Αριθμητική Επίλυση Συνήθων Διαφορίκών Εξισώσεων
1ο Εργαστήριο

26/02/2015

1 Variables

Μεταβλητές (Variables), μπορούν να ονοματοδοτηθούν χρησιμοποιώντας κεφαλαίους ή πεζούς λατινικούς χαρακτήρες σε συνδυασμό με αριθμούς. Αποδεκτά ονόματα μπορεί να έχουν τη μορφή:

`NetCost, Left2Pay, x3, X3, z25c5`

Δεν επιτρέπεται να δίνουμε ονόματα τα οποία περιέχουν ειδικούς χαρακτήρες ή μεταβλητές που ξεκινάνε με αριθμό. Για παράδειγμα μή αποδεκτά ονόματα μεταβλητών είναι:

`Net-Cost, 2pay, %x, *sign`

Επιπλέον δεν πρέπει να χρησιμοποιηθούν ονόματα τα οποία χρησιμοποιούνται από την ίδια τη Matlab όπως για παράδειγμα το $pi = 3.14159\ldots \simeq \pi$.

```
>> x=-13; y = 5*x, z = x^2+y
y=
     -65
z=
     104
>>
```

Η τιμή του x δεν εμφανίζεται διότι μετά την ανάθεση τιμής ακολουθεί το ερωτηματικό.

2 Διανύσματα

Στη Matlab η κατασκευή διανυσμάτων γίνεται πάρα πολύ εύκολα όπως φαίνεται στο επόμενο παράδειγμα:

```
>> v=[1 3, sqrt(5)]
v =
    1.0000    3.0000    2.2361
>> length(v)
ans =
     3
>>
```

Παρατηρούμε ότι μπορούμε να χωρίσουμε τις τιμές είτε με κενό είτε με κόμμα και η συνάρτηση `length` επιστρέφει το αριθμό των θέσεων (διάσταση) του διανύσματος. Ένας εναλακτικός τρόπος για να φτιάξουμε διανύσματα αποτελούμενα από ισαπέχοντα σημεία είναι ο εξής:
Ορίζω $v = \alpha : \beta : \gamma$ όπου α είναι η τιμή εκκίνησης, γ είναι η τιμή τερματισμού και β είναι το βήμα.

```
>> v=1:2:9
v =
     1     3     5     7     9
>> v=0.32:0.1:0.6
v =
    0.3200    0.4200    0.5200
>>
```

Τέλος η κατασκευή διανύσματος στήλη μπορεί να γίνει είτε με την χρήση ερωτηματικού για τον διαχωρισμό των τιμών είτε αναστρέφοντας ένα διανυσμα γραμμή χρησιμοποιώντας τον τόνο (′).

```
>> v=[1;2;3]
v =
     1
     2
     3
>> w=[4 5 6]'
w =
     4
     5
     6
>>
```

3 Πράξεις με διανύσματα

Μπορούμε να κάνουμε κάποιες πράξεις όπως για παράδειγμα την πρόσθεση και αφαίρεση διανυσμάτων όπως και τον πολλαπλασιασμό διανύσματος με βαθμωτό μέγεθος. Προϋπόθεση για να μπορεί να γίνει η πρόσθεση και αφαίρεση διανυσμάτων είναι τα διανύσματα να έχουν την ίδια διάσταση.

```
v2=[3 4 5]
v2 =
     3     4     5
>> v3=[7 4]
v3 =
     7     4
>> v+v2
Error using  +
Matrix dimensions must agree.
>> v'+v2
ans =
    4.0000    7.0000    7.2361
>> 3*v3
ans =
    21 12
```

3.1 Εσωτερικό γινόμενο

Έστω δύο διανύσματα $\vec{v}$ και $\vec{u}$, με μηκος n. Το εσωτερικό γίνόμενο ορίζεται ως η ποσότητα

$$\vec{v} \cdot \vec{u} = \sum_{i=1}^{n} v_i u_i$$

Στην περίπτωσή μας, έχοντας το διάνυσμα v να είναι στήλη και το v2 να είναι γραμμή, μπορούμε να υπολογίσουμε το εσωτερικό τους γινόμενο κάνοντας τον πολλαπλασιασμό τους, ως:

```
>> v2*v
ans =
    26
```

