το Βήμα 9B θα πρέπει να χρησιμοποιηθεί η συνάρτηση find που θα δώσει τους δείκτες των στοιχείων του πίνακα y που είναι 0 και 1 και να γίνει το γράφημα σύμφωνα μ' αυτά. Π.χ. η εντολή

```
Classa = find(y == 0);
```

θα επιστρέψει στον πίνακα classa τους δείκτες των στοιχείων του πίνακα y που είναι 0, οπότε στην εντολή plot θα παραστήσουμε με ένα σύμβολο όλα αυτά τα στοιχεία που ταξινομούνται στην κλάση 0.

Αν θέλουμε να σχεδιάσουμε και μια ευθεία που διαχωρίζει τα πρότυπα των 2 κλάσεων, χρησιμοποιούμε την εξίσωση ευθείας που είναι

$$w_1 \cdot x + w_2 \cdot y + w_3 = 0$$

οπότε λύνοντας ως προς y θα έχουμε

$$y = -(w_1 \cdot x + w_3)/w_2$$

Αν λοιπόν χρησιμοποιήσουμε τον πίνακα x = 1:10, αφού οι τιμές των προτύπων ανήκουν σ' αυτή την κλίμακα, μπορούμε στο ίδιο γράφημα, αφού σχεδιάσουμε τα πρότυπα, να σχεδιάσουμε και την ευθεία με τις παρακάτω εντολές :

```
plot(pats(classa,1),... );
hold on;
x = 1:10;
y = -(w1 . x + w3)/w2;
h = plot(x, y, 'EraseMode','none');
set(h, 'YData',y);
drawnow;
```

έτσι ώστε τα πρότυπα να μη σχεδιάζονται κάθε φορά στο γράφημα, αλλά να σχεδιάζεται μόνο η ευθεία που θα τα διαχωρίσει, όταν ολοκληρωθεί η εκπαίδευση.

2.3 Διαχωρισμός Μη-Γραμμικά Διαχωρίσιμων Κλάσεων με Perceptron και subplot

(Άσκηση 2b)

- Να τροποποιηθεί η άσκηση 2a, ώστε να διαχωρίζει στο επίπεδο τα πρότυπα 2 (δύο) **Γραμμικά Διαχωρίσιμων** Κλάσεων με ένα **Perceptron** 2 εισόδων (3 με την πόλωση) και 1 εξόδου, όπου τα 2 γραφήματα των Βημάτων 7 και 9B του Αλγορίθμου της Άσκησης 2a θα εμφανίζονται σε ένα με την εντολή subplot.

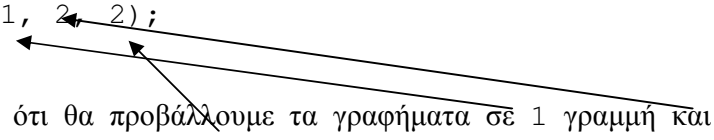
Παρατηρήσεις

- Με την εντολή `subplot` δηλώνουμε σε πόσες γραμμές και πόσες στήλες θα εμφανιστούν τα γραφήματα και το γράφημα στο οποίο αναφερόμαστε. Π.χ. με την εντολή :

```
Subplot(1, 2, 1);
```

Δηλώνουμε ότι θα προβάλλουμε τα γραφήματα σε `1` γραμμή και `2` στήλες και ότι αναφερόμαστε στο γράφημα `1`, ενώ με την εντολή :

```
Subplot(1, 2, 2);
```



Δηλώνουμε ότι θα προβάλλουμε τα γραφήματα σε `1` γραμμή και `2` στήλες και ότι αναφερόμαστε στο γράφημα `2`.

2.4 Διαχωρισμός Μη-Γραμμικά Διαχωρίσιμων Κλάσεων με Perceptron

(Άσκηση 2c)

- Να τροποποιηθεί η άσκηση `2b`, ώστε να διαχωρίζει (???) στο επίπεδο τα πρότυπα 2 (δύο) **Μη-Γραμμικά Διαχωρίσιμων** Κλάσεων με ένα **Perceptron** 2 εισόδων (3 με την πόλωση) και 1 εξόδου.

Παρατηρήσεις

- Για το Βήμα `5` θα πρέπει για τα πρότυπα της κλάσης `0` να χρησιμοποιηθούν τιμές : $1 \le x, y \le 3$ και $7 \le x, y \le 9$, ενώ για τα πρότυπα της κλάσης `1` να χρησιμοποιηθούν τιμές : $1 \le x \le 3, 7 \le y \le 9$ και $7 \le x \le 9, 1 \le y \le 3$.

2.5 Διαχωρισμός Γραμμικά Διαχωρίσιμων Κλάσεων με Perceptron 2 Εξόδων

(Άσκηση 2d)

- Να τροποποιηθεί η άσκηση `2b`, ώστε να διαχωρίζει στο επίπεδο τα πρότυπα 2 (δύο) **Γραμμικά Διαχωρίσιμων** Κλάσεων με ένα **Perceptron** 2 εισόδων (3 με την πόλωση) και 2 εξόδων.

Παρατηρήσεις

- Οι πίνακες **`W`** και **`D`** θα είναι δισδιάστατοι, ο **`W`** με `3` γραμμές και `2` στήλες και ο **`D`** με `n` γραμμές `2` στήλες.
- Ο στόχος για τα πρότυπα της $1^{ης}$ κλάσης θα είναι $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ και για τα πρότυπα της $2^{ης}$ κλάσης θα είναι $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

- Για τον υπολογισμό των διεγέρσεων u δεν θα έχουμε εσωτερικό γινόμενο, αλλά πολλαπλασιασμό πίνακα επί διάνυσμα, του πίνακα των συνάψεων `W` και της γραμμής του πίνακα των προτύπων (`u = W · p(i)`).

ΚΕΦΑΛΑΙΟ 3

Adaline

3.1 Εισαγωγή

- Το **Adaline**, όπως και το **Perceptron**, αποτελείται από ένα νευρώνα τύπου McCulloch και Pitts με n **εισόδους** και μία **έξοδο**.
- Χρησιμοποιεί σαν Συνάρτηση Ενεργοποίησης τη Γραμμική (Linear function) y = f(u) = u.
- Μπορεί να **διαχωρίζει** πρότυπα 2 κλάσεων οι οποίες είναι **γραμμικά ή μη γραμμικά διαχωρίσιμες**.
- Εκτός από τα **πρότυπα** χρειάζονται και **στόχοι** (0 για την πρώτη κλάση, 1 για τη δεύτερη).
- Όπως και το Perceptron, το Adaline είναι ένα δίκτυο που εκπαιδεύεται με **επίβλεψη**, ενώ στην εκπαίδευση εισάγονται τα πρότυπα με τη σειρά για κάθε **εποχή**.
- Οι συνάψεις τροποποιούνται σύμφωνα με τον **Κανόνα Δέλτα** (Delta Rule) :

$$\mathbf{w} = \mathbf{w} + \beta\,(\mathbf{d} - \mathbf{y})\,\mathbf{x}$$

όπου :

$\mathbf{w} = [w_1, w_2, ..., w_{n+1}]^T$ = το διάνυσμα των **συναπτικών** βαρών
$\mathbf{x} = [x_1, x_2, ... ,x_n, -1]^T$ = το πρότυπο που εισάγεται κάθε φορά
d = ο στόχος, η κλάση στην οποία ανήκει το πρότυπο με τιμές 0, 1.
y = η έξοδος του νευρώνα με τιμές συνεχείς.

- Η εκπαίδευση τελειώνει, όταν το Μέσο Τετραγωνικό Σφάλμα των Προτύπων πάρει μια επιθυμητή τιμή. Το Μέσο Τετραγωνικό Σφάλμα ορίζεται σαν :

$$mse = \frac{1}{\#\,of\ patterns} \sum_{i=1}^{\#\,of\ patterns} (d(i) - y(i))^2$$

3.2 Διαχωρισμός Γραμμικά Διαχωρίσιμων Κλάσεων με Adaline

(Άσκηση 3α)

Να γίνει **πρόγραμμα** - ***script*** στο ***Matlab*** που να διαχωρίζει στο επίπεδο τα πρότυπα 2 (δύο) Γραμμικά Διαχωρίσιμων Κλάσεων με ένα **Adaline** 2 εισόδων (3 με την πόλωση). Πιο αναλυτικά το ***script ask3a.m*** θα κάνει τα παρακάτω :

1. **Διαβάζει** τον **αριθμό** των Προτύπων *n (άρτιος αριθμός)*
2. **Διαβάζει** το Συντελεστή Εκπαίδευσης (Learning Rate) *beta*
3. **Διαβάζει** το Μέγιστο Αριθμό Επαναλήψεων max_num_of_epochs
4. **Διαβάζει** το Ελάχιστο Μέσο Τετραγωνικό Σφάλμα min_mean_squared_error
5. **Δημιουργεί** με τη συνάρτηση *randn* τυχαίες τιμές στις συνάψεις $\mathbf{w} = [w_1, w_2,..., w_{n+1}]^T$
6. **Δημιουργεί** με τη συνάρτηση *rand* τυχαίες τιμές για τα πρότυπα **p** (Το κάθε πρότυπο αποτελείται από 2 τιμές x, y: Για τα πρότυπα της $1^{ης}$ κλάσης 1,2,…,n/2, θα πρέπει να ισχύει $1 \le$ x, y ≤ 3, ενώ για τα πρότυπα της $2^{ης}$ κλάσης n/2+1, n/2+2,…,n, θα πρέπει να ισχύει $7 \le$ x, y ≤ 9, έτσι ώστε να είναι γραμμικά διαχωρίσιμα**.**
7. **Δίνει** τις τιμές 0, 1 για τους στόχους **d** = $[d_1, ..., d_n]^T$
8. **Εμφανίζει** το γράφημα των προτύπων των 2 κλάσεων
9. **Δίνει** αρχική τιμή στις εποχές 0.
10. **Για όσο** (Σφάλμα > min_mean_squared_error) **και** (εποχή < max_num_of_epochs)

    A. **Για κάθε πρότυπο i = 1:p**

       i. **Υπολογίζει** την έξοδο y(i) = u(i)

       ii. **Υπολογίζει** το delta(i) = d(i) – y(i)

       iii. **Προσθέτει** στο sfalma το delta(i)^2

       iv. **Διορθώνει** τις συνάψεις σύμφωνα με τον Κανόνα του Δέλτα

    Τέλος Για

    B. **Αυξάνει** την εποχή

    C. **Ενημερώνει** το mse

    D. **Εμφανίζει** με τη χρήση της εντολής subplot το **γράφημα** των **προτύπων** των 2 κλάσεων, ανάλογα με την κλάση στην οποία κατατάσσονται και το **γράφημα** του **Μέσου Τετραγωνικού Σφάλματος** στην κάθε εποχή.

11. **Για την Ανάκληση, Δημιουργεί** δύο τυχαία πρότυπα, ένα απ' την κάθε κλάση
12. **Για κάθε πρότυπο i = 1:2**

    i. **Υπολογίζει** την έξοδο y(i)

ii. **Εμφανίζει** το πρότυπο και την κλάση στην οποία κατατάσσεται.

Παρατηρήσεις

- Για το Βήμα `10` μπορεί να χρησιμοποιηθεί ένα `flag`, το οποίο θα ξεκινάει με την τιμή `0` και το οποίο θα γίνεται `1`, όταν το $mse = \frac{1}{n}\sum_{i=1}^{n}(d(i) - y(i))^2$ γίνει ≤ `min_mean_squared_error.`
- Για το Βήμα `10-A-iv` η διόρθωση των συνάψεων γίνεται κάθε φορά, χωρίς σύγκριση εξόδου – στόχου.
- Για το Βήμα `10-C` αποθηκεύουμε το `sfalma/n` στο `mse(epoch).`
- Για το Βήμα `10-D` μπορεί να χρησιμοποιηθεί η εντολή `plot( mse).`
- Για το Βήμα `10-D` θα πρέπει να χρησιμοποιηθεί η συνάρτηση `find` που θα δώσει τους δείκτες των στοιχείων του πίνακα `y` που είναι **κοντά** στο `0` και το `1` και να γίνει το γράφημα σύμφωνα μ' αυτά. Π.χ. η εντολή

  ```
  Classb = find( y > 0.5 );
  ```

  θα επιστρέψει στον πίνακα `classb` τους δείκτες των στοιχείων του πίνακα `y` που είναι > `0.5,` οπότε στην εντολή `plot` θα παραστήσουμε με ένα σύμβολο όλα αυτά τα στοιχεία που ταξινομούνται στην κλάση `1`.
- Παρόμοια συνθήκη θα χρησιμοποιηθεί και στο Βήμα `12-ii ( Αν y > 0.5`, το πρότυπο κατατάσσεται στην κλάση `1 )`.

3.3 Διαχωρισμός Μη-Γραμμικά Διαχωρίσιμων Κλάσεων με Adaline

(Άσκηση 3b)

- Να τροποποιηθεί η άσκηση `3a`, ώστε να διαχωρίζει στο επίπεδο τα πρότυπα 2 (δύο) **Μη-Γραμμικά Διαχωρίσιμων** Κλάσεων με ένα **`Adaline`** 2 εισόδων (3 με την πόλωση) και 1 εξόδου.

Παρατηρήσεις

- Για το Βήμα `5` θα πρέπει για τα πρότυπα της κλάσης `0` να χρησιμοποιηθούν τιμές : $1 \le x, y \le 3$ και $7 \le x, y \le 9$, ενώ για τα πρότυπα της κλάσης `1` να χρησιμοποιηθούν τιμές : $1 \le x \le 3, 7 \le y \le 9$ και $7 \le x \le 9, 1 \le y \le 3$.

3.4 Διαχωρισμός Γραμμικά Διαχωρίσιμων Κλάσεων με Adaline 2 Εξόδων (M-Adaline)

(Άσκηση 3c)

- Να τροποποιηθεί η άσκηση 3a, ώστε να διαχωρίζει στο επίπεδο τα πρότυπα 2 (δύο) **Γραμμικά Διαχωρίσιμων** Κλάσεων με ένα **Multiple-Adaline** 2 εισόδων (3 με την πόλωση) και 2 εξόδων.

Παρατηρήσεις

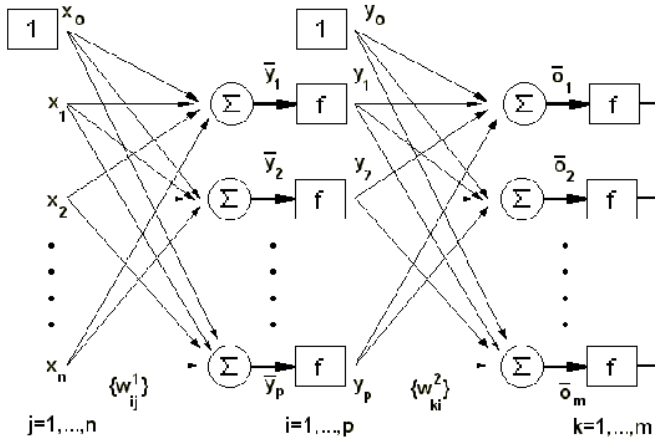
- Οι πίνακες **W** και **D** θα είναι δισδιάστατοι, ο **W** με 3 γραμμές και 2 στήλες και ο **D** με n γραμμές 2 στήλες.
- Ο στόχος για τα πρότυπα της $1^{ης}$ κλάσης θα είναι [1, 0] και για τα πρότυπα της $2^{ης}$ κλάσης θα είναι [0, 1].
- Για τον υπολογισμό των διεγέρσεων u δεν θα έχουμε εσωτερικό γινόμενο, αλλά πολλαπλασιασμό πίνακα επί διάνυσμα, του πίνακα των συνάψεων **W** και της γραμμής του πίνακα των προτύπων (**u = W · p(i)).**

ΚΕΦΑΛΑΙΟ 4

Back-Propagation

4.1 Εισαγωγή

- Το Δίκτυο **Back-Propagation** αποτελείται από το στρώμα **εισόδου** με n εισόδους $x_1, x_2, ..., x_n$, συν την εξωτερική διέγερση $x_0 = 1$, ένα τουλάχιστον **κρυφό** στρώμα με p νευρώνες και ένα στρώμα **εξόδου** με k νευρώνες, όπως φαίνεται στο επόμενο σχήμα :



- Χρησιμοποιεί **συνεχείς** Συναρτήσεις Ενεργοποίησης, όπως τη Σιγμοειδή, τη Γραμμική ή την Υπερβολική Εφαπτομένη.
- Μπορεί να **διαχωρίζει** πρότυπα 2 ή περισσότερων κλάσεων οι οποίες είναι **γραμμικά ή μη γραμμικά διαχωρίσιμες**.
- Εκτός από τα **πρότυπα** χρειάζονται και **στόχοι** (στην περίπτωση 2 κλάσεων 0 για την πρώτη κλάση, 1 για τη δεύτερη).
- Είναι ένα δίκτυο που εκπαιδεύεται με **επίβλεψη**.
- Στην **εκπαίδευση** :

a. Εισάγονται τα πρότυπα με τη σειρά (για κάθε **εποχή**), πρώτα στο κρυφό στρώμα, απ' το οποίο η **έξοδος** χρησιμοποιείται σαν **είσοδος** στο στρώμα εξόδου, απ' το οποίο βγαίνει και η **τελική** έξοδος. Πιο αναλυτικά :

- Υπολογίζεται η **διέγερση** και η **έξοδος** κάθε νευρώνα $i,\ i = 1,2,...,p$ του κρυφού στρώματος :

$$\bar{y}_i = \sum_{j=0}^{n} x_j w_{ij}^1, \quad y_i = f(\bar{y}_i), \quad i = 1,2,...,p$$

- Υπολογίζεται η **διέγερση** και η **έξοδος** κάθε νευρώνα $k, k = 1,2,...,m$ του στρώματος εξόδου :

$$\bar{o}_k = \sum_{i=0}^{p} y_i w_{ki}^2, \quad o_i = f(\bar{o}_i), \quad k = 1,2,...,m$$

b. Υπολογίζονται τα **Δέλτα** του στρώματος εξόδου και μετά τα **Δέλτα** του κρυφού στρώματος σύμφωνα με τους τύπους :

$\delta_k = (d_k - o_k) \cdot f'(\bar{o}_k), \quad k = 1,2,...,m$, τα **Δέλτα** του στρώματος εξόδου

$$\bar{\delta}_i = \left(\sum_{k=1}^{m} \delta_k w_{ki}^2 \right) \cdot f'(\bar{y}_i), \quad i = 1,2,...,p$$, τα **Δέλτα** του κρυφού στρώματος

c. Τροποποιούνται οι συνάψεις του στρώματος εξόδου και μετά του κρυφού στρώματος σύμφωνα με τους τύπους:

$w_{ki}^2 = w_{ki}^2 + beta \cdot \delta_k \cdot y_i = w_{ki}^2 + beta \cdot (d_k - o_k) \cdot f'(\bar{o}_k) \cdot y_i$, $k = 1,2,...,m$,
$i = 1,2,...,p$

$$w_{ij}^1 = w_{ij}^1 + beta \cdot \bar{\delta}_i \cdot x_j = w_{ij}^1 + beta \cdot \left(\sum_{k=1}^{m} \delta_k w_{ki}^2 \right) \cdot f'(\bar{y}_i) \cdot x_j,\ i = 1,2,...,p,\ j = 1,2,...,n$$

- Η εκπαίδευση τελειώνει, όταν το Μέσο Τετραγωνικό Σφάλμα των Προτύπων πάρει μια επιθυμητή τιμή. Το Μέσο Τετραγωνικό Σφάλμα ορίζεται σαν :

$$mse = \frac{1}{\#of\ patterns} \sum_{pat=1}^{\#of\ patterns} \sum_{i=1}^{m} (d_i^{(pat)} - o_i^{(pat)})^2$$

- Στην **ανάκληση** :

Εισάγονται τα πρότυπα με τη σειρά πρώτα στο κρυφό στρώμα, απ' το οποίο η **έξοδος** χρησιμοποιείται σαν **είσοδος** στο στρώμα εξόδου, απ' το οποίο βγαίνει και η **τελική** έξοδος, οπότε **εμφανίζει** το πρότυπο και την κλάση στην οποία κατατάσσεται.

4.2 Διαχωρισμός Μη Γραμμικά Διαχωρίσιμων Κλάσεων με Back-Propagation

(Άσκηση 4α)

Να γίνει **πρόγραμμα** - ***script*** στο ***Matlab*** που να διαχωρίζει στο επίπεδο τα πρότυπα 2 (δύο) **Μη Γραμμικά Διαχωρίσιμων** Κλάσεων με ένα Δίκτυο **Back-Propagation** 2 εισόδων (3 με την πόλωση) ενός κρυφού στρώματος με 2 νευρώνες και ενός στρώματος εξόδου με 1 νευρώνα. Πιο αναλυτικά το ***script ask4a.m*** θα κάνει τα παρακάτω :

1. **Διαβάζει** τον **αριθμό** των Προτύπων `n ( άρτιος αριθμός )`
2. **Διαβάζει** το Συντελεστή Εκπαίδευσης (Learning Rate) `beta`
3. **Διαβάζει** το Μέγιστο Αριθμό Επαναλήψεων `max_num_of_epochs`
4. **Δημιουργεί** με τη συνάρτηση `randn` τυχαίες τιμές στις συνάψεις $w2_i$, $i = 1,2$ και $w1_{ij}$, $i = 1,2$, $j = 1,2,3$ (Το δίκτυο θα έχει **2 εισόδους** συν την **πόλωση**, **2 νευρώνες** στο **κρυφό** στρώμα και **1 νευρώνα** στο στρώμα **εξόδου**).
5. **Δημιουργεί** με τη συνάρτηση `rand` τυχαίες τιμές για τα πρότυπα **p** (για τα πρότυπα της κλάσης `0` να χρησιμοποιηθούν τιμές : $1 \le x, y \le 3$ και $7 \le x, y \le 9$, ενώ για τα πρότυπα της κλάσης `1` να χρησιμοποιηθούν τιμές : $1 \le x \le 3, 7 \le y \le 9$ και $7 \le x \le 9, 1 \le y \le 3$, έτσι ώστε να είναι μη-γραμμικά διαχωρίσιμα**.**
6. **Δίνει** τις τιμές `0`, `1` για τους στόχους **d** = `[d1, ..., dn]'`
7. **Εμφανίζει** το γράφημα των προτύπων των `2` κλάσεων
8. **Δίνει** αρχική τιμή στις εποχές `0`.
9. **Για όσο** (Γίνονται αλλαγές στις συνάψεις) **και** (εποχή **<** `max_num_of_epochs`)

   A. **Για κάθε πρότυπο `i = 1:n`**

   - **Υπολογίζει** τη **διέγερση** `u(k)` και την **έξοδο** `y(k)` κάθε νευρώνα $k, k = 1,2$ του κρυφού στρώματος :

     $$u_k = p(i,:) \cdot w1(:,k), \quad y_k = f(u_k) = 1/(1 + e^{-u_k}), \quad k = 1,2$$

   - **Υπολογίζει** τη **διέγερση** `v` και την **έξοδο** `o` του νευρώνα εξόδου :

     $$v = y \cdot w2(1,:)', \quad o = f(v) = v$$

   - Υπολογίζονται τα **Δέλτα** του στρώματος εξόδου και μετά τα **Δέλτα** του κρυφού στρώματος σύμφωνα με τους τύπους :

     $\delta = (d_i - o) \cdot f'(v)$, το **Δέλτα** του στρώματος εξόδου

     $\delta 1_k = \delta \cdot w2_k \cdot f'(u_k), \quad k = 1,2$, τα **Δέλτα** του κρυφού στρώματος

   - Τροποποιούνται οι συνάψεις του στρώματος εξόδου και μετά του κρυφού στρώματος σύμφωνα με τους τύπους:

$$w2_j = w2_j + beta \cdot \delta \cdot y_j = w2_j + beta \cdot (d_i - o) \cdot f'(v) \cdot y_j, \quad j = 1,2$$

$$w1_{jk} = w1_{jk} + \delta 1_k \cdot p_i = w1_{jk} + beta \cdot \delta_k w2_{ki} \cdot f'(u_k), \quad i = 1,2,...,p, \quad j = 1,2,...,n$$

Τέλος Για

Β. **Εμφανίζει** το **γράφημα** των προτύπων των 2 κλάσεων, ανάλογα με την κλάση στην οποία κατατάσσεται και το **γράφημα** του **Μέσου Τετραγωνικού Σφάλματος** στην κάθε εποχή.

10. **Δημιουργεί** δύο τυχαία πρότυπα, ένα απ' την κάθε κλάση

11. **Για κάθε πρότυπο `i = 1:2`**

i. **Υπολογίζει** την έξοδο `o(i)`

ii. **Εμφανίζει** το πρότυπο και την κλάση στην οποία κατατάσσεται

Παρατηρήσεις

- Για το Βήμα `5` θα πρέπει για τα πρότυπα της κλάσης `0` να χρησιμοποιηθούν τιμές : $1 \le x, y \le 3$ και $7 \le x, y \le 9$, ενώ για τα πρότυπα της κλάσης `1` να χρησιμοποιηθούν τιμές : $1 \le x \le 3, 7 \le y \le 9$ και $7 \le x \le 9, 1 \le y \le 3$.

- Αν θέλουμε να σχεδιάσουμε και δύο ευθείες που να διαχωρίζουν τα πρότυπα των 2 κλάσεων, χρησιμοποιούμε δύο εξισώσεις ευθείας (μια εξίσωση για κάθε νευρώνα του κρυφού στρώματος) που είναι

$$w1_{11} \cdot x_1 + w1_{21} \cdot y_1 + w1_{31} = 0$$
$$w1_{12} \cdot x_2 + w1_{22} \cdot y_2 + w1_{32} = 0$$

οπότε λύνοντας ως προς y_1, y_2 θα έχουμε

$$y_1 = -(w1_{11} \cdot x_1 + w1_{31}) / w1_{21}$$
$$y_2 = -(w1_{12} \cdot x_2 + w1_{32}) / w1_{22}$$

- Στο Βήμα `9B` το **γράφημα** του **Μέσου Τετραγωνικού Σφάλματος** στην κάθε εποχή μπορεί να εμφανισθεί με την εντολή :

```
plot(mse);
```

- Στην Ανάκληση (Βήμα `11i`) κάνουμε ότι και στο Βήμα `9A`, **εκτός απ' τον Υπολογισμό των Δέλτα και την Τροποποίηση των Συνάψεων.**

4.3 Διαχωρισμός Γραμμικά Διαχωρίσιμων Κλάσεων με Back-Propagation 2 Εξόδων

(Άσκηση 4b)

- Να τροποποιηθεί η άσκηση `4a`, ώστε να διαχωρίζει στο επίπεδο τα πρότυπα 2 (δύο) **Μη Γραμμικά Διαχωρίσιμων** Κλάσεων με ένα Δίκτυο **Back-Propagation** 2 εισόδων (3 με την πόλωση) και 2 εξόδων.

Παρατηρήσεις

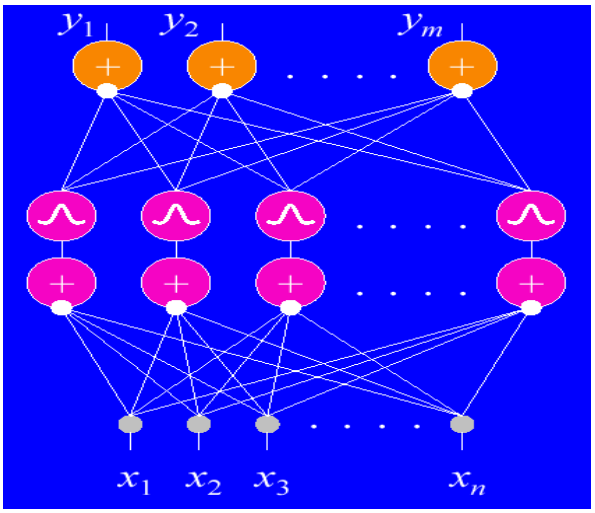
- Οι πίνακες **`W2`** και **`D`** θα είναι δισδιάστατοι, ο **`W2`** με `2` γραμμές και `2` στήλες και ο **`D`** με `n` γραμμές και `2` στήλες.
- Ο στόχος για τα πρότυπα της $1^{ης}$ κλάσης θα είναι $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ και για τα πρότυπα της $2^{ης}$ κλάσης θα είναι $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$.
- Για τον υπολογισμό των διεγέρσεων `v` δεν θα έχουμε εσωτερικό γινόμενο, αλλά πολλαπλασιασμό πίνακα επί διάνυσμα, του πίνακα των συνάψεων **`W2`** και του πίνακα `y` των εξόδων του κρυφού στρώματος.

ΚΕΦΑΛΑΙΟ 5

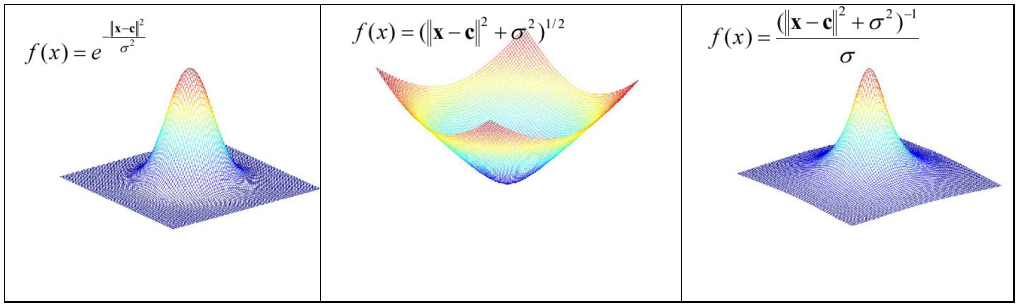
RBF-Δίκτυα Βάσης Ακτινικού Τύπου

Εισαγωγή

- Το Δίκτυο **RBF (Radial Basis Functions)** αποτελείται από το στρώμα **εισόδου** με n εισόδους $x_1, x_2, ..., x_n$, ένα **ενδιάμεσο** στρώμα με p νευρώνες και ένα στρώμα **εξόδου** με m νευρώνες, όπως φαίνεται στο επόμενο σχήμα :



- Μπορεί να **διαχωρίζει** πρότυπα 2 ή περισσότερων κλάσεων οι οποίες είναι **γραμμικά ή μη γραμμικά διαχωρίσιμες**.
- Εκτός από τα **πρότυπα** χρειάζονται και **στόχοι** (στην περίπτωση 2 κλάσεων 0 για την πρώτη κλάση, 1 για τη δεύτερη).
- Χρησιμοποιεί σαν Συναρτήσεις Ενεργοποίησης, τις συναρτήσεις ακτινικής βάσης (Radial Based Functions). Η τιμή των συναρτήσεων αυτών είναι συνάρτηση της απόστασης του διανύσματος εισόδου από ένα προκαθορισμένο κέντρο. Στα επόμενα σχήματα φαίνονται οι γραφικές παραστάσεις τριών συναρτήσεων RBF :



- Έστω π.χ. ότι μια συνάρτηση έχει κέντρο c. Τότε, αν x είναι η είσοδος (το όρισμα δηλαδή) της συνάρτησης, η έξοδος της συνάρτησης RBF είναι τόσο μεγαλύτερη όσο πιο κοντά είναι το x στο c. Δηλαδή, όσο η διαφορά x-c μικραίνει τόσο πιο μεγάλη έξοδος παράγεται. Η συνάρτηση RBF έχει δηλαδή την ενδογενή δυνατότητα να «αναγνωρίζει» εκείνα τα ορίσματα που είναι κοντά στο κέντρο της.
- Το πλάτος της συνάρτησης (spread), δηλαδή το πόσο γρήγορα ή αργά θα «πέφτει» δεξιά και αριστερά από το κέντρο μπορεί να καθοριστεί ελεύθερα. Συνήθως όμως φροντίζουμε τα πλάτη να είναι ίσα μεταξύ τους: $\sigma = \frac{max\_d}{\sqrt{2n}}$, όπου d το πλάτος του διαστήματος προτύπων και n το πλήθος τους.
- Το **ενδιάμεσο** στρώμα χρησιμοποιείται για την **ομαδοποίηση** των προτύπων, ανάλογα με τη θέση τους στο επίπεδο ή στο χώρο, με την **εύρεση** των **κέντρων** των **ομάδων**, του μέσου όρου δηλαδή των διανυσμάτων της κάθε ομάδας. Το κάθε πρότυπο-διάνυσμα εισόδου κατατάσσεται στην ομάδα εκείνη που απέχει τη **μικρότερη απόσταση** απ' το κέντρο της.
- Το τμήμα του δικτύου που αποτελείται απ' το ενδιάμεσο στρώμα και το στρώμα εξόδου εκπαιδεύεται με **επίβλεψη**.
- Για τους νευρώνες του στρώματος εξόδου χρησιμοποιείται η γραμμική συνάρτηση ενεργοποίησης.

- Στην **εκπαίδευση** :
  i. Για το κάθε πρότυπο υπολογίζεται η Συνάρτηση της απόστασης του προτύπου από το κάθε κέντρο, η οποία αποτελεί και την είσοδο στο ενδιάμεσο στρώμα..
  ii. Γίνεται η εκπαίδευση όπως και στο δίκτυο Adaline.

- Στην **ανάκληση** :
  i. Εισάγεται το πρότυπο για ταξινόμηση
  ii. Υπολογίζεται η Συνάρτηση της απόστασης του προτύπου από το κάθε κέντρο.
  iii. Γίνεται η ανάκληση όπως και στο δίκτυο Adaline.

5.2 Διαχωρισμός Γραμμικά Διαχωρίσιμων Κλάσεων με Δίκτυο RBF

(Άσκηση 5α)

Να γίνει **πρόγραμμα** - ***script*** στο ***Matlab*** που να διαχωρίζει στο επίπεδο τα πρότυπα 2 (δύο) **Γραμμικά Διαχωρίσιμων** Κλάσεων με ένα Δίκτυο **RBF** 2 εισόδων (η πόλωση θα χρησιμοποιηθεί στο ενδιάμεσο στρώμα) ενός ενδιάμεσου στρώματος με **τουλάχιστον** 2 νευρώνες και ενός στρώματος εξόδου με 1 νευρώνα. Πιο αναλυτικά το ***script ask5a.m*** θα κάνει τα παρακάτω :

1. **Διαβάζει** τον **αριθμό** των Προτύπων *n (άρτιος αριθμός)*
2. **Διαβάζει** τον **αριθμό** των Κέντρων *k (τουλάχιστον 2)*
3. **Διαβάζει** το Συντελεστή Εκπαίδευσης (Learning Rate) *beta*
4. **Διαβάζει** το Μέγιστο Αριθμό Επαναλήψεων max_num_of_epochs
5. **Δημιουργεί** με τη συνάρτηση *randn* τυχαίες τιμές στις συνάψεις w_j, $j = 1,2,...,k+1$. (Το δίκτυο θα έχει **k εισόδους** συν την **πόλωση** και **1 νευρώνα** στο στρώμα **εξόδου**).
6. **Δημιουργεί** με τη συνάρτηση *rand* τυχαίες τιμές για τα πρότυπα **p** (Το κάθε πρότυπο αποτελείται από 2 τιμές x, y: Για τα πρότυπα της $1^{ης}$ κλάσης 1,2,…,n/2, θα πρέπει να ισχύει $1 \le$ x, y ≤ 3, ενώ για τα πρότυπα της $2^{ης}$ κλάσης n/2+1, n/2+2,…,n, θα πρέπει να ισχύει $7 \le$ x, y ≤ 9, έτσι ώστε να είναι γραμμικά διαχωρίσιμα**.**
7. **Δίνει** τις τιμές 0, 1 για τους στόχους **d** = $[d_1, \ldots, d_n]^T$
8. **Εμφανίζει** το γράφημα των προτύπων των 2 κλάσεων.
9. Δημιουργούνται k τυχαία κέντρα $c(j,1:2),\ j = 1,2,...,k$
10. **Για όσο** τα κέντρα αλλάζουν (*c_old ~= c*)

    Α. Για κάθε πρότυπο $i = 1,2,...,n$

    i. Βρίσκει την **απόσταση** του προτύπου i απ' το κάθε κέντρο $j,\ j = 1,2,...,k$:

    $apostash(j) = norm(p(i,:) - c(j,:))\text{^}2$

    ii. Βρίσκει την **ελάχιστη απόσταση** του προτύπου i από όλα τα κέντρα $j,\ j = 1,2,...,k$ χρησιμοποιώντας τη συνάρτηση min, η οποία επιστρέφει την ελάχιστη απόσταση και το **δείκτη** του κέντρου με την ελάχιστη απόσταση, ο οποίος αποθηκεύεται στον πίνακα deiktes με την εντολή :

    ```
    [elaxisto deiktes(i)] = min(apostash);
    ```

    Β. Για κάθε πρότυπο $i = 1,2,...,n$

    i. Προσθέτει το πρότυπο i στον αντίστοιχο αθροιστή-κέντρο :

    ```
    c(deiktes(i),1:2)= c(deiktes(i),1:2)+p(i,1:2);
    ```

    ii. Αυξάνει τον αντίστοιχο μετρητή-κέντρο *count(deiktes(i))*

    Γ. Για κάθε κέντρο $j = 1,2,...,k$

    Αν ο αντίστοιχος μετρητής είναι $\neq 0$,

Βρίσκει το νέο κέντρο = αθροιστής/μετρητής

Δ. Εμφανίζει το γράφημα των προτύπων των 2 κλάσεων και των αντίστοιχων κέντρων με διαφορετικό χρώμα.

11. **Για κάθε** κέντρο *c(j):*
    i. **Υπολογίζει** τις αποστάσεις μεταξύ τους
    ii. **Βρίσκει** τη Μέγιστη Απόσταση *max_d* με τη συνάρτηση `max`
    iii. **Υπολογίζει** το $\sigma = \frac{max\_d}{\sqrt{2n}}$

12. **Για κάθε** πρότυπο *i = 1,2,...,n*

    Για κάθε κέντρο *c(j):*

    Βρίσκει την Απόσταση `x(i,j)` του προτύπου απ' το κάθε κέντρου με τη χρήση μιας RBF

13. **Δίνει** αρχική τιμή στις εποχές `0`.

