Εισαγωγή στο MATLAB

Αριθμητική Ανάλυση

Νικόλαος Πασσαλής

Διδάσκων: Αναστάσιος Τέφας

Τί είναι το MATLAB;

Τί είναι το MATLAB;

- "**Προγραμματιστικό περιβάλλον**" που επιτρέπει τη **γρήγορη ανάπτυξη εφαρμογών** (Rapid Prototyping Environment).
  - Λογική: Έχω μία ιδέα! Μπορώ γρήγορα (και χωρίς να σπαταλήσω χρόνο σε τεχνικές λεπτομέρειες) να δοκιμάσω αν δουλεύει.
  - Εύκολη έκφραση *μαθηματικών εννοιών* σε *κώδικα*.

Τί είναι το MATLAB;

- "**Προγραμματιστικό περιβάλλον**" που επιτρέπει τη **γρήγορη ανάπτυξη εφαρμογών** (Rapid Prototyping Environment).
  - Λογική: Έχω μία ιδέα! Μπορώ γρήγορα (και χωρίς να σπαταλήσω χρόνο σε τεχνικές λεπτομέρειες) να δοκιμάσω αν δουλεύει.
  - Εύκολη *έκφραση μαθηματικών εννοιών σε κώδικα*.
- Αποτελείται από:
  - μία **γλώσσα προγραμματισμού**
  - ένα **διαδραστικό περιβάλλον ανάπτυξης**
  - εργαλεία για **εύκολη οπτικοποίηση** των αποτελεσμάτων
  - **βιβλιοθήκες (toolboxes)** για διάφορες εργασίες (επεξεργασία σήματος, μηχανική μάθηση, υπολογιστική βιολογία, προσομοίωση φυσικών συστημάτων, ...)
  - και πολλά άλλα ...

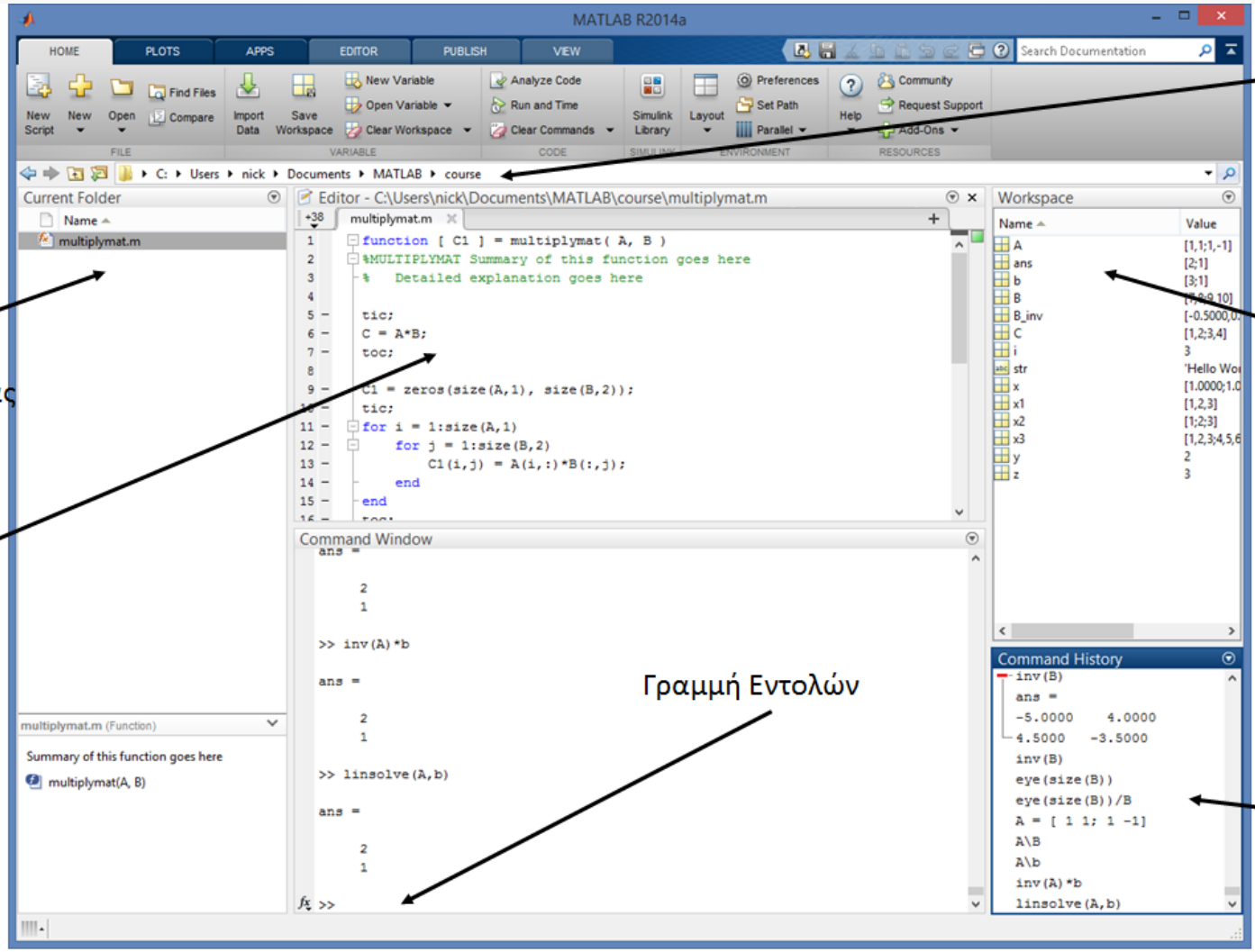
Τί είναι το MATLAB;

- Χρησιμοποιείται ευρέως τόσο στον ακαδημαικό τομέα όσο και στη βιομηχανία.
  - Πολλές third-party βιβλιοθήκες που επεκτείνουν τις δυνατότητες του.
  - Χρήσιμο προσόν ...

Τί είναι το MATLAB;

- Χρησιμοποιείται ευρέως τόσο στον ακαδημαικό τομέα όσο και στη βιομηχανία.
  - Πολλές third-party βιβλιοθήκες που επεκτείνουν τις δυνατότητες του.
  - Χρήσιμο προσόν ...
- MATLAB: MATrix LABoratory
  - Ξεκίνησε ως ένας wrapper συναρτήσεων Fortran για πράξεις γραμμικής άλγεβρας (LINPACK, EISPACK).
  - Βασική μονάδα υπολογισμού: Πίνακες
- Δεν είναι γλώσσα προγραμματισμού γενικού σκοπού (general purpose language), αλλά συνήθως είναι από τις καλύτερες επιλογές, όταν χρησιμοποιείται για αυτά που σχεδιάστηκε να κάνει.