Η σειρά με την οποία πολλαπλασιάζουμε είναι σημαντική, μιας και αν κάναμε τον πολλαπλασιασμό ως `v*v2` θα καταλήγαμε με έναν 3×3 πίνακα, αφού το v ως διάνυσμα στήλη έχει διάσταση 3×1, ενώ το `v2` διασταση 1×3

```
>> v*v2
ans =
     3     4     5
     6     8    10
     9    12    15
```

3.2 Dot Product (.*)

Ένας εναλλακτικός τρόπος να ορίσουμε το γινόμενο δύο διανυσμάτων είναι με τον τελεστή .* ο οποίος πολλαπλασιάζει δύο διανύσματα ιδίου μεγέθους ανα στοιχείο. Για παράδειγμα, αυτή η πράξη μεταξύ των v και w θα μας έδινε

$$\texttt{v.*w} = [v_1 w_1, v_2 w_2, v_3 w_3]$$

και συγκεκριμένα

```
>> v.*w
ans =
     4
    10
    18
```

Αντίστοιχα, μπορούμε να διαιρέσουμε διανύσματα ανά στοιχείο, ή να υψώσουμε στοιχεία ενός διανύσματος σε κάποια δύναμη:

```
>> v./w
ans =
    0.2500
    0.4000
    0.5000

>> v.^3
ans =
     1
     8
    27
```

4 Εσωτερικές συναρτήσεις της Matlab.

Στις εσωτερικές συναρτήσεις της Matlab συμπεριλαμβάνονται οι τριγωνομετρικές συναρτήσεις $\sin, \cos$ κ.α. καθώς και άλλες συναρτήσεις που χρησιμοποιούνται ευρέως όπως για παράδειγμα οι `sqrt, log, exp` κ.α.. Μερικά παραδείγματα χρήσης είναι:

```
>> x=9;
>> sqrt(x)
ans =
     3
>> sin(pi/6)
ans =
    0.5000
>>
```

Οι εσωτερικές συναρτήσεις μπορούν επίσης να εφαρμοστούν σε διανύσματα (αλλά και σε πίνακες όπως θα δούμε αργότερα), με τον ίδιο ακριβώς τρόπο. Μπορούμε για παράδειγμα πάρουμε την τετραγωνική ρίζα κάθε στοιχείου του v, χρησιμοποιώντας την εντολή `sqrt(v)` ή να υπολογίσουμε το ημίτονο σε κάθε στοιχείο του με το `sin(v)`:

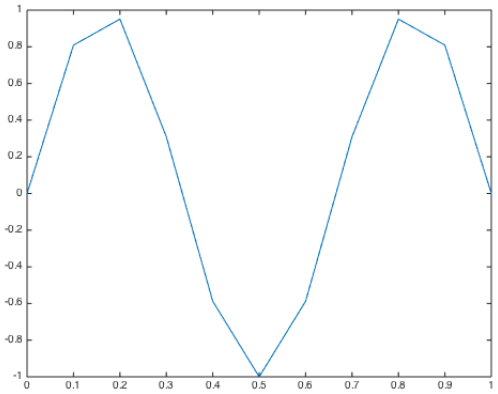
```
>> sqrt(v)
ans =
    1.0000
    1.4142
    1.7321

>> sin(v)
ans =
    0.8415
    0.9093
    0.1411
```

5 Plots

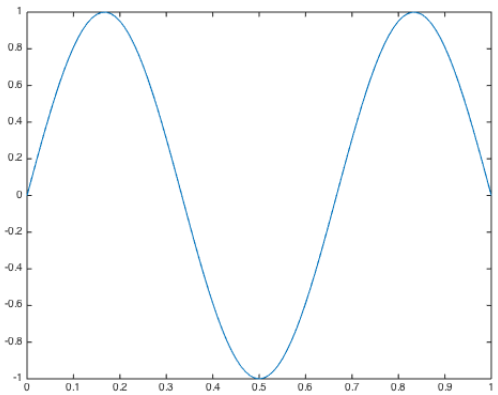
Με την Matlab μπορούμε να φτιάξουμε γραφικές παραστάσεις πάρα πολύ εύκολα, για παράδειγμα οι εντολές:

```
>> x=0:0.1:1;
>> y=sin(3*pi*x);
>> plot(x,y)
>>
```