14. **Για όσο** (`Σφάλμα > min_mean_squared_error`) **και** (`εποχή < max_num_of_epochs`)

    A. **Για κάθε πρότυπο `i = 1:n`**

    i. **Υπολογίζει** την έξοδο `y(i) = u(i)= x(i,:)* w`

    ii. **Υπολογίζει** το `delta(i) = d(i) - y(i)`

    iii. **Προσθέτει** στο `sfalma` το `delta(i)^2`

    iv. **Διορθώνει** τις συνάψεις σύμφωνα με τον Κανόνα του Δέλτα

    Τέλος Για

    B. **Αυξάνει** την εποχή

    C. **Ενημερώνει** το `mse`

    D. **Εμφανίζει** με τη χρήση της εντολής `subplot` το **γράφημα** των **προτύπων** των 2 κλάσεων **μαζί με τα κέντρα**, ανάλογα με την κλάση στην οποία κατατάσσονται και το **γράφημα** του **Μέσου Τετραγωνικού Σφάλματος** στην κάθε εποχή.

15. **Για την Ανάκληση, Δημιουργεί** δύο τυχαία πρότυπα, ένα απ' την κάθε κλάση

16. **Για κάθε πρότυπο `i = 1:2`**

    iii. **Υπολογίζει** την έξοδο `y(i)`

    ii. **Εμφανίζει** το πρότυπο και την κλάση στην οποία κατατάσσεται.

5.3 Διαχωρισμός Μη Γραμμικά Διαχωρίσιμων Κλάσεων με Δίκτυο RBF

(Άσκηση 5b)

- Να τροποποιηθεί η άσκηση 5a, ώστε να διαχωρίζει στο επίπεδο τα πρότυπα 2 (δύο) **Μη Γραμμικά Διαχωρίσιμων** Κλάσεων με ένα Δίκτυο **RBF** 2 εισόδων (η πόλωση θα χρησιμοποιηθεί στο ενδιάμεσο στρώμα), ενός ενδιάμεσου στρώματος με **τουλάχιστον** 4 νευρώνες και ενός στρώματος εξόδου με 1 νευρώνα.

Παρατηρήσεις

- Για το Βήμα 6 θα πρέπει για τα πρότυπα της κλάσης 0 να χρησιμοποιηθούν τιμές : $1 \le x, y \le 3$ και $7 \le x, y \le 9$, ενώ για τα πρότυπα της κλάσης 1 να χρησιμοποιηθούν τιμές : $1 \le x \le 3, 7 \le y \le 9$ και $7 \le x \le 9, 1 \le y \le 3$.
- Ο πίνακας **W** θα έχει 5 γραμμές και 1 στήλη.

5.4 Διαχωρισμός Μη Γραμμικά Διαχωρίσιμων Κλάσεων με RBF 2 Εξόδων

(Άσκηση 5c)

- Να τροποποιηθεί η άσκηση 5b ώστε να διαχωρίζει στο επίπεδο τα πρότυπα 2 (δύο) **Μη Γραμμικά Διαχωρίσιμων** Κλάσεων με ένα Δίκτυο **RBF** 2 εισόδων (η πόλωση θα χρησιμοποιηθεί στο ενδιάμεσο στρώμα) ενός ενδιάμεσου στρώματος με **τουλάχιστον** 4 νευρώνες και ενός στρώματος εξόδου με 1 νευρώνα. (3 με την πόλωση) ενός κρυφού στρώματος με **τουλάχιστον** 4 νευρώνες και ενός στρώματος εξόδου με 2 νευρώνες.

Παρατηρήσεις

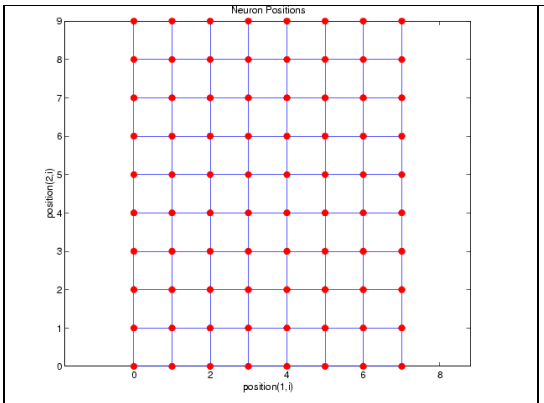
- Οι πίνακες **W** και **D** θα είναι δισδιάστατοι, ο **W** με 5 γραμμές και 2 στήλες και ο **D** με n γραμμές και 2 στήλες.
- Ο στόχος για τα πρότυπα της 1ης κλάσης θα είναι $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ και για τα πρότυπα της 2ης κλάσης θα είναι $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$.
- Για τον υπολογισμό των διεγέρσεων v δεν θα έχουμε εσωτερικό γινόμενο, αλλά πολλαπλασιασμό πίνακα επί διάνυσμα, του πίνακα των συνάψεων **W** και του πίνακα y των εξόδων του κρυφού στρώματος.

ΚΕΦΑΛΑΙΟ 6

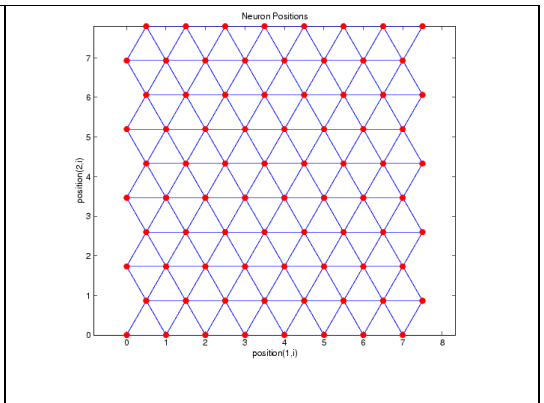
Αυτό-Οργανούμενοι Χάρτες
Self-Organizing Maps (SOM)

6.1 Εισαγωγή

- Το Δίκτυο **SOM** είναι ένα δίκτυο **χωρίς επίβλεψη**.
- Στηρίζεται στην **τοπογραφική οργάνωση** του εγκεφάλου.
- Συνήθως αποτελούνται από ένα **δισδιάστατο** πλέγμα από νευρώνες.
- Υπάρχουν δύο τρόποι να τοποθετηθούν οι νευρώνες στον χώρο, η **τετραγωνική** και η **εξαγωνική**.



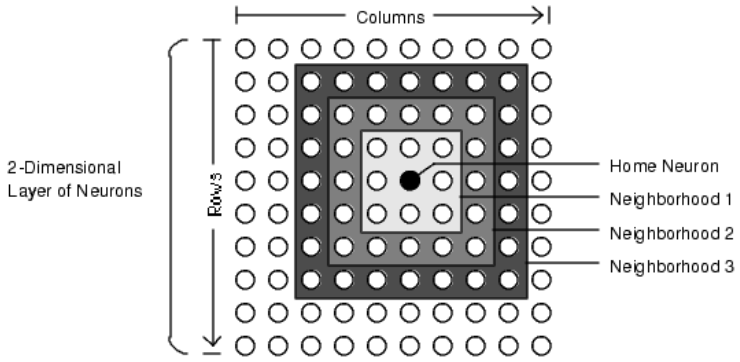
Τετραγωνική τοπολογία.



Εξαγωνική τοπολογία.

- Όταν εισάγεται ένα διάνυσμα εισόδου στο δίκτυο οι νευρώνες **ανταγωνίζονται** μεταξύ τους. **Νικητής (winner)** είναι εκείνος που ταιριάζει καλύτερα στο διάνυσμα εισόδου. Πιο αναλυτικά :
- :Υπολογίζεται η **Ευκλείδια** απόστασή του από όλους τους νευρώνες του δικτύου.

- Ο νευρώνας με την μικρότερη απόσταση ονομάζετει **winner**.
- Ο νευρώνας αυτός **μετακινείται** προς την κατεύθυνση του διανύσματος εισόδου.
- Το ίδιο γίνεται και για τους υπόλοιπους νευρώνες της **γειτονιάς** του.
- Το μήκος της μετακίνησης είναι ανάλογο του learning rate.
- Το learning rate στην αρχή είναι μεγάλο και μετά το μειώνουμε.
- Το ίδιο γίνεται και με το μέγεθος της γειτονιάς, όπως φαίνεται στο επόμενο σχήμα :.



- Στην ανάκληση : Όταν το SOM λάβει μια διέγερση ενεργοποιείται εκείνος ο νευρώνας που είναι πιο κοντά στο διάνυσμα εισόδου.

6.2 Διαχωρισμός Γραμμικά Διαχωρίσιμων Κλάσεων με Δίκτυο SOM

(Άσκηση 6α)

Να γίνει **πρόγραμμα** - ***script*** στο ***Matlab*** που να διαχωρίζει στο επίπεδο τα πρότυπα 2 (δύο) **Γραμμικά Διαχωρίσιμων** Κλάσεων με ένα Δίκτυο **SOM** 2 εισόδων (η πόλωση θα χρησιμοποιηθεί στο ενδιάμεσο στρώμα) ενός ενδιάμεσου στρώματος με **τουλάχιστον** 2 νευρώνες και ενός στρώματος εξόδου με 1 νευρώνα. Πιο αναλυτικά το ***script ask6a.m*** θα κάνει τα παρακάτω :

1. **Διαβάζει** τον **αριθμό** των Προτύπων *n (άρτιος αριθμός)*
2. **Διαβάζει** τον **αριθμό** των νευρώνων Kohonen στην κάθε γραμμή του δισδιάστατου πλέγματος neurons
3. **Διαβάζει** το Συντελεστή Εκπαίδευσης (Learning Rate) *beta*
4. **Διαβάζει** το Μέγιστο Αριθμό Επαναλήψεων max_num_of_epochs
5. **Διαβάζει** τον **αριθμό** των νευρώνων Kohonen στη γειτονιά
6. **Δημιουργεί** με τη συνάρτηση *randn* τυχαίες τιμές στις συνάψεις w_{ij}, $i = 1..2 \quad j = 1,2,...,neurons * neurons$. (Το δίκτυο θα έχει **k εισόδους** συν την **πόλωση** και **1 νευρώνα** στο στρώμα **εξόδου**).

7. **Δημιουργεί** με τη συνάρτηση *rand* τυχαίες τιμές για τα πρότυπα **p** (Το κάθε πρότυπο αποτελείται από 2 τιμές `x, y`: Για τα πρότυπα της $1^{ης}$ κλάσης `1,2,…,n/2`, θα πρέπει να ισχύει $1 \le$ `x, y` ≤ 3, ενώ για τα πρότυπα της $2^{ης}$ κλάσης `n/2+1, n/2+2,…,n`, θα πρέπει να ισχύει $7 \le$ `x, y` ≤ 9, έτσι ώστε να είναι γραμμικά διαχωρίσιμα**.**
8. **Δίνει** τις τιμές `0, 1` για τους στόχους **`d`** `= [d1, ..., dn]'`
9. **Εμφανίζει** το γράφημα των προτύπων των `2` κλάσεων.
10. **Για όσο** ο συντελεστής εκπαίδευσης δεν έχει μηδενισθεί και στη γειτονιά υπάρχουν περισσότεροι από ένας νευρώνες και (`epochs <= max_num_of_epochs`)

    Α. Για κάθε πρότυπο $i = 1,2,...,n$

    i. Βρίσκει την **ελάχιστη απόσταση** του προτύπου i απ' τον κάθε νευρώνα $j = 1,2,...,neurons * neurons$.

    ii. Μετακινεί κοντά στο πρότυπο εισόδου το νικητή νευρώνα και τους γείτονές του διορθώνοντας τις συνάψεις.

    Β. Εμφανίζει το γράφημα των προτύπων των `2` κλάσεων και του νικητή νευρώνα και των γειτόνων του με διαφορετικό χρώμα.

    C. **Αυξάνει** την εποχή

    D. **Ενημερώνει** το συντελεστή εκπαίδευσης

    Ε. **Ενημερώνει** τον αριθμό νευρώνων της γειτονιάς.

και το **γράφημα** του **Μέσου Τετραγωνικού Σφάλματος** στην κάθε εποχή.

11. **Για την Ανάκληση, Δημιουργεί** δύο τυχαία πρότυπα, ένα απ' την κάθε κλάση

12. **Για κάθε πρότυπο `i = 1:2`**

    i. **Υπολογίζει** την απόσταση του προτύπου από κάθε νευρώνα

    ii. **Εμφανίζει** το πρότυπο και την κλάση στην οποία κατατάσσεται.

ΜΕΡΟΣ Β

ΣΥΝΤΟΜΟ ΕΓΧΕΙΡΙΔΙΟ MATLAB

ΚΩΝΣΤΑΝΤΙΝΟΣ ΔΙΑΜΑΝΤΑΡΑΣ

ΚΕΦΑΛΑΙΟ 7

Εισαγωγή στο Matlab

7.1 Δυνατότητες

(Help: MATLAB->Getting Started ->Introduction->What Is MATLAB?)

Το MATLAB είναι ένα εργαλείο το οποίο αρχικά σχεδιάστηκε για μαθηματικό προγραμματισμό και το οποίο χρησιμοποιείται σήμερα ευρύτατα στην επιστημονική κοινότητα. Ωστόσο το MATLAB προσφέρει πάρα πολλές δυνατότητες στον προγραμματιστή, ώστε να μπορεί να θεωρηθεί πλέον ένα πολύ ισχυρό εργαλείο γενικού προγραμματισμού. Μερικά από τα κυριότερα χαρακτηριστικά του MATLAB περιγράφονται παρακάτω:

- Προγραμματισμός σε γλώσσα *scripting* που μοιάζει πολύ με τη γλώσσα *C*. Τα αρχεία script έχουν κατάληξη .m.
- Εκτέλεση εντολών σε command line από interpreter για γρήγορη δοκιμή ενός προγράμματος, μιας εφαρμογής, ή μιας ιδέας. Επίσης δυνατότητα εκτέλεσης ενός script από το command line γράφοντας απλώς το όνομα του αρχείου script με ή χωρίς την κατάληξη `.m`.
- Δυνατότητα δημιουργίας αρχείων exe με χρήση compiler.
- Εύκολη διαχείριση πινάκων (matrices) και διανυσμάτων (vectors).
- Ολοκληρωμένο περιβάλλον editor/debugger (medit.exe). Καλείται από το κεντρικό παράθυρο του MATLAB όταν ζητήσουμε να ανοίξουμε ή να δημιουργήσουμε ένα νέο αρχείο, μπορεί όμως να εκτελεστεί και ως ανεξάρτητο πρόγραμμα.
- Εξαιρετικές δυνατότητες δημιουργίας γραφικών παραστάσεων εύκολα και γρήγορα. Γραφικές παραστάσεις 2, και 3 διαστάσεων. Προχωρημένες δυνατότητες όπως 3-D φωτισμός, αλλαγή οπτικής γωνίας, δημιουργία εικονοσειρών, κα.
- Γραφικός προγραμματισμός. Δυνατότητα σχεδιασμού παραθύρων, κουμπιών, γραφικών μενού, κλπ. Πλήρης γκάμα δυνατοτήτων για σχεδιασμό Graphical User Interfaces (GUI).
- Εξαιρετικό εργαλείο βοήθειας Help. Με τόσες δυνατότητες που προσφέρει το MATLAB είναι πολύ εύκολο να ξεχάσεις τα ακριβή ορίσματα μιας συνάρτησης ή το

όνομα της συνάρτησης που εκτελεί μια συγκεκριμένη εργασία. Το εργαλείο Help του MATLAB είναι πλήρες, λεπτομερές, εύχρηστο και εύκολο στην αναζήτηση της πληροφορίας που χρειάζεται ο χρήστης, ενώ περιέχει παραδείγματα και demos.

7.2 Βιβλιοθήκες

(Help: MATLAB->Functions -- Categorical List)

Το MATLAB περιέχει μεγάλο πλήθος έτοιμων βιβλιοθηκών. Οι standard βιβλιοθήκες προσφέρουν έτοιμες συναρτήσεις για επεξεργασία αριθμών, διανυσμάτων, πινάκων, δημιουργία plots, κλπ. Υπάρχει επίσης μεγάλη σειρά ειδικών βιβλιοθηκών που λέγονται *εργαλειοθήκες* (*toolboxes*) και ειδικεύονται σε μια επιστημονική περιοχή. Χαρακτηριστικά τέτοια toolboxes είναι:

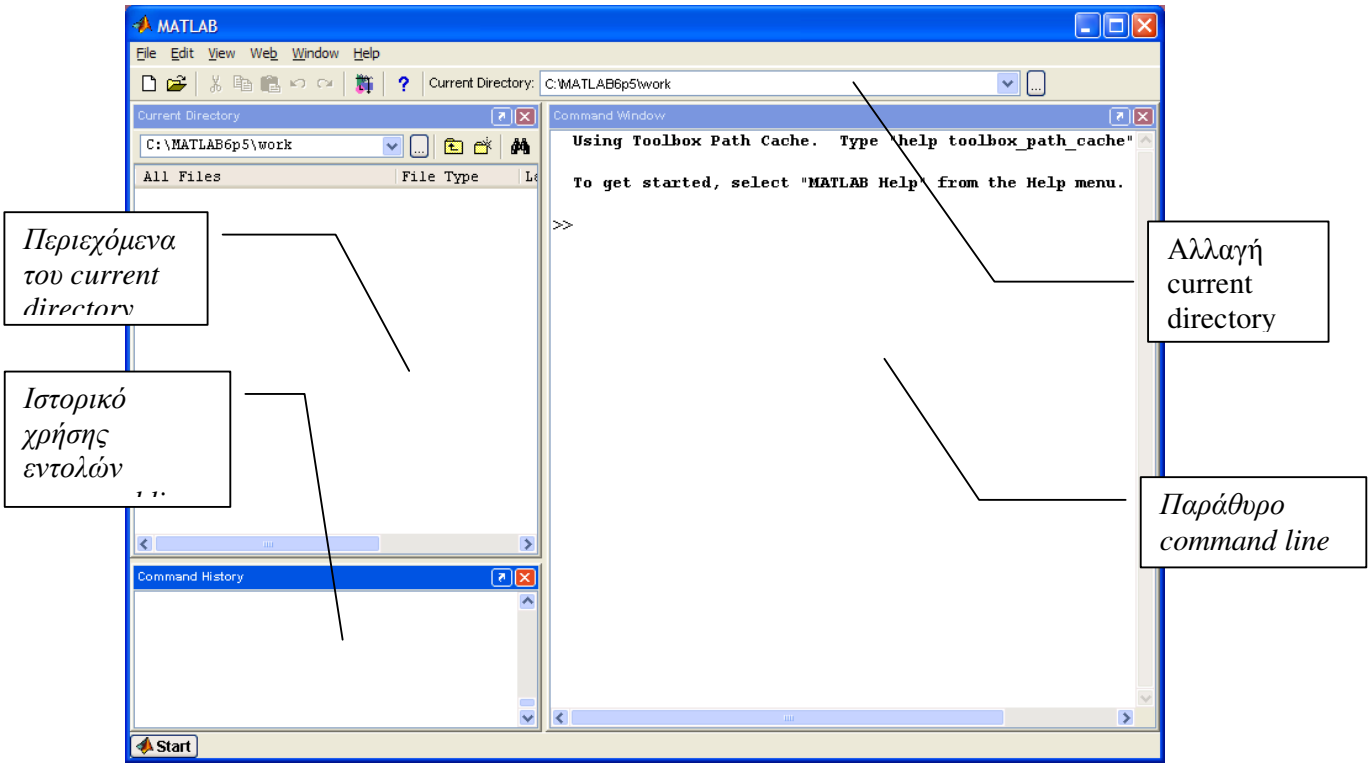
- Signal processing toolbox
- Image processing toolbox
- Neural networks toolbox
- Fuzzy Logic toolbox
- Statistics toolbox
- Optimization toolbox
- Communications toolbox
- Virtual Reality toolbox
- Database toolbox
- Control toolbox
- Symbolic math toolbox
- Financial toolbox
- Mapping toolbox
- Wavelet toolbox

και πολλά άλλα.

7.3 Περιβάλλον εργασίας

(Help: MATLAB->Getting Started ->Desktop Tools and Development Environment)

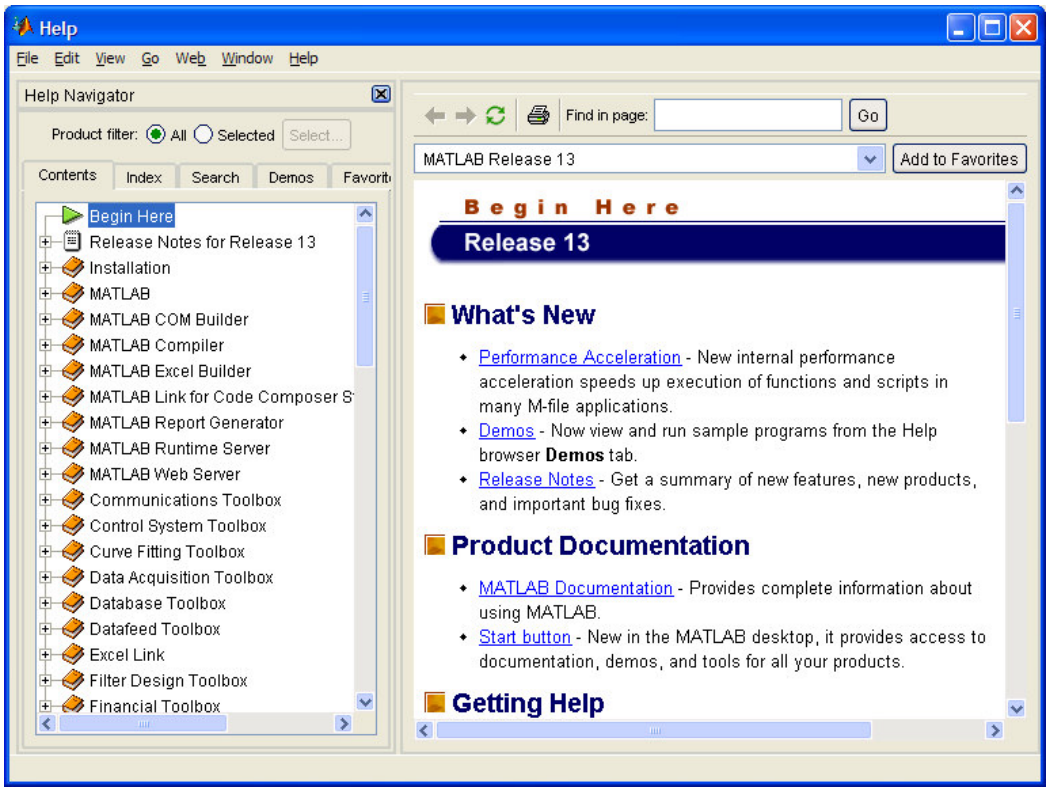
Η Εικόνα 1.1 δείχνει το βασικό παράθυρο εργασίας του MATLAB. Εκτός από τα εικονιζόμενα παράθυρα μπορεί κανείς να ανοίξει και άλλα χρησιμοποιώντας το μενού *View*. Άλλα χρήσιμα παράθυρα είναι τα *Workspace*, *Help*, και *Profiler*. Στο παράθυρο *Workspace* βλέπουμε τις μεταβλητές που έχουμε ορίσει και το χώρο που καταλαμβάνουν στη μνήμη.



Εικόνα 1.1

Ο *Profiler* είναι ένα χρήσιμο εργαλείο για τη βελτιστοποίηση του κώδικά μας. Για ένα συγκεκριμένο πρόγραμμα (script) μας δείχνει πόσος χρόνος CPU σπαταλήθηκε από κάθε γραμμή του κώδικα πόσες φορές εκτελέστηκε κάθε loop, κλπ.
Το παράθυρο *Help* είναι ένα ολοκληρωμένο περιβάλλον αναζήτησης πληροφορίας μέσα στο πλήθος των εργαλειοθηκών, των εντολών, και των συναρτήσεων του MATLAB (Εικόνα 1.2). Μέσα από το παράθυρο αυτό μπορεί κανείς να αναζητήσει πληροφορία

- κάνοντας πλοήγηση στον πίνακα περιεχομένων (*Contents*)
- μέσα από τον ονομαστικό κατάλογο των εντολών/συναρτήσεων (*Index*),
- μέσα από αναζήτηση με λέξεις κλειδιά (*Search*)
- μέσα από προγράμματα επίδειξης (*Demos*)


Help
File Edit View Go Web Window Help
Help Navigator
Product filter: All Selected Select...
Contents Index Search Demos Favorit
Begin Here
Release Notes for Release 13
Installation
MATLAB
MATLAB COM Builder
MATLAB Compiler
MATLAB Excel Builder
MATLAB Link for Code Composer S
MATLAB Report Generator
MATLAB Runtime Server
MATLAB Web Server
Communications Toolbox
Control System Toolbox
Curve Fitting Toolbox
Data Acquisition Toolbox
Database Toolbox
Datafeed Toolbox
Excel Link
Filter Design Toolbox
Financial Toolbox
Find in page:
Go
MATLAB Release 13
Add to Favorites
Begin Here
Release 13
What's New
Performance Acceleration - New internal performance acceleration speeds up execution of functions and scripts in many M-file applications.
Demos - Now view and run sample programs from the Help browser Demos tab.
Release Notes - Get a summary of new features, new products, and important bug fixes.
Product Documentation
MATLAB Documentation - Provides complete information about using MATLAB.
Start button - New in the MATLAB desktop, it provides access to documentation, demos, and tools for all your products.
Getting Help

Εικόνα 1.2

ΚΕΦΑΛΑΙΟ 8

Γλώσσα προγραμματισμού του Matlab

8.1 Βασικά χαρακτηριστικά

(Help: MATLAB->Programming)

Το MATLAB έρχεται εξοπλισμένο με μια γλώσσα προγραμματισμού που μοιάζει πολύ με τη γλώσσα C. Η γλώσσα αυτή μπορεί να χρησιμοποιηθεί είτε στο command line για να γράφουμε μια-μια τις εντολές που θέλουμε να εκτελέσουμε είτε σε ένα αρχείο script με την κατάληξη `.m`. Με το που ξεκινάμε το MATLAB στο παράθυρο command line εμφανίζεται το prompt

```
>>
```

έτοιμο να δεχτεί τις εντολές που θα δακτυλογραφήσουμε σε αυτό.

Γραμμές εντολών. Αν βρισκόμαστε στο command line μια γραμμή εντολών τελειώνει πατώντας <ENTER>. Αν γράφουμε κώδικα σε ένα αρχείο script μια γραμμή εντολών τελειώνει με new line. Αν μια εντολή είναι μεγάλη σε μήκος ή θέλουμε για οποιοδήποτε λόγο να την σπάσουμε σε πολλές γραμμές χρησιμοποιούμε τρεις τελείες. Οι τρεις τελείες σημαίνουν ότι η εντολή συνεχίζεται στην επόμενη γραμμή. Για παράδειγμα, ο παρακάτω κώδικας

```
fprintf('Τιμές: Δευτέρα=%d, Τρίτη=%d, Τετάρτη=%d\n', ...
        p1, ...
        p2, ...
        p3);
```

αντιστοιχεί στην εντολή

```
fprintf('Τιμές: Δευτέρα=%d, Τρίτη=%d, Τετάρτη=%d\n', p1, p2, p3);
```

Όπως το UNIX και η C, το MATLAB είναι ευαίσθητο στα μικρά και στα κεφαλαία γράμματα (case sensitive). Έτσι η μεταβλητή `A` δεν είναι ίδια με την μεταβλητή `a`.

Σχόλια. Τα σχόλια ξεκινούν με το σύμβολο `%` «επί τοις εκατό» και τελειώνουν στο τέλος της γραμμής. Μια ολόκληρη γραμμή είναι σχόλιο αν ξεκινάει με `%`.

Η χρήση του ; (semicolon). Το σύμβολο **;** στο τέλος μιας εντολής αποφεύγει το τύπωμα του περιεχομένου μιας μεταβλητής στην οθόνη. Για παράδειγμα, η εντολή

```
y = 3
```

θα κάνει την ανάθεση της τιμής 3 στο `y` και επί πλέον θα τυπώσει στην οθόνη το εξής μήνυμα:

```
y =
     3
```

ενώ η εντολή

```
y = 3;
```

θα κάνει την ανάθεση της τιμής 3 στο `y` χωρίς να τυπώσει τίποτα στην οθόνη. Περισσότερα στο *(Help: MATLAB->Programming->Basic Program Components -> Symbol Reference).*

Βασικές πράξεις
(Help: MATLAB->Programming->Basic Program Components->Operators)

Το MATLAB μπορεί να χρησιμοποιηθεί σαν μια απλή αριθμομηχανή χρησιμοποιώντας τις παρακάτω βασικές πράξεις(Πίνακας 1).

Σύμβολο	Πράξη
+	Πρόσθεση
-	Αφαίρεση
*	Πολλαπλασιασμός
/	Διαίρεση
^	Ύψωση σε δύναμη

Πίνακας 1

Παραδείγματα

```
>>  1 + 25.6

ans  =
      26.6

>>  8 - 15

ans =
```

```
        -7

>> 15 * 2

ans=
        30

>> 20 / 2

ans=
        10

>> 2 ^ 3

ans=
        8
```

Προτεραιότητα πράξεων
(Help: MATLAB->Programming->Basic Program Components->Operators->Operator Precedence)

Η προτεραιότητα των πράξεων στο MATLAB είναι ανάλογη με τους κανόνες προτεραιότητας των πράξεων της άλγεβρα.

1. Πρώτα εκτελούνται οι πράξεις στις παρενθέσεις από μέσα προς τα έξω.
2. Μετά εκτελούνται οι υψώσεις σε δύναμη.
3. Μετά εκτελούνται οι πολλαπλασιασμοί και οι διαιρέσεις από δεξιά προς τα αριστερά.
4. Τέλος εκτελούνται οι προσθέσεις και οι αφαιρέσεις από δεξιά προς τα αριστερά.

8.2 Scripts

(Help: Search for: script)

Ένα script είναι ένα εξωτερικό αρχείο που περιέχει μία σειρά από εντολές MATLAB. Με την πληκτρολόγηση του ονόματος του αρχείου στο command line εκτελούνται όλες οι εντολές του script με την σειρά. Τα scripts έχουν επέκταση ονόματος αρχείου .m και συνήθως ονομάζονται και M-Files. Τα script μπορούν να χρησιμοποιήσουν υπάρχοντα δεδομένα από το workspace ή να δημιουργήσουν νέα. Αν και τα script δεν επιστρέφουν τιμές όποια μεταβλητή δημιούργησαν παραμένει στο workspace.

8.3 Matlab Search Path

(Help: MATLAB->Programming->Programming Tips->MATLAB Path)

Το Matlab search path είναι τα directories στα οποία ψάχνει το MATLAB για υπάρχουσες μεταβλητές, functions, scripts, M-Files, αρχεία και άλλες δομές του MATLAB. Η σειρά κατά την οποία ψάχνει είναι η παρακάτω.

1. Μεταβλητές
2. Subfunctions
3. Private Functions

4. Class Constructors
5. Overloaded Methods
6. M-File που βρίσκεται στον παρόν κατάλογο
7. Τέλος, M-File που βρίσκεται στο path ή build-in functions του MATLAB

Παράδειγμα

Έστω ότι καλούμε ένα M-File με το όνομα trainnet.m.

```
>> trainnet
```

Αφού εκτελέσουμε αυτή την εντολή το MATLAB αρχικά ψάχνει στο search path για μία μεταβλητή με το όνομα trainnet, αν δεν βρει τότε ψάχνει για μία subfunction κτλ.

Το path είναι ο παρόν φάκελος(current directory) που ψάχνει το MATLAB. Στην Εικόνα 1.1 φαίνεται η θέση του current directory.

Για να εισάγουμε ένα φάκελο στο search path μπορούμε να το κάνουμε με δύο τρόπους, είτε από το toolbar File->Set Path είτε από το command window με την εντολή addpath.

```
>>addpath('c:\matlab_projects')
```

Τέλος για να δούμε όλο το search path του MATLAB πληκτρολογούμε την εντολή matlabpath στο command window.

8.4 Μεταβλητές

(Help: MATLAB->Programming-> Data Types)

Όπως και στην C, οι μεταβλητές μπορούν να έχουν διάφορους τύπους, για παράδειγμα double, char, string, logical, κλπ. Αντίθετα με την C ο τύπος των μεταβλητών του MATLAB που πρόκειται να χρησιμοποιηθούν παρακάτω στο πρόγραμμα δεν χρειάζεται να έχει δηλωθεί εκ των προτέρων. Έτσι πχ, η εντολή

```
x = 3;
```

δημιουργεί αυτομάτως μια μεταβλητή `x` στο χώρο της μνήμης του προγράμματος (workspace) που έχει τον τύπο double. Αντίστοιχα η εντολή

```
s = 'Hello!';
```

δημιουργεί αυτομάτως μια μεταβλητή τύπου string. Αν θέλουμε μπορούμε να αλλάξουμε τον τύπο μιας μεταβλητής δίνοντάς της μια τιμή άλλου τύπου, ασχέτως ποιος ήταν ο προηγούμενος τύπος της μεταβλητής. Το MATLAB αλλάζει δυναμικά τον τύπο της καθώς τρέχει το πρόγραμμα ανάλογα με τις εντολές μας. Για παράδειγμα, μπορούμε αρχικά να γράψουμε

```
a = 5.46;
```

οπότε η μεταβλητή `a` έχει την double τιμή 5.46 και κατόπιν μπορούμε να γράψουμε

```
a = 'King Kong';
```

οπότε η ίδια μεταβλητή γίνεται τύπου string με τιμή 'King Kong' (φυσικά η παλιά της τιμή τύπου double χάνεται).

8.5 Διανύσματα, πίνακες, cells

(*Help:* *MATLAB->Programming->Data Structures*)

Τα διανύσματα και οι πίνακες μπορούν πολύ εύκολα να αναπαρασταθούν στο MATLAB χωρίς να χρειάζεται να δηλωθούν εκ των προτέρων. Πχ, με την παρακάτω εντολή

```
d = [1, 1, 3, 0];
```

δημιουργείται αυτομάτως ένα διάνυσμα `d` μήκους 4 με στοιχεία τους double αριθμούς 1, 1, 3, και 0. Μπορούμε να προσπελάσουμε το στοιχείο `i` του διανύσματος `d` γράφοντας `d(i)`.

Αντίστοιχα με την εντολή

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

δημιουργείται ο πίνακας

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

Το «*κόμμα*» `,` διαχωρίζει τα στοιχεία ενός διανύσματος ή πίνακα που βρίσκονται στην ίδια γραμμή ενώ το *"semicolon"* `;` (*«ελληνικό ερωτηματικό»*) ενδεικνύει αλλαγή γραμμής. Μπορούμε να προσπελάσουμε το στοιχείο `(i,j)` του πίνακα `M` γράφοντας `M(i,j)`. Οι πίνακες πρέπει να αποτελούνται από γραμμές του ιδίου μήκους αλλιώς έχουμε σφάλμα.

Εντολή	Οθόνη
`vec = [1, 0, -2]`	`vec =` `1 0 -2`

`vec = [1; 0; -2]`	`vec =`
	`1`
	`0`
	`-2`

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

`M = [1, 2, 3; 4, 5]`	`??? Error using ==> vertcat` `All rows in the bracketed expression must have the same number of columns.`

Τα στοιχεία ενός διανύσματος ή ενός πίνακα δεν είναι απαραίτητα αριθμοί. Μπορεί να είναι, πχ, χαρακτήρες αφού οι χαρακτήρες αναπαριστώνται εσωτερικά με τους ASCII κωδικούς τους, δηλαδή με αριθμούς. Έτσι τα strings δεν είναι τίποτα άλλο παρά διανύσματα από χαρακτήρες. Παράδειγμα,

Εντολή	Οθόνη
`strvec = ['h','e','l','l','o']`	`strvec = hello`
`A = ['hello'; 'there']`	`A = hello there`

Στο παραπάνω παράδειγμα, ο πίνακας A έχει διαστάσεις 2x5:

$$A = \begin{bmatrix} h & e & l & l & o \\ t & h & e & r & e \end{bmatrix}$$

Δεν μπορούμε να δημιουργήσουμε έναν πίνακα με strings διαφορετικού μήκους το καθένα. Στην περίπτωση αυτή χρησιμοποιούμε τα «κελιά» "cells" όπως για παράδειγμα, στον παρακάτω κώδικα

Εντολή	Οθόνη
`C={'My','name','is','Bond.'}`	`C = 'My' 'name' 'is' 'Bond.'`
`C(1)`	`ans = 'My'`
`C(2)`	`ans = 'name'`
`C(3)`	`ans = 'is'`
`C(4)`	`ans = 'Bond.'`

Η μεταβλητή `C` είναι ένας πίνακας με διαστάσεις 1x4 όπου τα στοιχεία του είναι τύπου cell. Τα cells δεν είναι strings. Αν θέλουμε να χρησιμοποιήσουμε, πχ, το string `'name'` που αντιστοιχεί στο `C(2)` γράφουμε `C{2}` (προσέξτε τη διαφορά στις αγκύλες).

Δημιουργία διανύσματος από αριθμούς σε σειρά με αρχή (πχ. `a`), τέλος (πχ. `b`), και σταθερό βήμα (πχ. `step`). Αν παραβλέψουμε το βήμα τότε αυτό θεωρείται αυτομάτως ίσο με 1:

```
>> a=0; step=2; b=6;
>> x=a:step:b

x =
     0     2     4     6

>> x=a:b

x =
     0     1     2     3     4     5     6
```

```
>> y=35:-5:10

y =
      35    30    25    20    15    10
```

Σημειώστε ότι το MATLAB επιτρέπει επίσης την δημιουργία 3-διάστων, 4-διάστατων πινάκων, κλπ.