Περιβάλλον Ανάπτυξης



Αριθμητικές Πράξεις

- Το MATLAB είναι μία (πολύ) εξελιγμένη αριθμομηχανή...
- Μπορεί και εκτελεί απλές πράξεις:

```
>> 1+2
ans = 3
>> 3.141*5 + 8/4.5 + (2.3)*5
ans = 28.9828
>> 2^1.2
ans = 2.2974
```

Έτοιμες συναρτήσεις

- Οι περισσότερες βασικές μαθηματικές συναρτήσεις παρέχονται έτοιμες από το MATLAB:

```
>> sqrt(9)
ans = 3
>> cos(3.14)
ans = -1.0000
>> pi
ans = 3.1416
>> cos(pi)
ans = -1
>> abs(-3)
ans = 3
>> sign(-5)
ans = -1
>> exp(-1.3)
ans = 0.2725
```

Αριθμητικές Πράξεις

- Όλοι οι αριθμοί που εισάγονται θεωρούνται **αριθμοί κινητής υποδιαστολής διπλής ακρίβειας (double).**
- Η συνάρτηση *class()* επιστρέφει τον τύπο μίας μεταβλητής ή τιμής.

```
>> 1+2
ans = 3
>> 1.0 + 2.0
ans = 3
>> class(1)
ans = double
>> class(1.0)
ans = double
>> 0.99999 + 2
ans = 3.0000
```

Μεταβλητές

- Οι μεταβλητές στο MATLAB **ορίζονται αυτόματα την πρώτη φορά που ανατίθεται τιμή σε αυτές** (weakly typed).
  - Ο τύπος μίας μεταβλητής ενδέχεται να αλλάξει κατά την ανάθεση μίας νέας τιμής σε αυτή!
- Οι μεταβλητές είναι **case-sensitive!**
- Ισχύουν οι γνωστοί κανόνες για την ονοματολογία των μεταβλητών (να μην ξεκινά με αριθμό, να μην είναι δεσμευμένη λέξη, κτλ),

```
>> x = 1
x = 1
>> y = 2
y = 2
>> z = x + y
z = 3
>> class(z)
ans = double
```

Μεταβλητές

- Εκτός από αριθμούς μπορούμε να ορίσουμε ανάλογα και συμβολοσειρές (μέσα σε μονά εισαγωγικά).

```
>> str = 'Hello World'
str = Hello World
>> class(str)
ans = char
```

- Αν δεν οριστεί κάποια μεταβλητή όπου θα αποθηκευτεί το αποτέλεσμα, το MATLAB το αποθηκεύει αυτόματα στη μεταβλητή *ans*:

```
>> 1+2
ans = 3
>> ans
ans = 3
```

Μεταβλητές

- Παρατηρούμε ότι μετά απο κάθε εντολή το MATLAB επιστρέφει το αποτέλεσμα της πράξης.
- Αν αυτό δεν είναι επιθυμητό, μπορούμε να χρησιμοποιήσουμε τον χαρακτήρα ';' για να αποκρύψουμε το αποτέλεσμα:

```
1 >> x = 2 + 3;
2 >> x
3 x = 5
```

- Τα αποτελέσματα συνεχίζουν να αποθηκεύονται στη μεταβλήτη *ans*, αν δεν οριστεί μεταβλητή εξόδου:

```
1 >> 2+3;
2 >> ans
3 ans = 5
```

Πίνακες

- Ορισμός **διανύσματος-γραμμή** (τα κενά διαχωρίζουν στήλες):

```
>> x1 = [1 2 3]
x1 = 1     2     3
```

- Ορισμός **διάνυσματος-στήλη** (τα ερωτηματικά διαχωρίζουν γραμμές):

```
>> x2 = [1; 2; 3]
x2 = 1
     2
     3
```

- Ορισμός **πίνακα**:

```
>> x3 = [1 2 3; 4 5 6; 7 8 9 ]
x3 = 1     2     3
     4     5     6
     7     8     9
```

Πίνακες

- Μπορούμε να διαπιστώσουμε το μέγεθος ενός πίνακα με τη συνάρτηση *size()*:

```
>> size([1 2 3])
ans = 1     3
>> size([1; 2; 3])
ans = 3     1
>> size([1 2 3; 4 5 6; 7 8 9 ])
ans = 3     3
```

- Η συνάρτηση *size()* επιστρέφει ένα διάνυσμα με το μέγεθος κάθε διάστασης του πίνακα (πρώτα γραμμές και μετά στήλες)
- **Τα πάντα είναι πίνακες!** Οι μεταβλητές θεωρούνται πίνακες με ένα στοιχείο!

```
>> size(1)
ans = 1     1
```

- Μπορούμε να λάβουμε μόνο το μέγεθος της διάστασης που μας ενδιαφέρει προσδιορίζοντάς τη ως δεύτερο όρισμα στη συνάρτηση *size()*:
  - (1 για πλήθος γραμμών, 2 για πλήθος στηλών)

```
>> size([1 2 3], 1)
ans = 1
>> size([1 2 3], 2)
ans = 3
```

- Μπορούμε επίσης να χρησιμοποιήσουμε τη συνάρτηση *length()* η οποία επιστρέφει το μέγεθος της μεγαλύτερης διάστασης:

```
>> length([1 2 3])
ans = 3
>> length([1 2 3; 2 3 4; 5 6 7; 8 9 10])
ans = 4
```

Ορίζουμε δύο πίνακες που θα χρησιμοποιηθούν στα επόμενα παραδείγματα:

```
>> A = [ 1 2; 3 4; 5 6]
A = 1 2
    3 4
    5 6
>> B = [ 7 8; 9 10]
B = 7 8
    9 10
>> C = [ 1 2; 3 4]
C = 1 2
    3 4
```

Πράξεις Πίνακων

- Οι περισσότερες πράξεις λειτουργούν **και σε πίνακες.**

```
>> B + C
ans =   8     10
       12     14
```

- Οι διαστάσεις των πινάκων προς πρόσθεση/αφαίρεση/... **πρέπει να συμφωνούν!**

```
>> A + B
Error using  +
Matrix dimensions must agree.
```

Μαθηματικές Συναρτήσεις σε Πίνακες

- Οι περισσότερες μαθηματικές συναρτήσεις μπορούν να χρησιμοποιηθούν κατευθείαν σε πίνακες. Σε αυτή την περίπτωση εφαρμόζονται στα στοιχεία των πινάκων (element-wise):
  - Αυτό όμως δεν ισχύει πάντα!

```
>> sqrt(B)
ans =  2.6458    2.8284
       3.0000    3.1623
>> log2(B)
ans =  2.8074    3.0000
       3.1699    3.3219
>> exp(B)
ans =  1.0e+04 *
       0.1097    0.2981
       0.8103    2.2026
```

Πολλαπλασιασμός Πινάκων

- Οι τελεστές/συναρτήσεις δεν εφαρμόζονται element-wise, αν έχουν ειδική σημασία στη γραμμική άλγεβρα (π.χ. **πολλαπλασιασμός πίνακα με πίνακα**, δυνάμεις, νόρμα, κτλ):

```
>> A*B
ans =    25    28
         57    64
         89   100
```

- Αν επιθυμούμε τον **πολλαπλασιασμό στοιχείο με στοιχείο (element-wise)**, χρησιμοποιούμε την τελεία πριν τον τελεστή, π.χ. .* αντί για *

```
>> A .* A
ans =    1    4
         9   16
        25   36
```

Πράξεις Πινάκων

- Οι διαστάσεις των πινάκων πρέπει να ταιριάζουν για να εκτελεστεί ο πολλαπλασιασμός:

```
>> B*A
Error using  *
Inner matrix dimensions must agree.
```

- Άλλες γνωστές συναρτήσεις που ορίζονται πάνω σε πινάκες:

```
>> norm(A)
ans = 9.5255
>> det(B)
ans = -2.0000
>> det(A)
Error using det
Matrix must be square.
```

Πράξεις Πινάκων

- Επίσης, υποστηρίζεται ο **πολλαπλασιασμός/πρόσθεση αριθμού με πίνακα.**

```
>> A * 2
ans =   2     4
        6     8
       10    12
>> A - 1
ans =   0     1
        2     3
        4     5
>> [1 2 3] / 0.5
ans =   2     4     6
```

Βοήθεια!