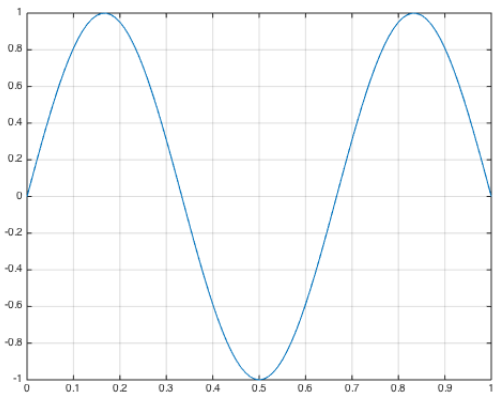
θα μας δώσουν:



Εαν μειώσουμε το βήμα θα πάρουμε πολύ αντιπροσωπευτικότερο σχήμα. Για παράδειγμα όταν το βήμα γίνει 0.01 τότε θα πάρω:



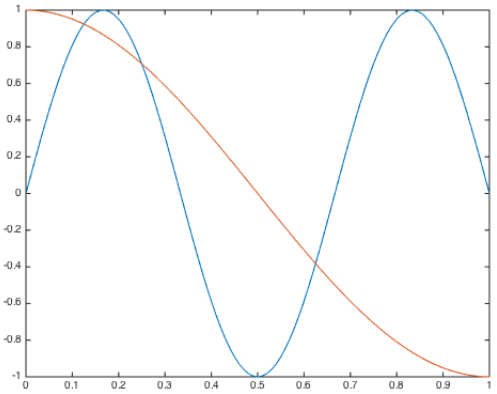
Ενώ μπορούμε να προσθέσουμε και grid πληκτρολογώντας την εντολή *grid*:



Μπορούμε να αλλάξουμε κάποια γραφική παράσταση με πολλούς τρόπους. Με τις εντολές **help plot** και **doc plot** μπορείτε να πάρετε περισσότερες πληροφορίες για το πώς μπορείτε να το κάνετε αυτό.

Με την εντολή **hold on** μπορείτε στο ίδιο γράφημα να τυπώσετε και άλλη γραφική παράσταση διατηρώντας την παλιά. Με την επιλογή **'r'**, επιλέγουμε η γραφική παράσταση του **z** να έχει κόκκινο χρώμα.

```
>> plot(x,y)
>> z=cos(pi*x);
>> hold on
>> plot(x,z,'r')
>>
```



6 Πίνακες

Τα διανύσματα γραμμής και στήλης είναι ειδικές περιπτώσεις πινάκων, όπου έχουν διάσταση $1 \times n$ και $n \times 1$ αντίστοιχα. Για να κατασκευάσουμε έναν πίνακα, χρησιμοποιούμε την ίδια μέθοδο με την εισαγωγή ενός διανύσματος, όπου εισάγουμε καινούρια γραμμή στον πίνακα αλλάζοντας γραμμή κατά την εισαγωγή των στοιχείων:

```
>> A = [5 7 9
        1 -3 -7]
A =
     5     7     9
     1    -3    -7
>> size(A)
ans =
     2     3
```

Ο διαχωρισμός των γραμμών μπορεί να γίνει επίσης με ερωτηματικό, όπως στο παράδειγμα που ακολουθεί:

```
>> B = [-1 2 5; 9 0 5]
B =
    -1     2     5
```

```
4        9     0     5
5  >> size(B)
6   ans =
7        2     3
```

και η συνάρτηση `size` επιστρέφει τις διαστάσεις του πίνακα.

6.1 Κατασκευή ειδικών πινάκων

Η αναστροφή ενός διανύσματος, το μετατρέπει από διάνυσμα γραμμή σε διάνυσμα στήλη και αντίστροφα, (βλ. §3). Ο ανάστροφος πίνακας ορίζεται με τον ίδιο τρόπο (′) και εναλλάσει τις γραμμές και στήλες του πίνακα.

```
1  >> D = [1:5; 6:10; 11:2:20]
2  D =
3        1     2     3     4     5
4        6     7     8     9    10
5       11    13    15    17    19
6  >> D'
7  ans =
8        1     6    11
9        2     7    13
10       3     8    15
11       4     9    17
12       5    10    19
```

Επιπλέον η Matlab παρέχει συναρτήσεις που με ύκολο τρόπο μπορούν να φτιάξουν πίνακες ειδικής μορφής.