Επιλογή τμήματος πίνακα στον οποίο έχουμε ήδη δώσει τιμή. Έστω πχ., ότι κάποιος πίνακας `A` έχει διαστάσεις 5x10. Μπορούμε να επιλέξουμε τμήματα του πίνακα ως εξής:

```
>> A1=A(4,:)        %Επιλέγουμε μόνο τη γραμμή 4
                    %και όλες τις στήλες

>> A2=A(:,9)        %Επιλέγουμε όλες τις γραμμές
                    %και μόνο τη στήλη 9

>> A3=A(1:3,4:10)   %Επιλέγουμε τις γραμμές 1 έως 3
                    %και τις στήλες 4 έως 10
```

8.6 Βασική επεξεργασία πινάκων και διανυσμάτων

(*Help: MATLAB->Functions by category->Programming and Data Type->Operators and Operations*)

Το MATLAB προσφέρει βασικές λειτουργίες δημιουργίας και επεξεργασίας πινάκων. Τα διανύσματα είναι απλώς ειδικές περιπτώσεις πινάκων όπου η μια διάσταση έχει πλάτος 1. Μερικές από τις βασικές δυνατότητες δίνονται παρακάτω

- Οι πράξεις μεταξύ διανυσμάτων και πινάκων είναι απλούστατες: πχ,
  - πολλαπλασιασμός του διανύσματος `x` με τον πίνακα `A` με αποτέλεσμα το διάνυσμα `b`:
    ```
    b = A*x;
    ```
  - πολλαπλασιασμός σταθεράς `a` με το διάνυσμα `x` με αποτέλεσμα το διάνυσμα `y`:
    ```
    y = a*x;
    ```
  - πολλαπλασιασμός πίνακα `A` με τον πίνακα `B` με αποτέλεσμα τον πίνακα `C`:
    ```
    C = A*B;
    ```
  - αντιστροφή πίνακα `A (matrix inversion)`:
    ```
    B = inv(A);
    ```
  - πολύπλοκη πράξη πινάκων και διανυσμάτων
    ```
    z = inv(A+B)*x – A*B*y;
    ```
  - ανάθεση τιμής σε διάνυσμα:
    ```
    x = [1, -3, 2, 4, 0];
    ```
  - ανάθεση τιμής σε πίνακα 2 γραμμών & 3 στηλών:

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

Ο παραπάνω πίνακας είναι ο

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

8.6.1 Τελεστής του ανάστροφου

(*Help:Search For: Transpose)*

Στα μαθηματικά ένα διάνυσμα συμβολίζεται με ένα μικρό γράμμα boldface, πχ., **a**, **b**, **c**, **d**, ... και αποτελείται από *n* στοιχεία γραμμένα ανάμεσα σε τετράγωνες αγκύλες [ ]. Κατά πάγια σύμβαση των μαθηματικών θεωρείται ότι ένα διάνυσμα είναι μια **στήλη** και όχι μια γραμμή. Φυσικά μπορούμε να μετατρέψουμε τη στήλη σε γραμμή και τη γραμμή σε στήλη χρησιμοποιώντας τον τελεστή του **αναστρόφου** (**transpose**) που συμβολίζεται στα μαθηματικά με το γράμμα *T* υψωμένο σε εκθέτη. Άλλος συνηθισμένος συμβολισμός του αναστρόφου είναι ο τόνος ′. Έτσι, για παράδειγμα,

$$\mathbf{a} = \begin{bmatrix} 3 \\ -2 \\ 4 \end{bmatrix} \quad \Leftrightarrow \quad \mathbf{a}^T = [3, -2, 4] \text{ ή } \mathbf{a}' = [3, -2, 4]$$

Εννοείται ότι η αναστροφή της αναστροφής μας δίνει το αρχικό διάνυσμα $(\mathbf{a}^T)^T = \mathbf{a}$.

Στο MATLAB για την αναστροφή χρησιμοποιείται το σύμβολο `'`:

```
>> a = rand(1,3)

a =
        0.9501    0.2311    0.6068

>> a'

ans =
        0.9501
        0.2311
        0.6068
```

8.6.2 Το εσωτερικό γινόμενο δύο διανυσμάτων

Η έννοια του εσωτερικού γινομένου είναι θεμελιώδης στη γραμμική άλγεβρα και θα μας φανεί ιδιαίτερα χρήσιμη στα νευρωνικά δίκτυα. Σύμφωνα με τον ορισμό του το **εσωτερικό γινόμενο** δύο διανυσμάτων **a** και **b** είναι το άθροισμα του γινομένου των στοιχείων τους. Προϋπόθεση:

- οι στήλες του αριστερού διανύσματος να είναι όσες και οι γραμμές του δεξιού. Με άλλα λόγια το αριστερό διάνυσμα πρέπει να είναι μια γραμμή *n* στοιχείων ενώ το δεξί πρέπει να είναι μια στήλη *n* στοιχείων. Προφανώς τα δύο διανύσματα πρέπει να έχουν το ίδιο πλήθος στοιχείων, δηλαδή το ίδιο μήκος.

Αν τα διανύσματα είναι στήλες, όπως συνηθίζεται κατά τη μαθηματική σύμβαση, το αριστερό (**a**) πρέπει να αναστραφεί για να γίνει γραμμή ενώ το δεξί (**b**) μένει ως έχει. Στην περίπτωση αυτή το εσωτερικό γινόμενο συμβολίζεται

$\mathbf{a}^T\mathbf{b}$

Το εσωτερικό γινόμενο είναι ένας *απλός αριθμός*, (<u>όχι</u> διάνυσμα).

Π.χ. Έστω τα δύο διανύσματα-στήλες: $\mathbf{w} = [2, 0, -1, 7]^T$, $\mathbf{x} = [-4, -1, -3, 1]^T$. Το εσωτερικό τους γινόμενο είναι

$\mathbf{w}^T\mathbf{x} = 2*(-4) + 0*(-1) + (-1)*(-3) + 7*1 = -8 + 0 + 3 + 7 = 2$

Στο MATLAB, αν τα `a`, `b` είναι δύο διανύσματα-στήλες τότε το εσωτερικό τους γινόμενο είναι απλά:

```
a'*b
```

Αν και τα δύο είναι διανύσματα-γραμμές, τότε το a πρέπει να παραμείνει ως έχει και το b να αναστραφεί

```
a*b'
```

8.6.3 Πράξεις με διανύσματα και πίνακες

(*<u>Help:</u> MATLAB -> Programming -> Basic Program Components -> Operators*)

Εκτός από το εσωτερικό γινόμενο μεταξύ διανυσμάτων ή το γινόμενο μεταξύ πινάκων ή το γινόμενο μεταξύ πινάκων και διανυσμάτων το MATLAB επιτρέπει σχεδόν όλοι οι τελεστές των πράξεων να εφαρμοστούν σε διανύσματα ή σε πίνακες. Για παράδειγμα, ο τελεστής ^ χρησιμοποιείται για να υψώσουμε σε δύναμη έναν αριθμό. Αν προσθέσουμε μια τελεία πριν από τον τελεστή αυτό μπορούμε να υψώσουμε σε μια δύναμη όλα τα στοιχεία ενός διανύσματος. Παράδειγμα:

```
>> x = 3

x =
        3

>> x^2

ans =
        9

>> a = [3, -2, 4, 1, 0, -1]
```

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

>> a.^2

ans =
       9     4    16     1     0     1
```

Ομοίως οι τελεστές `*` `/` χρησιμοποιούνται για τις γνωστές μας πράξεις μεταξύ αριθμών. Αν όμως προσθέσουμε μια τελεία πριν από έναν τέτοιο τελεστή (δηλαδή `.*` και `./`)τότε η πράξη γίνεται μεταξύ των στοιχείων δύο διανυσμάτων: αντίστοιχα πολλαπλασιασμός στοιχείο προς στοιχείο και διαίρεση στοιχείο προς στοιχείο. Παραδείγματα:

```
>> a = [3, -2, 4, 1];
>> b = [8, 6, 5, -7];
>> a.*b

ans =
       24   -12    20    -7

>> a./b

ans =
       0.3750   -0.3333    0.8000   -0.1429
```

Οι τελεστές `+` `−` δεν χρειάζονται την τελεία αφού από μόνοι τους λειτουργούν με τη λογική της πράξης στοιχείο προς στοιχείο, πχ:

```
>> a+b

ans =
       11     4     9    -6

>> a-b

ans =
       -5    -8    -1     8
```

Εντελώς αντίστοιχα οι λογικοί τελεστές λειτουργούν με τη λογική της πράξης στοιχείο προς στοιχείο. Το αποτέλεσμα μιας λογικής συνθήκης που έχει ορίσματα ένα ή περισσότερα διανύσματα είναι ένα διάνυσμα με 0 ή 1 στην θέση `i` ανάλογα αν το στοιχείο `i` ικανοποιεί τη λογική συνθήκη ή όχι, πχ:

```
>> a = [7, -2, 4, -3, -8];
>> a > 0


ans =
       1     0     1     0     0  %true, false, true, false, false
```

```
>> a==4

ans =
     0     0     1     0     0  %false, false, true, false, false

>> b = [1, 2, 3, 4, 5];
>> a < b

ans =
     0     1     0     1     1  %false, true, false, true, true
```

8.7 Συναρτήσεις rand, randn, ones, zeros, eye

(Help: MATLAB->Functions -- Categorical List->Mathematics->Arrays and Matrices)

Οι συναρτήσεις rand και randn δίνουν την δυνατότητα δημιουργίας τυχαίων πινάκων:

```
X = rand(3,4);
Y = randn(4,6);
x = rand(1,8);
y = randn(3,1);
a = randn(1,1);
```

Η συνάρτηση **`rand`** παράγει τυχαίους αριθμούς με **ομοιόμορφη** κατανομή μεταξύ 0 και 1.

Η συνάρτηση **`randn`** παράγει τυχαίους αριθμούς με **Γκαουσσιανή** κατανομή με μέση τιμή 0 και διασπορά 1 (λεγόμενη και «*κανονική κατανομή*»).

Τα δύο ορίσματα των συναρτήσεων αυτών δείχνουν το πλήθος των γραμμών και των στηλών του πίνακα που δημιουργείται. Έτσι ο πίνακας `X` που ορίσαμε παραπάνω έχει 3 γραμμές και 4 στήλες, ενώ ο πίνακας `y` είναι στην ουσία ένα διάνυσμα με 3 στοιχεία αφού έχει 3 γραμμές και μόνο μία στήλη. Ο πίνακας `a`, τέλος, είναι ένας απλός αριθμός (1 στήλη και 1 γραμμή).

Δημιουργία ειδικών πινάκων με επιλεγόμενες διαστάσεις χρησιμοποιώντας τις συναρτήσεις `zeros`, `ones`, `eye`.

```
>>Z=zeros(3,10);    %Πίνακας 3 γραμμών, 10 στηλών
                    %γεμάτος μηδενικά

>>Z=ones(4,1);      %Πίνακας 4x1 (δηλ. διάνυσμα)
                    %γεμάτος άσους


>> Z=eye(4,4)       % Μοναδιαίος πίνακας 4x4

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

8.8 Λογικοί τελεστές και if-then-else

(Help: MATLAB->Programming->Basic Program Components->Program Control Statements)
(Help: MATLAB->Programming->Basic Program Components->Operators)

Οι λογικοί τελεστές επεξεργάζονται συνήθως τις τιμές ενός ή δύο ορισμάτων (π.χ. συγκρίνουν δύο αριθμούς) και επιστρέφουν την τιμή *true* ή *false*. Στο MATLAB το true είναι ο αριθμός 1 και το false είναι ο αριθμός 0. Οι πιο συνηθισμένοι λογικοί τελεστές είναι:

Ισότητα: `==`

Όχι ισότητα: `~=`

Ανισότητες: > (μεγαλύτερο), < (μικρότερο), >= (μεγαλύτερο-ίσο),....<= (μικρότερο-ίσο)

Λογική άρνηση: ~

Λογικό OR: `|` Short-circuit OR: `||`

Λογικό AND: `&` Short-circuit AND: `&&`

Οι τελεστές τύπου *Short-circuit* χρησιμοποιούνται για να ενώσουν δύο λογικές εκφράσεις, πχ. `(a==2) || (b(2)>8)`
Η διαφορά από το κοινό `OR |` (ή `AND &`) είναι ότι η δεύτερη λογική έκφραση `(b(2)>8)` δεν υπολογίζεται αν δεν χρειάζεται, αν δηλαδή το αποτέλεσμα είναι γνωστό από την πρώτη κιόλας έκφραση. Πχ. αν η πρώτη έκφραση είναι *true* τότε το λογικό OR θα δώσει *true* άσχετα από την τιμή της δεύτερης έκφρασης ενώ, αντίστοιχα, αν η πρώτη έκφραση είναι *false* τότε το λογικό AND θα δώσει *false* άσχετα από την τιμή της δεύτερης έκφρασης.

Είσοδοι		*and*	*or*	*not*	*xor*
A	B	A & B	A | B	~A	xor(A,B)
0	0	0	0	1	0
0	1	0	1	1	1
1	0	0	1	0	1
1	1	1	1	0	0

Παραδείγματα:

Εντολή	Οθόνη	
`y = 3`	`y =` `3`	`%προσέξτε τη διαφορά μεταξύ %της ανάθεσης y = 3 και ...`
`y == 3`	`ans =` `1`	`%της Boolean σύγκρισης μεταξύ %του y και του 3 που είναι true`
`x = 2;`		
`(x==2) & (y==3)`	`ans =` `1`	
`v = (x~=2)`	`v =` `0`	

Οι λογικοί τελεστές παράγουν λογικές εκφράσεις που χρησιμοποιούνται

- μετά από το `if` στην εντολή `if-then-else`. Παράδειγμα:

```
if age<18
        fprintf('junior\n');
elseif age<65
        fprintf('adult\n');
else
        fprintf('senior citizen');
end
```

Παρατηρήστε ότι στην εντολή `if-then-else` αν έχουμε πάνω από δύο περιπτώσεις μπορούμε να χρησιμοποιήσουμε το `elseif`.

- μετά από το `while` στους βρόχους τύπου `while-end`. Παράδειγμα:

```
while (x<10)
        x = x+a(i);
        i= i+1;
end
```

- μέσα σε αριθμητικές εκφράσεις αφού οι λογικές εκφράσεις είναι αριθμοί (0 ή 1). Παράδειγμα:

```
a = 5*(x==2) + 4*(x~=2);
```

Εύκολα μπορείτε να διαπιστώσετε ότι η παραπάνω εντολή δίνει στο `a` την τιμή 5 αν `x==2` ή την τιμή 4 αν `x~=2`.

8.9 Βρόγχοι

(Help: MATLAB->Getting Started->Programming->Flow Control)

Οι κλασσικοί βρόγχοι που υπάρχουν και στην C μπορούν να υλοποιηθούν πολύ εύκολα και στο MATLAB. Τα παρακάτω παραδείγματα παρουσιάζουν την χρήση της for, while, continue και break.

```
%10 επαναλήψεις όπου σε κάθε επανάληψη το i παίρνει την αντίστοιχη τιμή επανάληψης
for i = 1:10
        x( i ) = i;
end

%επανάληψη όσο το i είναι μικρότερο του a
a=10;
i=0;
while i < a
        i++;
        fprintf( 'loop number %f.\n', i);
```

```
end

%επανάληψη for από 1 εώς 10 και στην επανάληψη 5 να μην εκτελεστεί ο κώδικας της συγκεκριμένης %επανάληψης
for i = 1:10
        if i == 5
                continue;
        end
        x( i ) = i;
end

%επανάληψη όσο το i είναι μικρότερου του a εκτός αν i^2 γίνει μεγαλύτερο του 60
a=10;
i=0;
while i < a
        i++;
        if (i^2) > 60
                break;
        end
        fprintf( 'loop number %f.\n', i);
end
```

8.10 Δομές (Structures)

(Help: MATLAB->External Interfaces->MATLAB Interface to Generic DLL->Data Conversion->Structures)

Το struct είναι ένας τύπος δεδομένων που μπορεί να δημιουργηθεί από τον χρήστη. Η struct() παίρνει ως ορίσματα το όνομα του πεδίου και δίπλα την τιμή του πεδίου. Τα πεδία είναι απλά ονόματα για τις τιμές. Οι τιμές μπορεί να είναι και πίνακες. Ο παρακάτω κώδικας είναι ένα παράδειγμα δημιουργίας ενός struct το οποίο περιέχει τρεις μεταβλητές την size, την color και την numbers.

```
s = struct('size', {'big','little'}, 'color', {'red'}, 'numbers', {3 4});
```

Για να προσπελάσουμε μία τιμή από ένα struct γράφουμε το εξής:

```
x = s.size;     %Η τιμή αποθηκεύεται στην μεταβλητή x
```

8.11 Συναρτήσεις (Functions)

(Help: MATLAB->Getting Started->Programming->Scripts and Functions->Functions)

Οι functions είναι M-Files πού δέχονται δεδομένα ως είσοδο και επιστρέφουν δεδομένα ως έξοδο. Το όνομα του M-File και της function πρέπει να είναι ίδιο. Οι functions χειρίζονται μεταβλητές που βρίσκονται στο δικό τους workspace και είναι ξεχωριστό από το workspace που είναι προσβάσιμο από το command window του MATLAB. Ο παρακάτω κώδικας είναι ένα παράδειγμα μιας function.

```
function sum = getsum(pinakas)
%   sum Το άθροισμα των στοιχείων του pinakas.
%   pinakas Μονοδιάστατος πίνακας.

pinakas = [1 2 3 4];
sum = 0;

for i=1:size(pinakas,2)
   sum = sum + pinakas(i);
end
```

Η πρώτη γραμμή του M-File ξεκινά με την λέξη function. Έπειτα δίνουμε τα ορίσματα εισόδου, το όνομα της function και τέλος τα ορίσματα εξόδου. Για να δηλώσουμε μία function με παραπάνω από δύο ορίσματα εισόδου και εξόδου γράφουμε το εξής:

```
function [a,b,c,d] = testFunction(s, g, t)
```

8.12 Είσοδος από το πληκτρολόγιο

(Help:Search For: Input)

Για να εισάγουμε μία τιμή σε μία μεταβλητή από το πληκτρολόγιο απλά γράφουμε την παρακάτω εντολή. Το πρώτο όρισμα της input είναι το μήνυμα που θέλουμε να εμφανιστεί πριν την είσοδο και το δεύτερο όρισμα δηλώνει ότι η είσοδος θα αποθηκευθεί σαν μεταβλητή κείμενου και όχι σαν αριθμός. Αυτό το όρισμα μπορεί να παραληφθεί αν θέλουμε να εισάγουμε μόνο αριθμούς.

```
reply = input('Do you want more? Y/N [Y]: ','s');
```

8.13 Επεξεργασία αρχείων

8.13.1 Δυαδικά αρχεία

(Help:Search For: Writing Binary Data)
(Help:Search For: Reading Binary Data)

Το MATLAB δίνει την δυνατότητα εγγραφής και ανάγνωσης δυαδικών αρχείων. Η εγγραφή σε ένα δυαδικό αρχείο γίνετε όπως φαίνεται στο παρακάτω παράδειγμα.

```
fwriteid = fopen('testfile.bin','w');          %ανοίγουμε αρχείο για
                                               %εγγραφή
count = fwrite(fwriteid,randn(5),'int32');     %εισάγουμε τα δεδομένα
fclose(fwriteid);                              %κλείνουμε το αρχείο
```

Η `fopen` ανοίγει ένα αρχείο για έξοδο ή εισαγωγή δεδομένων. Το πρώτο όρισμα δίνει το όνομα του αρχείου και το δεύτερο δηλώνει την ανάγνωση ή εγγραφή(`r` ή `w` αντίστοιχα). Η επιστρεφόμενη τιμή `fwriteid` είναι ο κωδικός ταυτότητας του αρχείου.

Η fwrite γράφει τα στοιχεία που παρήγαγε η συνάρτηση randn(5) που είναι τύπου int32. Επιστρέφει στην μεταβλητή count τον αριθμό των στοιχείων που γράφτηκαν.

Για να διαβάσουμε δεδομένα από δυαδικό αρχείο κάνουμε τα παρακάτω.

```
freadid = fopen('testfile.bin','r');          %ανοίγουμε αρχείο για
                                              %ανάγνωση
A = fread(freadid,'int32');                   %διαβάζουμε τα δεδομένα
fclose(freadid);                              %κλείνουμε το αρχείο
```

Η fread διαβάζει τα δεδομένα που ξέρουμε ότι είναι τύπου int32 και τα αποθηκεύει στην μεταβλητή A.

8.13.2 Αρχεία κειμένων

(Help:Search For: Writing Text Data)
(Help:Search For: Reading Text Data)
(Help:Search For: fprintf)

Η εγγραφή σε ένα αρχείο κειμένου γίνετε όπως φαίνεται στο παρακάτω παράδειγμα.

```
fwriteid = fopen('testfile.text','w');        %ανοίγουμε αρχείο για εγγραφή

x = 0:.1:1;                                   %δημιουργούμε κάποια δεδομένα
y = [x; exp(x)];

fprintf(fwriteid,'%6.2f %12.8f\n',y);         %εισάγουμε τα δεδομένα
fclose(fwriteid);                             %κλείνουμε το αρχείο
```

Για να διαβάσουμε δεδομένα από αρχείο κειμένου κάνουμε τα παρακάτω.

Για να ανακτήσουμε τα δεδομένα από το αρχείο θα χρησιμοποιήσουμε την function textscan. Η textscan προϋποθέτει ότι το αρχείο είναι σε text μορφή και ότι πρέπει σαν όρισμα να της δώσουμε ένα format string. Ένα string δηλαδή που θα δείχνει τον τρόπο που θα πάρει τα δεδομένα από κάθε γραμμή του αρχείου. Ακολουθεί ένα παράδειγμα.

Έστω ότι έχουμε ένα αρχείο της παρακάτω μορφής. Έχει 3 σειρές που έχουν από 7 στοιχεία που χωρίζονται από ένα κόμμα.

```
12,56,13,45,2,3,0
10,47,76,9,2,12,1
9,23,5,3,76,3,0
```

Άρα το format string θα είναι της μορφής

```
formatstr = '%[^,],%[^,],%[^,],%[^,],%[^,],%[^,],%[^\n]'
```

Το %[^,] δηλώνει διάβασε αυτό που υπάρχει μέχρι να συναντήσεις κόμμα. Εφόσον το διαβάσει υπάρχει ένα κόμμα και έπειτα διαβάζει κάτι άλλο που δεν είναι κόμμα και συνεχίζει

έτσι μέχρι ότου συναντήσει το τέλος της γραμμής με το %[^\n], όπου δηλώνει διάβασε μέχρι να συναντήσει το τέλος της γραμμής \n. Επειδή όμως η γραμμή μπορεί να είναι πολύ μεγάλη δημιουργούμε το format string με μία for. Ακολουθεί ο κώδικας δημιουργίας του ίδιου string που κάναμε παραπάνω αλλά με for.

```
formatstr=[];
for i=1:6
   formatstr=[formatstr,'%[^,],'];
end
formatstr=[formatstr,'%[^\n]'];
```

Έχοντας δημιουργήσει το format string μπορούμε να χρησιμοποιήσουμε την textscan για ανακτήσουμε τα δεδομένα από το αρχείο. Ο παρακάτω κώδικας ανακτά τα δεδομένα και τα αποθηκεύσει σε ένα cell array με το όνομα tempdata. Έπειτα η fclose κλείνει το αρχείο.

```
freadid = fopen('testfile.text','w');
tempdata=textscan(freadid,formatstr);
fclose(freadid);
```

8.14 Interpreter και Compiler

(Help: MATLAB Compiler)

Η γραμμή εντολών (command line) λειτουργεί ως interpreter των εντολών που γράφουμε εκεί. Κάθε εντολή που δακτυλογραφούμε μεταφράζεται σε γλώσσα μηχανής και εκτελείται μόλις πατήσουμε <ENTER>. Με την ίδια ακριβώς λογική λειτουργούν και τα προγράμματα script, δηλαδή είναι σαν να γράφαμε στο command line του MATLAB κάθε εντολή που είναι γραμμένη στο αρχείο script και πατούσαμε το <ENTER>. Έτσι κάθε εντολή μεταφράζεται και εκτελείται ξανά και ξανά ακόμη και εάν βρίσκεται σε ένα βρόχο. Αυτό δημιουργεί καθυστερήσεις στους βρόχους και είναι λιγότερο αποδοτική μέθοδος σε σχέση με την μετάφραση του κώδικα μια φορά στην αρχή και την δημιουργία binary εκτελέσιμου αρχείου τύπου exe. Η μέθοδος του interpreter είναι βέβαια ιδανική όταν θέλουμε να τεστάρουμε γρήγορα μια ιδέα ή έναν αλγόριθμο χωρίς να χρειάζεται κάθε φορά να μεσολαβεί το χρονοβόρο στάδιο της μετάφρασης και της εκτέλεσης του εκτελέσιμου αρχείου.

Το MATLAB προσφέρει έναν compiler με το όνομα `mcc` οποίος δημιουργεί εκτελέσιμο κώδικα σε μορφή αρχείου exe. Για παράδειγμα έστω ότι έχουμε γράψει το παρακάτω πρόγραμμα στο αρχείο `dokimi.m`

```
function main()

for (ii=1:10)
   fprintf('i=%f\n', ii);
end
```

Ο κώδικας του κυρίως προγράμματος πρέπει να είναι μια function με το όνομα `main()`. Εκτελούμε τον compiler μέσα από το MATLAB χρησιμοποιώντας την παράμετρο `-m`:

```
>> mcc -m dokimi
```

Select a compiler:
[1] Lcc C version 2.4 in C:\PROGRAM FILES\MATLAB704\BIN\WIN32\\..\..\sys\lcc
[2] Microsoft Visual C/C++ version 6.0 in C:\Program Files\Microsoft Visual Studio

[0] None

Compiler: 1

Try to update options file: C:\Documents and Settings\Kostas\Application Data\MathWorks\MATLAB\R14\compopts.bat
From template: C:\PROGRAM FILES\MATLAB704\BIN\WIN32\\..\WIN32\mbuildopts\lcccompp.bat

Done . . .

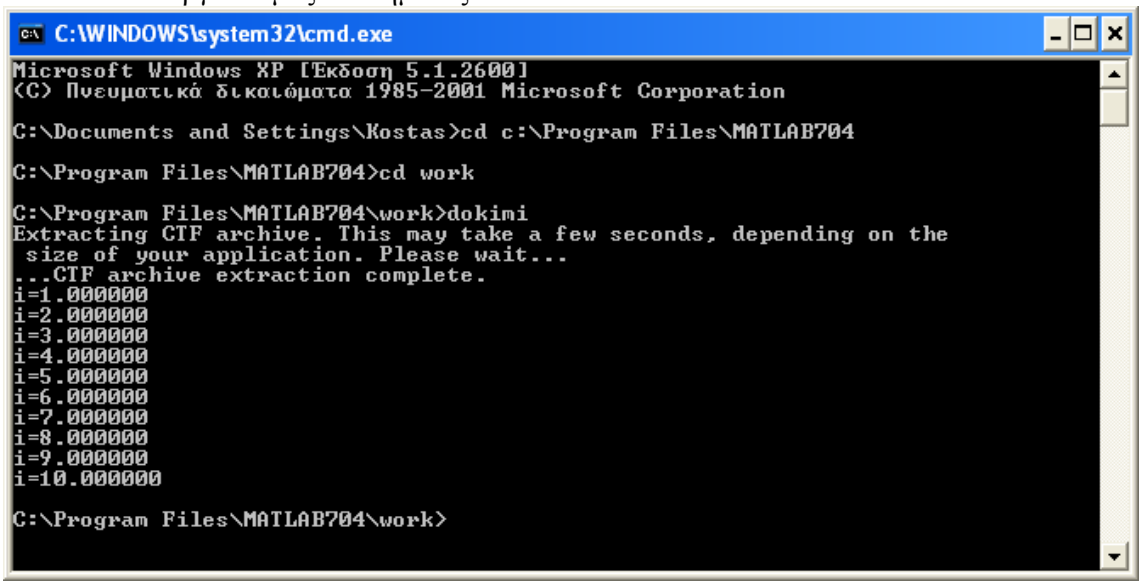
--> ""C:\Program Files\MATLAB704\bin\win32\mwregsvr" "C:\Program Files\MATLAB704\bin\win32\mwcomutil.dll""

DllRegisterServer in C:\Program Files\MATLAB704\bin\win32\mwcomutil.dll succeeded

--> ""C:\Program Files\MATLAB704\bin\win32\mwregsvr" "C:\Program Files\MATLAB704\bin\win32\mwcommgr.dll""

DllRegisterServer in C:\Program Files\MATLAB704\bin\win32\mwcommgr.dll succeeded

Δημιουργούμε έτσι το αρχείο `dokimi.exe` το οποίο μπορεί να εκτελεστεί στο command line του λειτουργικού μας συστήματος:



8.15 Ο Editor/Debugger

(*Help: MATLAB -> Desktop Tools and Development Environment -> Editing and Debugging M-Files*)

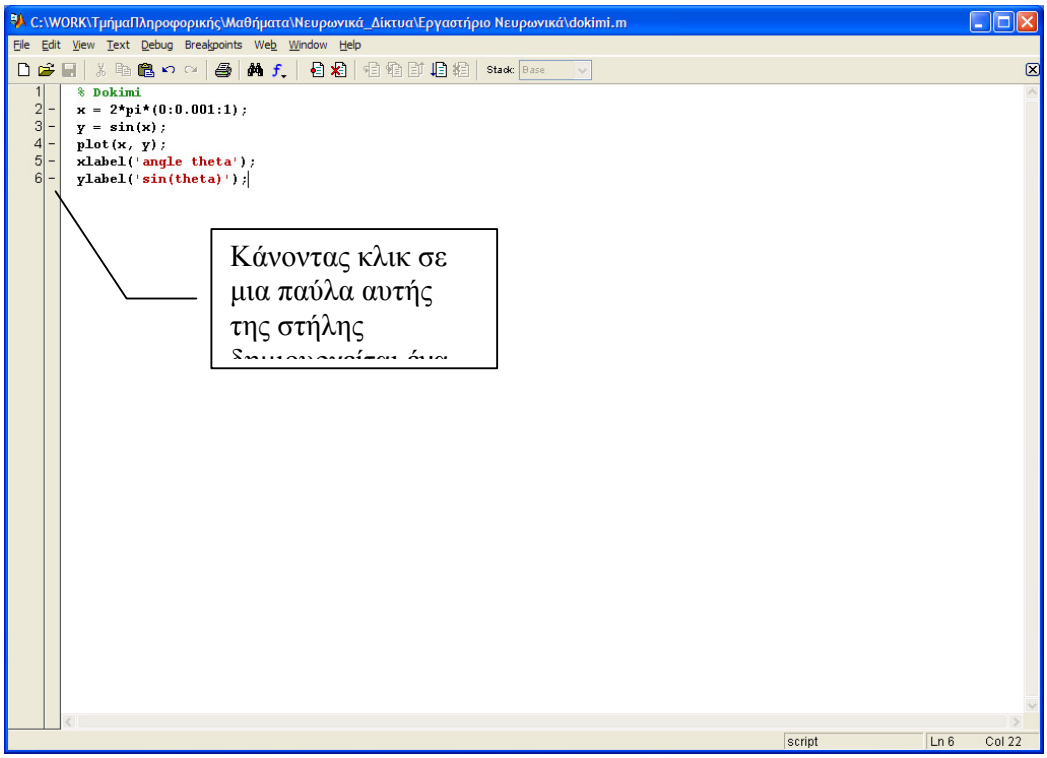
Ο editor/debugger είναι ένα ολοκληρωμένο περιβάλλον επεξεργασίας scripts. Προσφέρει αυτόματη μορφοποίηση του κώδικα που γράφετε ενώ με διαφορετικά χρώματα καταδεικνύονται οι διαφορετικές εντολές:

- πράσινο χρώμα: σχόλια. Ό,τι ακολουθεί το σύμβολο `%` σε μια γραμμή θεωρείται σχόλιο. Πχ. `%Dokimi`
- μπλε χρώμα: εντολές built-in. Πχ, οι εντολές `for`, `if`, `else`, και `end`

```
for (i=1:10)
        if (a(i) == 1)
                fprintf('Hello there!\n');
        else
                fprintf('---\n');
        end
end
```

- κόκκινο χρώμα: τα strings. Πχ, `'Hello there!\n'`

Στο παρακάτω Σχήμα φαίνεται το παράθυρο του editor/debugger.



Ο *debugger* προσφέρει τη δυνατότητα δημιουργίας σημείων στον κώδικα όπου σταματάει προσωρινά η εκτέλεση του script ώστε να μπορεί να γίνει επισκόπηση των μεταβλητών και

του συστήματος γενικότερα. Τα σημεία αυτά λέγονται *breakpoints* και ο debugger προσφέρει τη δυνατότητα δημιουργίας και καθαρίσματος των breakpoints με τις λειτουργίες: *set/clear breakpoint*, *clear all breakpoints*. Αφού σταματήσει η εκτέλεση σε κάποιο breakpoint τότε αυτή μπορεί να συνεχιστεί βηματικά γραμμή-γραμμή χρησιμοποιώντας τη λειτουργία *step*. Αν η επόμενη γραμμή καλεί μια συνάρτηση τότε μπορούμε να μπούμε μέσα στη συνάρτηση και να την εκτελέσουμε γραμμή-γραμμή με τη λειτουργία *step in*, ή να βγούμε από τη συνάρτηση πίσω στο αρχικό επίπεδο με τη λειτουργία *step out*. Επίσης η λειτουργία *run/continue* συνεχίζει κανονικά (όχι βηματικά) την εκτέλεση μέχρι το επόμενο breakpoint.

Επίσης προσφέρονται λειτουργίες *stop-if-*condition όπου το πρόγραμμα σταματάει να εκτελείται ακόμα κι αν δεν υπάρχει breakpoint εφόσον εμφανιστεί κάποια συνθήκη σφάλματος. Τέτοιες συνθήκες είναι: *error*, *warning*, *Not a Number* (*NaN*) ή *Infinity* (*Inf*). Το πρόγραμμα σταματάει στη γραμμή που εμφανίστηκε το σφάλμα ώστε να μπορούμε να επιθεωρήσουμε τις τιμές των μεταβλητών και να βοηθηθούμε στην ανακάλυψη της αιτίας του σφάλματος. Όταν σταματήσει το πρόγραμμα σε debugging mode τότε στο command line εμφανίζεται το εξής prompt:

```
K>>
```

Στο prompt αυτό μπορούμε να γράψουμε το όνομα μιας μεταβλητής χωρίς να ακολουθεί το σύμβολο ; οπότε τυπώνεται στην οθόνη το περιεχόμενο της μεταβλητής αυτής. Πχ,

```
K>> x

x =
        3

K>> y

y =
        1   -2   3   0.1   5
```

Με την εντολή `return` επιστρέφουμε στην κανονική εκτέλεση του προγράμματος (βγαίνουμε από τον debugger).