- Υπάρχουν **έτοιμες συναρτήσεις** για τα περισσότερα πράγματα που θα θελήσετε να κάνετε.
- Πώς θα βρείτε πώς ονομάζεται και πώς χρησιμοποιείται αυτή που θέλετε?
  - Στο **internet** `http://www.mathworks.com/help/matlab` ή στο **built-in reference** του MATLAB
  - Στα "γρήγορα" με την εντολή *help*

```
>> help norm
norm   Matrix or vector norm.
norm(X,2) returns the 2-norm of X.
norm(X) is the same as norm(X,2).
norm(X,1) returns the 1-norm of X.
norm(X,Inf) returns the infinity norm of X.
norm(X,'fro') returns the Frobenius norm of X.
In addition, for vectors...
norm(V,P) returns the p-norm of V defined as SUM(ABS(V).
...
```

Ανάστροφος πίνακας

- Μπορούμε να υπολογίσουμε τον **συζυγή ανάστροφο** ενός πίνακα με τον τελεστή '
- Αν επιθυμούμε τον απλό **ανάστροφο πίνακα** χρησιμοποιούμε τον τελεστή .'

```
>> A.'
ans = 1     3     5
      2     4     6
>> size(A)                  | >> size(A.')
ans = 3     2                | ans =  2     3
```

- *Υπενθύμιση*: Αν ο πίνακας A περιέχει μόνο πραγματικές τιμές, οι παραπάνω δύο πίνακες ταυτίζονται.

Χρήσιμες συναρτήσεις

- *zeros(M,N)*: Δημιουργεί έναν πίνακα διάστασης $M \times N$ και τον αρχικοποιεί με 0.
- *ones(M,N)*: Δημιουργεί έναν πίνακα διάστασης $M \times N$ και τον αρχικοποιεί με 1.
- Προσοχή:
  - Οι συναρτήσεις *zeros(M)* και *ones(M)* δέχονται και 1 όρισμα. Σε αυτή την περίπτωση κατασκευάζουν **τετραγωνικούς πίνακες** $M \times M$, **όχι διανύσματα**!
  - Επίσης, οι συναρτήσεις αυτές δέχονται και διανύσματα. Αυτό είναι ιδιαίτερα χρήσιμο, αν θέλουμε να κατασκευάσουμε έναν πίνακα ίδιας διάστασης με κάποιον άλλο.

```
1 zeros(size(A))
2 ans =     0     0
3           0     0
4           0     0
```

Χρήσιμες συναρτήσεις

- *eye(M)*: Δημιουργεί έναν **μοναδιαίο πίνακα** $M \times M$

```
>> eye(4)
ans =  1     0     0     0
       0     1     0     0
       0     0     1     0
       0     0     0     1
```

- *inv(X)*: Υπολογίζει τον **αντίστροφο** του πίνακα X

```
>> inv(B)
ans = -5.0000    4.0000
       4.5000   -3.5000
```

Διαίρεση πινάκων

- ... δηλαδή **πολλαπλασιασμός με τον αντίστροφο**
  - Διαίρεση απο τα δεξιά $P/Q = P * inv(Q)$
  - Διαίρεση απο τα αριστερά $P\backslash Q = inv(Q) * P$

```
>> eye(size(B))/B
ans = -5.0000    4.0000
       4.5000   -3.5000
>> eye(size(B))\B
ans = 7    8
      9   10
>> B/eye(size(B))
ans = 7    8
      9   10
```

Προσπέλαση Πίνακων

- **Η αρίθμηση των δεικτών στους πίνακες ξεκινά από το 1** (όχι απο το 0, όπως π.χ. στη C/C++/Java)!

```
>> A(1,1)
ans = 1
>> A(1,2)
ans = 2
>> A(3,2)
ans = 6
>> A(3,3)
Index exceeds matrix dimensions.
>> A(0,0)
Subscript indices must either be real positive
    integers or logicals.
```

- Ο πρώτος δείκτης αφορά στη γραμμή και ο δεύτερος στη στήλη (subscript indexing).

Προσπέλαση Πίνακων

- Χρησιμοποιώντας το keyword *end* μπορούμε να αναφερθούμε στην **τελευταία θέση της στήλης/γραμμής**

```
>> A(1, end)
ans = 1
```

- Μπορούμε επίσης να **ορίσουμε ένα range** το οποίο θέλουμε να προσπελάσουμε ορίζοντας την αρχή και το τέλος του

```
>> A(1:2, end)
ans = 2
      4
>> A(2:3, 1:end)
ans = 3     4
      5     6
```

Προσπέλαση Πίνακων

- Μπορούμε επίσης να τροποποιήσουμε τις τιμές ενός πίνακα είτε μία-μία είτε συγκεντρωτικά:

```
>> A(1,1) = 0;
>> A(2,1:2) = -1;
>> A(3,1:2) = [-2 -3]
A =  0   2
    -1  -1
    -2  -3
```

- Όταν έχουμε ορίσει εύρος τιμών στους δείκτες ενός πίνακα, πρέπει να παρέχουμε ως τιμή
  - είτε **μια μοναδική τιμή** που θα τοποθετηθεί **σε όλες τις θέσεις του εύρους**,
  - είτε **έναν πίνακα** που θα **αντικαταστήσει** όλες τις τιμές του εύρους (πρέπει να έχει την ίδια διάσταση).

Προσπέλαση Πίνακων

- Τα στοιχεία ενός πίνακα επίσης αριθμούνται διαδοχικά από πάνω προς τα κάτω και έπειτα από αριστερά προς τα δεξιά.
- Μπορούμε να αναφερθούμε σε αυτά προσδιορίζοντας μόνο μια τιμή (linear indexing)

```
>> A = [1 4; 2 5; 3 6];
>> A(3)
ans = 3
>> A(1:end)
ans = 1     2     3     4     5     6
>> A(:)
ans = 1
2
3
4
5
6
```

Προσπέλαση Πίνακων

- Παρατηρούμε ότι η εντολή *A(:)* είναι ισοδύναμη με την *A(1:end).'*. **Γιατί**;
- Τα ranges, π.χ. 1:5, παράγουν διανύσματα που μπορούν να χρησιμοποιηθούν (μεταξύ άλλων) για την προσπέλαση των τιμών των πινάκων

```
>> 1:5
ans = 1     2     3     4     5
>> 1:2:5
ans = 1     3     5
>> 5:-1:2
ans = 5     4     3     2
>> A=[1 2; 3 4; 5 6]
>> A([1 1 3])
ans = 1     1     5
```

- ΑΠΑΝΤΗΣΗ: Το 1:end παράγει ένα διάνυσμα γραμμή, ενώ το : ένα διάνυσμα στήλη (ισοδύναμο του (1:end)')

Προσπέλαση Πίνακων

- Μπορούμε να λάβουμε/τροποποιήσουμε διανύσματα ενός πίνακα με τη χρήση του τελεστή :

```
>> x=A(1,:)
x =  1     2
>> y=A(:,2)
y =  2
     4
     6
>> z=A(1:2,:)
z =
     1     2
     3     4
```