`ones(m,n)`: επιστρέφει ένα $m \times n$ πίνακα με άσους παντού:

```
1  >> P = ones(2,3)
2  P =
3        1     1     1
4        1     1     1
```

`zeros(m,n)`: επιστρέφει ένα $m \times n$ πίνακα με μηδέν παντού:

```
1  >> Z = zeros(2,3)
2  Z =
3        0     0     0
4        0     0     0
```

`eye(m,n)`: επιστρέφει ένα $m \times n$ πίνακα με άσους στη διαγώνιο και μηδέν παντού αλλού:

```
1  >> K=eye(3,4)
2  K =
3        1     0     0     0
4        0     1     0     0
5        0     0     1     0
```

Σε περίπτωση που θέλουμε να δημιουργήσουμε έναν τετραγωνικό $n \times n$ πίνακα, τότε μπορούμε να δώσουμε ως όρισμα μόνο μία φορα τη διάστασή του. Π.χ., η εντολή `zeros(3,3)` και η εντολή `zeros(3)` θα μας δώσουν το ίδιο αποτέλεσμα.

`diag(v)`: επιστρέφει ένα διαγώνιο πίνακα με με τα στοιχεία του διανύσματος `v` στη διαγώνιο:

```
1  >> v=[1 2 -5];
2  >> P=diag(v)
3  P =
4        1     0     0
5        0     2     0
6        0     0    -5
```

Αντίστροφα το `diag(M)` επιστρέφει ένα διανυσμα με με τα στοιχεία της διαγωνίου του πίνακα `M` :

```
1  >> M = [0 1 8 7; 3 -2 -4 2; 4 2 1 1]
2  M =
3        0     1     8     7
4        3    -2    -4     2
5        4     2     1     1
6  >> v=diag(M)
7  v =
8        0
9       -2
10       1
```

6.2 Πράξεις πινάκων

6.2.1 Dot product (.*)

Πρόκειται για γινόμενο κατα στοιχείο δύο πινάκων και όχι για τον κλασσικό πολλαπλασιασμό πινάκων. Η μόνη προϋπόθεση είναι οι πίνακες να έχουν τις ίδιες διαστάσεις.

```
1  >> A, B
2  A =
3       5     7     9
4       1    -3    -7
5  B =
6      -1     2     5
7       9     0     5
8  >> A.*B
9  ans =
10         -5    14    45
11          9     0   -35
12 >> A.*C
13 ??? Error using ==> .*
14 Matrix dimensions must agree.
15 >> A.*C'
16 ans =
17         0    21    36
18         1     6   -14
```

Αντίστοιχα με τα διανύσματα, τα αποτελέσματα που παίρνουμε για την κατά στοιχείο διαίρεση και ύψωση σε δύναμη είναι παρόμοια και υπακούν τους ίδιους κανόνες.

6.2.2 Πολλαπλασιασμός πίνακα με πίνακα και πίνακα με διάνυσμα (*)

Ισχύουν φυσικά και σε αυτή τη περίπτωση οι κανόνες που γνωρίζουμε για τις διαστάσεις για τις οποίες είναι δυνατόν να πραγματοποιηθεί ένα γινόμενο.

```
1  >> A = [5 7 9; 1 -3 -7]
```

```
A =
     5     7     9
     1    -3    -7
>> x = [8; -4; 1]
x =
     8
    -4
     1
>> A*x
ans =
    21
    13
```

```
>> A=[5 7 9; 1 -3 -7];
>> B = [0, 1; 3, -2; 4, 2];
>> C = A*B
C =
    57     9
   -37    -7

>>D = B*A
D =
     1    -3    -7
    13    27    41
    22    22    22

>>E=B*A
E =
     1    -3    -7
    13    27    41
    22    22    22
```

7 Δομή ελέγχου (if)

Η Matlab αναπαριστά τα `true` και `false` μέσω των ακεραίων 0 και 1, όπου

`true = 1,  false = 0`

Άν σε κάποιο σημείο έχουμε αναθέσει κάποια τιμή στην μεταβλητή x, τότε μπορούμε να κάνουμε ελέγχους σε αυτό, όπως

x == 2 είναι το x ίσο με 2;
x = 2 **δεν** είναι το x ίσο με 2;
x > 2 είναι το x μεγαλύτερο από 2;
x < 2 είναι το x μικρότερο από 2;
x >= 2 είναι το x μεγαλύτερο από ή ίσο με 2;
x <= 2 είναι το x μικρότερο από ή ίσο με 2;

Ιδιαίτερη προσοχή πρέπει να δωθεί στο γεγονός ότι ο έλεγχος για την ισότητα απαιτεί δύο σύμβολα ισότητας ==.

```
>> x = pi;

>> x ≠3
ans =
     1
>> x ≠pi
ans =
     0
```

