Εισαγωγή στους Η/Υ (MATLAB)

11η διάλεξη

MatLab II

Αναφορά σε στοιχεία πίνακα

Εάν δώσουμε ως δείκτη, αντί για αριθμό ένα διάνυσμα με δείκτες, τότε προκύπτει ένας νέος πίνακας που έχει μόνο τα συγκεκριμένα στοιχεία του πίνακα. Πχ.

```
z = 2:2:20
z([2 5 8])
```

ίδια διάσταση

επιστρέφει:

```
[4 10 16]
```

Σε κάθε περίπτωση ο αρχικός πίνακας δεν επιρρεάζεται
-
Μόνο γίνεται αναφορά σε κάποια στοιχεία του!

όμοια για να πάρω τα 4 πρώτα στοιχεία:

`z(1:4)` → [2 4 6 8]

και για να πάρω τα στοιχεία από το 7 ως το τέλος, χρησιμοποιώντας την ειδική λέξη **end**

`z(7:end)` → [14 16 18 20]

και για τα 3 τελευταία:

`z((end-2):end)` → [16 18 20]

Τέλος εάν δώσετε έναν πίνακα που έχει το πολύ διαστάσεις όσες ο πίνακας πχ z αλλά τα στοιχεία του είναι logical (true ή false – όχι 0 ή 1 αριθμητικά) τότε από τον πίνακα επιστρέφονται τα στοιχεία που αντιστοιχούν στο true. Πχ

`z([true true false true])` → [2 4 8]

Αναφορά σε στοιχεία πίνακα

Εάν δώσουμε ως δείκτη σε μονοδιάστατο πίνακα, έναν πίνακα 2 διαστάσεων, τότε προκύπτει πίνακας ίδιων διαστάσεων με αυτόν των δεικτών, όπου το κάθε στοιχείο αντιστοιχεί όμοια με το αν είχε δοθεί μόνο του. Άρα:

```
z = 2:2:20
z([1 2 3 4])   →   [2 4 6 8]
z([1 2; 3 4])  →   [2 4; 6 8]
```

και αντίστοιχα/αντίστροφα ένα δώσουμε μονοδιάστατο πίνακα σαν δείκτη σε πίνακα δύο διαστάσεων παίρνουμε:

$$a=\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{bmatrix}$$

```
a([1 2 3]) = [1 4 7]
```

Εάν για δείκτη δώσουμε 1:end πχ στον z, παίρνουμε τον ίδιο τον πίνακα ως μονοδιάστατο:

```
z(1:end)   →   [2 4 6 8 10 12 14 16 18 20]
```

αλλά και:

```
a(1:end, 1:end)   →   [1 2 3; 4 5 6; 7 8 9]
```

Δηλαδή το κάθε end παριστάνει τον τελευταίο δείκτη της κάθε διάστασης ξεχωριστά. Επίσης όταν θέλουμε να γράψουμε 1:end μπορούμε να γράψουμε συνοπτικά και σκέτο : δηλαδή

`z(:)` → **[2 4 6 8 10 12 14 16 18 20]** *και* **`a(:, :)`** → **[1 2 3; 4 5 6; 7 8 9]**

αλλά και **`a(:)`** → **[1; 4; 7; 2; 5; 8; 3; 6; 9]**

Λογικές συγκρίσεις πινάκων

Οι λογικές συγκρίσεις μεταξύ πίνακα και βαθμωτού μεγέθους γίνονται μεταξύ κάθε στοιχείου του πίνακα και του βαθμωτού μεγέθους και έτσι προκύπτει ένας πίνακας ίδιων διαστάσεων που περιέχει το αποτέλεσμα της σύγκρισης του κάθε αντίστοιχου στοιχείου με το βαθμωτό, πχ

```
[1 2 3] > 2    →    [0 0 1]
```

Αντίστοιχα γίνεται και όταν έχουμε σύγκριση μεταξύ πινάκων (που θα πρέπει βέβαια να έχουν κοινές διαστάσεις), πχ

```
[1 2 3] >= [3 2 1]    →    [0 1 1]
```