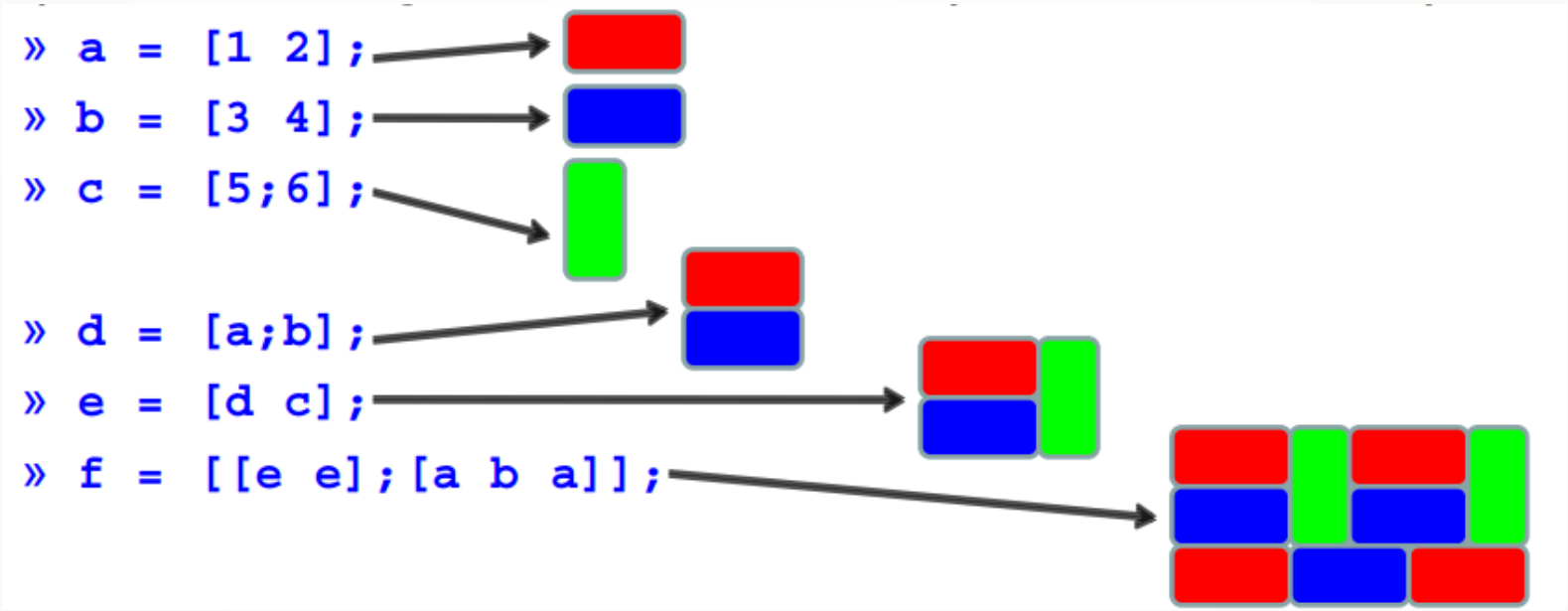
Διαγραφή τιμών

- Μπορούμε να διαγράψουμε τιμές/διανύσματα αναθέτοντάς τους τον κενό πίνακα []:

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

Συνένωση Πίνακων

- Μπορούμε να ορίσουμε νέους πίνακες χρησιμοποιώντας άλλους πίνακες!
- Θα πρέπει οι πίνακες που χρησιμοποιούμε να ορίζουν ένα ορθογώνιο:



Άσκηση

- Θέλουμε να επιλύσουμε το γραμμικό σύστημα

$$x + y = 3, x - y = 1 \tag{1}$$

Άσκηση

- Θέλουμε να επιλύσουμε το γραμμικό σύστημα

$$x + y = 3, x - y = 1 \tag{1}$$

- Λύση:

```
>> A = [ 1 1; 1 -1 ]
>> b = [ 3; 1 ]
>> inv(A)*b
ans = 2
      1
>> A\b
ans = 2
      1
>> linsolve(A,b)
ans = 2
      1
```

Λογικές Συνθήκες

- Υπάρχουν **δύο λογικές τιμές**: true (1), false (0)
- Όπως και στη C/C++, **οποιαδήποτε μη μηδενική τιμή θεωρείται αληθής**, ενώ μόνο **το 0 θεωρείται λογικό false.**
- Λογικοί Τελεστές
  - Λογική Σύζευξη: and(x,y) ή x & y
  - Λογική Διάζευξη: or(x,y) ή x | y
  - Λογική Άρνηση: not(x) ή x
  - XOR: xor(x,y)
- Οι προηγούμενοι τελεστές and και or δεν χρησιμοποιούν short-circuit evaluation (δηλαδή υπολογίζουν και τα δύο ορίσματα ακόμη και αν δεν χρειάζεται).
- Οι αντίστοιχες short-circuited εκδοχές χρησιμοποιούνται με τους τελεστές && και ||

Λογικές Συνθήκες

```
>> 1 < 3
ans = 1
>> 1 < 3 && abs(3) ~= 0
ans = 1
>> 5<=3
ans = 0
>> i = 3;
>> res = size(A,2) > 5 | i == 3
res = 1
>> class(res)
ans = logical
>> fprintf('first\n') || fprintf('second\n');
first
>> fprintf('first\n') | fprintf('second\n');
first
second
```

Δομή If

- Λειτουργεί όπως και στη C, όμως διαφέρει στη σύνταξη.
- Δεν χρησιμοποιούνται brackets {, } και το τέλος του μπλοκ ορίζεται με το keyword *end*.

```
>> a = 4;
if a > 0
    fprintf('positive (value = %2.3f)\n', a);
elseif a == 0
    fprintf('zero\n');
else
    fprintf('negative (value = %2.3f)\n', a);
end

positive (value = 4.000)
```

- Δεν είναι αναγκαστικό να εμφανίζονται και τα *elseif* και *else*.

Δομή While

- Έχει την αναμενόμενη λειτουργικότητα (εκτελεί επαναλήψεις έως ότου σταματήσει να ισχύει η συνθήκη) και στη σύνταξη μοιάζει με την if:

```
>> t=0;
while t < 4
fprintf('t is %d\n', t);
t = t + 1;
end

t is 0
t is 1
t is 2
t is 3
```

Δομή For

- Δομή for:

```
for i = start_value:step(optional):end_value
...code...
end
```

- Παράδειγμα:

```
>> ac = 0;
>> for t = 1:5
ac = ac + t;
end
>> ac
ac = 15
```

- **Προσοχή**: Δεν τροποποιούμε ποτέ τη μεταβλητή της for εντός του loop!
  - Αν θέλουμε να το κάνουμε αυτό, χρησιμοποιούμε ένα while loop.

Δομή For

- Στην πραγματικότητα μοιάζει περισσότερο με τις δομές foreach παρά με την κλασική for.
- Δεν υπάρχει δυνατότητα τροποποίησης.
- Διατρέχει όλα τα διανύσματα-στήλες του πίνακα που της δίνεται:

```
>> acc = 0;
>> for t = eye(4)
acc = acc + t;
end
>> acc
acc = 1
      1
      1
      1
```

- Παρατηρούμε ότι **αλλάζει αυτόματα ο τύπος της μεταβλητής** *acc*.