ΚΕΦΑΛΑΙΟ 9

ΓΡΑΦΙΚΕΣ ΠΑΡΑΣΤΑΣΕΙΣ

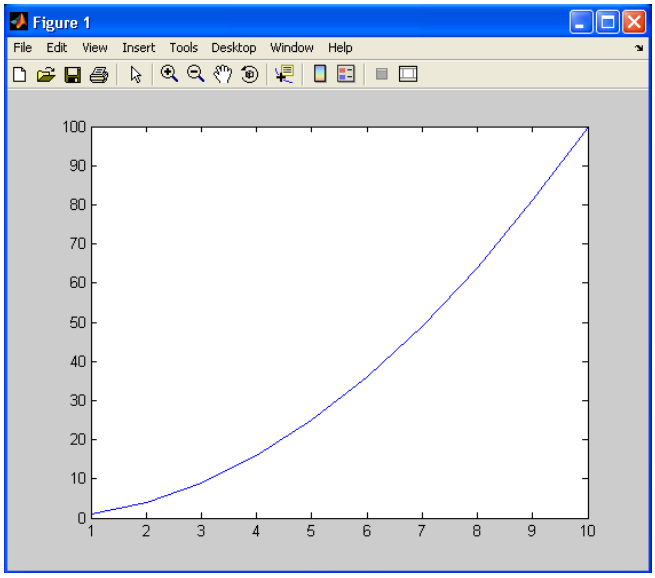
9.1 2Δ γραφικές παραστάσεις

9.1.1 Plot

(Help: Matlab->Functions – Categorical List->Graphics->Basic Plots and Graphs> Plot)

Η βασική συνάρτηση για 2Δ γραφικές παραστάσεις είναι η plot. Η plot σαν κύριο όρισμα παίρνει δύο vectors ή matrices με σημεία στον 2Δ χώρο και σχεδιάζει την γραφική παράσταση που ορίζουν τα σημεία. Ο παρακάτω κώδικας δίνει ένα παράδειγμα χρήσης της plot. Το αποτέλεσμα φαίνεται στην Εικόνα 3.1.

```
x=1:1:10;      %θέση σημείου στον άξονα x
y=x.^2;        %θέση σημείου στον άξονα y
plot(x,y);     %δημιουργία του figure με την γραφική παράσταση
```



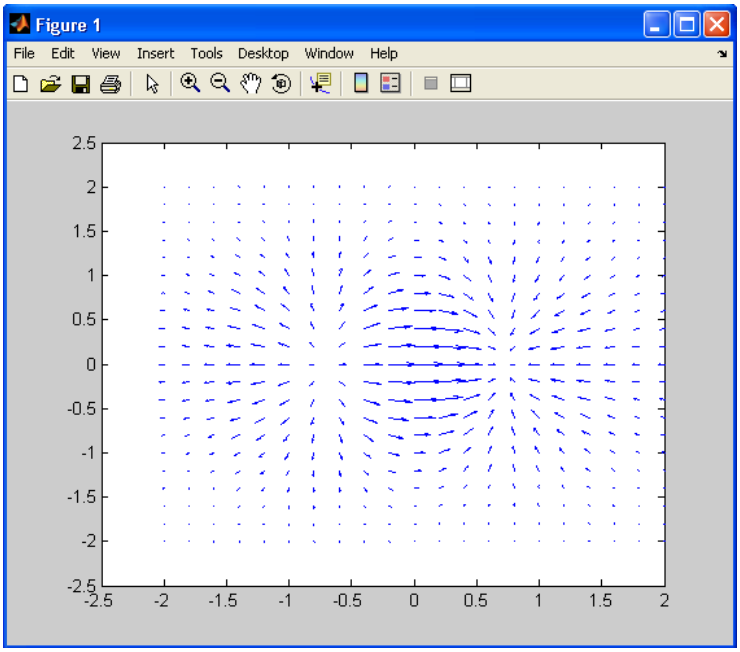
Εικόνα 3.3

9.1.2 Quiver

(Help: Matlab->Functions – Categorical List->Graphics->Basic Plots and Graphs> Quiver)

Η quiver σχεδιάζει ένα γράφημα διανυσμάτων. Το κάθε διάνυσμα αναπαριστάται ως ένα βέλος. Ο παρακάτω κώδικας δίνει ένα παράδειγμα χρήσης της quiver. Το αποτέλεσμα φαίνεται στην Εικόνα 3.2.

```
[X,Y] = meshgrid(-2:.2:2);   %ορισμός πλέγματος
Z = X.*exp(-X.^2 - Y.^2);
[DX,DY] = gradient(Z,.2,.2);
quiver(X,Y,DX,DY)
```



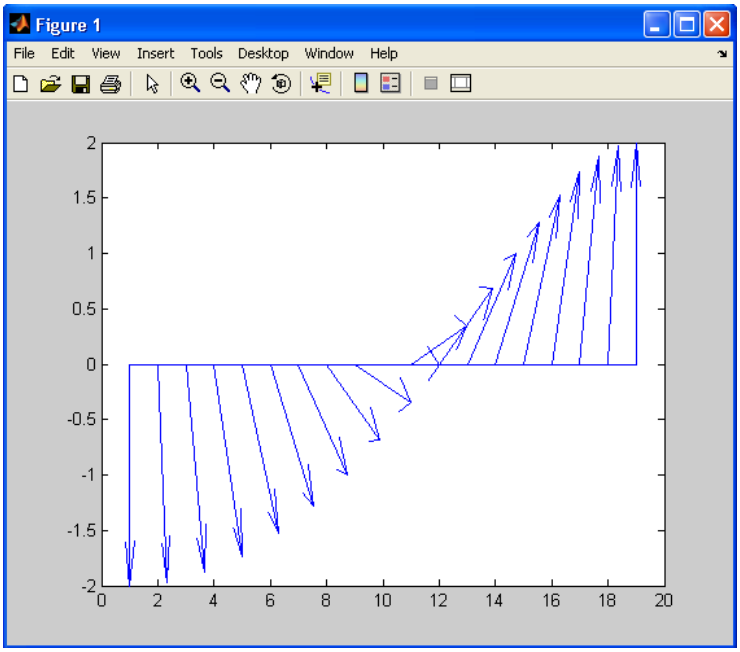
Εικόνα 3.4

9.1.3 Feather

(Help: Matlab->Functions – Categorical List->Graphics->Basic Plots and Graphs> Feather)

Η feather σχεδιάζει διανύσματα που προέρχονται σημεία που ισαπέχουν στον οριζόντιο άξονα. Ο παρακάτω κώδικας δίνει ένα παράδειγμα χρήσης της feather. Το αποτέλεσμα φαίνεται στην Εικόνα 3.3.

```
theta = (-90:10:90)*pi/180;          %κατεύθυνση διανυσμάτων
r = 2*ones(size(theta));             %μέτρο διανυσμάτων
[u,v] = pol2cart(theta,r);           %μετατροπή πολικών συντεταγμένων σε
                                     %καρτεσιανών

feather(u,v);
```



Εικόνα 3.5

9.2 3Δ γραφικές παραστάσεις

9.2.1 Meshgrid

(Help: Matlab->Functions – Categorical List->3-D Visualization-> Surface and Mesh Plots ->Mesh)

Η συνάρτηση meshgrid δημιουργεί matrices που ορίζουν το πλέγμα των γραφημάτων. Η meshgrid μπορεί να χρησιμοποιηθεί μόνο για δισδιάστατο ή τρισδιάστατο καρτεσιανό χώρο. Παίρνει για ορίσματα δύο(x,y) ή τρία vectors(x,y,z) και επιστρέφει δύο(X,Y) ή αντίστοιχα τρία(X,Y,Z) vectors. Οι γραμμές του X είναι αντίγραφα του x και οι στήλες του Y είναι αντίγραφα του y. Ο παρακάτω κώδικας δίνει ένα παράδειγμα χρήσης της meshgrid.

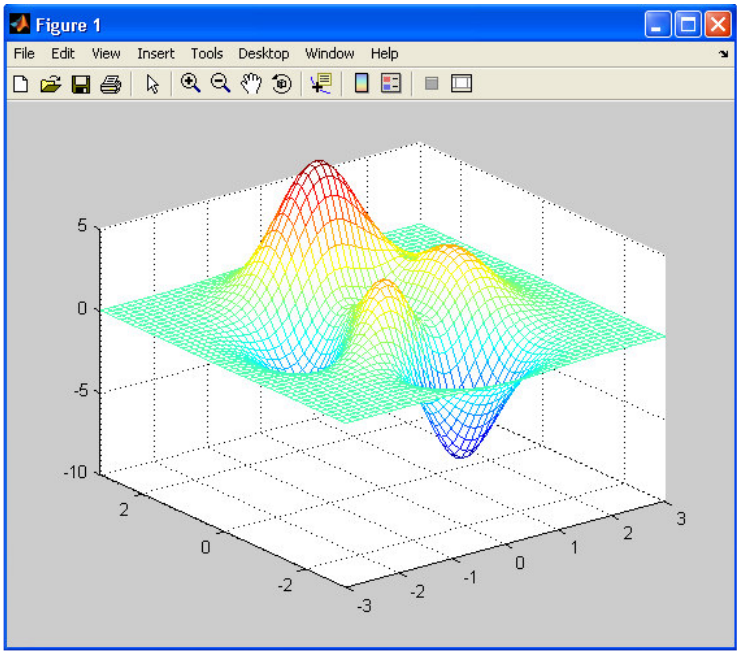
```
[X,Y] = meshgrid(1:3,10:14)

X =

     1     2     3
     1     2     3
     1     2     3
     1     2     3
     1     2     3

Y =

    10    10    10
    11    11    11
    12    12    12
    13    13    13
    14    14    14
```

9.2.2 Mesh

(Help: Matlab->Functions – Categorical List->3-D Visualization-> Surface and Mesh Plots ->Mesh)

Η συνάρτηση mesh σχεδιάζει την γραφική παράσταση μιας επιφάνειας. Παίρνει τρία ορίσματα, δύο vectors ή matrices που περιέχουν τις συντεταγμένες του οριζόντιου επιπέδου και ένα τρίτο που περιέχει τις συντεταγμένες της επιφάνειας στην τρίτη διάσταση. Ο παρακάτω κώδικας δίνει ένα παράδειγμα χρήσης της mesh. Το αποτέλεσμα φαίνεται στην Εικόνα 3.4.

```
[X,Y] = meshgrid(-3:.125:3);
Z = peaks(X,Y);                    %δίνουμε τιμές στα ύψη σε σχέση
                                   %με τα x και y
mesh(X,Y,Z);
axis([-3 3 -3 3 -10 5]);           %δίνουμε τα όρια των axes
```



Εικόνα 3.6

9.2.3 Contour

(Help: Matlab->Functions – Categorical List->Graphics -> Specialized Plotting -> Contour)

Η συνάρτηση contour σχεδιάζει διαγράμματα ισοϋψών. Η σύνταξη της φαίνεται παρακάτω.

```
[C,h] =contour(X,Y,Z,n);
```

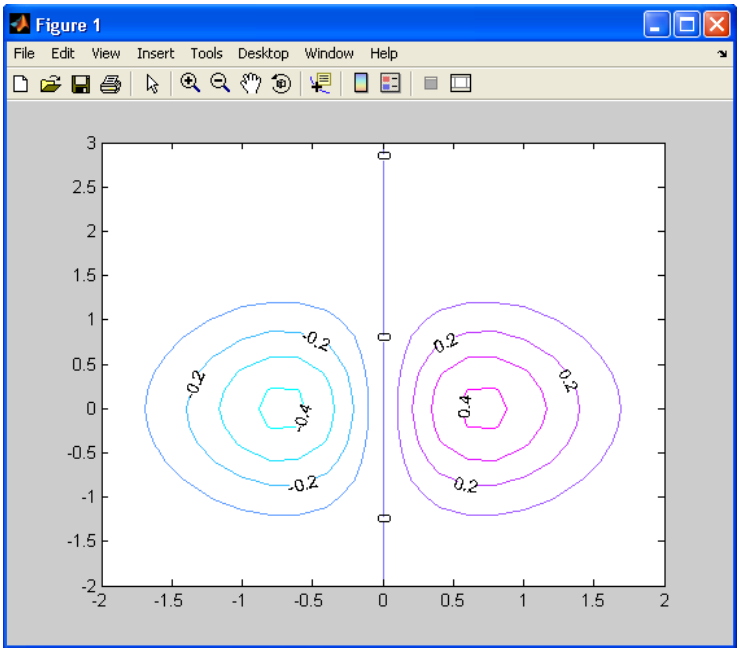
όπου:
X -- τα x του γραφήματος
Y -- τα y του γραφήματος

Z -- τα ύψη του γραφήματος σχέση με τα x και y
n -- τα επίπεδα του γραφήματος

και επιστρέφει:
C -- το matrix με τα ισοϋψή
h -- το handle για τα contourgroup αντικείμενα του γραφήματος

Ο παρακάτω κώδικας δίνει ένα παράδειγμα χρήσης της contour. Το αποτέλεσμα φαίνεται στην Εικόνα 3.5.

```
[X,Y] = meshgrid(-2:.2:2,-2:.2:3);
Z = X.*exp(-X.^2-Y.^2);                    %δίνουμε τιμές στα
                                           %ύψη σε σχέση  με τα
                                           %x και y

[C,h] = contour(X,Y,Z);
set(h,'ShowText','on','TextStep',get(h,'LevelStep')*2)   %εισάγουμε labels σε
                                                         %κάθε επίπεδο
colormap cool                                            %αλλάζουμε τα
                                                         %χρώματα του
                                                         %γραφήματος
```



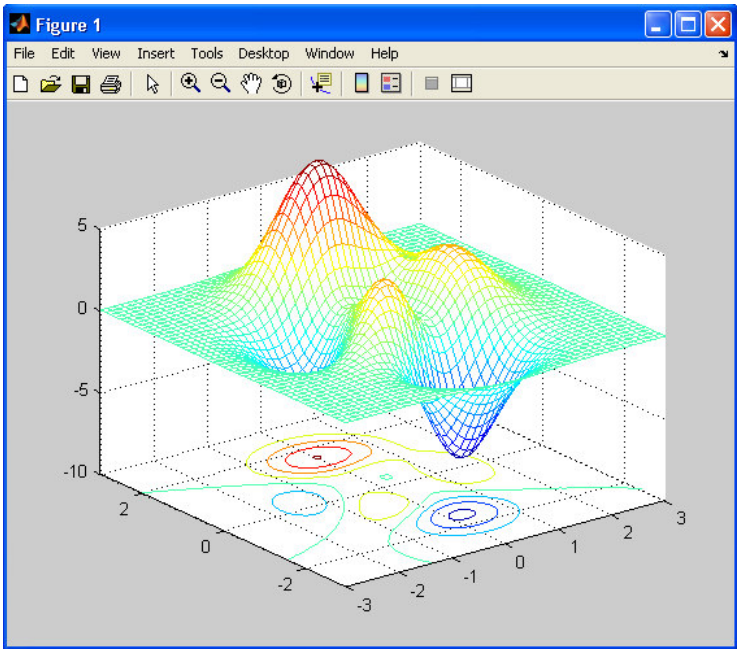
Εικόνα 3.7

9.2.4 Meshc

(Help: Matlab->Functions – Categorical List->3-D Visualization ->Surface and Mesh Plots-> Meshc)

Η meshc είναι απλά ο συνδυασμός της mesh με την contour. Ο παρακάτω κώδικας δίνει ένα παράδειγμα χρήσης της meshc. Το αποτέλεσμα φαίνεται στην Εικόνα 3.6.

```
[X,Y] = meshgrid(-3:.125:3);
Z = peaks(X,Y);
meshc(X,Y,Z);
axis([-3 3 -3 3 -10 5]);
```



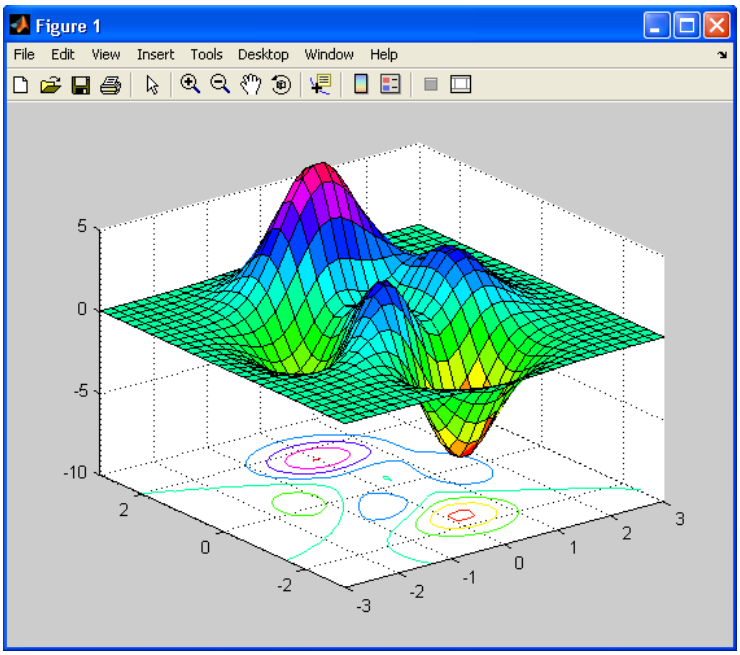
Εικόνα 3.8

9.2.4 Surf

(Help: Matlab->Functions – Categorical List->3-D Visualization ->Surface and Mesh Plots->Surf)

Η surf σχεδιάζει την γραφική παράσταση μιας επιφάνειας με την διαφορά ότι η επιφάνεια θα είναι σκιασμένη. Ο παρακάτω κώδικας δίνει ένα παράδειγμα χρήσης της surf. Το αποτέλεσμα φαίνεται στην Εικόνα 3.7.

```
[X,Y] = meshgrid(-3:.125:3);
Z = peaks(X,Y);
surfc(X,Y,Z);
axis([-3 3 -3 3 -10 5]);
```



Εικόνα 3.9

9.3 Σύμβολα, χρώματα και γραμμές

(Help: Matlab->Functions – Categorical List->Graphics->Basic Plots and Graphs-> LineSpec)

Στα plots μπορούμε να χρησιμοποιήσουμε διαφορετικά χαρακτηριστικά όσον αφορά την γραμμή. Αυτά τα χαρακτηριστικά είναι τα εξής:

- Το είδος της γραμμής
- Το πάχος της γραμμής
- Το χρώμα
- Το τύπο κουκίδας
- Το μέγεθος της κουκίδας
- Τα χρώματα της κουκίδας(περίγραμμα και εσωτερικό)

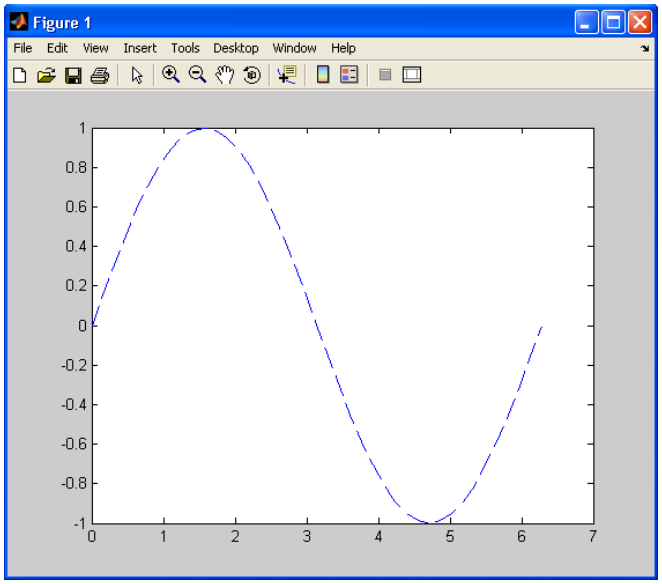
Ο πίνακας 1 δείχνει τις τιμές των ειδών της γραμμής.

Κωδικός	**Τύπος γραμμής**
-	Μονοκόμματη γραμμή
--	Διακεκομμένη γραμμή
:	Γραμμή με τελίτσες
-.	Γραμμή με παύλα-τελεία

Πίνακας 2

Ο παρακάτω κώδικας δημιουργεί ένα plot με διακεκομμένη γραμμή. Το plot φαίνεται στην Εικόνα 3.8.

```
x=0:pi/20:2*pi;
plot(x,sin(x),'--');
```



Εικόνα 3.10

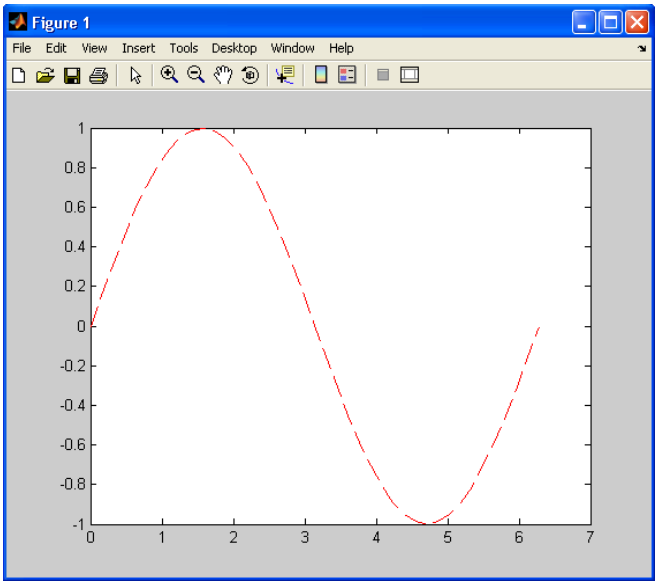
Ο πίνακας 2 δίχνει τις τιμές των χρωμάτων της γραμμής.

Κωδικός	**Χρώμα γραμμής**
r	κόκκινο
g	πράσινο
b	μπλε
c	cyan
m	magenta
y	κίτρινο
k	μαύρο
w	άσπρο

Πίνακας 3

Ο παρακάτω κώδικας δημιουργεί ένα plot με διακεκομμένη γραμμή και χρώματος κόκκινο. Το plot φαίνεται στην Εικόνα 3.9.

```
x=0:pi/20:2*pi;
plot(x,sin(x),'r--');
```



Εικόνα 3.11

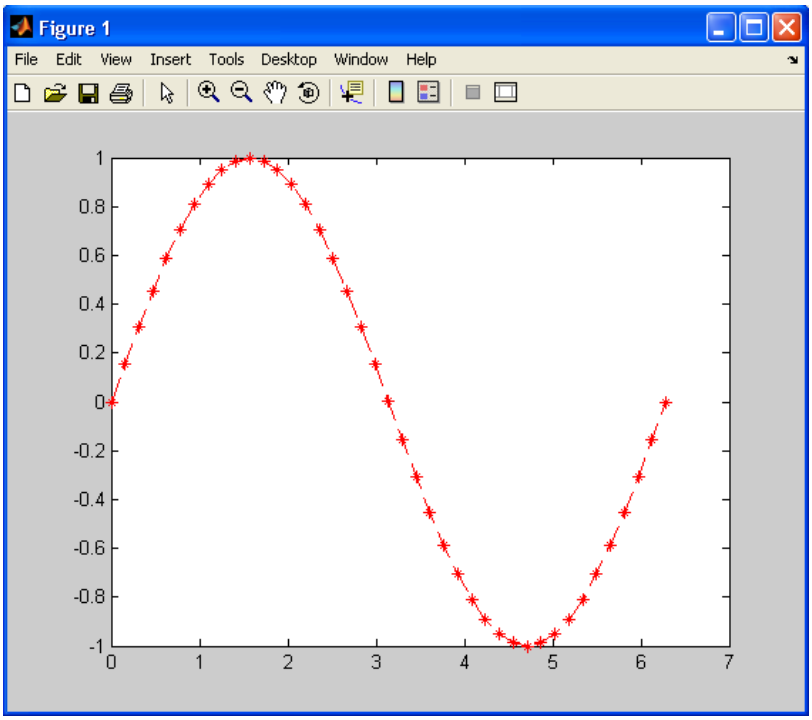
Ο πίνακας 3 δείχνει τις τιμές των κουκίδων.

Κωδικός	**Τύπος κουκίδας**
+	σύμβολο πρόσθεσης
o	κύκλος
*	αστερίσκος
.	σημείο
x	σταυρός
‘square’ or s	τετράγωνο
‘diamond’ or d	διαμάντι
^	τρίγωνο με μύτη επάνω
v	τρίγωνο με μύτη κάτω
>	τρίγωνο με μύτη προς τα δεξιά
<	τρίγωνο με μύτη προς τα αριστερά
'pentagram' or p	πεντάγραμμο
'hexagram' or h	εξάγραμμο

Πίνακας 3.4

Ο παρακάτω κώδικας δημιουργεί ένα plot με διακεκομμένη γραμμή και χρώματος κόκκινο και με κουκίδα τύπου αστερίσκου. Το plot φαίνεται στην Εικόνα 3.10.

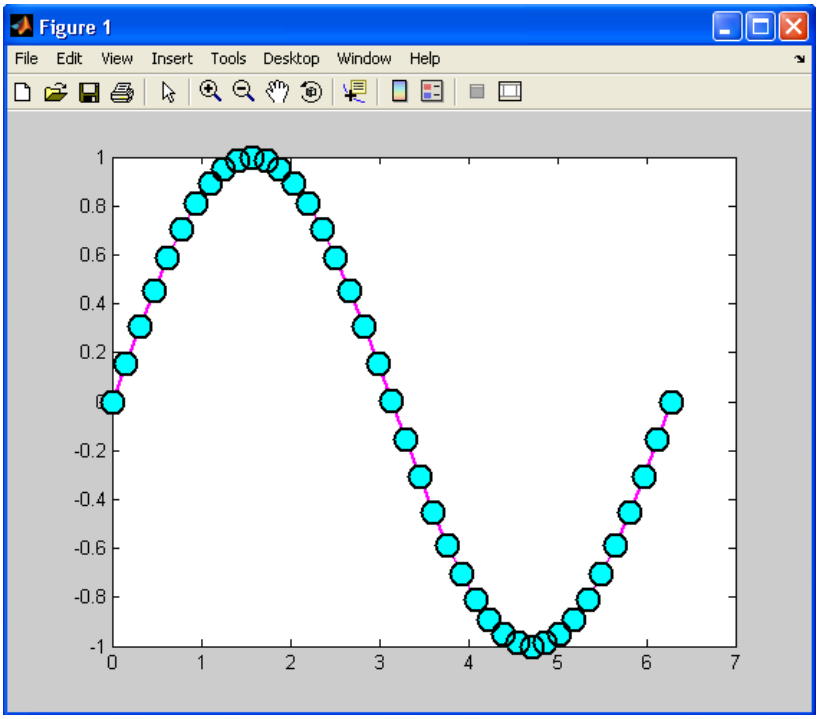
```
x=0:pi/20:2*pi;
plot(x,sin(x),'r--*');
```



Εικόνα 3.12

Ο παρακάτω κώδικας είναι παράδειγμα για την αλλαγή του πάχους της γραμμής, του μεγέθους της κουκίδας και του χρωματισμού της κουκίδας(Εικόνα 3.11).

```
plot(x,sin(x),'-mo',...                  %Solid Line, color magenta, marker circle
        'LineWidth',2,...                %Line width 2
        'MarkerEdgeColor','k',...        %Marker edge color black
        'MarkerFaceColor','c',...        %Marker face color cyan
        'MarkerSize',12);                %Marker size 12
```

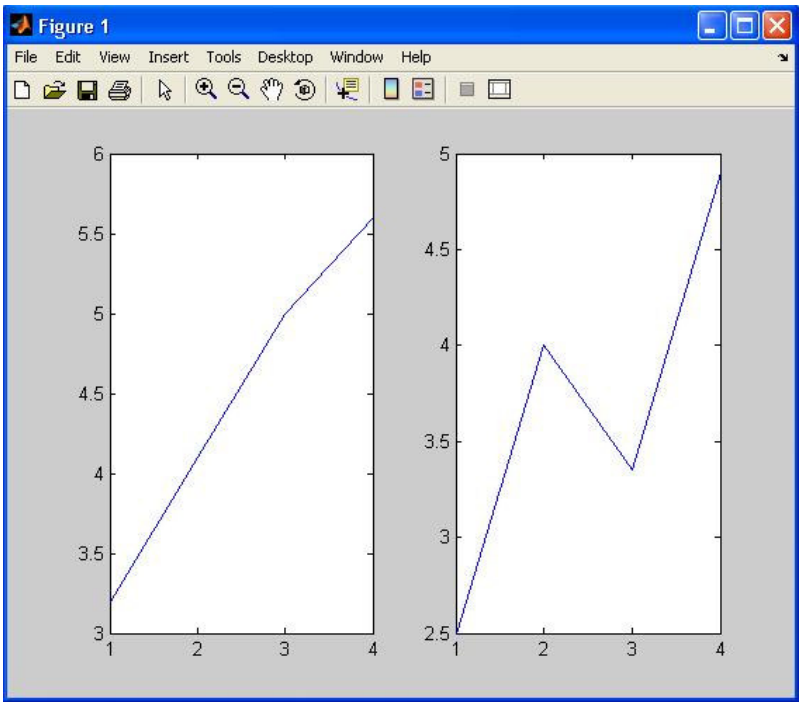


Εικόνα 3.13

9.4 Subplot

(Help: Matlab->Functions – Categorical List->Graphics->Basic Plots and Graphs-> Subplot)

Η subplot χωρίζει το figure σε ορθογώνια τμήματα, που αριθμούνται κατά σειρά, όπου περιέχουν άξονες και μπορούμε σε κάθε ένα από αυτά να κάνουμε διαφορετικά διαγράμματα. Στην Εικόνα 3.12 φαίνονται μία γραμμή με δύο plots.



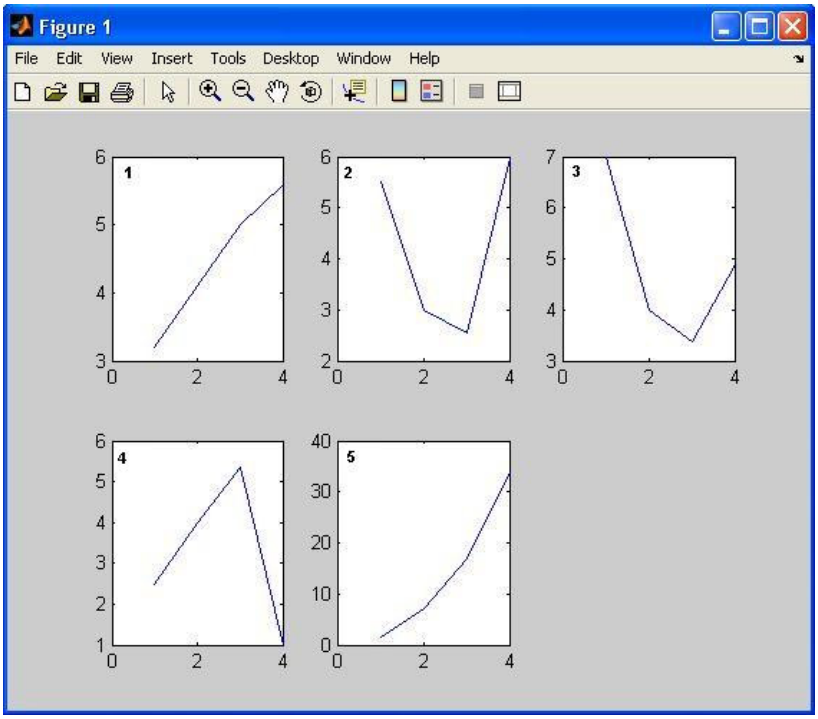
Εικόνα 3.14

Τα παραπάνω plots δημιουργήθηκαν με το παρακάτω script.

```
x = [3.2 4.1 5.0 5.6];%δεδομένα διανύσματος x
y = [2.5 4.0 3.35 4.9];        %δεδομένα διανύσματος y

subplot(1,2,1);
plot(x);
subplot(1,2,2);
plot(y);
```

Το πρώτο όρισμα της subplot είναι ο αριθμός των γραμμών και η δεύτερη ο αριθμών των στηλών που θα χωριστεί το figure. Το τρίτο όρισμα δηλώνει σε ποιο τμήμα του figure θα εργαστούμε.

Στην Εικόνα 3.13 φαίνεται ένα figure που με την subplot έχει χωριστεί σε δύο γραμμές και τρεις στήλες. Επίσης σημειώνετε η αρίθμηση του κάθε plot.

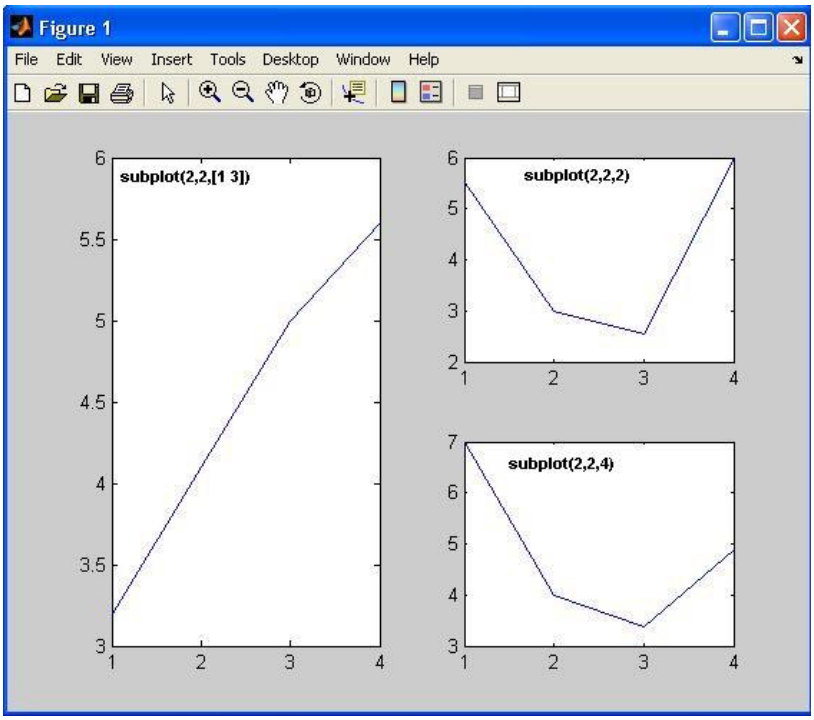


Εικόνα 3.15

Η subplot δίνει τη δυνατότητα να δημιουργήσουμε ασύμμετρες διατάξεις subplot. Το παρακάτω script κάνει ακριβώς αυτό.

```
subplot(2,2,[1 3]);
plot(x);
subplot(2,2,2);
plot(y);
subplot(2,2,4);
plot(z);
```

Στην πρώτη εντολή στο τρίτο όρισμα το [1 3] δηλώνει ότι το συγκεκριμένο subplot θα είναι στο χώρο του 1^{ου} και 3^{ου} plot (Εικόνα 3.14).



Εικόνα 3.16

9.5 Plotyy

(Help: Matlab->Functions – Categorical List->Graphics->Basic Plots and Graphs-> Plotyy)

Η plotyy δίνει την δυνατότητα να δημιουργήσουμε γραφικές παραστάσεις με δύο διαφορετικούς άξονες y. Ο ένας άξονας βρίσκεται στην δεξιά μεριά του γραφήματος και ο άλλος στην αριστερή. Η σύνταξη της plotyy φαίνεται παρακάτω.

```
[AX,H1,H2] = plotyy(X1,Y1,X2,Y2);
```

όπου:
X1 -- τα x του πρώτου γραφήματος
Y1 -- τα y του πρώτου γραφήματος
X2 -- τα x του δεύτερου γραφήματος
Y2 -- τα y του δεύτερου γραφήματος

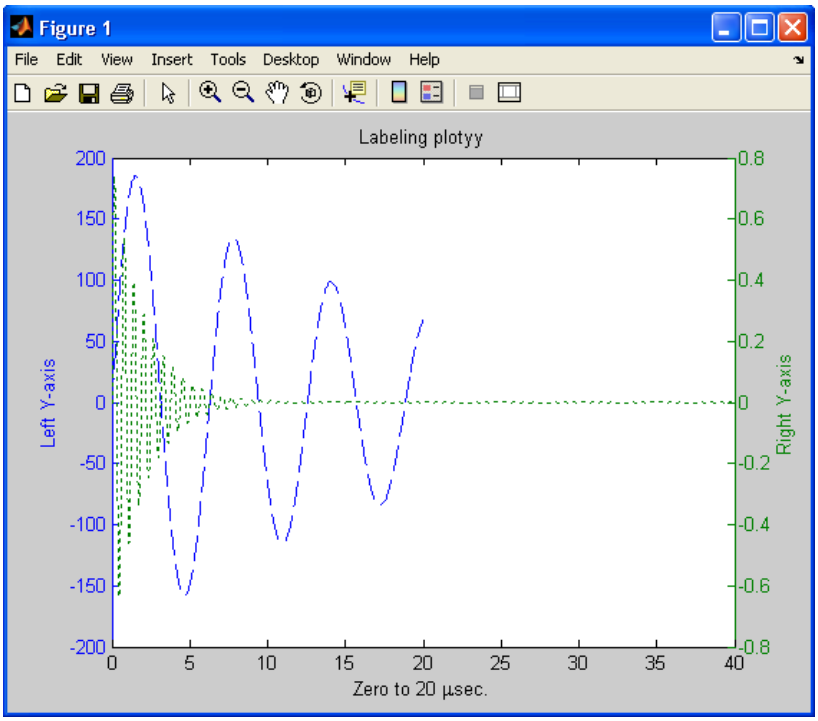
και επιστρέφει:
AX -- το handle για τους άξονες y1 και y2
H1 -- το handle για την γραμμή της πρώτης γραφικής παράστασης
H2 -- το handle για την γραμμή της δεύτερης γραφικής παράστασης

Μπορούμε να χρησιμοποιήσουμε τα handles για να αλλάξουμε τις ιδιότητες των αξόνων και των γραμμών.

Ο παρακάτω κώδικας δημιουργεί ένα plotyy. Επίσης ονομάζει όλους του άξονες και αλλάζει τις ιδιότητες των γραμμών(Εικόνα 3.15). Παρατηρούμε ότι η πρώτη γραφική παράσταση

σχεδιάζεται μέχρι το μισό του άξονα x. Αυτό συμβαίνει γιατί δώσαμε τιμές στο x1 από 0 μέχρι 20 και όχι μέχρι 40 που έχει το x2.

```
x1 = 0:0.01:20;
x2 = 0:0.01:40;
y1 = 200*exp(-0.05*x1).*sin(x1);
y2 = 0.8*exp(-0.5*x2).*sin(10*x2);
[AX,H1,H2] = plotyy(x1,y1,x2,y2);                %create plotyy

set(get(AX(1),'Ylabel'),'String','Left Y-axis');  %set left y label
set(get(AX(2),'Ylabel'),'String','Right Y-axis'); %set right y label

xlabel('Zero to 20 \musec.');                     %set x label
title('Labeling plotyy');                         %set title of plotyy

set(H1,'LineStyle','--');                         %change y1 line settings
set(H2,'LineStyle',':');                          %change y2 line settings
```



Εικόνα 3.17

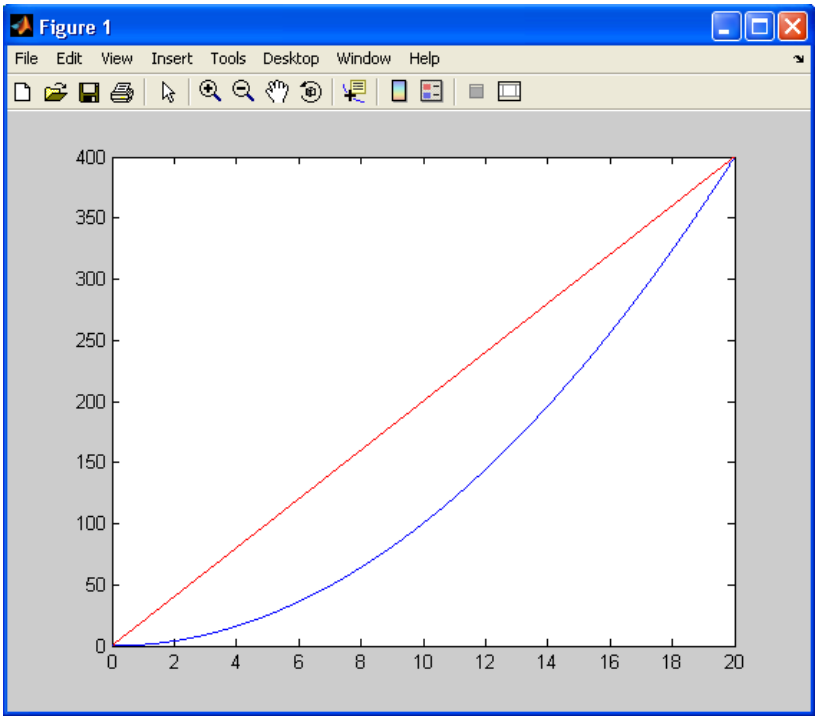
9.6 Hold on/off

(Help: Matlab->Functions – Categorical List->Graphics->Basic Plots and Graphs-> Hold)

Η χρησιμότητα της hold είναι ότι μας δίνει την δυνατότητα να σχεδιάσουμε πολλαπλά γραφήματα σε ένα figure. Η function hold καθορίζει το αν στην επόμενη φορά που θα εισάγουμε μία νέα γραφική παράσταση θα αντικαταστήσει την προηγούμενη ή θα σχεδιαστεί από πάνω. Όταν καλούμε την hold με το όρισμα on να την ακολουθεί τότε η γραφική

παράσταση που έχουμε σχεδιάσει διατηρείται. Η δράση της hold on διαρκεί μέχρι να καλέσουμε την hold off. Τότε ότι έχουμε σχεδιάζει θα σβηστεί από την επόμενη γραφική παράσταση που θα σχεδιάσουμε. Ο παρακάτω κώδικας είναι ένα παράδειγμα της χρήσης της hold(Εικόνα 3.16).

```
x = 0:0.01:20;
y = x.^2;

plot(x,y);
hold on;
y = 20*x;
plot(x,y,'r');
hold off;
```



Εικόνα 3.18

Στο παράδειγμα πρώτα καλούμε την plot για να σχεδιάζουμε την μπλε γραφική παράσταση έπειτα καλούμε την hold on και τέλος καλούμε την plot και σχεδιάζουμε την κόκκινη γραφική παράσταση. Η hold off στο τέλος δηλώνει ότι σχεδιαστεί μετά στο figure θα σβήσει τα προηγούμενα.