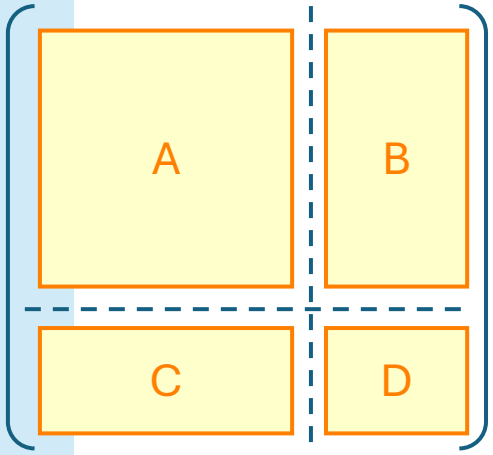
Και στις δύο περιπτώσεις ο τύπος δεδομένων του αποτελέσματος είναι logical, το οποίο σε συνδυασμό με τη δυνατότητα να δίνουμε logical δείκτες στους πίνακες μας επιτρέπει να γράφουμε:

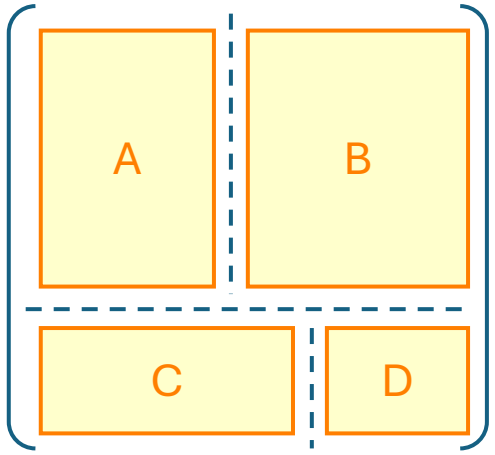
```
x = 1:5;
x(x>3)    →    [4 5]
```

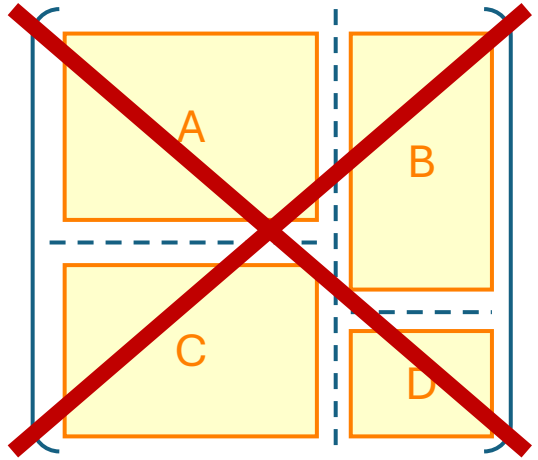
Δηλαδή σε αυτό το παράδειγμα βρίσκουμε τα στοιχεία του διανύσματος (με τη σειρά που υπάρχουν μέσα του) τα οποία ικανοποιούν κάποια συνθήκη.

Σύνθεση πινάκων

Πίνακες μπορούν να προκύψουν και με σύνθεση υπαρχόντων πινάκων, αρκεί να ταιριάζουν οι διαστάσεις τους. Πχ








```
x = [ A B ; C D ];
```

είτε απλούστερα για γραμμή ή για στήλη:

```
r = [ A , B ];
c = [ A ; B ];
```

Επειδή η σύνθεση των πινάκων γίνεται κατά γραμμές – όπως φαίνεται και από την εντολή – ένα τέτοιο σχήμα **δεν** είναι δυνατόν να γίνει με μία τέτοια εντολή.

Γίνεται όμως να δημιουργηθεί:

- είτε ως ανάστροφος και με τον σχετικό τελεστή να ξανα-αναστραφεί,
- είτε με τις συναρτήσεις **`horzcat`** και **`vertcat`**.

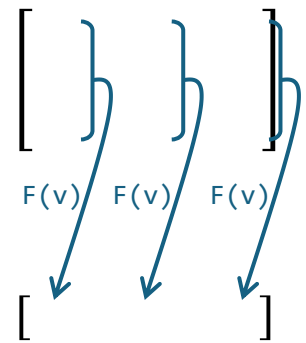
Αλγεβρικές συναρτήσεις πινάκων

Στα πλαίσια της Άλγεβρας πινάκων υπάρχουν οι παρακάτω διαθέσιμες συναρτήσεις.