Δομή For

- Γιατί δεν λειτουργεί ο παρακάτω κώδικας;

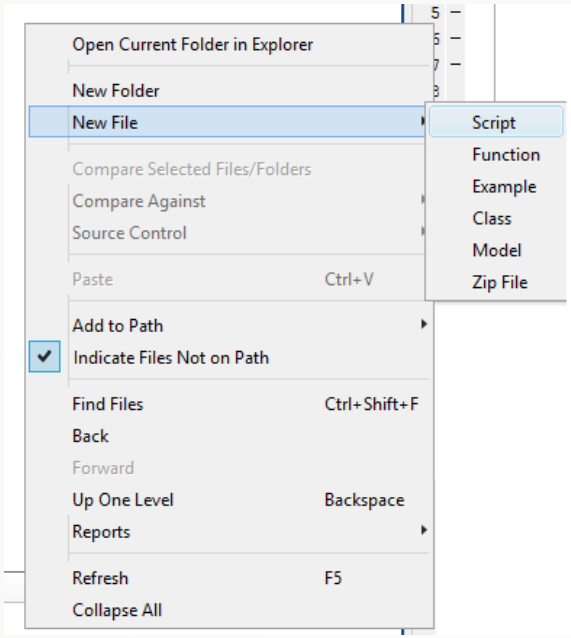
```
>> for t = eye(4)
acc = acc + t;
end
Undefined function or variable 'acc'.
```

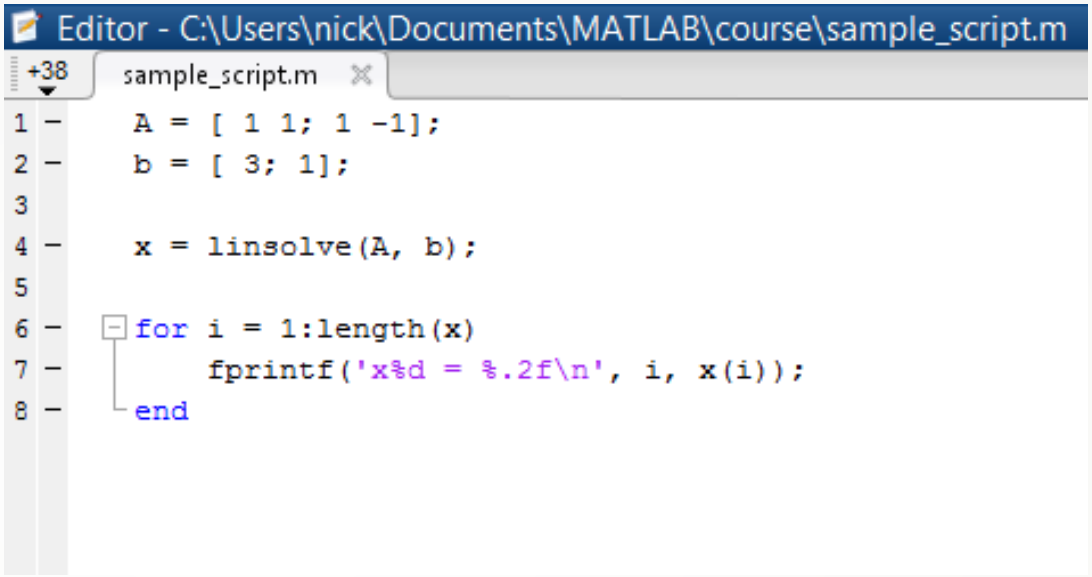
- Οι μεταβλητές:
  - **δεν χρειάζεται να οριστούν πριν την πρώτη ανάθεση τιμής** σε αυτές
  - **αλλάζουν αυτόματα τύπο**
  - ΟΜΩΣ, πρέπει **να περιέχουν μία τιμή πριν την πρώτη αναφορά** σε αυτές.

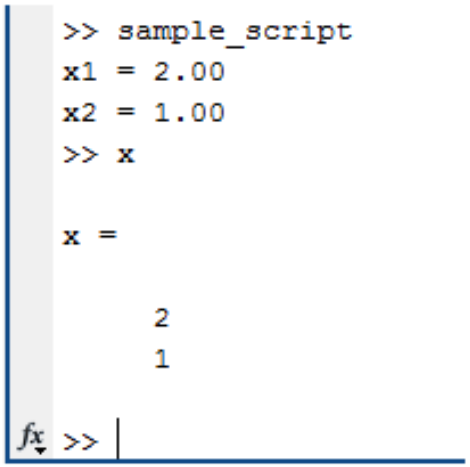
Scripts

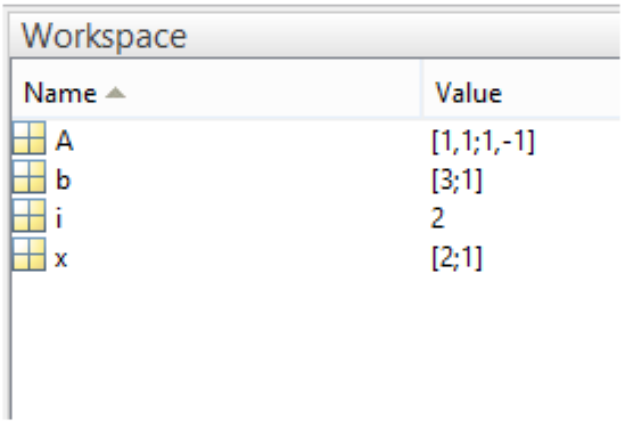
- Τα προηγούμενα παραδείγματα ήταν μικρά και εκτελούσαμε τις εντολές μία προς μία στη γραμμή εντολών.
- Αυτό **δεν είναι πρακτικό**, όταν πρόκεται να γράψουμε μεγαλύτερα προγράμματα.
- Μπορούμε να **ομαδοποιήσουμε τον κώδικα σε ένα script** και μετά να το **εκτελέσουμε**.
  - **Τα scripts είναι αρχεία** με την κατάληξη .m
  - Για να **εκτελέσουμε ένα script** πρέπει να **βρισκόμαστε στον φάκελο που το περιέχει** (είτε να έχουμε προσθέσει τον φάκελο στο PATH του MATLAB)
  - Εκτελούμε το script **πληκτρολογώντας το όνομα του αρχείου** του (χωρίς την κατάληξη)
  - Όλες οι μεταβλητές που ορίζονται κατά τη διάρκεια εκτέλεσης του script **προστίθενται στο workspace** και μπορούμε να τις χρησιμοποιήσουμε από τη γραμμή εντολών

Scripts


Open Current Folder in Explorer
New Folder
New File
Compare Selected Files/Folders
Compare Against
Source Control
Paste Ctrl+V
Add to Path
Indicate Files Not on Path
Find Files Ctrl+Shift+F
Back
Forward
Up One Level Backspace
Reports
Refresh F5
Collapse All
Script
Function
Example
Class
Model
Zip File


Editor - C:\Users\nick\Documents\MATLAB\course\sample_script.m
+38
sample_script.m
1 A = [1 1; 1 -1];
2 b = [3; 1];
3
4 x = linsolve(A, b);
5
6 for i = 1:length(x)
7 fprintf('x%d = %.2f\n', i, x(i));
8 end