9.7 Pause

(Help: Matlab->Functions – Categorical List->Programming and Data Types-> Programming in MATLAB->Pause)

Η pause σταματά την εκτέλεση του κώδικα για όσο χρονικό διάστημα έδωσε ο χρήστης σαν όρισμα. Ο παρακάτω κώδικας σχεδιάζει μία νέα γραφική παράσταση κάθε 1 δευτερόλεπτο. Αν βγάλουμε και το σχόλιο από την hold on τότε η κάθε νέα γραφική δεν θα σβήνει την προηγούμενη.

```
x = 0:0.01:20;
%hold on
for i=1:10
    y = x.^rand();      %set new y
    plot(x,y);          %plot new graph
    pause(1);           %pause for 1 second
end
```

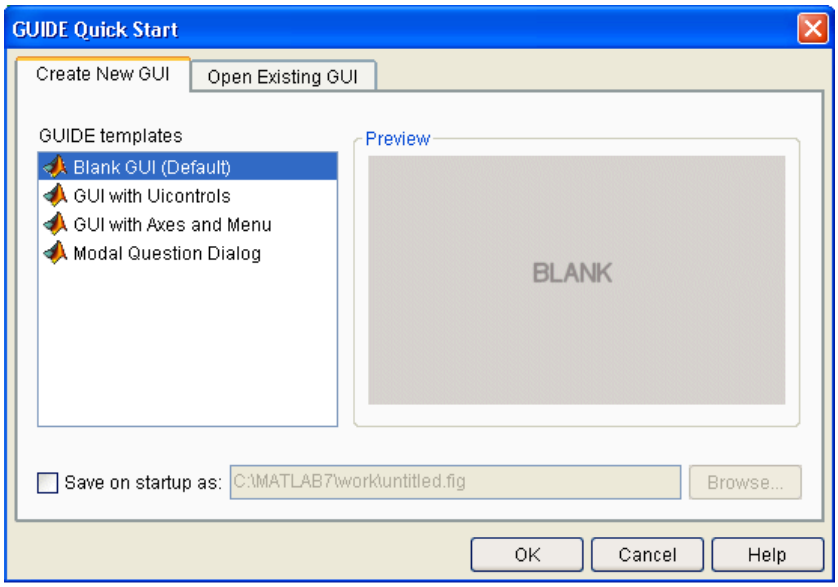
ΚΕΦΑΛΑΙΟ 10

ΑΝΑΠΤΥΞΗ ΓΡΑΦΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

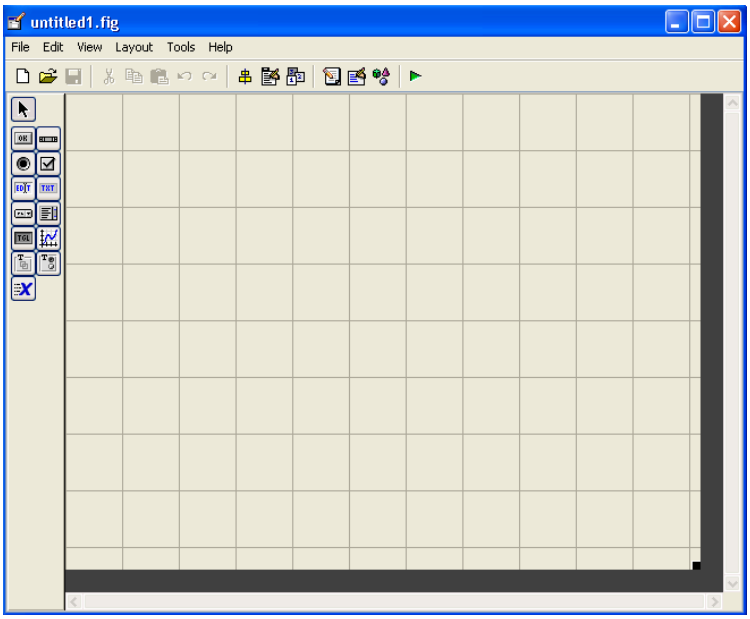
10.1 Παρουσίαση του GUIDE (GUI Layout Editor)

(Help: Matlab->Getting Started ->Creating Graphical User Interfaces->What Is GUIDE?)

Το Matlab δίνει την δυνατότητα στο χρήστη να δημιουργήσει γραφικό περιβάλλον για τις εφαρμογές του. Το εργαλείο GUIDE περιέχει όλες τις ευκολίες για το σκοπό αυτό. Εκτελώντας την εντολή guide στο command line αρχίζει η εκτέλεση του GUIDE. Αμέσως εμφανίζεται ένα dialog(Εικόνα 4.1) που μας επιτρέπει να διαλέξουμε τον τύπο figure που θέλουμε. Επιλέγουμε το «Blank GUI» και πατάμε το κουμπί «OK». Έπειτα εμφανίζεται ο GUIDE(Εικόνα 4.2).



Εικόνα 4.19



Εικόνα 4.20

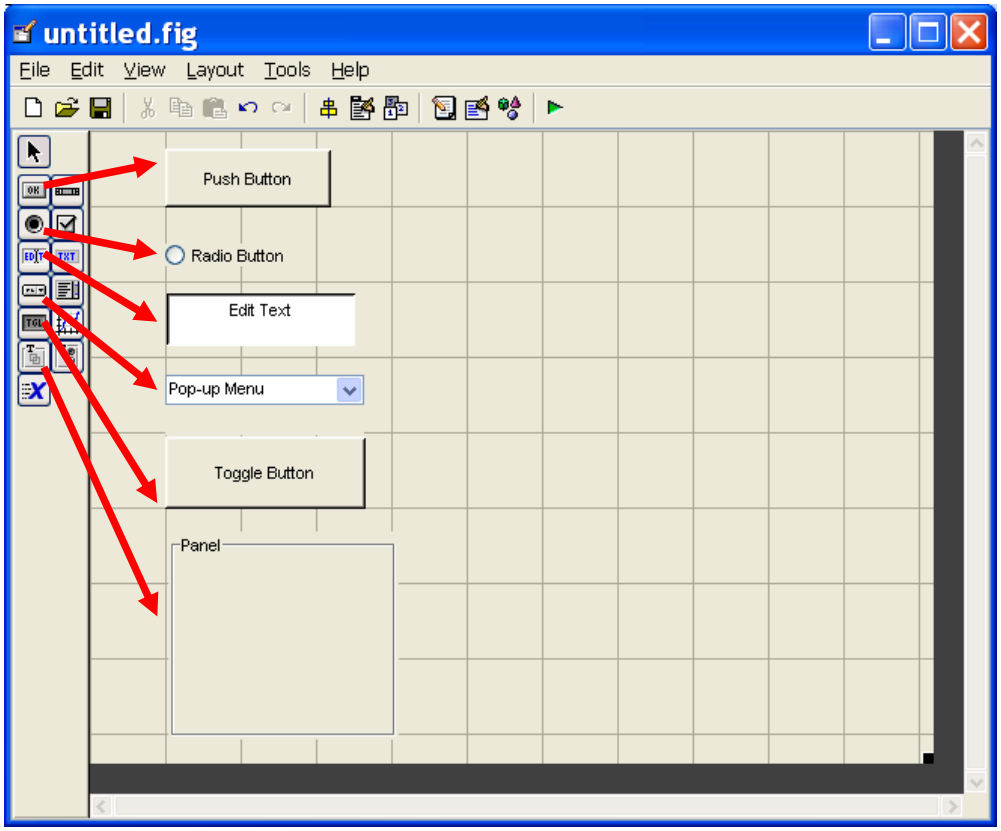
10.2 Παρουσίαση και δημιουργία gui αντικειμένων

(Help: Matlab->Getting Started ->Creating Graphical User Interfaces->The Layout Editor)

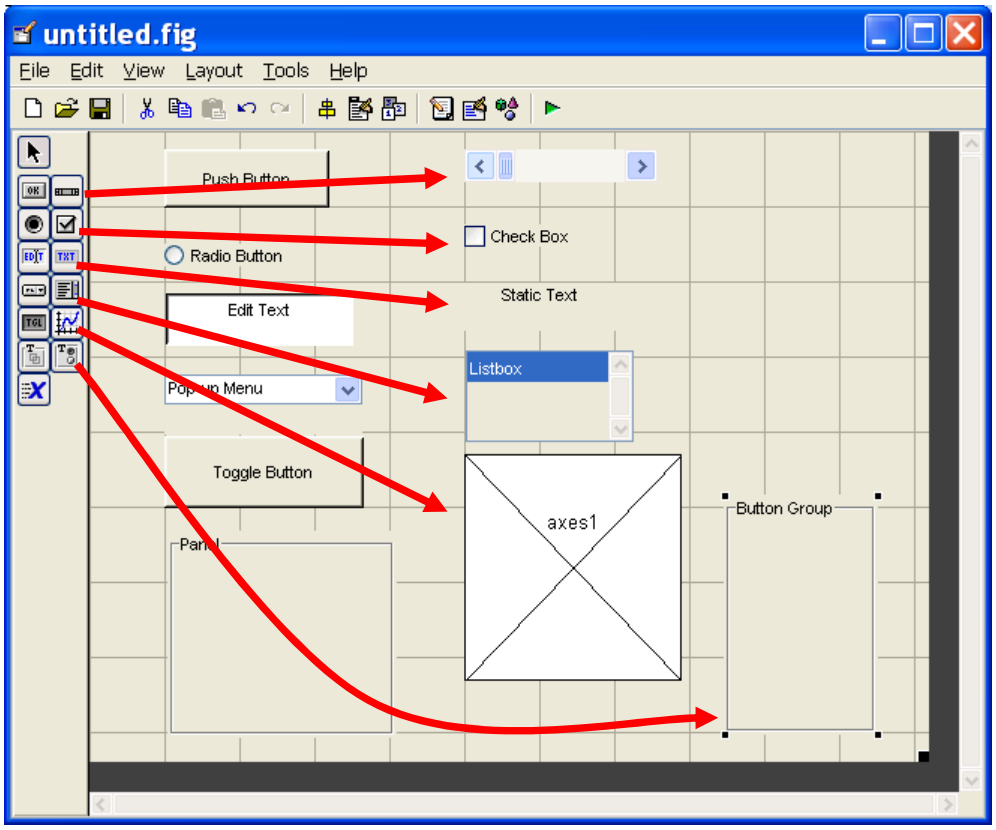
Στο κεντρικό παράθυρο του GUIDE περιέχετε ένα figure το οποίο είναι αρχικά άδειο. Στα αριστερά υπάρχει η μπάρα εργαλείων (toolbar) η οποία περιέχει διάφορα αντικείμενα (objects) –ως επί το πλείστον κουμπιά– τα οποία μπορούμε να τοποθετήσουμε στην εικόνα μας. Στις Εικόνες 4.3 και 4.4 φαίνονται ποια αντικείμενα αντιστοιχούν σε κάθε κουμπί της μπάρας εργαλείων. Τα αντικείμενα – κουμπιά στην μπάρα εργαλείων είναι τα εξής:

- *Push Button*: ορθογώνιο κουμπί. Αφού πατηθεί, επανέρχεται στην αρχική του θέση.
- *Slider*: φούσκα κυλιόμενη μεταξύ μιας ελάχιστης τιμής (0) και μιας μέγιστης τιμής (1).

- *Radio Button*: στρογγυλό κουμπί. Με κλικ αλλάζει η κατάστασή του από ○ (*off*) σε ⊙ (*on*) και αντίστροφα.
- *Check Box*: τετράγωνο κουμπί. Με κλικ αλλάζει η κατάστασή του από ☐ (*off*) σε ☑ (*on*) και αντίστροφα.
- *Edit Text*: ορθογώνια πινακίδα όπου μέσα ο χρήστης μπορεί να γράψει κείμενο
- *Static Text*: ορθογώνια πινακίδα (label). Το κείμενο είναι στατικό και δεν αλλάζει
- *Popup Menu*: λίστα που ξετυλίγεται σαν μενού επιλογών
- *List Box*: λίστα επιλογών
- *Toggle Button*: ορθογώνιο κουμπί. Με κλικ αλλάζει η κατάστασή του από πατημένο σε μη-πατημένο και αντίστροφα.
- *Axis*: άξονες συντεταγμένων για γραφικές παραστάσεις (plots)
- *Panel*: ορθογώνια περιοχή με περίφραξη και τίτλο
- *Button Group*: ομάδα από κουμπιά
- *ActiveX Control*: αντικείμενο ελέγχου ActiveX



Εικόνα 4.21

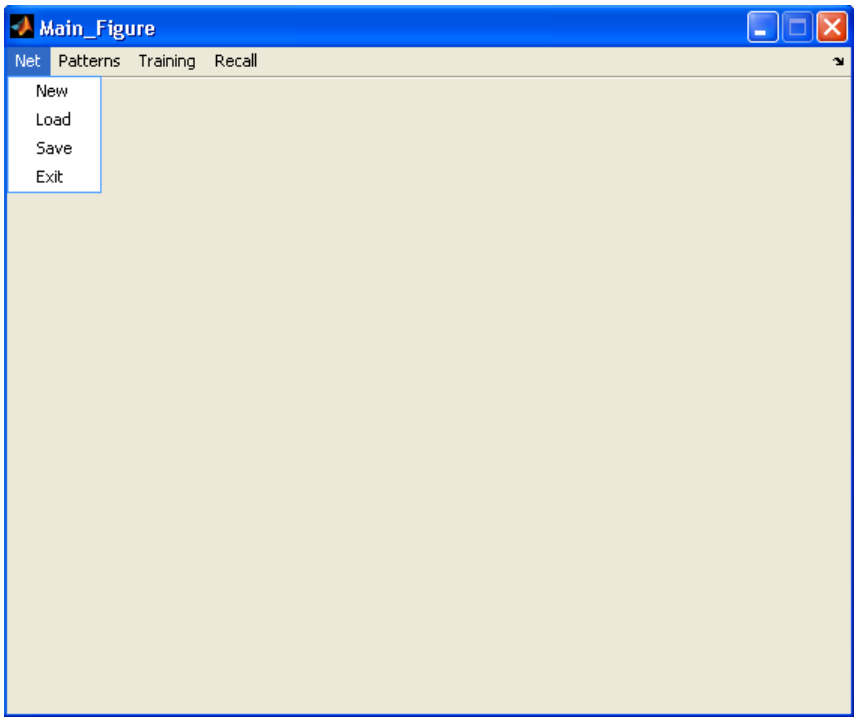


Εικόνα 4.22

10.2.1 Menu bar

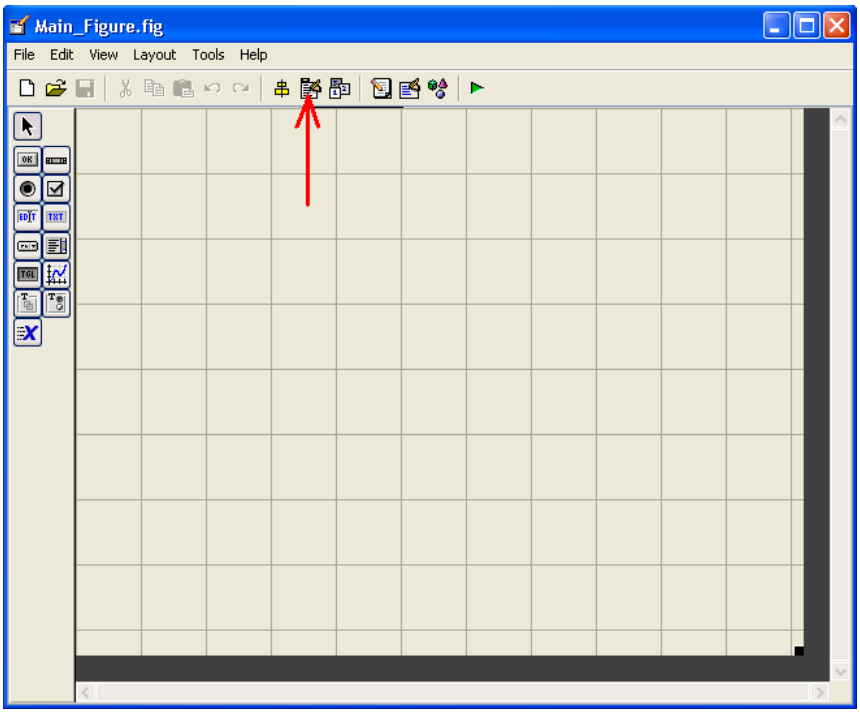
(Help: Matlab->Getting Started ->Creating Graphical User Interfaces-> Laying Out GUIs and Setting Properties-> Creating Menus -- The Menu Editor)

Το menu bar είναι το menu επιλογών που εμφανίζεται στο άνω τμήμα του figure όπως φαίνεται στην Εικόνα 4.5.



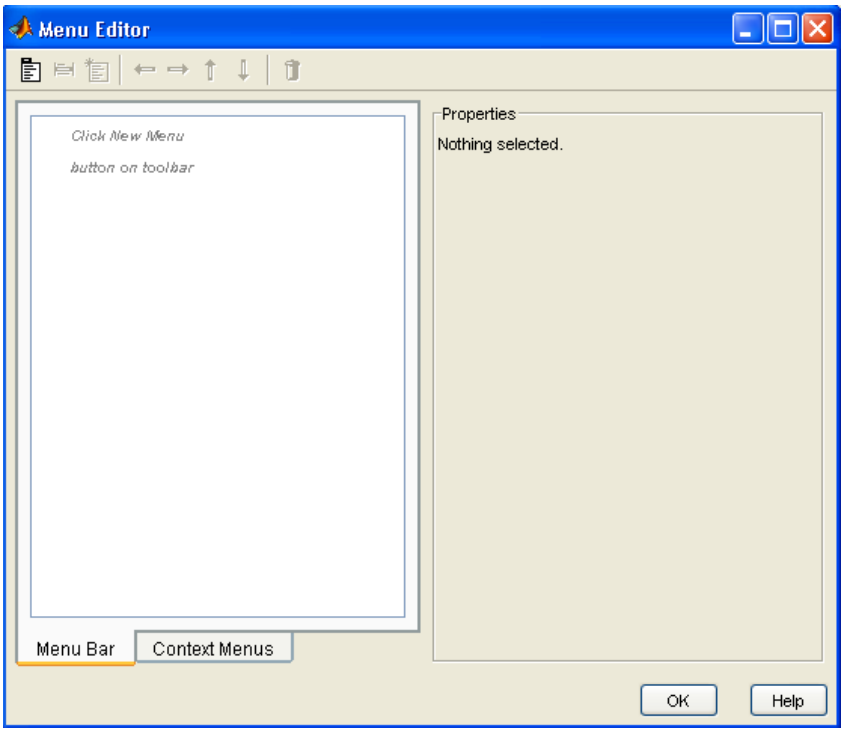
Εικόνα 4.23

Για να εκκινήσουμε το menu editor πατάμε το αντίστοιχο κουμπί στο άνω toolbar του GUIDE όπως φαίνεται στην Εικόνα 4.6.



Εικόνα 4.24

Έπειτα εμφανίζεται ο menu editor(Εικόνα 4.7).



Εικόνα 4.25

Τα κουμπιά που είναι στο πάνω toolbar κάνουν τα εξής:

- Το πρώτο κουμπί δημιουργεί ένα νέο Menu.
- Το δεύτερο κουμπί δημιουργεί ένα νέο Menu Item. Το νέο Menu Item θα είναι παιδί κάτω από το Menu που θα είναι επιλεγμένο.
- Το τρίτο κουμπί δημιουργεί Context Menu. Τα Context Menu είναι αυτά που εμφανίζονται όταν πατάει ο χρήστης το δεξί πλήκτρο που ποντικιού.
- Τα επόμενα τέσσερα κουμπιά μετακινούν το Menu ή Menu Item που έχουμε επιλεγμένο σε άλλη θέση.
- Το τελευταίο κουμπί διαγράφει το Menu ή Menu Item που έχουμε επιλεγμένο.

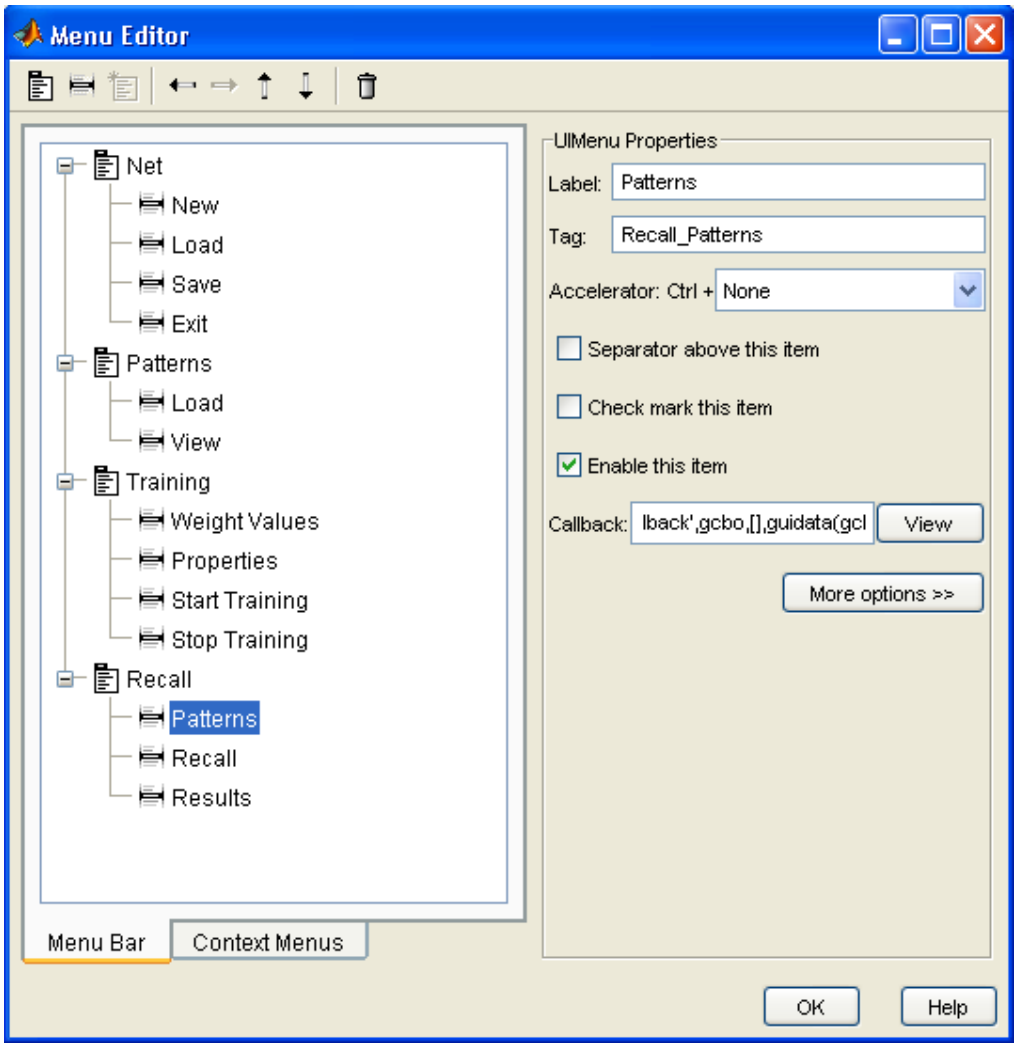
Στο δεξί panel μπορούμε αλλάξουμε τις ιδιότητες του κάθε Menu ή Menu Item. Ιδιαίτερη προσοχή πρέπει να δοθεί στα ονόματα που θα δοθούν στο tag. Το label απλά είναι το string που εμφανίζεται πάνω στο Menu ή Menu Item ενώ το tag είναι το όνομα που έχει η μεταβλητή που περιέχει όλες τις ιδιότητες του Menu ή Menu Item(ή οποιοδήποτε άλλου γραφικού αντικειμένου γενικότερα). Αν δώσουμε ίδιο tag σε δύο αντικείμενα τότε θα υπάρχουν σοβαρά προβλήματα όταν θα τα επιλέγουμε.

Εκτός από το label και το tag του αντικειμένου μπορούμε να του δώσουμε και επιπλέον ιδιότητες. Η ιδιότητα “accelerator” προσφέρει την δυνατότητα να επιλέγουμε το Menu που θέλουμε με τον συνδυασμό των πλήκτρων Ctrl συν κάποιου άλλου της επιλογής μας π.χ. Ctrl + A. Η “Separator above this item” προσθέτει μία γραμμή πάνω από το Menu Item. Αυτό

χρησιμοποιείται για λόγους ομαδοποίησης διάφορων Menu Items. Η "Check mark this item" προσθέτει ένα mark μπροστά από το Menu Item. Όταν είναι ενεργοποιημένη η "Enable this item" δηλώνει ότι το Menu μπορεί να επιλεχθεί.

Η "callback" είναι η συνάρτηση ή το script που καλείται όταν πατηθεί το κουμπί του Menu. Η λειτουργία της είναι ίδια και με την αντίστοιχη callback των υπόλοιπων αντικειμένων GUI του Matlab. Αν πατήσουμε το κουμπί View τότε θα μεταφερθούμε στο M-file του figure στο σημείο που είναι το function της callback ώστε να την τροποποιήσουμε αν θέλουμε. Επιπλέον αν δεν θέλουμε να χρησιμοποιήσουμε την default function που δημιουργεί το Matlab μπορούμε να γράψουμε το όνομα του script που επιθυμούμε. Τέλος το κουμπί "More options" εμφανίζει επιπλέον ιδιότητες για το Menu οι οποίες είναι ανάλογες των υπολοίπων GUI αντικειμένων. Σχεδόν όλες αυτές οι ιδιότητες μπορούν να αλλάξουν κατά την εκτέλεση της εφαρμογής με την χρήση κώδικα.

Στην Εικόνα 4.8 είναι ένα παράδειγμα ενός menu bar. Στην παρούσα κατάσταση όποιο Menu ή Menu Item επιλέξουμε δεν θα γίνει τίποτα διότι δεν έχουμε προσθέσει κώδικα στα callbacks.

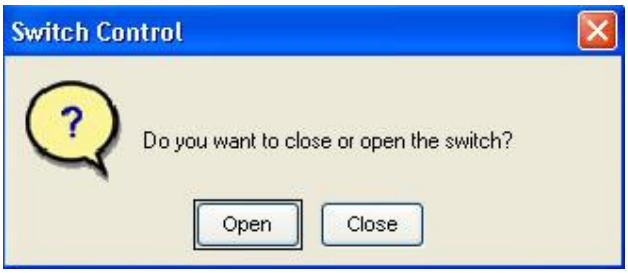


Εικόνα 4.26

10.2.2 Default Dialogs

(Help: Matlab->Functions - By Category->Creating Graphical User Interfaces-> Predefined Dialog Boxes)

Το Matlab προσφέρει την δυνατότητα δημιουργίας dialogs. Τα dialogs είναι ειδικού τύπου παράθυρα τα οποία χρησιμοποιούνται για να δώσουν πληροφορίες στον χρήστη ή να ζητήσουν εισαγωγή πληροφορίας π.χ. την εμφάνιση ενός μηνύματος λάθους ή την επιβεβαίωση για την διαγραφή ενός αρχείου. Στην Εικόνα 4.9 φαίνεται ένα dialog που ρωτά τον χρήστη αν θέλει να κλείσει την εφαρμογή.



Εικόνα 4.27

Το παραπάνω dialog δημιουργήθηκε με την εντολή:

```
button = questdlg('Do you want to close the application?', 'Question')
```

Η πρώτη παράμετρος είναι η ερώτηση που θέλουμε να εμφανίζεται και η δεύτερη παράμετρος είναι ο τίτλος του dialog. Η τιμή που επιστρέφεται μπορεί να είναι “Yes”, “No” ή “Cancel”, δηλαδή το κουμπί που πάτησε ο χρήστης. Στο παράδειγμα μας αποθηκεύσαμε αυτή την τιμή στην μεταβλητή button. Το παρακάτω script δημιουργεί ένα question dialog και αναλόγως το κουμπί που πατήσει ο χρήστης εμφανίζει το αντίστοιχο μήνυμα.

```
button = questdlg('Press a button','title');          %Δημιουργία question dialog

if strcmp(button,'Yes')
   fprintf('You clicked Yes\n');                      %Αν ο χρήστης πατήσει το πλήκτρο
Yes
end

if strcmp(button,'No')
   fprintf('You clicked No\n');                       %Αν ο χρήστης πατήσει το πλήκτρο
No
end

if strcmp(button,'Cancel')
   fprintf('You clicked  Cancel\n');          %Αν ο χρήστης πατήσει το πλήκτρο Cancel
end
```

Τα string ‘Yes’, ‘No’ και ‘Cancel’ είναι τα default της εντολής questdlg. Μπορούμε να τα αντικαταστήσουμε βάζοντας επιπλέον παραμέτρους στην questdlg. Η μέθοδος strcmp είναι η κατάλληλη για την σύγκριση strings. Η παρακάτω εντολή δημιουργεί ένα question dialog με δύο κουμπιά, το ένα γράφει ‘Open’ και το άλλο ‘Close’.

```
button = questdlg('Do you want to close or open the switch?', 'Switch Control', 'Open',
'Close', 'Open')
```

Η πρώτη παράμετρος είναι η ερώτηση, η δεύτερη ο τίτλος του dialog, η τρίτη και τέταρτη παράμετρος είναι τα strings που θα εμφανίζονται στα δύο κουμπιά και η τελευταία παράμετρος είναι ποιο κουμπί θέλουμε να είναι επιλεγμένο κατά την δημιουργία του dialog. Σημειώνεται ότι μπορούμε να δημιουργήσουμε question dialogs που διαθέτουν το πολύ τρία κουμπιά. Στην Εικόνα 4.10 φαίνεται το dialog της παραπάνω εντολής.

Εικόνα 4.28

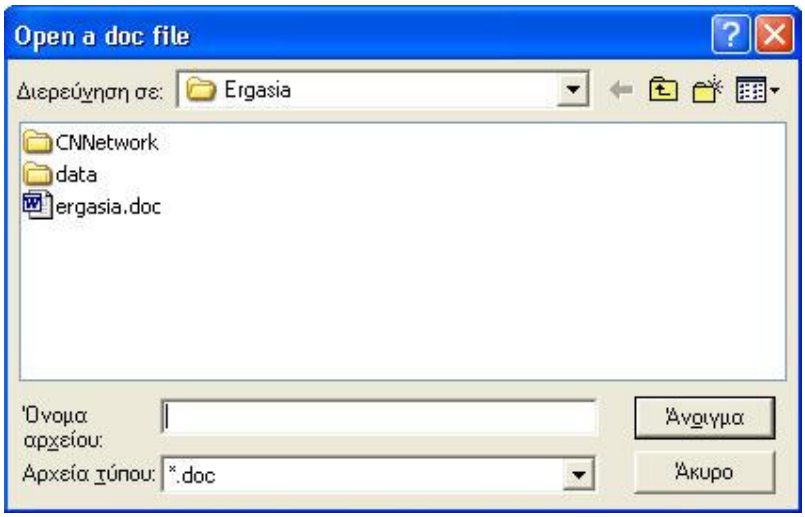
uigetfile και uiputfile dialogs

Δύο χρήσιμα dialogs είναι τα uigetfile και uiputfile. Χρησιμοποιούνται για την επιστροφή των ονομάτων και θέσης των αρχείων που θέλουμε να διαβάσουμε ή να εισάγουμε δεδομένα. Το παρακάτω script δημιουργεί ένα dialog που ζητά από τον χρήστη να διαλέξει το αρχείο που θέλει να διαβάσει. Έπειτα μπορούμε να ανοίξουμε το αρχείο και να το επεξεργαστούμε. Επιπλέον επιστρέφει το όνομα και την θέση του αρχείου σε δύο μεταβλητές για μελλοντική επεξεργασία.

```
[filename, pathname] = uigetfile('*.doc' , 'Open a doc file');     %Δημιουργία
                                                                   %dialog

%Η παρακάτω if ελέγχει  αν ο χρήστης επέλεξε κάποιο αρχείο ώστε να γίνει
επεξεργασία
if isequal(filename,0)                                             %Ο χρήστης δεν έχει
                                                                   %επιλέξει αρχείο
  disp('User selected Cancel')
else
  disp(['User selected', fullfile(pathname, filename)])            %Ο χρήστης επέλεξε
                                                                   %αρχείο
  fid=fopen([pathname,filename],'r');                              %Ανοίγουμε το αρχείο
                                                                   %για διάβασμα
end
```

Η παραπάνω if χρησιμοποιείται γιατί αν ο χρήστης δεν έχει επιλέξει κάποιο αρχείο και προχωρήσουμε στον κώδικα της επεξεργασίας αρχείου τότε θα εμφανιστεί σφάλμα εκτέλεσης.

Το πρώτο όρισμα της uigetfile είναι το φίλτρο, δηλαδή εμφανίζονται στο dialog μόνο τα αρχεία με κατάληξη doc. Το δεύτερο όρισμα είναι ο τίτλος του dialog. Στις μεταβλητές filename και pathname επιστρέφονται το όνομα και θέση του αρχείου που επιλέχθηκε. Στην Εικόνα 4.11 φαίνεται το dialog που δημιούργησε το παραπάνω script.



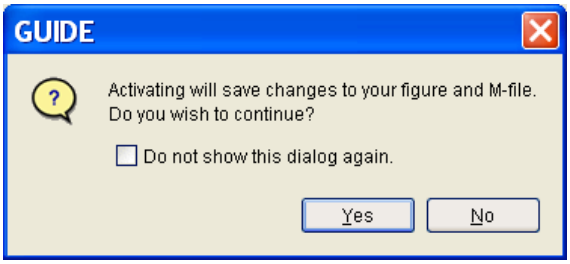
Εικόνα 4.29

Αντίστοιχα η uiputfile χρησιμοποιείτε για να αποθηκεύσουμε ένα αρχείο. Το παρακάτω script δημιουργεί ένα dialog που ζητά από τον χρήστη το όνομα του αρχείου στο οποίο θέλουμε να αποθηκεύσουμε. Όταν δεν υπάρχει το αρχείο τότε δημιουργείται. Στην περίπτωση που υπάρχει ήδη το αρχείο ο χρήστης ερωτείται αν θέλει να αντικαταστήσει το αρχείο ή όχι.

```
[filename,pathname] = uiputfile('*.doc' , 'Save a doc file');
```

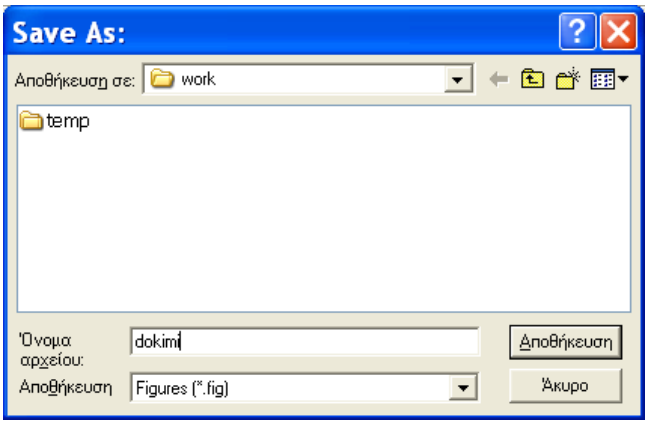
10.3 Αποθήκευση και έναρξη εφαρμογής

Αφού κατασκευάσαμε το παράθυρο της εφαρμογής μας με τα κουμπιά που επιθυμούμε μπορούμε να τρέξουμε την εφαρμογή μας πατώντας το κουμπί "*Run*"  . Το `guide` θα μας ζητήσει να επιβεβαιώσουμε τις αλλαγές που κάναμε



Εικόνα 4.30

Επιλέγοντας "Yes" το `guide` θα μας ζητήσει να δώσουμε ένα όνομα στην εφαρμογή μας. Έστω ότι επιλέγουμε το όνομα `dokimi`



Εικόνα 4.31

Κατόπιν το `GUIDE` θα σώσει την εφαρμογή μας σε δύο αρχεία: `dokimi.fig` και `dokimi.m`. Ταυτόχρονα θα ανοίξει η εφαρμογή στο παρακάτω παράθυρο. Προσέξτε ότι η μπάρα του τίτλου, επάνω αριστερά, γράφει το όνομα της εφαρμογής, δηλαδή "`dokimi`". Αφού κλείσουμε την εφαρμογή μπορούμε να την ξανατρέξουμε με οποιονδήποτε από τους παρακάτω τρόπους:

- Αν τρέχουμε ακόμη το `GUIDE` πατάμε το κουμπί *Run*.
- Αν έχουμε βγει από το `GUIDE` κάνουμε ένα από τα παρακάτω:
  - Γράφουμε στο *command line* την εντολή
    >> dokimi
    οπότε ανοίγει η εφαρμογή αλλά όχι το παράθυρο σχεδίασης
  - Γράφουμε στο *command line* την εντολή
    >> guide(dokimi)
    οπότε ανοίγει και η εφαρμογή και το παράθυρο σχεδίασης
  - Γράφουμε στο *command line* την εντολή
    >> guide
    και όταν βγει το παράθυρο του guide επιλέγουμε *"Open Existing GUI"* για να ανοίξει το παράθυρο σχεδίασης και κατόπιν η εφαρμογή τρέχει με *Run*.