`trace(a)`	Επιστρέφει το ίχνος του τετράγωνου πίνακα a
`det(a)`	Επιστρέφει την ορίζουσα του τετράγωνου πίνακα a
`eig(a)`	Επιστρέφει ένα διάνυσμα με τις ιδιοτιμές του τετράγωνου πίνακα a
`rank(a)`	Επιστρέφει την τάξη (ή βαθμό) του πίνακα a
`inv(a)`	Επιστρέφει τον αντίστροφο του πίνακα a
`norm(a)`	Επιστρέφει το Ευκλείδειο μέτρο του πίνακα (ή του διανύσματος) – για άλλα μέτρα δείτε την τεκμηρίωση
`dot(a,b)`	Επιστρέφει το εσωτερικό γινόμενο των a, b
`cross(a,b)`	Επιστρέφει το εξωτερικό γινόμενο των a, b
`linsolve(a, b)`	Επιστρέφει τη λύση του γραμμικού συστήματος a*x = b

Οι τυπικές – στατιστικού χαρακτήρα – συναρτήσεις διανυσμάτων είναι οι ακόλουθες. Εάν αυτές εφαρμοστούν σε πίνακα (2D) τότε εφαρμόζονται αυτόματα σε κάθε στήλη του και προκύπτει ένα διάνυσμα όπου κάθε στοιχείο του είναι το αποτέλεσμα της συνάρτησης.

`sum(v)`	Επιστρέφει το άθροισμα των στοιχείων του v
`prod(v)`	Επιστρέφει το γινόμενο των στοιχείων του v
`max(v)`	Επιστρέφει το μέγιστο των στοιχείων του v
`min(v)`	Επιστρέφει το ελάχιστο των στοιχείων του v
`cumsum cumprod cummax cummin`	Επιστρέφει διάνυσμα/πίνακα ίδιας διάστασης με το v, το σωρρευτικό αντίστοιχο των στοιχείων του v από 1 έως i, δηλαδή για κάθε `v(i) = cum...(v(1:i))`
`mean(v)`	Επιστρέφει τη μέση τιμή των στοιχείων του v
`median(v)`	Επιστρέφει την κεντρική τιμή των στοιχείων του v
`std(v)`	Επιστρέφει την τυπική απόκλιση των στοιχείων του v



Συναρτήσεις is...

Υπάρχει μια σειρά από συναρτήσεις που επιστρέφουν logical (δηλαδή boolean) και εξετάζουν διάφορες ιδιότητες μιας ποσότητας.

```
 isempty  - εξετάζει αν ένας πίνακας είναι κενός
 isequal  - εξετάζει αν δύο πίνακες είναι ίσοι (προσοχή! Το == δεν είναι το ίδιο)
isequaln  - εξετάζει όπως η isequal , αλλά θεωθεί το NaN ίσο με τον εαυτό του
isfinite  - εξετάζει εάν η τιμή που του δίνεται είναι πεπερασμένη
   isinf  - εξετάζει εάν η τιμή που του δίνεται είναι άπειρη
   isnan  - εξετάζει εάν η τιμή που του δίνεται είναι NaN
isletter  - εξετάζει εάν η τιμή που του δίνεται είναι χαρακτήρας (ως κωδικός)
 isprime  - εξετάζει εάν η τιμή που του δίνεται είναι πρώτος αριθμός
   isrow  - εξετάζει εάν η τιμή που του δίνεται είναι πίνακας-γραμμή
iscolumn  - εξετάζει εάν η τιμή που του δίνεται είναι πίνακας-στήλη
```

Φανταστικοί και μιγαδικοί αριθμοί

Ως μαθηματικό λογισμικό η MATLAB έχει πλήρη υποστήριξη για μιγαδικούς αριθμούς. Η φανταστική μονάδα παριστάνεται με το 1i ή το 1j (καθώς και με το i ή το j μόνα τους, αλλά αυτή η πρακτική είναι προβληματική στην περίπτωση που τα i ή j χρησιμοποιηθούν ως μεταβλητές – πράγμα σχεδόν βέβαιο).