>> sample_script
x1 = 2.00
x2 = 1.00
>> x
x =
2
1
>>


Workspace
Name
Value
A [1,1;1,-1]
b [3;1]
i 2
x [2;1]

Συναρτήσεις

- Εκτός από scripts, το MATLAB παρέχει τη **δυνατότητα ορισμού συναρτήσεων.**
- **Για κάθε συνάρτηση** δημιουργούμε ένα **ξεχωριστό αρχείο** .m, όπως και στα scripts.
- Οι συναρτήσεις μπορούν να δέχονται και να επιστρέφουν κανένα, ένα, ή περισσότερα ορίσματα/τιμές.

```
1 function [ x_squared, x_abs ] = myfun( x )
2 %MYFUN Summary of this function goes here
3 %   Detailed explanation goes here
4
5 x_squared = x * x;
6 x_abs = abs(x);
7 end
```

  - Σχόλια μίας γραμμής: %,
  - Σχόλια πολλών γραμμών: %{, }%

Συναρτήσεις

- Χρησιμοποιούμε τις συναρτήσεις που ορίσαμε, όπως τις built-in συναρτήσεις του MATLAB.
- Οι **μεταβλητές** που ορίζονται **εντός των συναρτήσεων δεν είναι ορατές έξω από αυτές**.
- Για να λάβουμε παραπάνω από μία τιμή εξόδου χρησιμοποιούμε ένα διάνυσμα-λίστα με τις μεταβλητές όπου θέλουμε να τοποθετηθούν τα αποτελέσματα.
- Αν δεν οριστεί κάποια λίστα, επιστρέφεται μόνο η πρώτη τιμή.

```
1 >> res = myfun( 5 )
2 res = 25
3 >> [res1, res2] = myfun( 5 )
4 res1 = 25
5 res2 = 5
6 >> myfun( 5 )
7 ans = 25
```

Συναρτήσεις

- Οι συναρτήσεις μπορούν επίσης να **λαμβάνουν και να επιστρέφουν πίνακες**.
- Προσοχή χρειάζεται, όταν οι πράξεις που εκτελούμε δεν ορίζονται σε πίνακες:

```
1 >> x= [1  2  3]
2 x =
3 1      2      3
4 >> myfun(x)
5 Error using  *
6 Inner matrix dimensions must agree.
7
8 Error in myfun (line 5)
9 x_squared = x * x;
```

- Γιατί συμβαίνει αυτό;

Συναρτήσεις

```
1 function [ x_squared, x_abs ] = myfun( x )
2 %MYFUN Summary of this function goes here
3 %   Detailed explanation goes here
4
5 x_squared = x .* x;
6 x_abs = abs(x);
7
8 end
```

```
1 >> x = [1 2 3];
2 >> [y1, y2] = myfun(x)
3 y1 = 1     4     9
4 y2 = 1     2     3
```

Built-in variables

- Το MATLAB χρησιμοποιεί κάποιες μεταβλητές/συναρτήσεις **για ειδικούς σκοπούς**:
  - *i*, *j*: για να συμβολίσει μιγαδικούς
  - pi: η σταθερή π
  - Inf: άπειρο
  - NaN: Not a Number (προκύπτει όταν το αποτέλεσμα μιας πράξης είναι απροσδιόριστο)
- Καλό είναι **να μην χρησιμοποιούμε αυτά τα ονόματα ως μεταβλητές** (το MATLAB όμως δεν το απαγορεύει)

Άλλες χρήσιμες συναρτήσεις

- Επανάληψη πίνακα: *repmat()*
- Διαγραφή όλων των μεταβλητών του workspace: *clear*
- *linspace(a,b,N)*: Δημιουργεί N σημεία που ισαπέχουν μεταξύ των σημείων *a* και *b*

```
>> linspace(1, 5, 10)
ans = 1.0000    1.4444    1.8889    2.3333
      2.7778    3.2222    3.6667    4.1111
      4.5556    5.0000
```

Άλλες χρήσιμες συναρτήσεις

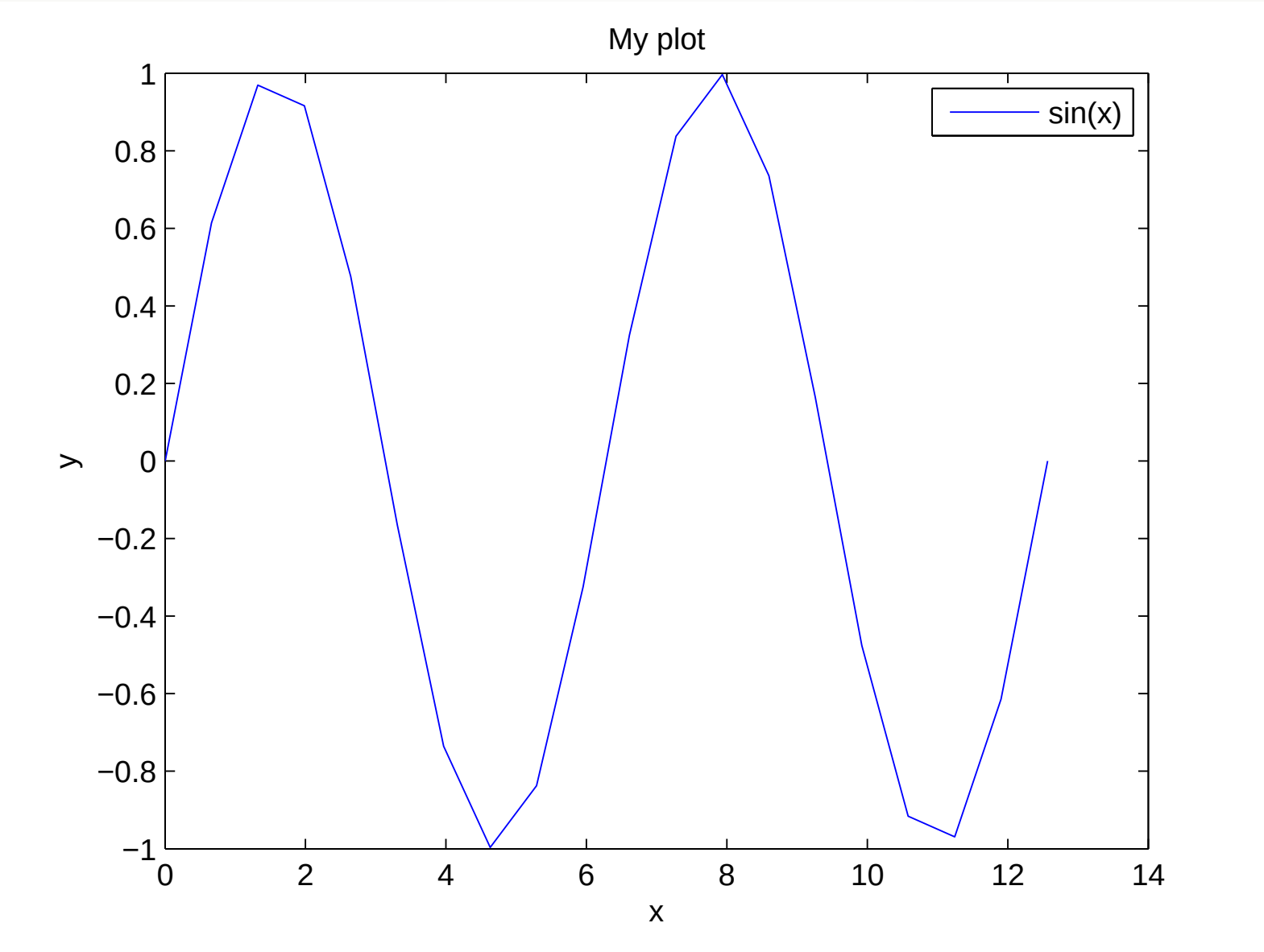
- Λογάριθμοι: *log(), log2(), log10()*
- Τυχαίες τιμές: *rand(), randn(), randi(), randperm(), rng()*
- Διάφορες βοηθητικές συναρτήσεις: *sum(), cumsum(), linspace(), logspace(), mean(), median(), isinf(), isnan(), ...*
  - *...bsxfun(), arrayfun(), cell arrays!*
- Strings: *strcat(), sprintf(), ischar(), strcmp(), strcmpi(), regex(), strfind(), ...*
  - Τα strings αναπαριστώνται ως πίνακες χαρακτήρων!