10.4 Fig και M-File του figure

(Help: Matlab->Getting Started ->Creating Graphical User Interfaces-> Laying Out a GUI-> Programming a GUI)

Αν ανοίξουμε τον φάκελο που αποθηκεύσαμε το dokimi.fig θα δούμε ότι έχει γίνει ένα επιπλέον αρχείο, το dokimi.m. Το αρχείο fig περιέχει μία ολοκληρωμένη περιγραφή του GUI figure και όλων των GUI αντικειμένων του (uicontrols και axes) και τις τιμές όλων των ιδιοτήτων των αντικειμένων που μπορεί να υπάρχουν σε αυτό. Μπορούμε να κάνουμε αλλαγές σε αυτό το αρχείο με τον GUIDE. Το m αρχείο περιέχει τις μεθόδους που τρέχουν και ελέγχουν το GUI και τις callbacks. Αυτό το αρχείο αναφέρεται σαν GUI M-file.

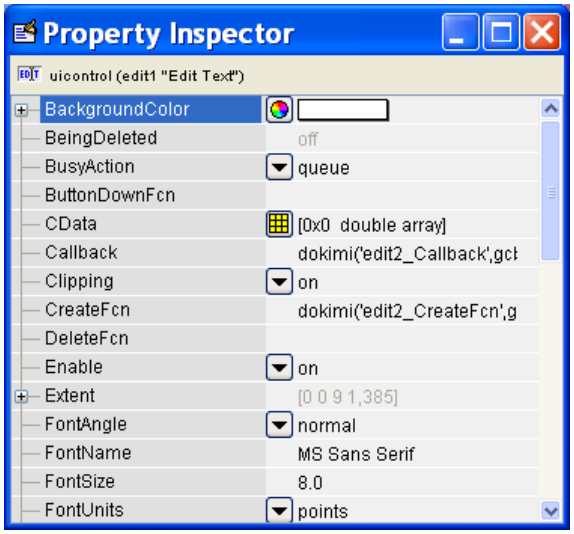
Ο GUIDE για λόγους ευκολίας δημιουργεί το M-file απευθείας από το layout(το σχέδιο του figure που έχουμε σχεδιάσει με τον GUIDE όπως τα κουμπιά, τα plot ή οποιοδήποτε άλλο γραφικό αντικείμενο που έχουμε προσθέσει). Επίσης παράγονται αυτόματα και οι callbacks για τα GUI αντικείμενα που τους χρειάζονται. Αρχικά δημιουργούνται δύο callbacks η Opening function και η Output function. Η πρώτη εκτελείται πριν το figure γίνει ορατό στο

χρήστη. Η Output function επιστρέφει δεδομένα στο command line. Τα default ονόματα των δύο αυτών μεθόδων είναι my_gui_OpeningFcn και my_gui_OutputFcn αντίστοιχα(όπου my_gui είναι το όνομα που δώσαμε στο figure μας, στην περίπτωσή μας dokimi). Και στις δύο μεθόδους μπορούμε να προσθέσουμε δικό μας κώδικα όπως π.χ. στην Opening function μπορούμε να αρχικοποιήσουμε δεδομένα. Κάθε φορά που εισάγουμε ένα γραφικό αντικείμενο με το GUIDE παράγετε και στο M-File το αντίστοιχο callback.

10.5 Ιδιότητες των gui αντικειμένων

(Help: Matlab->Getting Started ->Graphics->Handle Graphics)

Όλα τα κουμπιά που μπορούν να μπουν σε ένα παράθυρο, ακόμη και το κεντρικό παράθυρο αυτό καθ' αυτό, είναι αντικείμενα (objects) και ως τέτοια έχουν κάποιες ιδιότητες (properties). Ειδικά τα κουμπιά Push button, Slider, Radio button, Check box, Editable text, Static text, Pop-up menu, List box, και Toggle button ανήκουν στην κατηγορία αντικειμένων uicontrol (uicontrol objects – βλ. MATLAB Help). Το κεντρικό παράθυρο ανήκει στην κατηγορία figure. Με δεξί κλικ επάνω στο αντικείμενο που μας ενδιαφέρει μπορούμε να επιλέξουμε τον "Property Inspector" ανοίγοντας ένα παράθυρο, όπως της Εικόνας 5.14.



Εικόνα 4.32

Το συγκεκριμένο παράθυρο περιέχει τις ιδιότητες του αντικειμένου *"Edit Text"* με όνομα `edit1`. Το όνομα του αντικειμένου βρίσκεται στο πεδίο *"Tag"* και φυσικά μπορούμε να το αλλάξουμε. Κάποια σημαντικά πεδία ιδιοτήτων που συχνά χρειάζονται αλλαγή είναι:

- *Background Color*, *Foreground Color* = χρώμα background και χρώμα foreground (συνήθως το χρώμα του κειμένου). Τα χρώματα μπορούν να οριστούν
  - είτε σαν τριάδες [red, green, blue] όπου κάθε ένα από τα τρία χρώματα έχει τιμή μεταξύ 0 και 1
  - είτε χρησιμοποιώντας την παλέττα.
- *FontName*, *FontSize* = ιδιότητες του Font του κειμένου που περιέχεται στο κουμπί. Για ελληνικά μπορείτε να επιλέξετε, πχ., *FontName* ← `MS Sans Serif Greek`, `Times New Roman Greek`, `Arial Greek`, κα.

- *String* = το κείμενο που εμφανίζεται στο κουμπί.
- *Callback* = η συνάρτηση ή το script που καλείται όταν πατηθεί το κουμπί αυτό
- *κλπ*

Φυσικά το πλήθος και το είδος των πεδίων εξαρτάται από τον τύπο του κουμπιού. Η ανάγκη μας να διαβάσουμε ή να αλλάξουμε τις τιμές που περιέχει ένα πεδίο εξαρτάται προφανώς από την συγκεκριμένη εφαρμογή που θέλουμε να υλοποιήσουμε.

10.6 Δέντρο αντικειμένων(handles, findall, `guihandles`)

(Help: Matlab->Getting Started ->Graphics->Handle Graphics)

Η δημιουργία μιας παραθυρικής εφαρμογής GUI δημιουργεί αυτομάτως μια δομή δέντρου ανάμεσα στα αντικείμενα που περιλαμβάνει η εφαρμογή αυτή. Η ρίζα του δέντρου είναι το κεντρικό παράθυρο το οποίο είναι πάντα αντικείμενο τύπου figure. Παιδιά του αντικειμένου αυτού είναι όλα τα αντικείμενα που περιέχει το παράθυρο. Σε κάθε αντικείμενο αντιστοιχεί ένας δείκτης (handle). Κάποιοι χρήσιμοι αυτόματοι δείκτες (δεσμευμένες μεταβλητές στο MATLAB) είναι οι παρακάτω:

- `gcf` = δείκτης στο τρέχον figure (get current figure)
- `gco` = δείκτης στο τρέχον object (get current object). Τρέχον είναι το παραθυρικό αντικείμενο (κουμπί, παράθυρο, μενού, κλπ) που έχει το focus. Ένα αντικείμενο αποκτάει focus όταν κάνουμε κλικ επάνω του.
- `gca` = δείκτης στο τρέχον axis (get current axis)
- `gcbo` = δείκτης στο object του οποίου το callback εκτελείται. Το αντικείμενο gcbo είναι ίδιο με το αντικείμενο gco εαν το callback αντιστοιχεί στην ενέργεια «κλικ». Αν όμως το callback αντιστοιχεί σε άλλο event εκτός από κλικ μπορεί τα δύο αντικείμενα να μην είναι ίδια.

Η εντολή findall βρίσκει όλα τα handles όλων των παραθυρικών αντικειμένων είτε αυτά είναι figures είτε είναι παιδιά άλλων αντικειμένων. Τα handles αυτά επιστρέφονται στο array `h`.

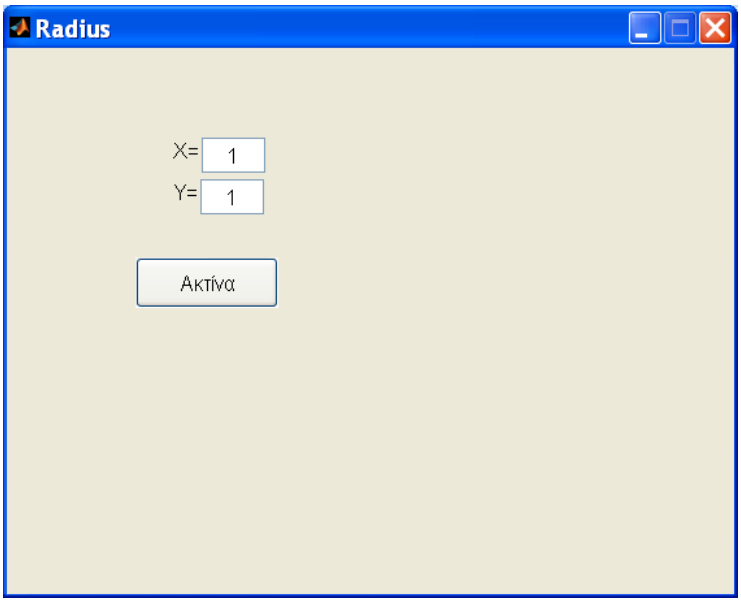
```
>> h = findall(0);
```

Η εντολή guihandles φορτώνει στη δομή `handles` τους δείκτες όλων των αντικειμένων που βρίσκονται μέσα στο δέντρο του τρέχοντος *figure*.

```
>> handles = guihandles(gcf);
```

Παράδειγμα.

Στην Εικόνα 4.16 φαίνεται ένα παράδειγμα παραθυρικής εφαρμογής με ακριβώς αυτό το πλήθος και τύπο παιδιών, δηλαδή δύο static texts (οι πινακίδες `'X='` και `'Y='`), δύο editable texts (τα δύο κουτιά στα δεξιά με τιμές `1` και `1`), και ένα push button (το κουμπί κάτω με String = `'Ακτίνα'`).



Εικόνα 4.33

Έστω ότι κάθε φορά που κάνουμε κλικ στο push button `'Ακτίνα'` επιθυμούμε να υπολογίζουμε το άθροισμα των τετραγώνων των τιμών που είναι γραμμένες στα δύο editable boxes. Ένας απλός τρόπος να το πετύχουμε αυτό είναι να ορίσουμε το παρακάτω script ως callback για το push button:

```
handles = guihandles(gcf);
stringx = get(handles.edit1,'String');
stringy = get(handles.edit2,'String');
x = str2num(stringx);
y = str2num(stringy);
aktina = x^2 + y^2
```

Κατ' αρχήν καλούμε την `handles = guihandles(gcf)` έτσι ώστε να πάρουμε τα handles από όλα τα αντικείμενα της εφαρμογής. Κατόπιν διαβάζουμε την ιδιότητα String τόσο από το αντικείμενο `edit1` όσο και από το αντικείμενο `edit2`. Οι απαντήσεις που θα πάρουμε στις μεταβλητές `stringx` και `stringy`, θα είναι τύπου string. Συνεπώς, για να χρησιμοποιήσουμε τους αντίστοιχους αριθμούς ώστε να υψώσουμε στο τετράγωνο, κλπ, πρέπει να καλέσουμε τη συνάρτηση `str2num()` δύο φορές ώστε να πάρουμε τους αριθμούς `x` και `y`. Κατόπιν εκτελούμε την πράξη `aktina = x^2 + y^2` ώστε να πάρουμε το επιθυμητό αποτέλεσμα.

10.7 Οι εντολές `set`, `get`. Διάβασμα και αλλαγή των ιδιοτήτων από το command line

(Help: Matlab->Getting Started ->Graphics->Handle Graphics->Using the Handle)

Οι εντολές `set` και `get` επιτρέπουν την πρόσβαση στις ιδιότητες ενός παραθυρικού αντικειμένου μέσα από το command line αρκεί να γνωρίζουμε το handle του αντικειμένου. Για παράδειγμα, εκτελώντας την εντολή

```
>> x = get(fh, 'Tag')
```

παίρνουμε ως απάντηση την τιμή της ιδιότητας '*Tag*' του αντικειμένου με *handle*=`fh`.

```
x =
        figure1
```

Αντίστοιχα εκτελώντας την εντολή

```
>> x = get(fh);
```

παίρνουμε μια ολόκληρη δομή `x` που περιέχει όλες τις ιδιότητες του αντικειμένου με *handle*=`fh`.

Εντελώς ανάλογα, η εντολή

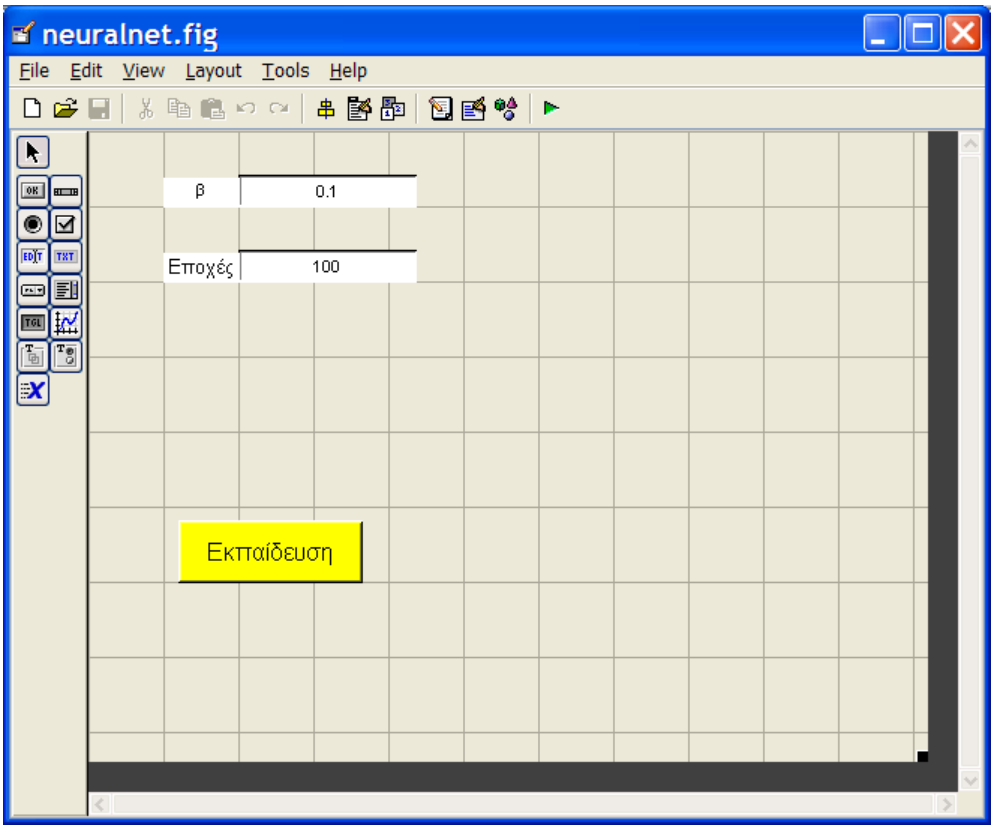
```
>> set(fh, 'Tag', 'button1')
```

αναθέτει την τιμή `'button1'` στην ιδιότητα '*Tag*' του αντικειμένου με *handle*=`fh`.

10.8 Callbacks

(Help: Matlab->Creating Graphical User Interfaces->Creating GUIs Programmatically-> Programming the GUI->Callbacks: An Overview)

Τα κουμπιά που βάλαμε στο παράθυρο έχουν κάποιες ελάχιστες (default) λειτουργίες: πχ. το *toggle button* αφού πατηθεί θα παραμείνει πατημένο. Ωστόσο δεν έχουμε δώσει ακόμα στα κουμπιά τη λειτουργικότητα που θα θέλαμε έτσι ώστε να κάνουν κάτι χρήσιμο για την εφαρμογή μας. Η λειτουργία που εκτελείται όταν πατηθεί ένα κουμπί, καλείται *callback*. Το MATLAB προσφέρει έτοιμες συναρτήσεις *callback* για κάθε κουμπί της εφαρμογής μας. Αν υποθέσουμε ότι η εφαρμογή μας είναι η `neuralnet.fig` τότε οι *συναρτήσεις-callback* βρίσκονται μέσα στο αρχείο `neuralnet.m`. Έστω, π.χ., ότι το `neuralnet.fig` είναι το παρακάτω παράθυρο



Εικόνα 4.34

Η εφαρμογή μας έχει ένα κεντρικό παράθυρο με *Tag* = '`figure1`' έχει το οποίο περιέχει τα εξής παιδιά

A/A	*Τύπος παιδιού*	*Tag*
1	Push button	'pushbutton1'
2	Static text	'text1'
3	Static text	'text2'
4	Editable text	'edit1'
5	Editable text	'edit2'

Αν ανοίξουμε το αρχείο neuralnet.m θα δούμε ότι περιέχει τις παρακάτω συναρτήσεις:

function varargout = neuralnet(varargin): είναι η κεντρική συνάρτηση που αν την εκτελέσουμε τρέχει η παραθυρική μας εφαρμογή

function neuralnet_OpeningFcn(hObject, eventdata, handles, varargin): εκτελείται ακριβώς πριν να εμφανιστεί στην οθόνη το παράθυρο neuralnet

function varargout = neuralnet_OutputFcn(hObject, eventdata, handles): οι έξοδοι αυτής της συνάρτησης επιστρέφονται στο command line

function pushbutton1_Callback(hObject, eventdata, handles): εκτελείται όταν πατηθεί το κουμπί 'pushbutton1'

function edit1_Callback(hObject, eventdata, handles): εκτελείται όταν γράψουμε κάτι στο edit box 'edit1'

function edit1_CreateFcn(hObject, eventdata, handles): εκτελείται κατά τη δημιουργία του κουμπιού 'edit1' και αφού έχουν γίνει set όλες οι ιδιότητές του

function edit2_Callback(hObject, eventdata, handles): εκτελείται όταν γράψουμε κάτι στο edit box 'edit2'

function edit2_CreateFcn(hObject, eventdata, handles): εκτελείται κατά τη δημιουργία του κουμπιού 'edit2' και αφού έχουν γίνει set όλες οι ιδιότητές του

Σημειώστε ότι τα αντικείμενα τύπου static text δεν έχουν callbacks.

Αν θέλουμε να ορίσουμε ή να αλλάξουμε τη λειτουργία ενός κουμπιού, για παράδειγμα του `'pushbutton1'`, δεν έχουμε παρά να γράψουμε τον κώδικα που επιθυμούμε μέσα στην συνάρτηση pushbutton1_Callback. **Προσέξτε**! Όποιες μεταβλητές χρησιμοποιηθούν μέσα στη συνάρτηση αυτή είναι τοπικές και δεν μπορούν να χρησιμοποιηθούν από άλλες συναρτήσεις ή scripts εκτός αν δηλωθούν `global`.

Οι παραπάνω συναρτήσεις παίρνουν αυτόματα παραμέτρους εισόδου τις μεταβλητές

- hObject: είναι το `handle` στο κουμπί του οποίου το callback εκτελείται
- handles: είναι μια δομή struct που περιέχει τα handles σε όλα τα παραθυρικά αντικείμενα (κουμπιά, παράθυρα, μενού, κλπ). Τα ονόματα των πεδίων της δομής είναι τα *Tags* των αντικειμένων. Ετσι έχουμε την ακόλουθη *δομή* (*struct*):

handles =

figure1: 160.0018

text2: 11.0029

edit2: 10.0029

text1: 9.0029

edit1: 8.0039

pushbutton1: 161.0018

Αυτός ο «περίεργος» αριθμός είναι pointer στο αντικείμενο `'figure1'`. Το MATLAB

Για παράδειγμα, το πεδίο `handles.edit1` περιέχει το *handle* του αντικειμένου με *Tag* = `'edit1'` (δηλαδή του πρώτου edit box), το πεδίο `handles.pushbutton1` περιέχει το *handle* του αντικειμένου με *Tag* = `'pushbutton1'` (δηλαδή του μοναδικού pushbutton), κλπ.

Εναλλακτικά, ο χρήστης μπορεί να ορίσει τη δική του συνάρτηση ή script ως *callback* σε κάποιο κουμπί. Το *callback* ορίζεται στο αντίστοιχο πεδίο ιδιότητας και υπάρχει πάντα μια ιδιότητα *callback* σε κάθε κουμπί. Αν λοιπόν για ένα *Push button* ορίσουμε το πεδίο ιδιότητας *callback* να έχει την τιμή

Callback ← fprintf('Hello!\n')

τότε κάθε φορά που θα πατάμε το κουμπί αυτό τότε στο *Command Window* του MATLAB θα τυπώνεται το μήνυμα

>> Hello!

Αντίστοιχα αν έχουμε γράψει ένα αρχείο script με το όνομα `mycb.m` και ορίσουμε την ιδιότητα

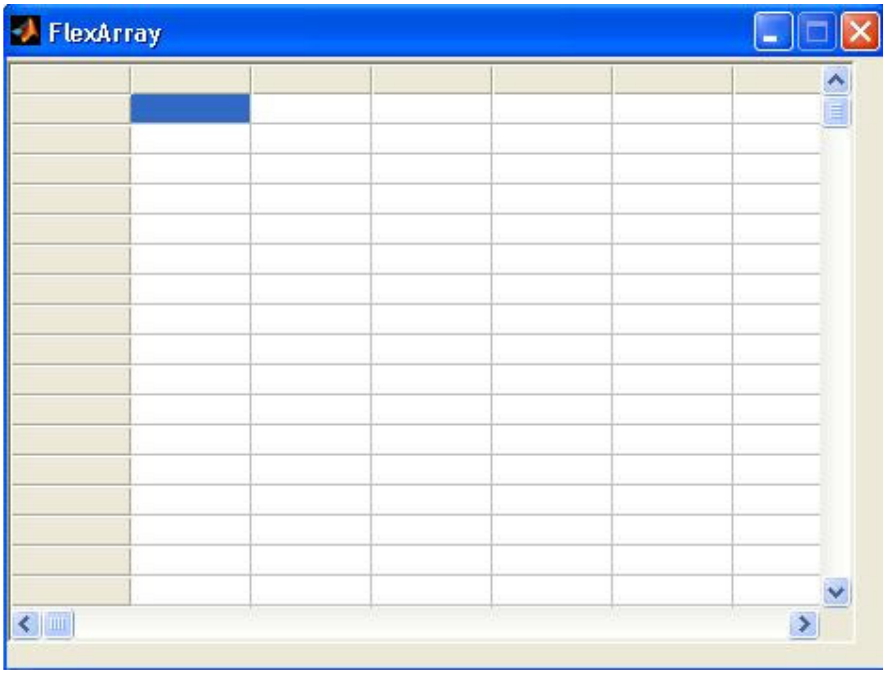
Callback ← mycb

τότε κάθε φορά που θα πατάμε το κουμπί αυτό θα εκτελείται το `mycb.m`.

10.9 Flexarrays (ActiveX control)

(Help: Matlab->MATLAB Release Notes->Version 7 (R14) MATLAB->Creating Graphical User Interfaces (GUIs), MATLAB Version 7 (R14)-> ActiveX Controls)

Το ActiveX Control είναι όρος της Microsoft που χρησιμοποιείται για να υποδηλώσει επαναχρησιμοποιούμενα κομμάτια λογισμικού που βασίζονται στο Component Object Control (COM) της Microsoft. Μία χρήση του είναι η δημιουργία ενός spreadsheet παρόμοιου με του Excel για την παρουσίαση δεδομένων πίνακα. Το συγκεκριμένο ονομάζεται 'VideoSoft FlexArray'. Στην Εικόνα 4.17 φαίνεται ένα άδειο FlexArray.



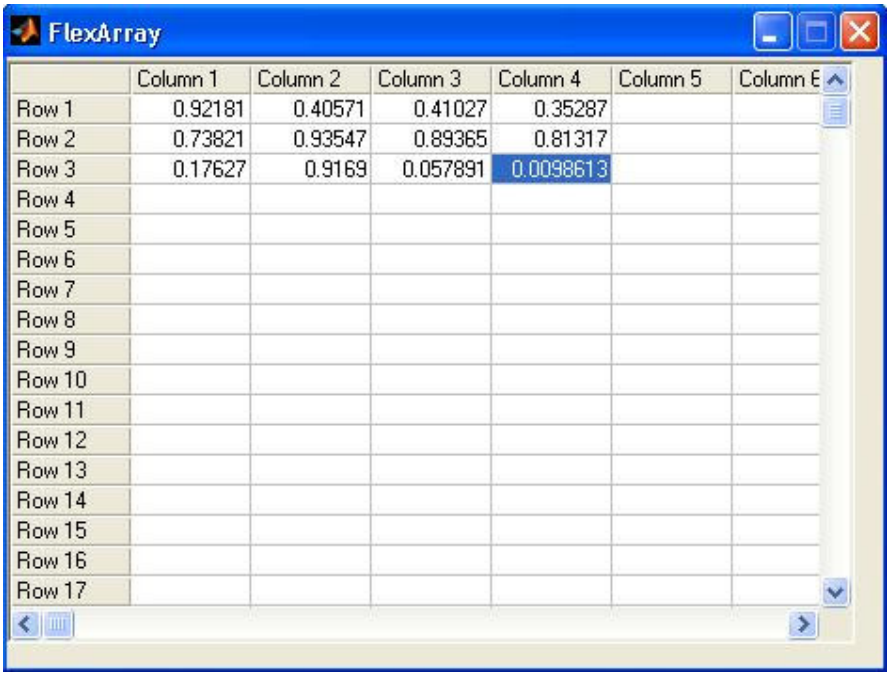
Εικόνα 4.35

Αφού δημιουργήσουμε ένα FlexArray μπορούμε να εισάγουμε κώδικα για να δώσουμε ονόματα στις στήλες και γραμμές του. Ο παρακάτω κώδικας κάνει αυτή τη δουλειά.

```
% Ονομάζουμε τις γραμμές
set(handles.activex1,'Col',0);
for k=1:get(handles.activex1, 'rows')-1
   set(handles.activex1,'Row',k);
   set(handles.activex1,'Text',['Row ' num2str(k)]);
end;

% Ονομάζουμε τις στήλες
set(handles.activex1,'row',0);
```

```
for k = 1:get(handles.activex1, 'cols')-1
   set(handles.activex1,'Col',k);
   set(handles.activex1,'Text',['Column' num2str(k)]);
end;

%Έστω πίνακας c με τυχαία δεδομένα
c = rand(3,4);

nC=size(c);   %Αποθηκεύουμε το μέγεθος του c στο nC

%Έπειτα μπορούμε να εισάγουμε τα δεδομένα
for i=1:nC(2)
   set(handles.activex1,'Col',i);
   for j=1:nC(1)
      k = c(j,i);
      set(handles.activex1, 'Row', j);
      set(handles.activex1, 'Text', [num2str(k)]);
   end
end
```

Στο παρακάτω σχήμα φαίνεται το FlexArray με τις στήλες και σειρές του ονομασμένες και τα δεδομένα του πίνακα c.



Εικόνα 4.36

Με τις παρακάτω εντολές μπορούμε να αλλάξουμε το μέγεθος του FlexArray.

```
set(handles.activex1,'Rows', arithmosGrammwn );
set(handles.activex1,'Cols', arithmosSthlwn );
```

Αξίζει να σημειωθεί ότι στον αριθμό των γραμμών και στηλών συμπεριλαμβάνεται και τα κελιά με τα ονόματα τους, δηλαδή τα Row 1, Row 2,…, και τα Column 1, Column 2,… Άρα στην περίπτωση που θέλουμε να εμφανίσουμε ένα πίνακα 2x4 σε ένα FlexArray τότε δημιουργούμε ένα με 3 γραμμές και 5 στήλες.

ΚΕΦΑΛΑΙΟ 11

NEURAL NETWORK TOOLKIT

11.1 Παρουσίαση και Δυνατότητες

(Help: Neural Network Toolbox)

Το NNToolKit απλοποιεί την δημιουργία και χρησιμοποίηση των νευρωνικών δικτύων στο Matlab. Το toolkit περιέχει functions και structures που χειρίζονται τα νευρωνικά δίκτυα ώστε να μην χρειάζεται να γράψουμε κώδικα για τις functions ενεργοποίησης, για τους αλγόριθμους εκπαίδευσης, κτλ που θέλουμε να χρησιμοποιήσουμε.

11.2 Δομές και Συναρτήσεις του Neural Network Toolkit

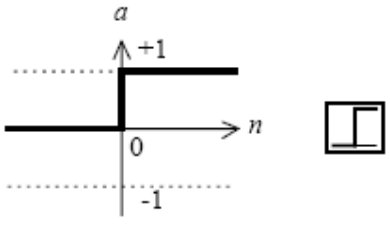
11.2.1 Συναρτήσεις Ενεργοποίησης

(Help: Neural Network Toolbox->Transfer Function Graphs)

Οι συναρτήσεις ενεργοποίησης των νευρώνων που περιέχει το NNToolKit είναι πάρα πολλές. Μία πλήρες λίστα υπάρχει στο help του Matlab. Παρακάτω παρουσιάζονται οι πιο κοινές από αυτές.

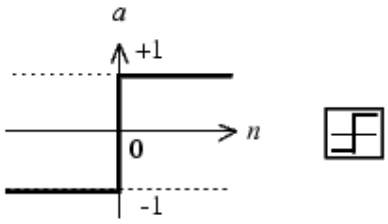
Βηματική(0/1)

a=hardlim(n)



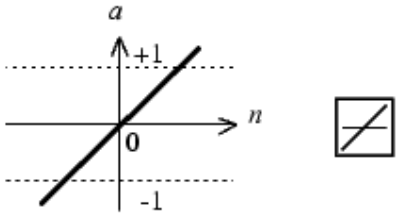
Βηματική(-1/1)

a=hardlims(n)



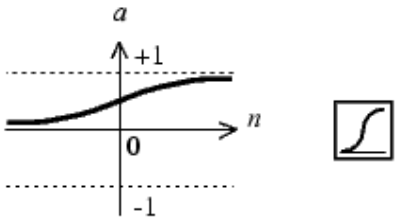
Γραμμική

a=purelin(n)



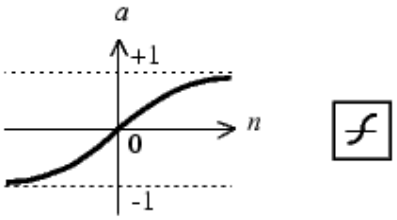
Σιγμοειδείς

a=logsig(n)



Υπερβολική Εφαπτομένη

a=tansig(n)



5.2.2 Δομές και Συναρτήσεις δημιουργίας δικτύων

(Help: Neural Network Toolbox->Functions – Categorical List->New Network Functions)

Στο NNToolKit περιέχονται δομές(structs) για την αποθήκευση των ιδιοτήτων των δικτύων. Αυτές οι δομές δημιουργούνται από συγκεκριμένες συναρτήσεις.

11.2.2.1 Perceptron

Ένα δίκτυο Perceptron δημιουργείται με την συνάρτηση newp.

```
net = newp(PR, S ,TF, LF)
```

όπου:
PR -- ένας πίνακας Rx2 όπου περιέχει το ελάχιστο και μέγιστο για R εισόδους.
S -- ο αριθμός των νευρώνων.
TF -- η συνάρτηση ενεργοποίησης (προεπιλογή είναι η hardlim)
LF -- η συνάρτηση εκπαίδευσης (προεπιλογή είναι η learnp)

Η TF μπορεί να πάρει τις τιμές 'hardlim' ή 'hardlims' και η LF τις τιμές 'learnp' και 'learnpn'. Στη μεταβλητή net επιστρέφεται το struct με τις ιδιότητες του δικτύου που δημιουργήσαμε. Περιέχει την αρχιτεκτονική του δικτύου, τις συναρτήσεις ενεργοποίησης, εκπαίδευσης, τις παραμέτρους εκπαίδευσης, τα βάρη κτλ. Στο command window μπορείτε να γράψετε το όνομα του δικτύου που δημιουργήσαμε για να δείτε λεπτομερώς τη δομή του struct.

Παράδειγμα

Έστω ότι θέλουμε να δημιουργήσουμε ένα perceptron που θα προσομοιώνει μία πύλη OR. Το πρώτο βήμα είναι να δημιουργήσουμε τα δεδομένα εισόδου και στόχων του δικτύου. Ο πίνακας P θα περιέχει τις εισόδους και ο πίνακας T τις εξόδους. Παρακάτω είναι ο κώδικας δημιουργίας των δύο πινάκων.

```
P = [0 0 1 1;
     0 1 0 1];          %inputs
T = [0 1 1 1];          %targets
```

Θα πρέπει να δοθεί μεγάλη προσοχή στην δομή αυτών των πινάκων. Κάθε γραμμή του πίνακα εισόδων συμβολίζει την κάθε είσοδο του νευρώνα και κάθε στήλη το κάθε πρότυπο. Ο πίνακας εξόδου έχει τόσες γραμμές όσοι είναι και οι νευρώνες της εξόδου του δικτύου, στο παράδειγμα μας έχουμε μία γραμμή διότι έχουμε ένα νευρώνα. Ο αριθμός των στηλών είναι ίσος με τον αριθμό των προτύπων εισόδων.

	$1^{ο}$ πρότυπο	$2^{ο}$ πρότυπο	$3^{ο}$ πρότυπο	$4^{ο}$ πρότυπο
$1^{η}$ είσοδος	0	0	1	1
$2^{η}$ είσοδος	0	1	0	1

Πίνακας Προτύπων - Είσοδοι

	$1^{ο}$ πρότυπο	$2^{ο}$ πρότυπο	$3^{ο}$ πρότυπο	$4^{ο}$ πρότυπο

Έξοδος $1^{ου}$ νευρώνα	0	1	1	1

Πίνακας Στόχων - Εξόδοι

Το επόμενο βήμα είναι να δημιουργήσουμε τον πίνακα PR ο οποίος περιέχει την ελάχιστη και μέγιστη τιμή για κάθε είσοδο. Άρα από κάθε γραμμή του πίνακα εισόδου βρίσκουμε την ελάχιστη και μέγιστη τιμή και τις εισάγουμε στον πίνακα PR. Κάθε γραμμή του πίνακα PR αντιστοιχεί σε κάθε είσοδο του νευρώνα. Παρακάτω είναι ο πίνακας και η εντολή για την δημιουργία τους στο Matlab.

	Ελάχιστο	Μέγιστο
$1^{η}$ είσοδος	0	1
$2^{η}$ είσοδος	0	1

Πίνακας Ελαχίστων – Μεγίστων Εισόδων

```
PR = [0 1;
      0 1];    %min and max
```

Πλέον όλα τα δεδομένα που χρειάζονται για την δημιουργία του δικτύου perceptron είναι έτοιμα. Η παρακάτω εντολή δημιουργεί το δίκτυο.

```
net = newp(PR,1);
```

Πλέον έχουμε δημιουργήσει το δίκτυο με ένα νευρώνα και με συναρτήσεις ενεργοποίησης και εκπαίδευσης τις προεπιλεγμένες.

11.2.2.2 Adaline

Ένα δίκτυο adaline δημιουργείται με την συνάρτηση newlin (linear layer).

```
net = newlin(PR,S,ID,LR)
```

όπου:
PR -- ένας πίνακας Rx2 όπου περιέχει το ελάχιστο και μέγιστο για R εισόδους.
S -- ο αριθμός των νευρώνων.
ID -- ένα διάνυσμα που περιέχει την χρονική καθυστέρηση (προεπιλογή είναι το διάνυσμα [0]).
LR -- ο ρυθμός εκπαίδευσης (προεπιλογή είναι η τιμή 0.01).

11.2.2.3 Multi-layer Perceptron

Ένα δίκτυο MLP δημιουργείται με την συνάρτηση newff.

```
net = newff(PR,[S1 S2...SNl],{TF1 TF2...TFNl},BTF,BLF,PF)
```

όπου:
PR -- ένας πίνακας Rx2 όπου περιέχει το ελάχιστο και μέγιστο για R εισόδους.
Si -- το μέγεθος του $ι^{th}$ στρώματος, για Nl στρώματα.

TFi -- η συνάρτηση ενεργοποίησης του ιth στρώματος (προεπιλογή είναι η συνάρτηση tansig).
BTF -- ο Back – Propagation αλγόριθμος εκπαίδευσης (προεπιλογή είναι η συνάρτηση traingdx).
BLF -- ο Back – Propagation weight/bias αλγόριθμος εκπαίδευσης (προεπιλογή είναι η συνάρτηση learngdm).
PF -- η συνάρτηση μέτρησης απόδοσης του δικτύου (προεπιλογή είναι η συνάρτηση mse).

Οι συναρτήσεις ενεργοποίησης TFi μπορεί να πάρει τις τιμές tansig, logsig ή η purelin. Η συνάρτηση εκπαίδευσης BTF μπορεί να πάρει τις τιμές trainlm, trainbfg, trainrp, traingd, κτλ. Η συνάρτηση BLF μπορεί να πάρει τις τιμές learngd ή learngdm.