Οι μιγαδικοί δημιουργούνται γράφοντας:

```
5+8i        3-4i        7+9j
x+y*1i      complex(5)  complex(7,9)
```

Από έναν μιγαδικό c μπορούμε να βρούμε τα μέρη του με:

```
x = real(z)        y = imag(z)
```

το μέτρο και τη φάση με:

```
abs(z)        angle(z)
```

και τον συζυγή του με:

```
z2 = conj(z)
```

οι υπόλοιπες μαθηματικές συναρτήσεις, που στα μαθηματικά δέχονται μιγαδικούς υποστηρίζουν και στη MATLAB μιγαδικούς ως ορίσματα.

Διάφορα αλγεβρικά χρήσιμα

Συναρτήσεις βαθμωτών μεγεθών

Όταν σε έναν πίνακα εφαρμοστούν συναρτήσεις που αφορούν βαθμωτά μεγέθη, τότε προκύπτει πίνακας ίδιων διαστάσεων όπου το κάθε στοιχείο είναι το αποτέλεσμα της εφαρμογής της συνάρτησης στο αντίστοιχο στοιχείο του αρχικού πίνακα.

```
sqrt([1 2; 4 16])   →   [sqrt(1) sqrt(2); sqrt(4) sqrt(16)]
```

Αλγεβρικοί τελεστές

Οι αλγεβρικοί τελεστές + - * / ^ λειτουργούν όπως στην άλγεβρα πινάκων (με ό,τι αυτό συνεπάγεται για τις διαστάσεις των τελεσταίων).

Εάν θέλουμε να γίνει ο πολλαπλασιασμός ή η διαίρεση των στοιχείων όπως η πρόσθεση (δηλαδή στοιχείο προς στοιχείο) τότε χρησιμοποιούμε τους τελεστές .* , ./ και .^

Επίσης ο τελεστής ' επιστρέφει τον ανάστροφο πίνακα (αλλά στην περίπτωση μιγαδικών τιμών προκύπτουν οι συζυγείς τους). Ενώ το .' απλά επιστρέφει τον ανάστροφο.

Συναρτήσεις πινάκων logical ποσοτήτων

Όταν ένας πίνακας έχει στοιχεία με τύπο δεδομένο logical, τότε μπορούμε να χρησιμοποιήσουμε τις συναρτήσεις all (και any) που μας επιστρέφουν αληθές όταν όλα (ή έστω ένα) τα στοιχεία του πίνακα είναι αληθή. Δηλαδή τα κάνουν and (και or αντίστοιχα) μεταξύ τους.

Ολοκληρωτικά και διαφορικά εργαλεία

Υπάρχουν οι συναρτήσεις που υπολογίζουν με αριθμητικούς υπολογισμούς:

`integral(f,a,b)`	το ολοκλήρωμα της συνάρτησης f στο διάστημα a,b
`trapz(Y, X)`	το ολοκλήρωμα της της καμπύλης που έχει σημεία x,y τα στοιχεία των X και Y αντίστοιχα
`diff(X)`	τις διαφορές των διαδοχικών στοιχείων του X, οπότε προκύπτει ένα διάνυσμα κατά 1 κοντύτερο

Συναρτήσεις (Ορισμός)

Οι συναρτήσεις που δημιουργούμε εμείς στη MATLAB προκειμένου να είναι διαθέσιμες τις γράφουμε μία σε κάθε αρχείο, με την ίδια λογική της ονομασίας των scripts.

Το αρχείο πρέπει να ξεκινά με τη δήλωση της συνάρτησης ως εξής:

```
function [y1,...,yN] = myfun(x1,...,xM)
```

Όπου x_i είναι οι είσοδοι (παράμετροι) της συνάρτησης και y_i είναι οι έξοδοι (αποτελέσματα) της, ενώ το όνομά της ακολουθεί τους γνωστούς κανόνες και πρέπει να συμπίπτει με το όνομα του αρχείου.