Αντίστοιχοι έλεγχοι μπορούν να γίνουν και σε διανύσματα, όπου πλέον το αποτέλεσμα του ελέγχου θα είναι και αυτό διάνυσμα. Για παράδειγμα,

```
>> x = [-2 3 5 6 2 4];
>> x>2
ans =
     0     1     1     1     0     1

>> x>2 & x<5
ans =
     0     1     0     0     0     1
```

8 Επαναληπτικές δομές

8.1 for

Σε αρκετές περιπτώσεις, χρειάζεται να επαναλάβουμε ένα τμήμα κώδικα αρκετές φορές, όπως για παράδειγμα να φτιάξουμε τα στοιχεία μίας ακολουθίας. Ένα κλασσικό παράδειγμα είναι η δημιουργία της ακολουθίας *Fibonnaci*, όπου οι αρχικοί αριθμοί της είναι το 0 και το 1 και οι επόμενοι όροι είναι το άθρισμα των δύο αμέσως προηγούμενων όρων. Μαθηματικά, έχουμε ότι $f_1 = 0$, $f_2 = 1$ και

$$f_n = f_{n-1} + f_{n-2}, \quad n = 3, 4, 5, \ldots$$

Η υλοποίηση των πρώτων 10 όρων της ακολουθίας στη Matlab θα γινόταν ως εξής:

```
>> f(1) = 0;
>> f(2) = 1;
>> for i = 3:10
f(i) = f(i-1) + f(i-2);
end
>> f
f =
     0     1     1     2     3     ...
           5     8    13    21    34
```

8.2 while

Υπάρχουν όμως και περπτώσεις στις οποίες δεν ξέρουμε εξάρχής το πόσες επαναλήψεις θα χρειαστούμε μέχρι να φτάσουμε στο επιθυμητό αποτέλεσμα, αλλά το κριτήριο τερματισμού της επανάληψης είναι κάποιος λογικός έλεγχος.
Για παράδειγμα, ως ε της μηχανής, ορίζουμε τον μεγαλύτερο αριθμό ο οποίος έχει την ιδιότητα πως αν προστεθεί στο 1, δεν θα έχει κάποια επίδραση, δηλαδή (στον υπολογιστή) $1+\varepsilon = 1$. Παρόλο που η Matlab έχει ήδη αποθηκευμένο στην μεταβλητή `eps` αυτόν τον αριθμό, αν θέλαμε να τον προσεγγίσουμε, τότε θα μπορούσαμε να το κάνουμε ως

```
>> me = 1;
>> while (1+me>1)
me = me/2;
end
>> me
me =
    1.1102e-16
```

```
>> eps
ans =
    2.2204e-16
```

Παρατηρούμε ότι η προσέγγισή μας είναι υποδιπλάσια του ε (κάτι που μπορούμε να επαληθεύσουμε κάνοντας τον έλεγχο `2*me==eps`) , λόγω της δομής του ελέγχου μας.

9 Scripts και Συναρτήσεις σε M-files

Σε περίπτωση που έχουμε μια σειρά από εντολές, τις οποίες χρειάζεται να επαναλάβουμε λόγω κάποιου λάθους ή αλλαγή στις επιλογές μας, τότε είναι πιο βολικη προς το χρήστη η δημιουργία ενός αρχείου με κατάληξη `.m`, στο οποίο θα ¨περαστούν' οι εντολές και θα μπορούν να εκτελούνται αυτόματα

9.1 Scripts

Τα Scripts είναι μία ειδική κατηγορία m-files, τα οποία εκτελούν μία σειρά εντολών και εκτελούνται χρησιμοποιώντας μόνο το όνομα του αρχείου. Για παράδειγμα, αν θέλαμε να περάσουμε τις εντολές της παραγράφου 5, όπου κάναμε την γραφική παράσταση ενός ημιτόνου, σε ένα αρχείο (ας το ονομάσουμε `sinplot.m`), τότε θα έπρεπε να γράψουμε στην γραμμή εντολών

```
edit sinplot.m
```

και αυτό θα μας εμφάνιζε ένα καινούριο παράθυρο ενός επεξεργαστή κειμένου, στο οποίο θα περνούσαμε τις εντολές

```
x=0:0.1:1;
y=sin(3*pi*x);
plot(x,y)
```

Για να εκτελέσουμε τώρα το αρχείο μας (αφού το αποθηκεύσουμε), γυρναμε στην γραμμή εντολών και πληκτρολογούμε

```
>>sinplot
```

όπου θα πρέπει να προσέξουμε ότι δεν συμπεριλαμβάνουμε την κατάληξη `.m` κατά την εκτέλεση.