Plotting

- Το MATLAB παρέχει πάρα πολλές δυνατότητες για **τη δημιουργία γραφικών παραστάσεων.**
- Η πιο απλή μορφή γραφικής παράστασης δημιουργείται με τη συνάρτηση *plot(x,y)* η οποία δέχεται ένα διάνυσμα με τις τιμές του άξονα x και τις αντίστοιχες τιμές στον άξονα y.

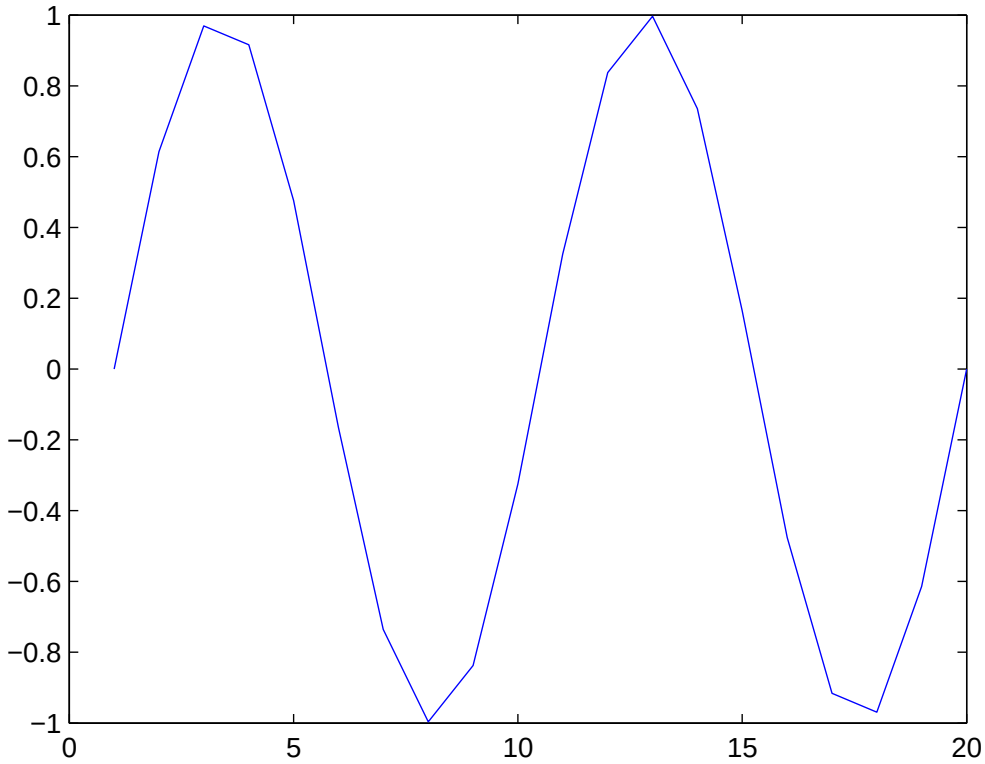
```
1 >> x=linspace(0, 4*pi, 20);
2 >> y=sin(x);
3 >> plot(x,y)
4 >> title('My plot');
5 >> xlabel('x');
6 >> ylabel('y');
7 >> legend('sin(x)');
```

Plotting



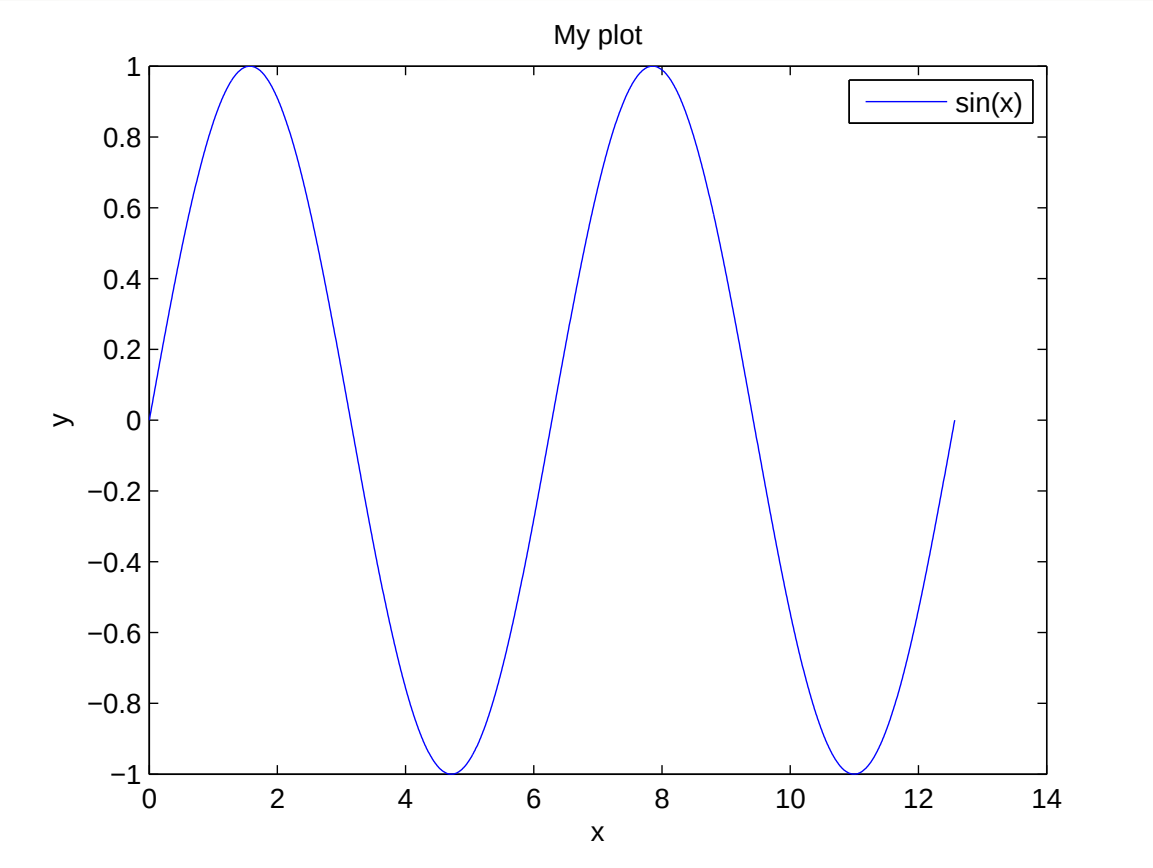
Plotting

- Τι συμβαίνει αν δεν ορίσω τιμές στον άξονα x (δηλαδή καλέσω την *(plot(y))*;
- Το MATLAB υποθέτει ότι αυτές είναι οι διαδοχικοί ακέραιοι ξεκινώτας απο το 1.