Το αποτέλεσμα της συνάρτησης επιστρέφεται δίνοντας τιμές στις μεταβλητές που έχουν δηλωθεί ως έξοδοι. Η `return` χρησιμοποιείται μόνο για να τερματίσει τη συνάρτηση πρόωρα. Δεν συνοδεύεται όμως (όπως στη C) με το αποτέλεσμα, αφού άλλωστε εδώ υποστηρίζονται πολλαπλά αποτελέσματα.

Η παραπάνω γενική σύνταξη της δήλωσης μπορεί να απλοποιηθεί ως εξής:

- Εάν έχουμε λιγότερα από δύο αποτελέσματα, μπορούμε να παραλείψουμε τις αγκύλες
- Εάν δεν έχουμε κανένα αποτέλεσμα μπορούμε να παραλείψουμε και το =
- Εάν δεν έχουμε καμία είσοδο (παράμετρο) μπορούμε να παραλείψουμε και τις παρενθέσεις

Σημειώστε ότι οι μεταβλητές που δηλώνονται μέσα στη συνάρτηση είναι όλες **τοπικές** και δεν έχουν σχέση με τις μεταβλητές του workspace όπως συμβαίνει στα scripts.

Εάν θέλουμε να υπάρχει «επικοινωνία» μέσω εξωτερικών μεταβλητών, θα πρέπει (πριν τη χρήση τους) να δηλωθούν ως global, γράφοντας στη γραμμή που ακολουθεί το function, τη λέξη `global` και δίπλα της χωρισμένες με , τις μεταβλητές που θέλουμε. Όπου χρησιμοποιούνται global μεταβλητές, θα πρέπει να δηλώνονται και εκεί, ακόμα και στη γραμμή εντολών.

Παράδειγμα

```
function [r1,r2] = aFunction(A)
% aFunction to create two matrices with the same size as A
%     A - a matrix that is used only to indicate the size of the result
%         Returns two matrices with the sum and the prod of the indices
%         of each element
    r1 = zeros(size(A));
    r2 = zeros(size(A));
    for i=1:size(A,1)
        for j=1:size(A,2)
            r1(i,j) = i+j;
            r2(i,j) = i*j;
        end
    end
end
```

Τεκμηρίωση συνάρτησης
Εμφανίζεται γράφοντας help και το όνομα της συνάρτησης

```
[q1,q2] = aFunction(zeros(3,5));
```

```
function [r1,r2] = bFunction
% bFunction to create two matrices with the same size as global A
%     A - a matrix that is used only to indicate the size of the result
%         Returns two matrices with the sum and the prod of the indices
%         of each element
global A

    r1 = zeros(size(A));
    r2 = zeros(size(A));
    for i=1:size(A,1)
        for j=1:size(A,2)
            r1(i,j) = i+j;
            r2(i,j) = i*j;
        end
    end
end
```

Πολύ καλή πρακτική/συνήθεια για λόγους που δεν μπορούν να παρουσιαστούν εδώ, να τερματίζεται η έκταση της συνάρτησης με την end

```
global A
A = zeros(3,5);
[q1,q2] = bFunction;
```

Συναρτήσεις (Κλήση)

Η κλήση μιας συνάρτησης αντίστοιχα γίνεται ακολουθώντας τη λογική της δήλωσης. Δηλαδή γράφουμε:

```
[y1, y2] = f(x1,x2,x3);
```

Και πιο ειδικά μπορεί να ισχύει και:

```
f(x,y);
```

ή

```
y1 = f;
```

Επίσης μπορούν να δηλωθούν και οι ανώνυμες συναρτήσεις (anonymous functions) ως εξής:

```
half = @(x) x/2;
```

οπότε είτε καλούνται ως συναρτήσεις «κατά τα γνωστά», είτε μπορούν να δοθούν και ως παράμετροι σε άλλες συναρτήσεις!

```
half(100)
```

ή

```
integral(half,0,1)
```

Γραφήματα

Με όλα αυτά τα δεδομένα που διαχειριζόμαστε στη MATLAB, συχνά χρειάζεται να τα παραστήσουμε με γραφικό τρόπο. Οι γραφικές παραστάσεις είναι 2 ή 3 διαστάσεων. Για να περιγράψουμε γραμμές, χρειαζόμαστε μονοδιάστατα δεδομένα πχ

```
x = 1:30; y = x.*sin(x); plot(x,y);
```

Στην ίδια λογική λειτουργεί και η `area` και η `scatter` και η `stem`.