11.2.2.4 RBF

Ένα δίκτυο RBF δημιουργείται με την συνάρτηση newrb.

```
net = newrb(P,T,goal,spread,MN, DF)
```

όπου:
P -- ένας πίνακας RxQ με Q διανύσματα εισόδου.
T -- ένας πίνακας SxQ με Q διανύσματα κλάσεων στόχων.
goal -- το μέσο τετραγωνικό σφάλμα (MSE) (προεπιλογή είναι η τιμή 0.0).
spread -- η διασπορά της συνάρτησης ακτινικού τύπου (προεπιλογή 1.0).
MN -- ο μέγιστος αριθμός νευρώνων (προεπιλογή είναι η τιμή Q).
DF -- ο αριθμός των νευρώνων που θα προσθέτει μεταξύ εμφανίσεων.

Η συγκεκριμένη συνάρτηση δημιουργεί δίκτυο RBF με αριθμό νευρώνων που πληρούν τις προϋποθέσεις που θέσαμε στις MN και DF. Υπάρχει ένα θέμα όσον αφορά την τιμή της spread(διασπορά). Όσο πιο μεγάλη είναι η διασπορά τόσο πιο ομαλή θα είναι η προσέγγιση της συνάρτησης. Αν εισάγουμε μεγάλη διασπορά τότε θα χρειαστεί μεγάλος αριθμός νευρώνων για να ταιριάξει σε μία ταχέως μεταβαλλόμενη συνάρτηση. Στην αντίθετη περίπτωση πολλοί νευρώνες θα χρειαστούν για να ταιριάξουν σε μία ομαλή συνάρτηση. Το πρόβλημα λύνεται με την δοκιμή της newrb με διάφορες τιμές διασποράς ώστε να βρούμε την καλύτερη τιμή για το συγκεκριμένο πρόβλημα.

11.2.2.5 SOM

Ένα δίκτυο SOM δημιουργείται με την συνάρτηση newsom.

```
net = newsom (PR,[D1,D2,...],TFCN,DFCN,OLR,OSTEPS,TLR,TND)
```

όπου:
PR -- ένας πίνακας Rx2 όπου περιέχει το ελάχιστο και μέγιστο για R εισόδους.
Di -- το μέγεθος του ιth στρώματος (προεπιλογή είναι το διάνυσμα [5 8]).
TFCN -- η τοπογραφική συνάρτηση (προεπιλογή είναι η hextop).
DFCN -- η συνάρτηση απόστασης (προεπιλογή είναι η linkdist).
OLR -- ο ρυθμός εκπαίδευσης της φάσης ταξινόμησης (προεπιλογή είναι η τιμή 0.9).
OSTEPS -- τα βήματα της φάσης ταξινόμησης (προεπιλογή είναι η τιμή 1000).
TLR -- ο ρυθμός εκπαίδευσης της φάσης εκπαίδευσης (προεπιλογή είναι η τιμή 0.02).
TND -- το μέγεθος της γειτονίας στην φάση εκπαίδευσης(προεπιλογή είναι η τιμή 1).

Η τοπογραφική συνάρτηση μπορεί να είναι η hextop, gritop ή η randtop. Η συνάρτηση απόστασης μπορεί να είναι η linkdist, dist ή η mandist.

11.2.3 Εκπαίδευση Δικτύου

(Help: Neural Network Toolbox->Functions – Categorical List->Network Use Functions)

Η εκπαίδευση ενός δικτύου γίνεται με την συνάρτηση train. Η train εκπαιδεύει το δίκτυο ανάλογα με την συνάρτηση εκπαίδευσης που είναι δηλωμένη στο struct του δικτύου δηλαδή το net.trainFcn και σύμφωνα με τις παραμέτρους εκπαίδευσης που είναι δηλωμένες στο net.trainParam. Όταν τελειώσει η εκπαίδευση εμφανίζεται ένα figure με την γραφική παράσταση του σφάλματος σε σχέση με την εποχή εκπαίδευσης.

```
[net,tr,Y,E,Pf,Af] = train(net,P,T,Pi,Ai,VV,TV)
```

όπου:
net -- το νευρωνικό δίκτυο
P – οι είσοδοι του δικτύου
T – οι στόχοι του δικτύου, (προεπιλογή zeros)
Pi -- αρχικές χρονικές καθυστερήσεις των εισόδων (προεπιλογή zeros).
Ai -- αρχικές χρονικές καθυστερήσεις των στρωμάτων (προεπιλογή zeros).
VV -- Structure of validation vectors, default = [ ]
TV -- Structure of test vectors, default = [ ]

και επιστρέφει:
net -- το νέο δίκτυο.
TR -- πληροφορίες της εκπαίδευσης (εποχές και απόδοση).
Y -- οι έξοδοι του δικτύου.
E -- τα σφάλματα του δικτύου.
Pf -- οι τελικές χρονικές καθυστερήσεις των εισόδων.
Af -- οι τελικές χρονικές καθυστερήσεις των στρωμάτων.

11.2.4 Ανάκληση Δικτύου

(Help: Neural Network Toolbox->Functions – Categorical List->Network Use Functions)

Η ανάκληση ενός νευρωνικού δικτύου γίνεται με την συνάρτηση sim.

```
[Y,Pf,Af,E,perf] = sim(net,P,Pi,Ai,T)
```

όπου:
net -- το νευρωνικό δίκτυο.
P -- οι είσοδοι του δικτύου.
Pi -- περιέχει τις αρχικές χρονικές καθυστερήσεις των εισόδων (προεπιλογή zeros).
Ai -- περιέχει τις αρχικές χρονικές καθυστερήσεις των στρωμάτων (προεπιλογή zeros).
T -- οι στόχοι του δικτύου (προεπιλογή zeros).

και επιστρέφει:
Y -- οι έξοδοι του δικτύου.
Pf -- οι τελικές χρονικές καθυστερήσεις των εισόδων.
Af -- οι τελικές χρονικές καθυστερήσεις των στρωμάτων.
E -- τα σφάλματα του δικτύου.

perf -- η απόδοση του δικτύου.

11.3 Neural Network Toolkit Plots

(Help: Neural Network Toolbox->Functions – Categorical List->Plotting Functions)

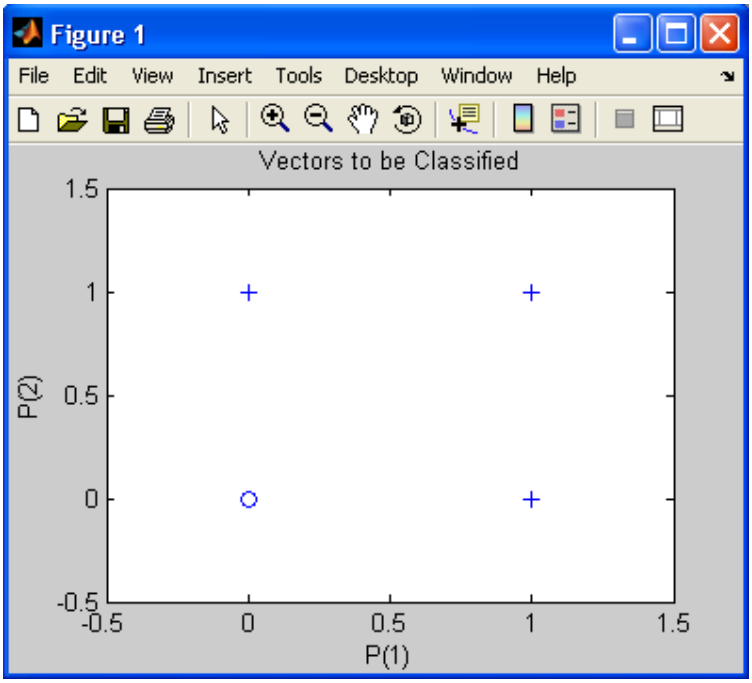
Το NNToolKit προσφέρει έτοιμα plots ειδικά για νευρωνικά δίκτυα.

Plotpv

Το plotpv σχεδιάζει τα πρότυπα εισόδου πάνω σε ένα επίπεδο. Στην Εικόνα 5.1 παρουσιάζονται τα δεδομένα που χρησιμοποιήσαμε στο παράδειγμα του δικτύου perceptron. Παρακάτω παρουσιάζεται η σύνταξη της plotpv.

```
plotpv(P,T);
```

όπου:
P – RxQ πίνακας με τα δεδομένα εισόδου (το R πρέπει να είναι το πολύ 3).
T – SxQ πίνακας με τους στόχους (το S πρέπει να είναι το πολύ 3).



Εικόνα 5.37

Plotpc

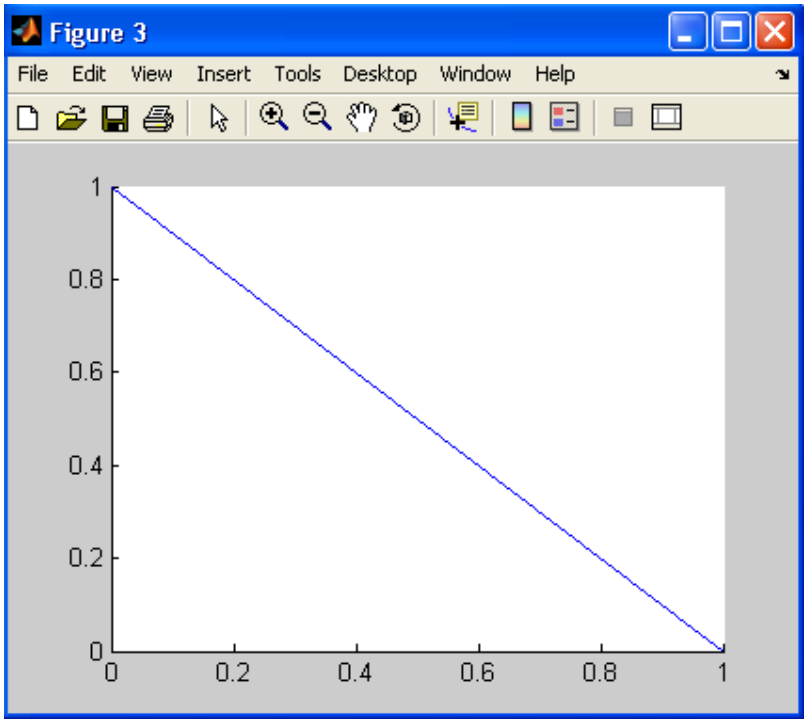
Το plotpc σχεδιάζει την γραμμή διαχωρισμού των προτύπων σε ένα δίκτυο perceptron. Στην Εικόνα 5.2 φαίνεται η γραμμή διαχωρισμού του παραδείγματος του perceptron. Παρακάτω παρουσιάζεται η σύνταξη της plotpc.

```
plotpc(W,B);
```

όπου:
W -- SxR πίνακας με τα βάρη (το R πρέπει να είναι το πολύ 3).
B -- Sx1 πίνακας με την πόλωση των νευρώνων.

Η παρακάτω εντολή δημιουργεί το plot της Εικόνας 5.2.

```
plotpc(net.iw{1,1},net.b{1});
```



Εικόνα 5.38

Ίσως θα ήταν πιο χρήσιμο σε αυτό το figure να εμφανίζονται και τα πρότυπα. Αυτό μπορεί να γίνει με τον παρακάτω κώδικα.

```
plotpv(P,T);
plotpc(net.iw{1,1},net.b{1});
```

Plotperf

Η plotperf σχεδιάζει την απόδοση του δικτύου. Παρακάτω παρουσιάζεται η σύνταξη της plotperf.

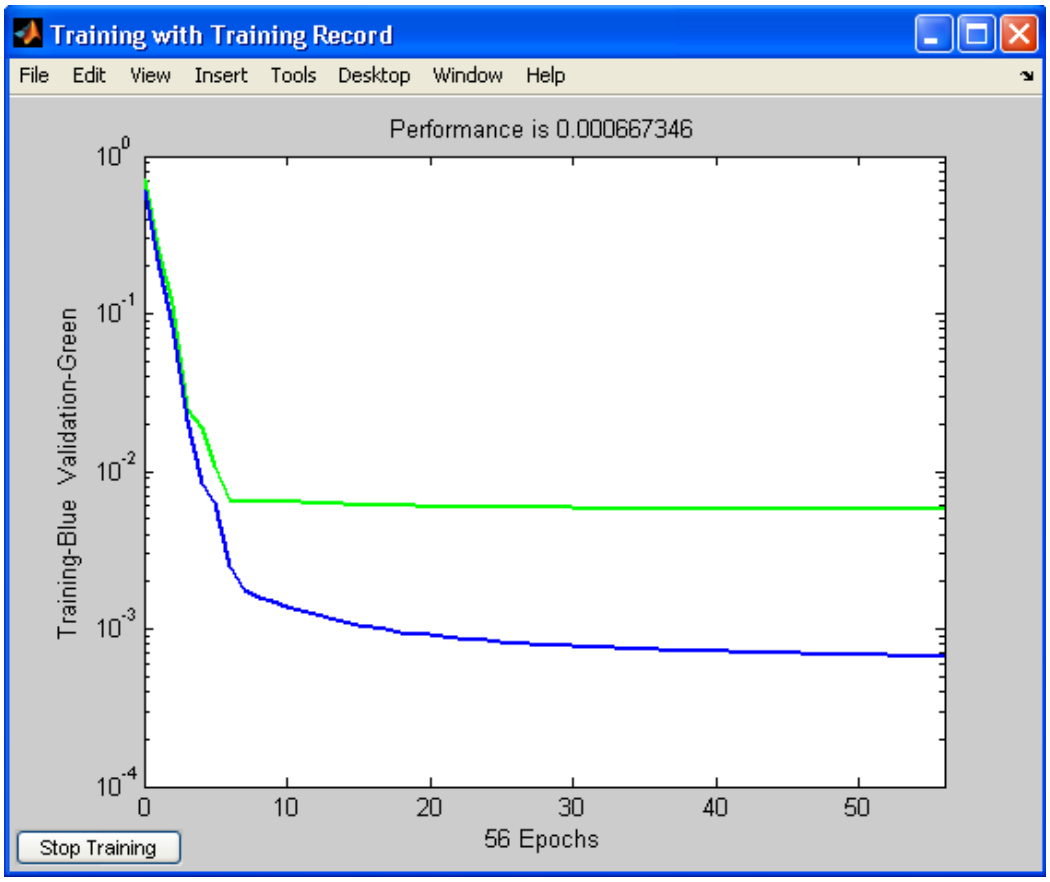
```
plotperf(TR,goal,name,epoch)
```

όπου:
TR -- Η δομή της εκπαίδευση που επιστρέφει η train.
goal -- Στόχος απόδοσης (προεπιλογή NaN).
name -- Όνομα συνάρτησης εκπαίδευσης (προεπιλογή ' ').
epoch -- Αριθμός εποχών, (προεπιλογή μήκος της δομής εκπαίδευσης).

Η παρακάτω εντολή δημιουργεί το plot της Εικόνας 5.3

Έστω η δομή tr περιέχει τις πληροφορίες μίας εκπαίδευσης ενός δικτύου.

```
plotperf(tr);
```



Εικόνα 5.39

Plotsom

Το plotsom σχεδιάζει τον τοπογραφικό χάρτη ενός δικτύου SOM. Παρακάτω παρουσιάζεται η σύνταξη της plotsom.

```
plotsom(pos);
```

όπου:
pos -- NxS πίνακας με S N-διαστατές συντεταγμένες θέσεων νευρώνων.

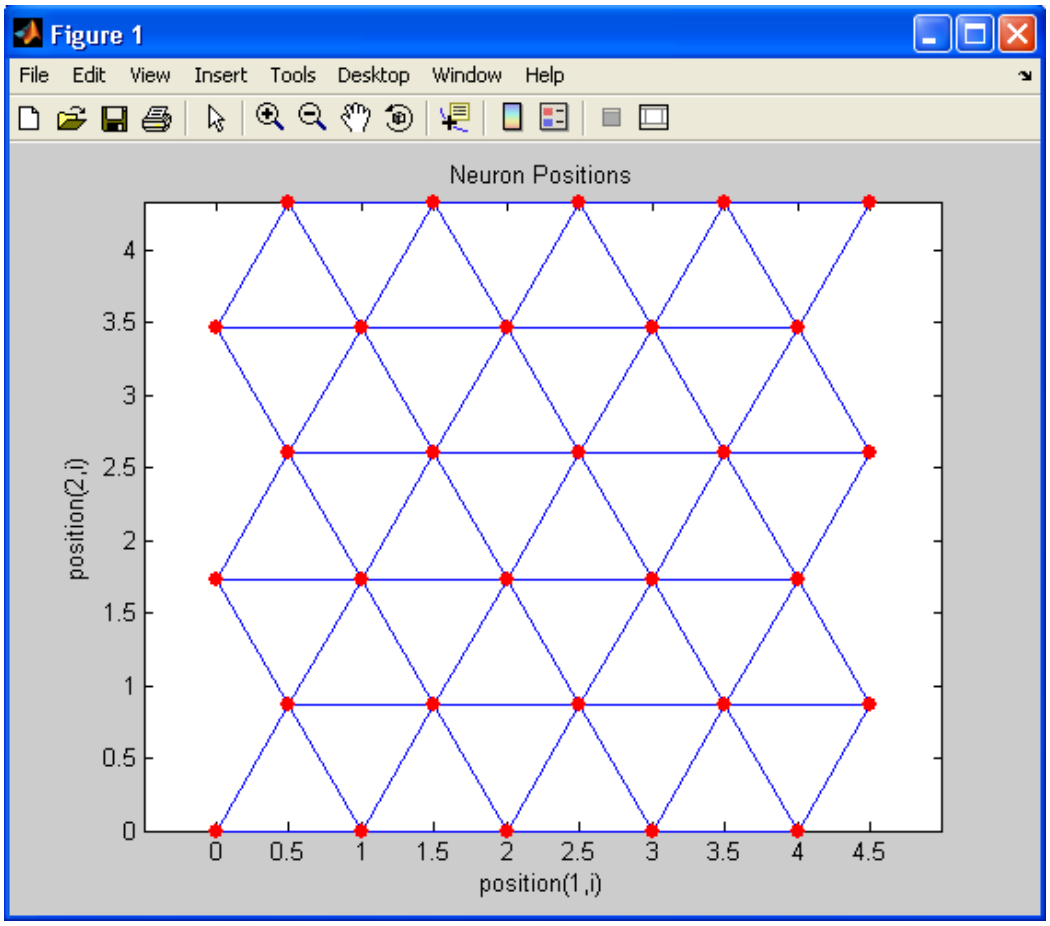
Η παρακάτω εντολή δημιουργεί το plot της Εικόνας 5.4. Σχεδιάζει τις θέσεις των νευρώνων με κόκκινες βούλες και ενώνει του νευρώνες με απόσταση 1.

```
pos = hextop(5,6);
plotsom(pos);
```

Επίσης υπάρχει και μία εναλλακτική μορφή σύνταξης της plotsom.

```
plotsom(W,D,ND);
```

όπου:
W -- SxR πίνακας βαρών.
D -- SxS distance matrix πίνακας αποστάσεων.
ND -- Απόσταση γειτονιάς (προεπιλογή 1).



Εικόνα 5.40

11.4 Παράδειγμα δημιουργίας δικτύου με χρήση κώδικα

Έστω ότι θέλουμε να δημιουργήσουμε ένα Multi-layer Perceptron δίκτυο. Το δίκτυο θα έχει 2 εισόδους, 1 κρυφά στρώμα και 1 στρώμα εξόδου. Παρακάτω παρουσιάζεται η σύνταξη της newff η οποία δημιουργεί το δίκτυο MLP.

net = newff(PR,[S1 S2...SNl],{TF1 TF2...TFNl},BTF,BLF,PF)

όπου:
PR -- ένας πίνακας Rx2 όπου περιέχει το ελάχιστο και μέγιστο για R εισόδους.
Si -- το μέγεθος του ι^{th} στρώματος, για Nl στρώματα.
TFi -- η συνάρτηση ενεργοποίησης του ι^{th} στρώματος (προεπιλογή είναι η συνάρτηση tansig).
BTF -- ο Back – Propagation αλγόριθμος εκπαίδευσης (προεπιλογή είναι η συνάρτηση traingdx).
BLF -- ο Back – Propagation weight/bias αλγόριθμος εκπαίδευσης (προεπιλογή είναι η συνάρτηση learngdm).
PF -- η συνάρτηση μέτρησης απόδοσης του δικτύου (προεπιλογή είναι η συνάρτηση mse).

Θα πρέπει να δημιουργήσουμε όλες τις παραμέτρους της newff. Ο παρακάτω κώδικας κάνει αυτή τη δουλειά, επίσης δημιουργεί και τα πρότυπα και τους στόχους.

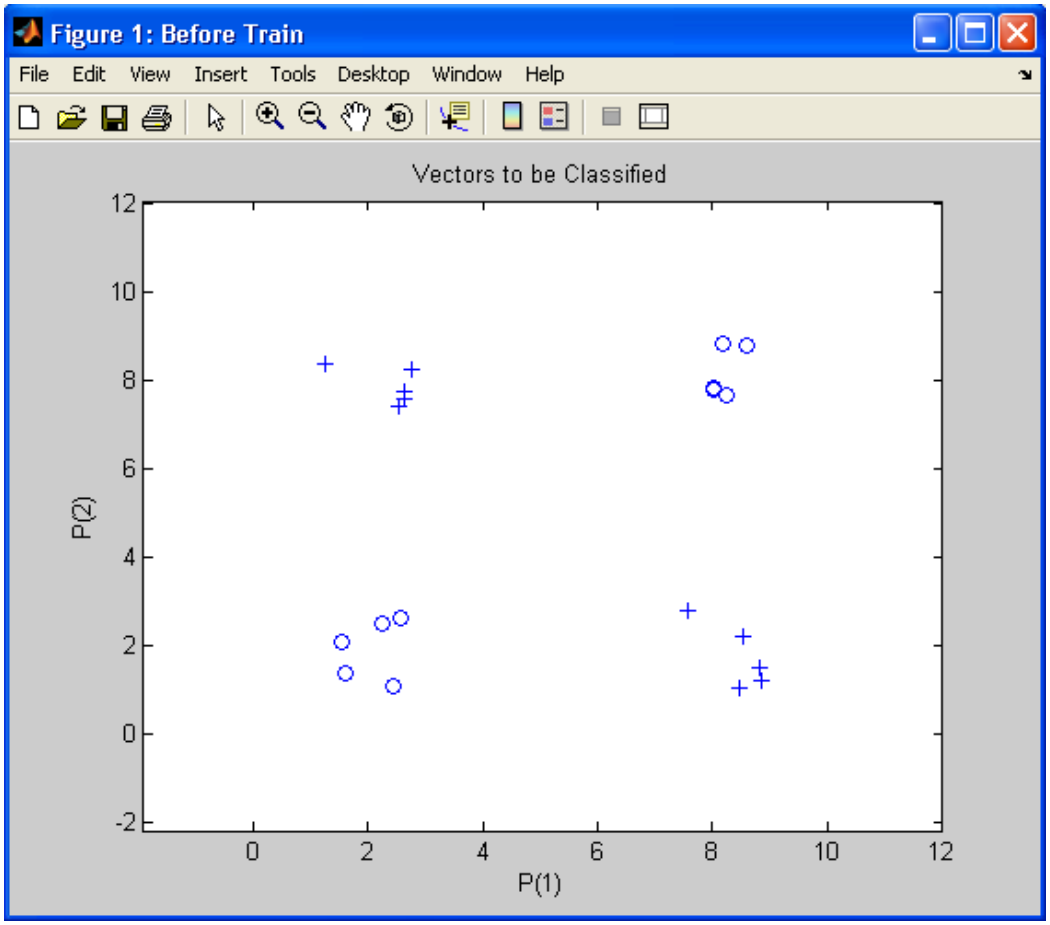
```
%Create inputs
```

```
n=20;                                   %Number of patterns
pats(1:2, 1:n/4) = 2*rand(2,n/4)+1;     %pattern class A
pats(1:2, n/4+1:n/2) = 2*rand(2,n/4)+7; %pattern class A
pats(1, n/2+1:3*n/4) = 2*rand(1,n/4)+7; %pattern class B
pats(2, n/2+1:3*n/4) = 2*rand(1,n/4)+1; %pattern class B
pats(1, 3*n/4+1:n) = 2*rand(1,n/4)+1;   %pattern class B
pats(2, 3*n/4+1:n) = 2*rand(1,n/4)+7;   %pattern class B

%Create targets

d(1,1:n/2) = zeros(1,n/2);              %targets class A
d(1,n/2+1:n) = ones(1,n/2);             %targets class B

%PR matrix contains the mins and maxs of the inputs
for i=1:2
   PR(i,1) = min(pats(i,:));
   PR(i,2) = max(pats(i,:));
end

%Number of neuron of each layer
S = [2 3 1];

%Training function of each layer
TF = {'tansig' 'tansig' 'tansig'};

%Back-propagation training function
BTF = 'traingdx';

%Backpropagation weight/bias learning function
BLF = 'learngdm';

%Performance function
PF = 'mse';

%Create the network
net = newff(PR,S,TF,BTF,BLF,PF);
```

Σε αυτό το σημείο μπορούμε να χρησιμοποιήσουμε ένα plot από το Neural Network Toolkit για να αναπαραστήσουμε τα πρότυπα στο επίπεδο. Ο παρακάτω κώδικας δημιουργεί το plotpv σε ένα figure. Το αποτέλεσμά του φαίνεται στην Εικόνα 5.5.

```
figure('name','Before Train');
%Draw patterns
plotpv(pats,d);
```

Το επόμενο βήμα είναι η εκπαίδευση του δικτύου.



Εικόνα 5.41

Πριν καλέσουμε την συνάρτηση εκπαίδευσης θα πρέπει να θέσουμε τον μέγιστο αριθμό εποχών και το ρυθμό εκπαίδευσης. Ο παρακάτω κώδικας κάνει αυτή τη δουλειά και καλεί την συνάρτηση εκπαίδευσης.

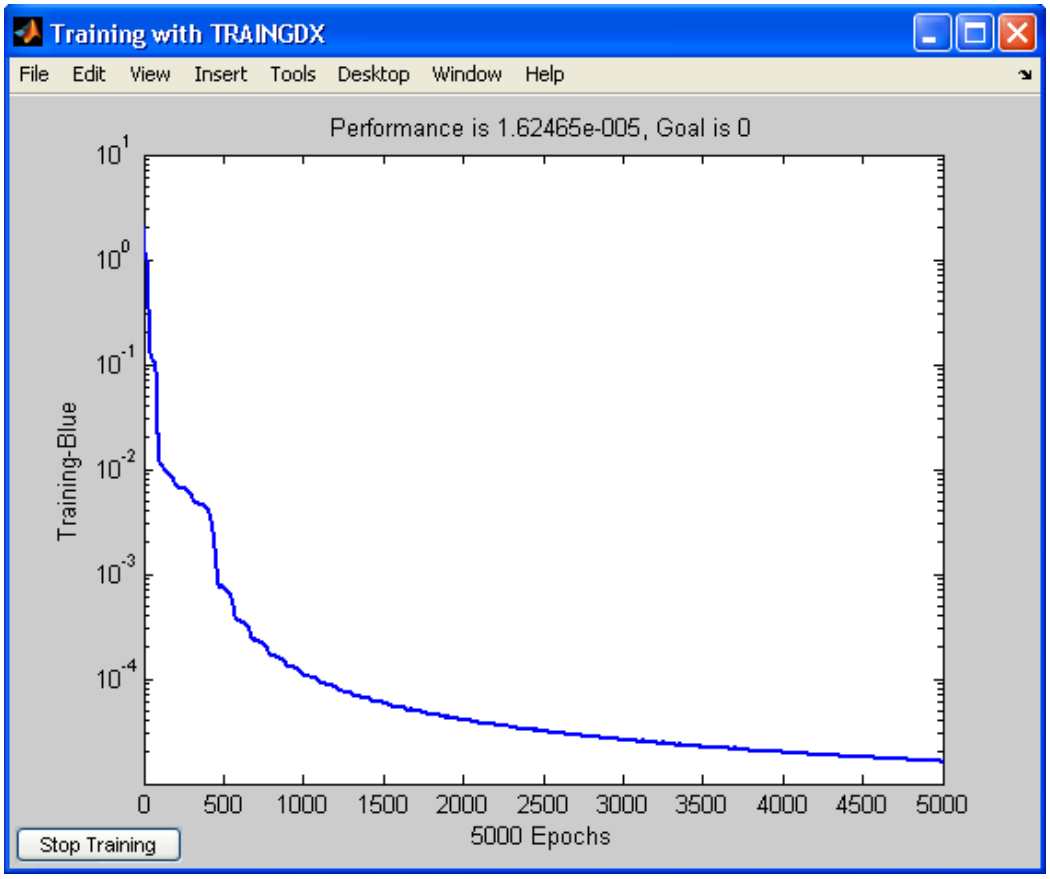
```
%train the network
net.trainParam.epochs = 10000;
net.trainParam.lr = 0.001;
[net,tr] = train(net,pats,d);
```

Όταν γίνει η κλήση της train τότε εμφανίζεται ένα figure με το σφάλμα προς την εποχή εκπαίδευσης (Εικόνα 5.6). Επίσης εμφανίζεται και η ίδια πληροφορία και στο command window του Matlab. Παρατηρούμε ότι η εκπαίδευση τελειώνει όταν φτάσει η συνάρτηση εκπαίδευσης στο μέγιστο αριθμό εποχών.

Τέλος μπορούμε να κάνουμε ανάκληση το δίκτυο με τα δεδομένα εισόδου και να συγκρίνουμε τις εξόδους με τους στόχους ώστε να δούμε το ποσοστό επιτυχίας. Ο παρακάτω κώδικας κάνει ακριβώς αυτή τη δουλειά.

```
[Y] = sim(net,pats);

correct = 0;
for i = 1:n
    if Y(i) >= 0.5
```

```
        y=1;
    else
        y=0
    end
    if y==d(i)    %check if current output is equal to equivalent target
        correct=correct+1;
    end
end
percent = (correct/n) * 100
```



Εικόνα 5.42

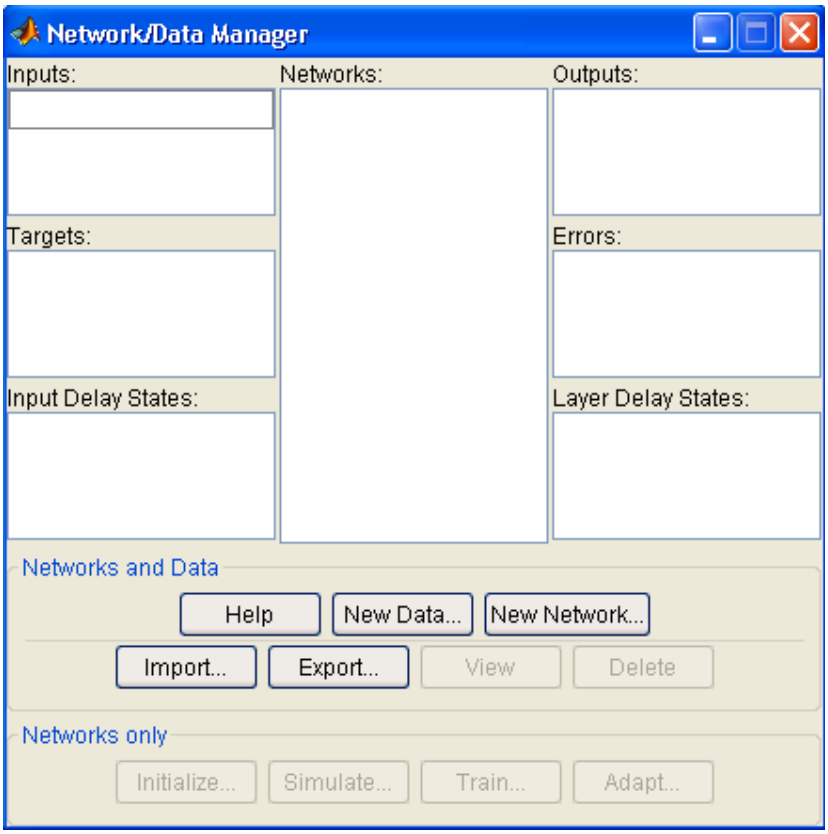
11.5 GUI Interface του Neural Network Toolkit

11.5.1 Παρουσίαση Network/Data Manager

Υπάρχει η δυνατότητα να δημιουργήσουμε και να χρησιμοποιήσουμε νευρωνικά δίκτυα με την βοήθεια ενός γραφικού περιβάλλοντος. Για να ξεκινήσουμε το GUI του NNToolKit γράφουμε στο command window του Matlab την εντολή

```
>> nntool
```

Αμέσως εμφανίζεται το κεντρικό παράθυρο του toolkit (‘Network/Data Manager’) όπως φαίνεται στην Εικόνα 5.8.



Εικόνα 5.43

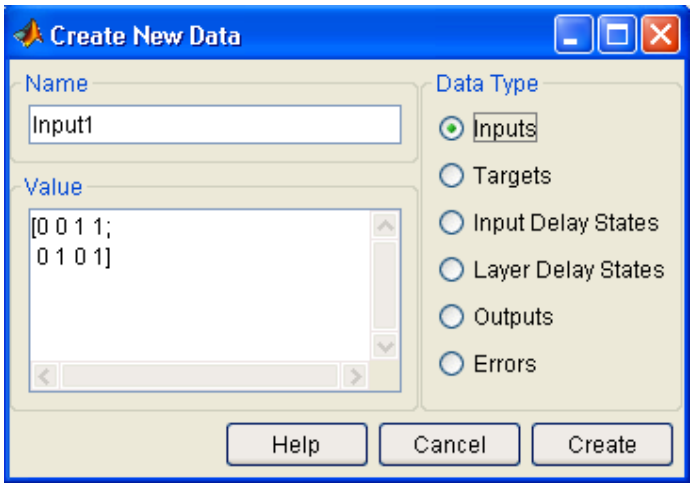
Όπως βλέπουμε υπάρχουν λειτουργίες που μας επιτρέπουν να δημιουργήσουμε δίκτυα και δεδομένα. Με το κουμπί 'New Network' προφανώς μπορούμε να δημιουργήσουμε ένα δίκτυο. Με τα κουμπιά 'New Data' και 'Import' μπορούμε να δημιουργήσουμε ή να εισάγουμε αντίστοιχα δεδομένα εκπαίδευσης ή ανάκλησης για το δίκτυο ή δίκτυα που μπορεί να έχουμε δημιουργήσει.

11.5.2 Παραδείγματα δημιουργίας δικτύων με το GUI Interface

Δημιουργία δικτύου Perceptron

Εισαγωγή Δεδομένων Εισόδου

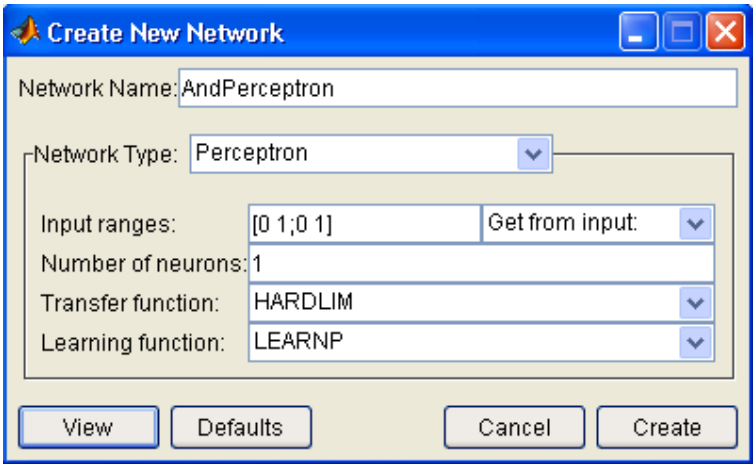
Το πρώτο βήμα είναι να δημιουργήσουμε τα δεδομένα εισόδου του δικτύου μας. Πατάμε το κουμπί 'New Data' και εμφανίζεται το παρακάτω dialog(Εικόνα 5.9). Τα δεδομένα που έχουμε εισάγει στο παρακάτω παράδειγμα είναι για ένα δίκτυο δύο εισόδων με τέσσερα σετ τέτοιων εισόδων. Άρα κάθε στήλη του πίνακα δεδομένων αντιστοιχεί σε ένα σετ εισόδου για το δίκτυο. Επίσης έχουμε επιλέξει και ως Data Type inputs.



Εικόνα 5.44

Δημιουργία Δικτύου

Πατώντας το κουμπί 'New Network' εμφανίζεται το παρακάτω dialog(Εικόνα 5.10) όπου μπορούμε να επιλέξουμε το δίκτυο που θέλουμε να δημιουργήσουμε. Ανάλογα με το δίκτυο που επιλέγουμε να δημιουργήσουμε το dialog αλλάζει μορφή και εμφανίζει ιδιότητες που αφορούν το ανάλογο δίκτυο. Στην Εικόνα 5.10 έχουμε επιλέξει δίκτυο Perceptron. Με τα δεδομένα εισόδου που δημιουργήσαμε μπορούμε να δημιουργήσουμε ένα δίκτυο Perceptron που θα υλοποιεί μία λογική πύλη AND. Γράφουμε το όνομα του δικτύου και για input ranges διαλέγουμε από το pop-up menu τα δεδομένα 'Input1' που δημιουργήσαμε πριν. Τον αριθμό νευρώνων και τις Transfer και Learning function τις αφήνουμε όπως έχουν. Η Transfer function 'HARDLIM' είναι η βηματική συνάρτηση. Στο help του Matlab μπορείτε να βρείτε αναλυτικές πληροφορίες για τις συγκεκριμένες functions.