Πλέον, αν δεν είμαστε ευχαριστημένοι το βήμα της διακριτοποίησης του `x`, τότε μπορούμε απλά να πάμε στην πρώτη γραμμή του αρχείου `sinplot.m`), να αλλάξουμε το βήμα, να αποθηκεύσουμε το αρχείο και απλά να ξανατρέξουμε το script όπως προηγουμένως.

9.2 Συναρτήσεις

Οι συναρτήσεις σε m-files, είναι ένας συνδυασμός των Scripts και των μαθηματικών συναρτήσεων. Ένα παράδειγμα είναι ο υπολογισμός του εμβαδού ενός κύκλου δεδομένης της ακτίνας του.

Τα κυρίως βήματα για την δημιουργία μίας συνάρτησης στη Matlab είναι

1. Ορίζουμε ένα όνομα για την συνάρτηση, το οποίο δεν είναι ήδη δεσμευμένο απο την Matlab, π.χ. plot, hold, κ.λ.π. Στην περίπτωσή μας το όνομα της συνάρτησης είναι areac, οπότε θα δημιουργήσουμε το αρχείο areac.m.

2. Η πρώτη γραμμή του αρχείου θα πρέπει να έχει τη μορφή:
   `function` [ορίσματα εξόδου]
   = όνομα_συνάρτησης(ορίσματα εισόδου)
   Στην περίπτωσή μας έχουμε έξοδο το εμβαδόν `E`, ως συνάρτηση της ακτίνας `r`, οπότε η πρώτη γραμμή μας θα είναι

   ```
   function [E] = areac(r)
   ```

3. Τεκμηριώνω την συνάρτηση, περιγράφοντας σύντομα τη λειτουργία της και τον τρόπο με τον οποίο χρησιμοποιείται. Αυτές οι γραμμές θα πρέπει να προηγούνται απο το `%`, που υποδηλώνει ότι αυτές είναι γραμμές σχολίων και δεν εμπλέκονται στην εκτέλεση της συνάρτησης.

4. Τέλος, συμπεριλαμβάνουμε τον κώδικα που ορίζει την συνάρτηση

Ο ολοκήρωμένος κώδικας θα μοιάζει με

```
function [E] = areac(r)
% Computes the area of a circle of
radius r
% Input:  r (radius of the circle)
% Output:  E (area of the circle)
% Usage:  Emb = areac(10);
%
A = pi*r*r;
```

9.2.1 Συναρτήσεις στην γραμμή εντολών

Στην περίπτωση που θέλουμε να ορίσουμε μόνο μία μαθηματική συνάρτηση, είναι ευκολότερο να την ορίσουμε στην γραμμή εντολών, παρά να χρησιμοποιήσουμε ένα αρχείο για την αποθήκευσή της. Για παράδειγμα, αν θέλω να υπολογίσω την συνάρτηση

$$f(x) = \sin\left(\frac{\pi x}{2}\right) e^{2x+1},$$

μπορώ να την ορίσω στην γραμμή εντολών ως

```
f = @(x) sin(pi*x/2)*exp(2*x+1)
```

και να την χρησιμοποιήσω στην γραμμή εντολών, μετά την δήλωσή της

```
>> f = @(x) sin(pi*x/2)*exp(2*x+1)
f =
    @(x)sin(pi*x/2)*exp(2*x+1)

>> f(3)
ans =
  -1.0966e+03
```

Στην δήλωση της συνάρτηση, είναι αρκετά σημαντικός ο ορισμός της μεταβλητής, που στην περίπτωσή μας είναι το `x` και ορίζεται από το `@(x)`. Σε περίπτωση που είχαμε συνάρτηση δύο ή παραπάνω μεταβλητών, π.χ.

$$g(x, y) = \sin(x^2) + \cos^2(x),$$

τότε θα ορίζαμε την συνάρτηση ως

```
g = @(x,y) sin(x^2)+cos(x)^2
```

Τέλος, να σημειώσουμε ότι αυτές οι μορφές συναρτήσεων (που στη βιβλιογραφία αναφέρονται ως `inline` συναρτήσεις) μπορούν επίσης να συμπεριληφθούν και μέσα σε ένα m-files.