Η κάθε μία από αυτές παίρνει και άλλες παραμέτρους, ανάλογα με τη φύση της. Π.χ.

- Η `plot` παίρνει ως 3η παράμετρο χαρακτήρες που καθορίζουν το σημάδι των σημείων, την καμπύλη και το χρώμα.
- Η `area` έχει ως 3η παράμετρο μια τιμή που ορίζει την τιμή (κατά y) από την οποία γίνεται η σκίαση.
- Η `scatter` έχει ως επόμενες παραμέτρους τιμές που ορίζουν το μέγεθος, το χρώμα, τον τύπο (γεμάτο ή περίγραμμα) και το σχήμα των σημαδιών.
- Η `stem` έχει ως επόμενες παραμέτρους τιμές που ορίζουν τον τύπο των σημαδιών και της γραμμής.

Ειδικά η `plot` έχει τη δυνατότητα να πάρει και πρόσθετες τριάδες (ή δυάδες) παραμέτρων ώστε να προστίθενται στο ίδιο γράφημα.

Γραφήματα

Στο επίπεδο γίνονται και οι ακόλουθες παραστάσεις.

```
bar(y) ή bar(x,y)
```

και

```
pie(y)
```

και

```
t=0:0.1:(2*pi); r=3*t; polarplot(t,r);
```

και

```
loglog(x,y) ή semilogx(x,y) ή semilogy(x,y)
```

όπου και αυτές παίρνουν πρόσθετες παραμέτρους σχετικά με την εμφάνιση των γραφημάτων.

- Η bar παίρνει παραμέτρους που ρυθμίζουμε το πλάτος της μπάρας, το πως συνδυάζονται πρόσθετα δεδομένα, το χρώμα.
- Η pie παίρνει παραμέτρους που ρυθμίζουμε ποια κομμάτια της πίτας προεξέχουν και τις ετικέτες γραμμένες σε διπλά εισαγωγικά.
- Η polarplot λειτουργεί όμοια με την plot, απλά μόνο σε πολικές συντεταγμένες.
- Οι loglog , semilogx και semilogy λειτουργούν όμοια με την plot σε λογαριθμική και ημιλογαριθμική κλίμακα

Γραφήματα

Η μορφοποίηση της γραμμής και των σημαδιών γίνεται με τις ακόλουθες επιλογές.

Για τη γραμμή: `- -- : -.`

Για το σημάδι: `o + * . x s d ^ v < >`

Για το χρώμα: `r g b  c m y k  w`

Γραμμικές παραστάσεις γίνονται και στον χώρο ως τρισδιάστατες καμπύλες.

Π.χ.:

```
t = 0:0.1:(8*pi); x=t.*cos(t); y= t.*sin(t);
                  z = t; plot3(x,y,z);
```

και αντίστοιχα η `scatter3`.

Ενώ για απλό 3D εφέ μπορούμε να έχουμε και τις `bar3` και `pie3`.

Γραφήματα

Οι γραφικές παραστάσεις επιφανειών απαιτούν 3D δεδομένα. Τα δεδομένα αυτά θα πρέπει να δίνονται ως 3 πίνακες δύο διαστάσεων όπου το αντίστοιχο στοιχείο των πινάκων δημιουργεί μία τριάδα συντεταγμένων στον χώρο.

Προκειμένου από μια απλή κατανομή x και y να πάρουμε δισδιάστατους πίνακες συντεταγμένων, χρησιμοποιούμε τη βοηθητική συνάρτηση meshgrid.

```
x=-10:10; y=x; [X Y] = meshgrid(x,y);
```

όπου ο X έχει όσες γραμμές καθορίζει ο y, αλλά η κάθε μία είναι αντίγραφο της x και αντίστοιχα ο Y. Έτσι:

```
[a1,a2]=meshgrid(1:3,7:9)
a1 =                        a2 =
     1     2     3               7     7     7
     1     2     3               8     8     8
     1     2     3               9     9     9
```

οπότε μπορούμε εύκολα να υπολογίσουμε τις κατά Z τιμές για αρκετές συναρτήσεις. Π.χ.