Plotting

- Πώς θα γίνει το γράφημα πιο ομαλό?
- Προσθέτω παραπάνω σημεία! *x=linspace(0, 4*pi, 1000);*



Προτεραιότητα πράξεων

1. Parentheses ()
2. Transpose (.'), power (.^), complex conjugate transpose ('), matrix power (^)
3. Unary plus (+), unary minus (-), logical negation (~)
4. Multiplication (.*), right division (./), left division (.\), matrix multiplication (*), matrix right division (/), matrix left division (\)
5. Addition (+), subtraction (-)
6. Colon operator (:)
7. Less than (<), less than or equal to (<=), greater than (>), greater than or equal to (>=), equal to (==), not equal to (~=)
8. Element-wise AND (&)
9. Element-wise OR (|)
10. Short-circuit AND (&&), 11. Short-circuit OR (||)

Συχνά λάθη - Συμβουλές

- Δεν επιτρέπεται ο "σιωπηρός" (implicit) πολλαπλασιασμος!
  - Πρέπει πάντα να υπάρχει το σύμβολο *

```
>> x = 2;
>> y = 4x;
y = 4x;
      |
Error: Unexpected MATLAB expression.
>> y = 4*x;
```

- Όταν προκύπτουν σφάλματα με διαστάσεις (π.χ. πολλαπλασιασμός, κτλ,) μπορείτε να τυπώνετε τις διαστάσεις με τη συνάρτηση *size()*, ώστε να διαπιστώσετε πού είναι το σφάλμα.

Συχνά λάθη - Συμβουλές

- Βεβαιωθείτε ότι **γνωρίζετε πώς συμπεριφέρονται οι συναρτήσεις** που χρησιμοποιείτε!
- Στη *sum()* θα μπορούσε κάποιος να υποθέσει ότι αυτή αθροίζει *πάντα τις στήλες*. Όμως αυτό δεν ισχύει!

```
>> sum([1 2 3; 4 5 6])
ans = 5     7     9
>> sum([1 2 3])
ans = 6
>> sum([1 2 3], 1)
ans = 1     2     3
```

- Το δεύτερο όρισμα δηλώνει τον άξονα της άθροισης (στήλες ή γραμμές)

Άσκηση

- Να γραφεί συνάρτηση που θα δέχεται δύο αριθμούς N και m και θα:
  1. κατασκευάζει έναν τυχαίο τετραγωνικό πίνακα A διάστασης $M \times M$
  2. μηδενίζει τυχαία το $m\%$ των τιμών του A
  3. προσθέτει στον A τον μοναδιαίο πίνακα $M \times M$
  4. κατασκευάζει ένα διάνυσμα b διάστασης $M \times 1$
  5. επιλύει το γραμμικό σύστημα $Ax = b$ και επιστρέφει το διάνυσμα x

Άσκηση

- Λύση 1:

```
1  function [ x ] = exercise1( N, m  )
2
3  A = rand(N, N);
4  for k = 1:N
5      for l = 1:N
6          if rand(1) <= (m/100)
7              A(k, l) = 0;
8          end
9      end
10 end
11 A = A + eye(size(A));
12 b = rand(N,1);
13 x = linsolve(A,b);
14 end
```

Άσκηση

- Λύση 2:

```
1  function [ x ] = exercise1( N, m  )
2
3  A = rand(N, N);
4
5  M = rand(N,N) <= (m/100);
6  %Logical Indexing
7  A(M) = 0;
8
9  A = A + eye(size(A));
10 b = rand(N,1);
11 x = linsolve(A,b);
12
13 end
```

Extra: Το MATLAB είναι αργό...

- Χμ...
- Ας υλοποιήσουμε τον πολλαπλασιασμό πίνακα με δομή for!

Extra: Το MATLAB είναι αργό...

- Χμ...
- Ας υλοποιήσουμε τον πολλαπλασιασμό πίνακα με δομή for!

```
1 C = zeros(size(A,1), size(B,2));
2 tic;
3 for i = 1:size(A,1)
4     for j = 1:size(B,2)
5         for k = 1:size(A,2)
6             C(i,j) = C(i,j) + A(i,k)*B(k,j);
7         end
8     end
9 end
```

Extra: Το MATLAB είναι αργό...

- Μία εναλλακτική υλοποίηση:

Extra: Το MATLAB είναι αργό...

- Μία εναλλακτική υλοποίηση:

```
1 C = zeros(size(A,1), size(B,2));
2 for i = 1:size(A,1)
3       for j = 1:size(B,2)
4           C(i,j) = A(i,:)*B(:,j);
5       end
6 end
```

Extra: Το MATLAB είναι αργό...

- Μία εναλλακτική υλοποίηση:

```
C = zeros(size(A,1), size(B,2));
for i = 1:size(A,1)
    for j = 1:size(B,2)
        C(i,j) = A(i,:)*B(:,j);
    end
end
```

- Και μία ακόμη!

```
C = A*B
```

- Ποιά από τις τρεις τρέχει γρηγορότερα?

Extra: Το MATLAB είναι αργό...

```
>> multiplymat( 100, 100, 100 );
Implementation 1: 0.00786 sec
Implementation 2: 0.01041 sec
Implementation 3: 0.00014 sec
>> multiplymat( 1000, 1000, 1000 );
Implementation 1: 7.56350 sec
Implementation 2: 6.01486 sec
Implementation 3: 0.01582 sec
>> multiplymat( 1000, 5000, 1000 );
Implementation 1: 58.20716 sec
Implementation 2: 55.02268 sec
Implementation 3: 0.07013 sec
```

Συμβουλές

- Καλύψαμε λιγότερο από το 1% των δυνατοτήτων του MATLAB.
- Παρόλα αυτά, αναφέρθηκαν οι περισσότερες έννοιες που πρέπει να γνωρίζετε για να αρχίσετε να γράφετε προγράμματα σε αυτό.
- Όπως και στις υπόλοιπες γλώσσες προγραμματισμού, ο μόνος τρόπος να τις μάθετε είναι να γράψετε κώδικα.
  - Εκμεταλευτείτε τον interactive χαρακτήρα του MATLAB! Πειραματιστείτε με τις εντολές!
- Δεν χρειάζεται να μάθετε εξ' αρχής όλες τις δυνατότητες του. Αρκεί να κατανοήσετε τη λογική.
  - Όταν χρειάζεστε κάτι εξειδικευμένο θα ανατρέχετε στην τεκμηρίωση.
  - Με τον καιρό θα μάθετε όλες τις δυνατότητες που σας χρειάζονται και θα ανατρέχετε όλο και λιγότερο στη βοήθεια.

Πηγές

- MATLAB Documentation - MathWorks
- Introduction to MATLAB - MathWorks
- Introduction to Programming in MATLAB, MIT 6.094, Danilo Šćepanović
- MATLAB - Εισαγωγικά στοιχεία, Αριθμητική Ανάλυση, Σωτήρης Καραβαρσάμης
- Internet ...