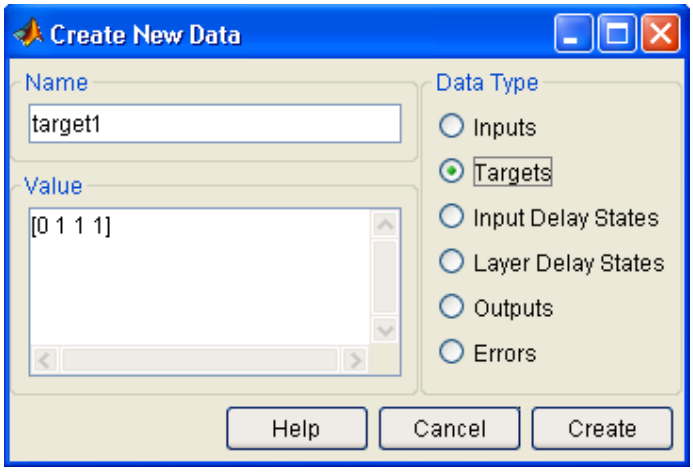
Εικόνα 5.45

Πατάμε 'Create' και το δίκτυό μας έχει δημιουργηθεί.

Δημιουργία Στόχων

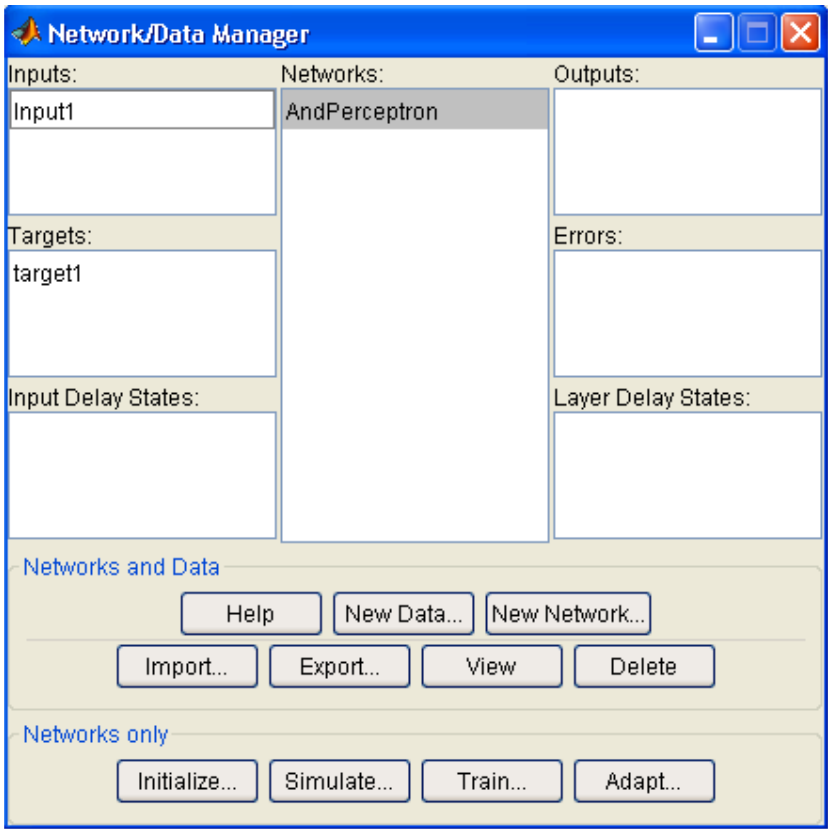
Αυτό που χρειαζόμαστε τώρα για να εκπαιδεύσουμε το δίκτυό μας είναι να δημιουργήσουμε δεδομένα με στόχους. Πατάμε πάλι το κουμπί 'New Data'. Το dialog με συμπληρωμένα τα

δεδομένα στόχων φαίνεται στην Εικόνα 5.11. Ο πίνακας των στόχων έχει μία σειρά και τέσσερις στήλες. Πατάμε 'Create' για να δημιουργηθούν τα δεδομένα.



Εικόνα 5.46

Πλέον το κύριο dialog του NNToolKit με τα δεδομένα μας και το δίκτυο μας φαίνεται στην Εικόνα 5.12.

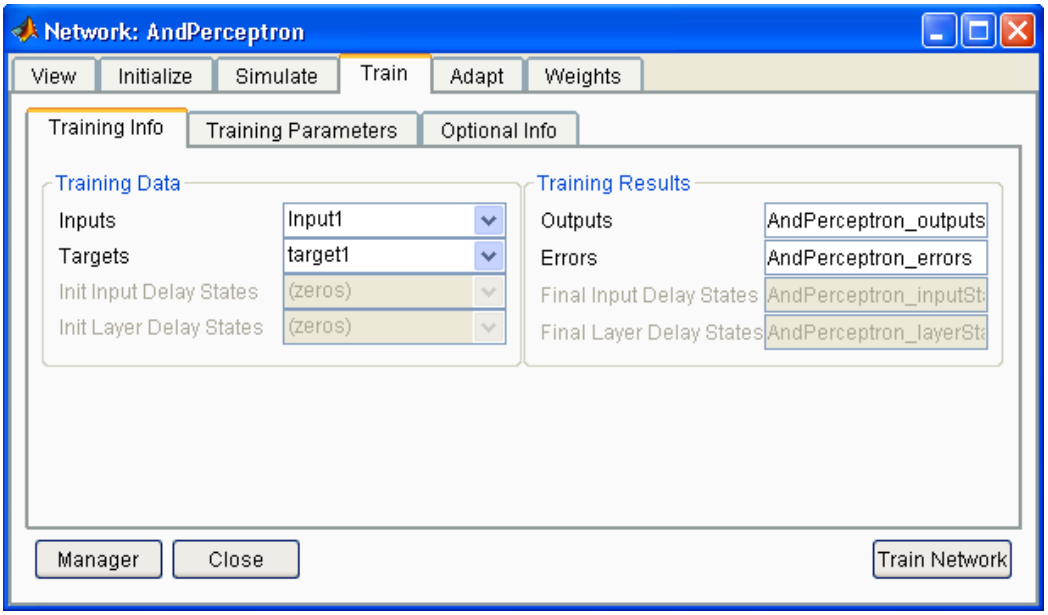


Εικόνα 5.47

Εκπαίδευση δικτύου

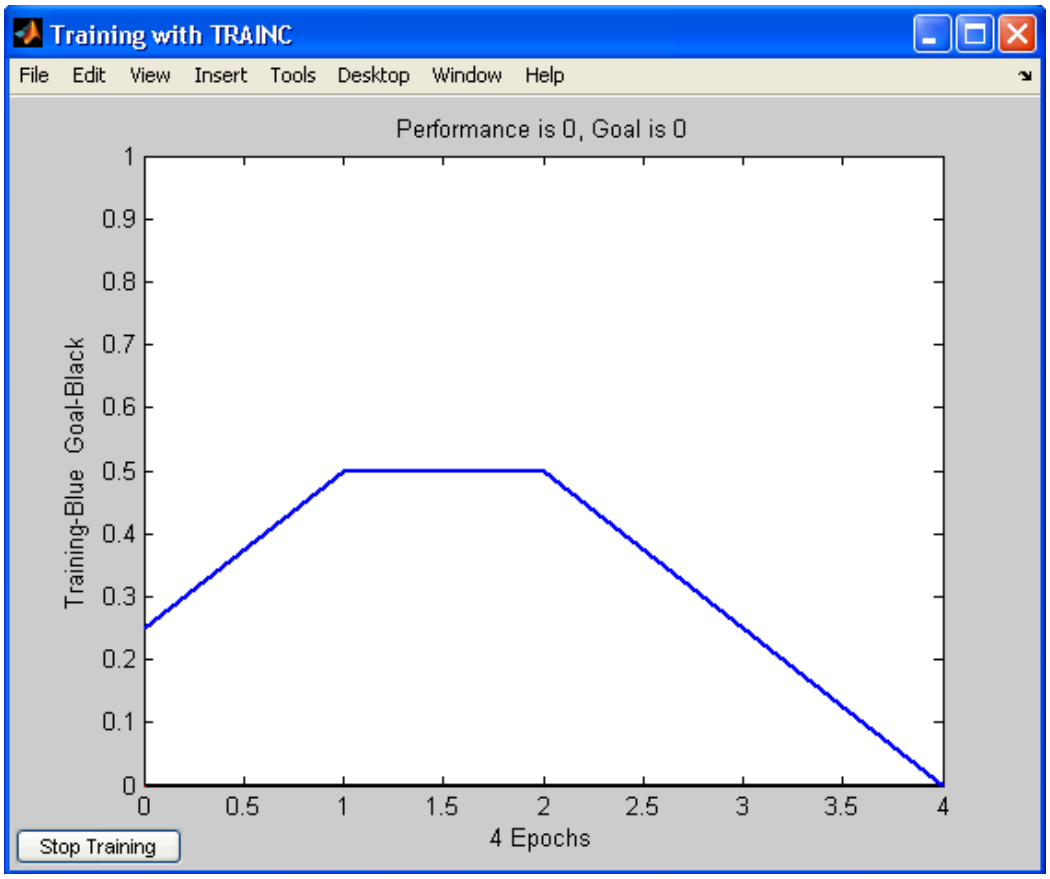
Το επόμενο βήμα είναι η εκπαίδευση του δικτύου μας. Επιλέγουμε το δίκτυο 'ANDPerceptron' και πατάμε το κουμπί 'Train'. Αμέσως εμφανίζεται το dialog με τις

πληροφορίες του δικτύου. Διαλέγουμε την καρτέλα ‘Train’και μετά την καρτέλα ‘Training Info’(Εικόνα 5.13).



Εικόνα 5.48

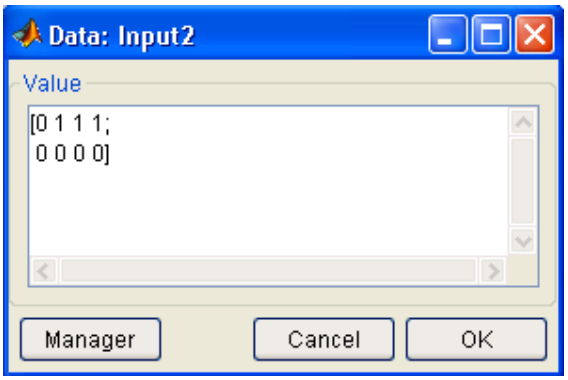
Έχουμε επιλέξει για δεδομένα εκπαίδευσης τα δεδομένα που είχαμε δημιουργήσει. Ως δεδομένα εισόδου το Input1 και ως δεδομένα στόχων το target1. Στην καρτέλα Training Parameters μπορούμε να επιλέξουμε πόσες εποχές και για πόσο χρόνο θα γίνεται η εκπαίδευση του δικτύου. Πλέον μπορούμε να εκπαιδεύσουμε το δίκτυό μας. Πατάμε το κουμπί ‘Train Network’. Όταν τελειώσει η εκπαίδευση εμφανίζεται ένα διάγραμμα με το μέσο τετραγωνικό σφάλμα ανά εποχή εκπαίδευσης(Εικόνα 5.14). Η εκπαίδευση του δικτύου όπως φαίνεται από την γραφική παράσταση τελείωσε σε τέσσερις εποχές. Επίσης επιστρέφονται και δύο πίνακες με επιπλέον πληροφορίες για την πορεία της εκπαίδευσης. Πλέον το δίκτυο μας έχει εκπαιδευτεί και είμαστε έτοιμοι να το κάνουμε ανάκληση ή να το εκπαιδεύσουμε πάλι με επιπλέον δεδομένα.



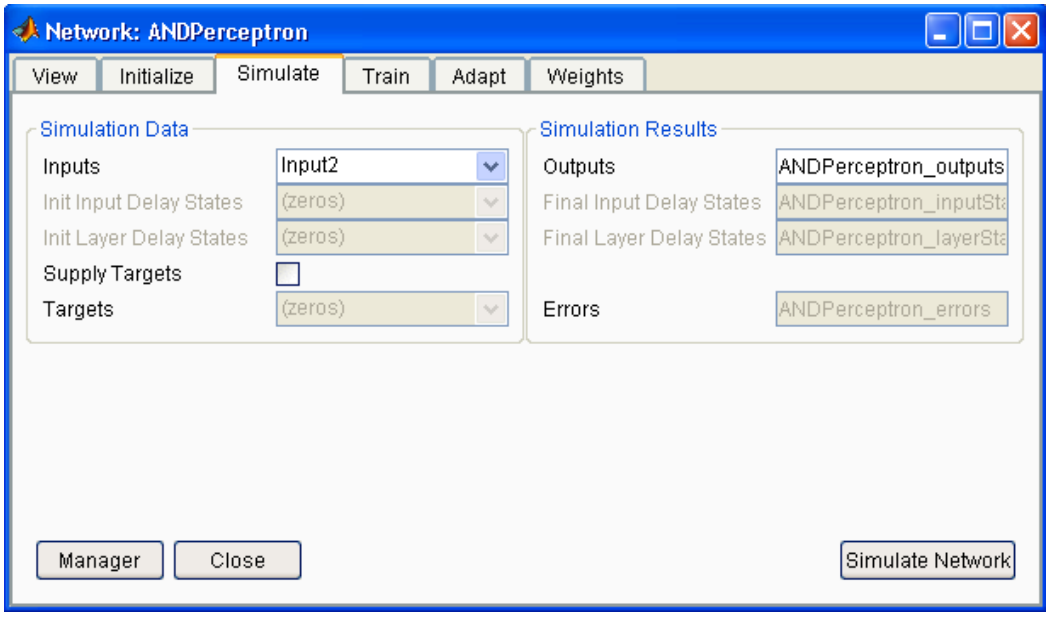
Εικόνα 5.49

Ανάκληση Δικτύου

Πατάμε το κουμπί 'New Data' για να δημιουργήσουμε νέα δεδομένα εισόδου. Αυτά τα δεδομένα θα τα χρησιμοποιήσουμε για την ανάκληση του δικτύου μας. Π.χ. δημιουργούμε τα δεδομένα της Εικόνας 5.15. Τα δεδομένα εισόδου είναι έτοιμα και μπορούμε να προχωρήσουμε στην ανάκληση. Επιλέγουμε το δίκτυο 'ANDPerceptron' και πατάμε το κουμπί 'Simulate'. Στην καρτέλα 'Simulate' (Εικόνα 5.16) επιλέγουμε σαν inputs τα δεδομένα 'inputs2' που δημιουργήσαμε πριν. Πατάμε το κουμπί 'Simulate Network'. Έχει γίνει η ανάκληση του δικτύου και οι έξοδοι έχουν αποθηκευθεί στον πίνακα 'ANDPerceptron_outputs'. Κλείνουμε το παράθυρο της ανάκλησης και επιστρέφουμε στο 'Network/Data Manager' επιλέγουμε τον πίνακα 'ANDPerceptron_outputs" και πατάμε 'View' και εμφανίζονται οι έξοδοι του δικτύου για κάθε σετ εισόδων που περιείχαν τα δεδομένα 'input2'.



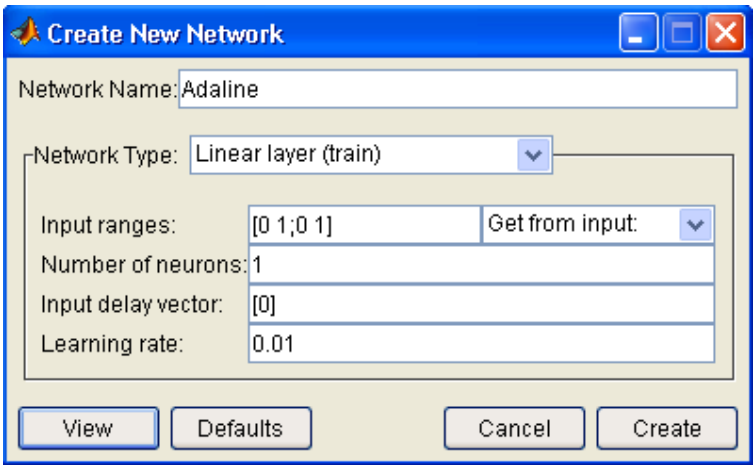
Εικόνα 5.50



Εικόνα 5.51

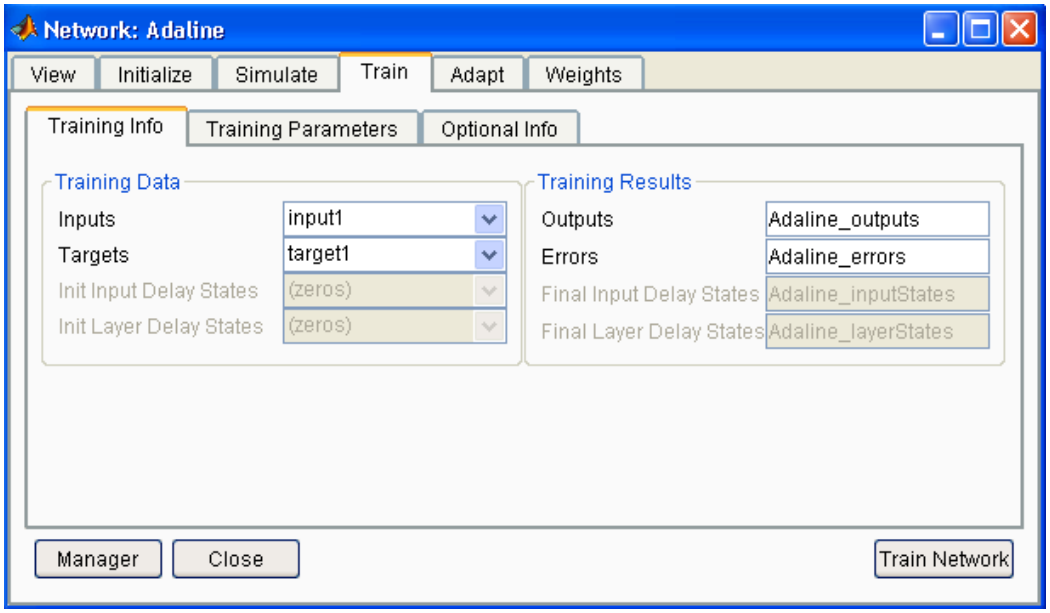
Δημιουργία δικτύου Adaline

Για την δημιουργία ενός δικτύου adaline δημιουργούμε τα ίδια δεδομένα με το δίκτυο perceptron και στην δημιουργία του δικτύου επιλέγουμε ‘Linear Layer(Train)’ όπως φαίνεται στην Εικόνα 5.17.



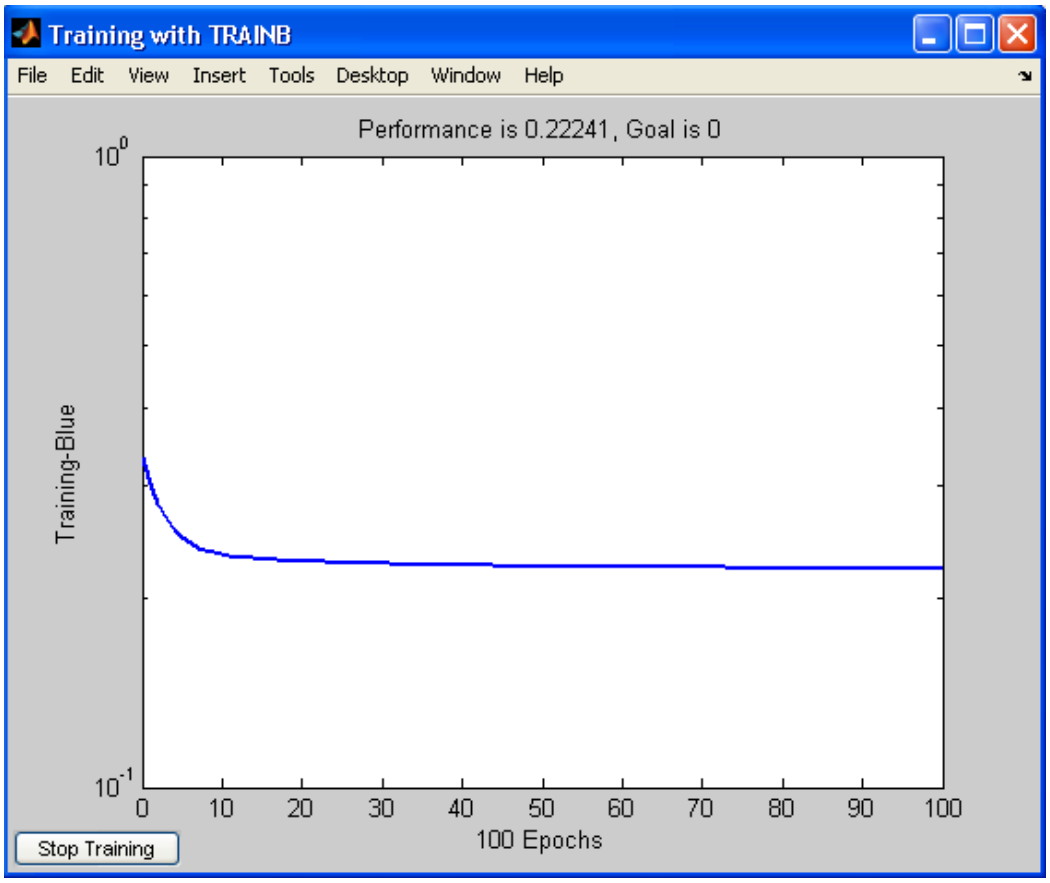
Εικόνα 5.52

Έπειτα μπορούμε να εκπαιδεύσουμε το δίκτυο. Όπως και στο perceptron στην καρτέλα 'Training info' επιλέγουμε τα δεδομένα εισόδου και των στόχων που δημιουργήσαμε(Εικόνα 5.18).



Εικόνα 5.53

Πατάμε το κουμπί 'Train Network'. Όταν τελειώσει η εκπαίδευση εμφανίζεται ένα διάγραμμα με το μέσο τετραγωνικό σφάλμα ανά εποχή εκπαίδευσης(Εικόνα 5.19). Η εκπαίδευση του δικτύου τελείωσε σε 100 εποχές. Όσες δηλαδή είχαμε δηλώσει στην καρτέλα Training Parameters.



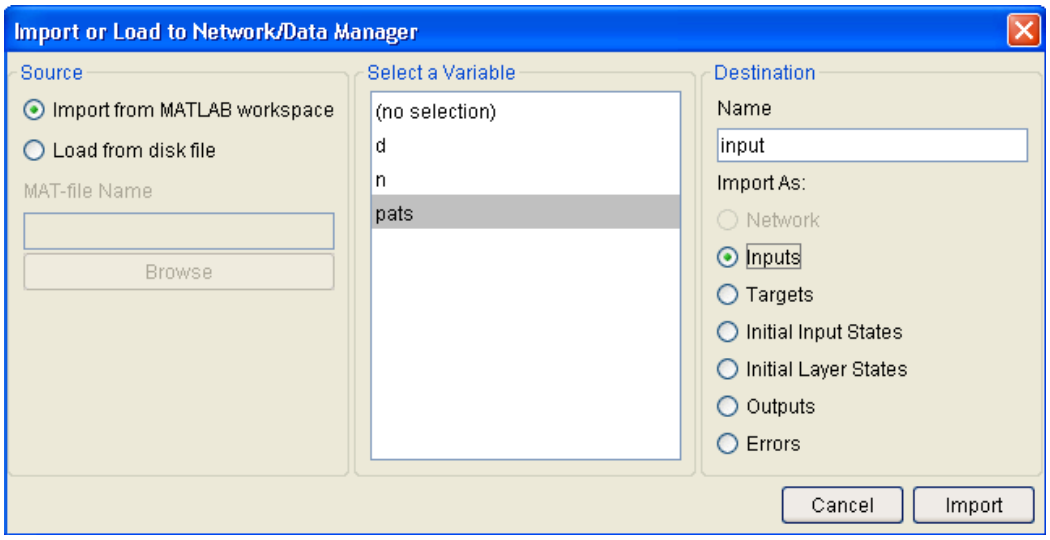
Εικόνα 5.54

Δημιουργία δικτύου Multi-Layer Perceptron

Για την εκπαίδευση του Multi-Layer Perceptron θα χρησιμοποιήσουμε μη γραμμικά διαχωρίσιμα δεδομένα. Τα δεδομένα θα τα δημιουργήσουμε στο workspace του Matlab και τα εισάγουμε στο δίκτυο μας από εκεί. Το δίκτυο θα περιέχει δύο εισόδους και ένα κρυφό στρώμα. Οι παρακάτω εντολές δημιουργούν τα δεδομένα εισόδου και εξόδου στο workspace. Θα πρέπει να δοθεί προσοχή στο ότι οι γραμμές του πίνακα εισόδων να είναι όσες οι είσοδοι του δικτύου και οι στήλες του πίνακα στόχων να είναι όσες και τα σετ εισόδου που έχει ο πίνακας εισόδων. Το αρχείο εισόδων θα περιλαμβάνει 20 πρότυπα για τρεις εισόδους.

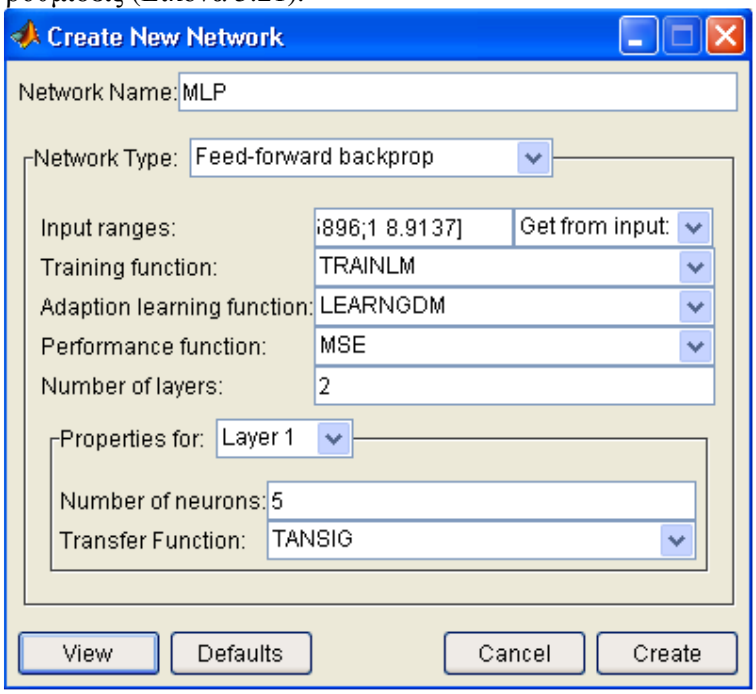
```
n=20;                                   %20 Protypa
pats(1:2,1:n/4) = 2*rand(2,n/4)+1;      %Protypa class A
pats(1:2,n/4+1:n/2) = 2*rand(2,n/4)+7;  %Protypa class A
pats(1,n/2+1:3*n/4) = 2*rand(1,n/4)+7;  %Protypa class B
pats(2,n/2+1:3*n/4) = 2*rand(1,n/4)+1;  %Protypa class B
pats(1,3*n/4+1:n) = 2*rand(1,n/4)+1;    %Protypa class B
pats(2,3*n/4+1:n) = 2*rand(1,n/4)+7;    %Protypa class B
pats(3,:) = 1;

d(1,1:n/2) = zeros(1,n/2);              %Stoxoi class A
d(1,n/2+1:n) = ones(1,n/2);             %Stoxoi class B
```

Για να εισάγουμε τα δεδομένα από το workspace στο 'Network/Data Manager' πατάμε το κουμπί 'Import'. Εμφανίζεται το dialog της Εικόνας 5.20. Παρατηρούμε ότι περιέχει όλες τις μεταβλητές του workspace που έχουμε δημιουργήσει. Επιλέγουμε την μεταβλητή 'pats' και στο 'Destination' γράφουμε σαν όνομα 'input' και επιλέγουμε 'Import As: Inputs'. Πατάμε το κουμπί 'Import' για να εισάγουμε τα δεδομένα στο 'Data Manager' του NNToolKit. Κάνουμε το ίδιο για να εισάγουμε και τα δεδομένα στόχων.



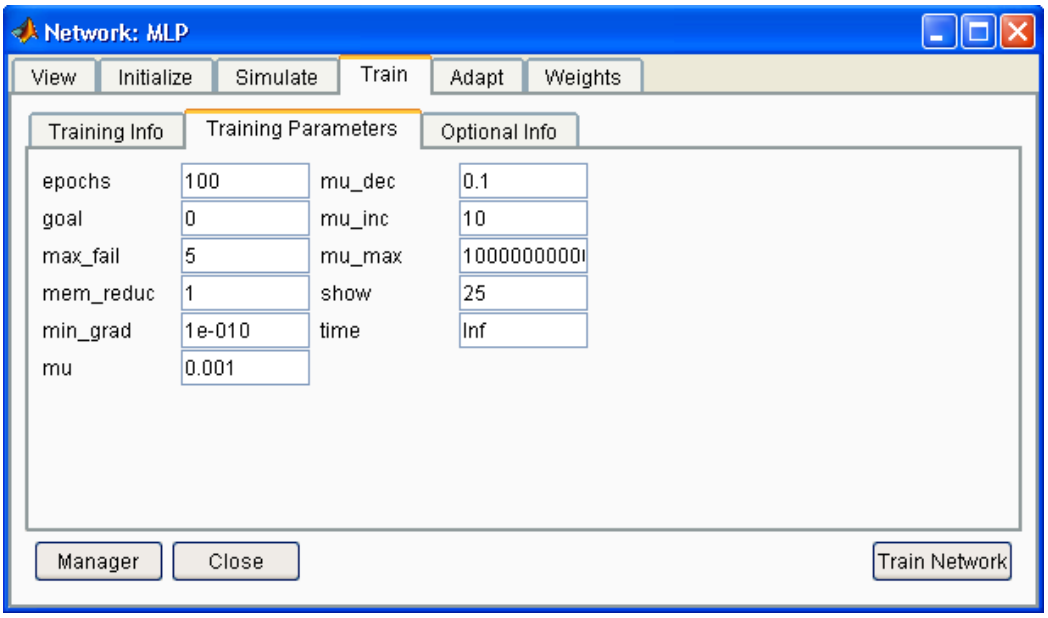
Εικόνα 5.55

Για την δημιουργία του δικτύου μας στο dialog 'Create New Network' επιλέγουμε 'Feed-Forward Network'. Τις 'input ranges' τις επιλέγουμε από τα δεδομένα που εισάγαμε από το workspace. Επιλέγουμε την συνάρτηση εκπαίδευσης όπως και την συνάρτηση απόδοσης. Μας δίνεται η δυνατότητα να επιλέξουμε τον αριθμό των στρωμάτων και την συνάρτηση ενεργοποίησης για τους νευρώνες του κάθε στρώματος. Στο παράδειγμά μας αφήνουμε τις προεπιλεγμένες ρυθμίσεις (Εικόνα 5.21).



Εικόνα 5.56

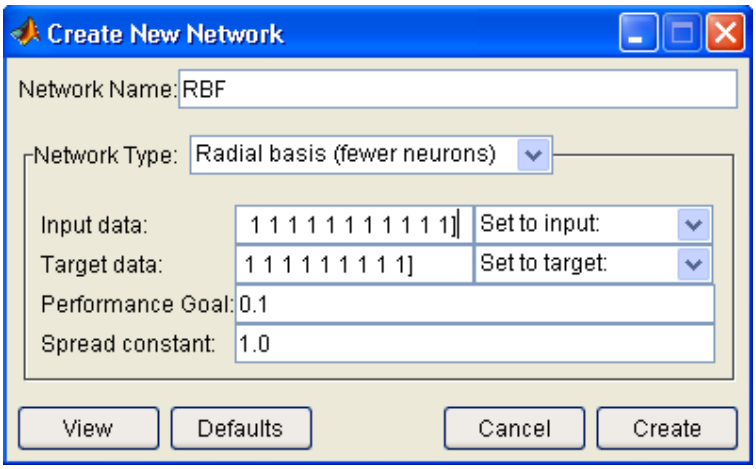
Πλέον μπορούμε να εκπαιδεύσουμε το δίκτυό μας. Η εκπαίδευση γίνεται όπως και στα προηγούμενα δίκτυα. Επιλέγουμε το δίκτυο και πατάμε στο κουμπί 'Train'. Στην καρτέλα 'Training Info' επιλέγουμε τα δεδομένα που εισάγαμε από το workspace. Στην καρτέλα 'Training Parameters' (Εικόνα 5.22) μπορούμε να αλλάξουμε τις διάφορες παραμέτρους εκπαίδευσης όπως το μέγιστο αριθμό εποχών, τον ελάχιστο σφάλμα κτλ.



Εικόνα 5.57

Δημιουργία δικτύου RBF

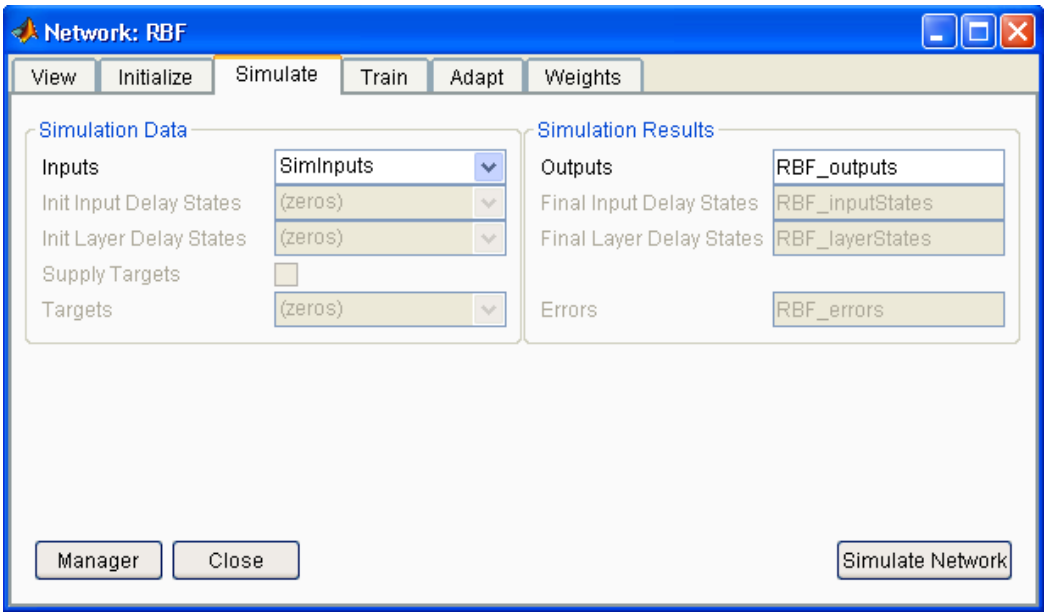
Για την εκπαίδευση του RBF θα χρησιμοποιήσουμε τα ίδια δεδομένα που χρησιμοποιήσαμε στο δίκτυο Multi-Layer Perceptron. Ακολουθούμε την ίδια διαδικασία για την εισαγωγή των δεδομένων. Για την δημιουργία του δικτύου μας στο dialog 'Create New Network' επιλέγουμε 'Radial basis(fewer neurons)'. Υπάρχει η δυνατότητα να επιλέξουμε 'Radial basis(exact fit)', αυτό δημιουργεί ένα δίκτυο RBF όπου κάθε πρότυπο είναι και ένα κέντρο. Εκτός από τα δεδομένα εισόδου και στόχων του δικτύου μπορούμε να επιλέξουμε το ελάχιστο σφάλμα και την διασπορά(Εικόνα 5.23).



Εικόνα 5.58

Με την δημιουργία του δικτύου έχει γίνει και η εκπαίδευσή του. Έχουν βρεθεί τα κέντρα και έχει εκπαιδευτεί το Adaline τμήμα του δικτύου.

Μπορούμε να κάνουμε ανάκληση στο δίκτυο πατώντας το κουμπί Simulate (Εικόνα 5.24).

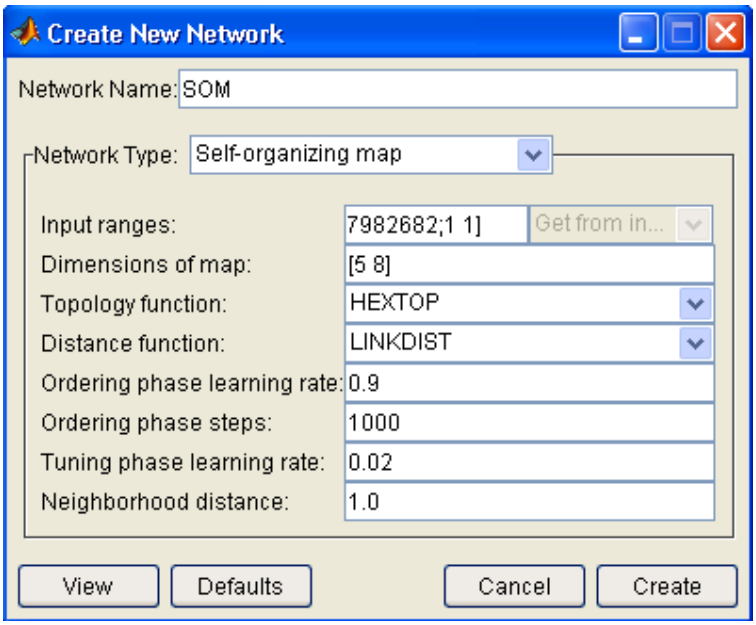


Εικόνα 5.59

Οι έξοδοι της ανάκλησης θα αποθηκευτούν στην δομή RBF_outputs.

Δημιουργία δικτύου SOM

Για την εκπαίδευση του SOM θα χρησιμοποιήσουμε τα ίδια δεδομένα που χρησιμοποιήσαμε στο δίκτυο Multi-Layer Perceptron. Για την δημιουργία του δικτύου μας στο dialog 'Create New Network' επιλέγουμε 'Self-organizing map' (Εικόνα 5.25).



Εικόνα 5.60

Η εκπαίδευση του δικτύου γίνεται στην καρτέλα Train όπως και στα άλλα δίκτυα. Επιλέγουμε τα δεδομένα εισόδου και στην καρτέλα Training Parameters τις παραμέτρους τις εκπαίδευσης. Η ανάκληση του δικτύου γίνεται στην καρτέλα Simulate. Όπως και στα άλλα δίκτυα τα αποτελέσματα τις ανάκληση αποθηκεύονται στην δομή SOM_outputs.