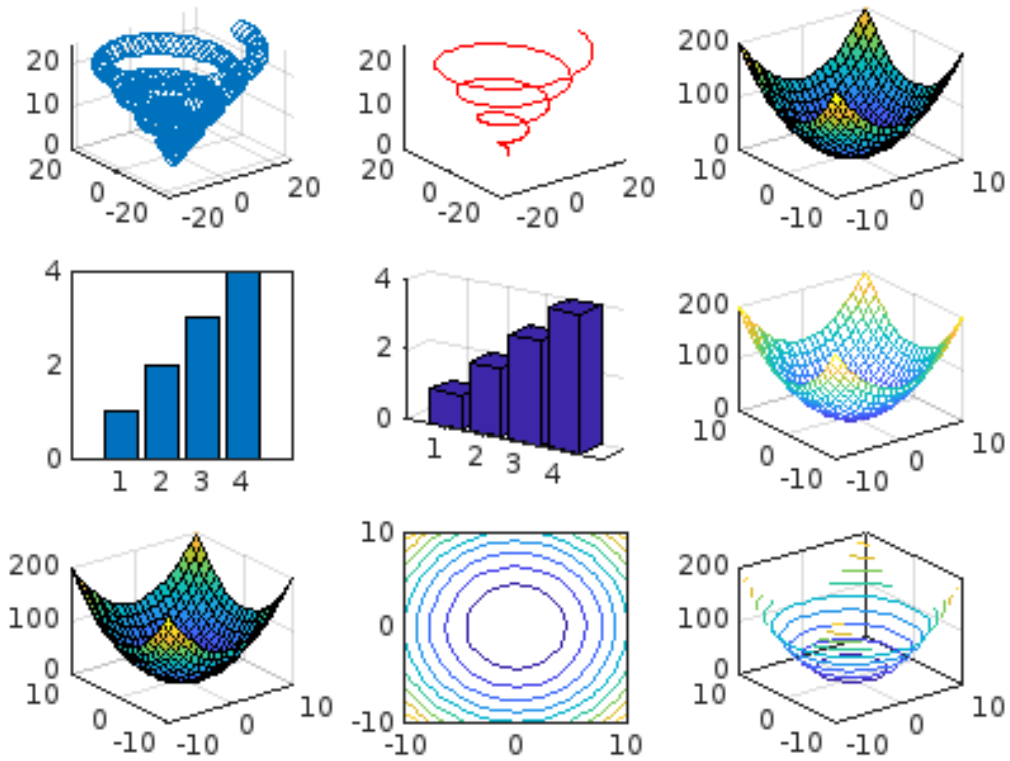
```
z = X.^2 + Y.^2;
```

Γραφήματα

Με δεδομένους αυτούς τους πίνακες μπορούμε να κάνουμε τους ακόλουθους τύπους γραφικών παραστάσεων:

- `mesh(X,Y,z)`
- `surf(X,Y,z)`
- `contour(X,Y,z)`
- `contour3(X,Y,z)`

Δίπλα βλέπουμε διάφορους τύπους γραφικών παραστάσεων.



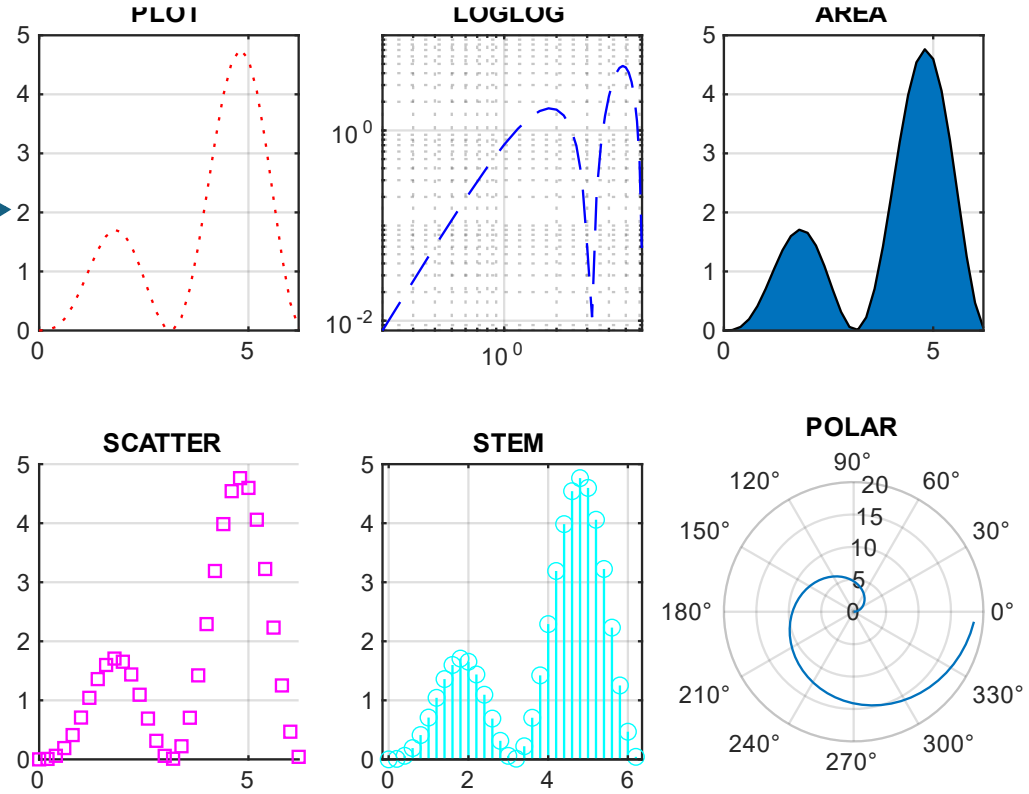
Βοηθητικές συναρτήσεις

Για κάθε γραφική παράσταση (η οποία εμφανίζεται σε ένα figure), να προσθέσουμε τα παρακάτω:

`title`	Τίτλο (και υπότιτλο)
`xlabel`	Ετικέτα άξονα (και ylabel και zlabel)
`legend`	Υπόμνημα
`colormap`	Καθορισμός χρωματικής κλίμακας
`colorbar`	Εμφάνιση/απόκρυψη της χρωματικής κλίμακας
`grid`	Εμφάνιση/απόκρυψη του πλέγματος
`hold`	Διατήρηση αντί για αντικατάσταση επόμενου γραφήματος
`subplot`	Ορίζει μία υποπεριοχή του figure βάσει ενός πλέγματος
`fplot`	Σχεδιάζει μία γραφική παράσταση βάσει του τύπου π.χ. `fplot(@(x) sin(x),[0 2*pi],'r:')`
`figure`	Εμφανίζει την περιοχή ενός γραφήματος

Παράδειγμα script

```
% Δεδομένα
x = 0:0.2:(2*pi);
y = x.*sin(x).^2;
t=0:0.1:(2*pi);
r=3*t;

% Σχεδιασμός
subplot(2,3,1)
plot(x,y,':r');
title('PLOT');
grid on;

subplot(2,3,2);
loglog(x,y,'--b');
title('LOGLOG');
grid on;

subplot(2,3,3);
area(x,y);
title('AREA');
grid on;

subplot(2,3,4);
scatter(x,y,'sm');
title('SCATTER');
grid on;

subplot(2,3,5);
stem(x,y,'c');
title('STEM');
grid on;

subplot(2,3,6);
polarplot(t,r);
title('POLAR');
grid on;
```



Παράδειγμα script

```
% Δεδομένα
x = 0:0.1:10;
y = x;
[X Y] = meshgrid(x,y);
Z = X.^2 + Y.^2;

% Σχεδιασμός
subplot(2,2,1);
mesh(X,Y,Z);
title('Mesh’);
grid on;

subplot(2,2,2);
surf(X,Y,Z);
title('Surf’);
grid on;

subplot(2,2,3);
contour3(X,Y,Z);
title('Contour 3D’);
grid on;

subplot(2,2,4);
contour(X,Y,Z);
title('Contour (2D)');
grid on;
```