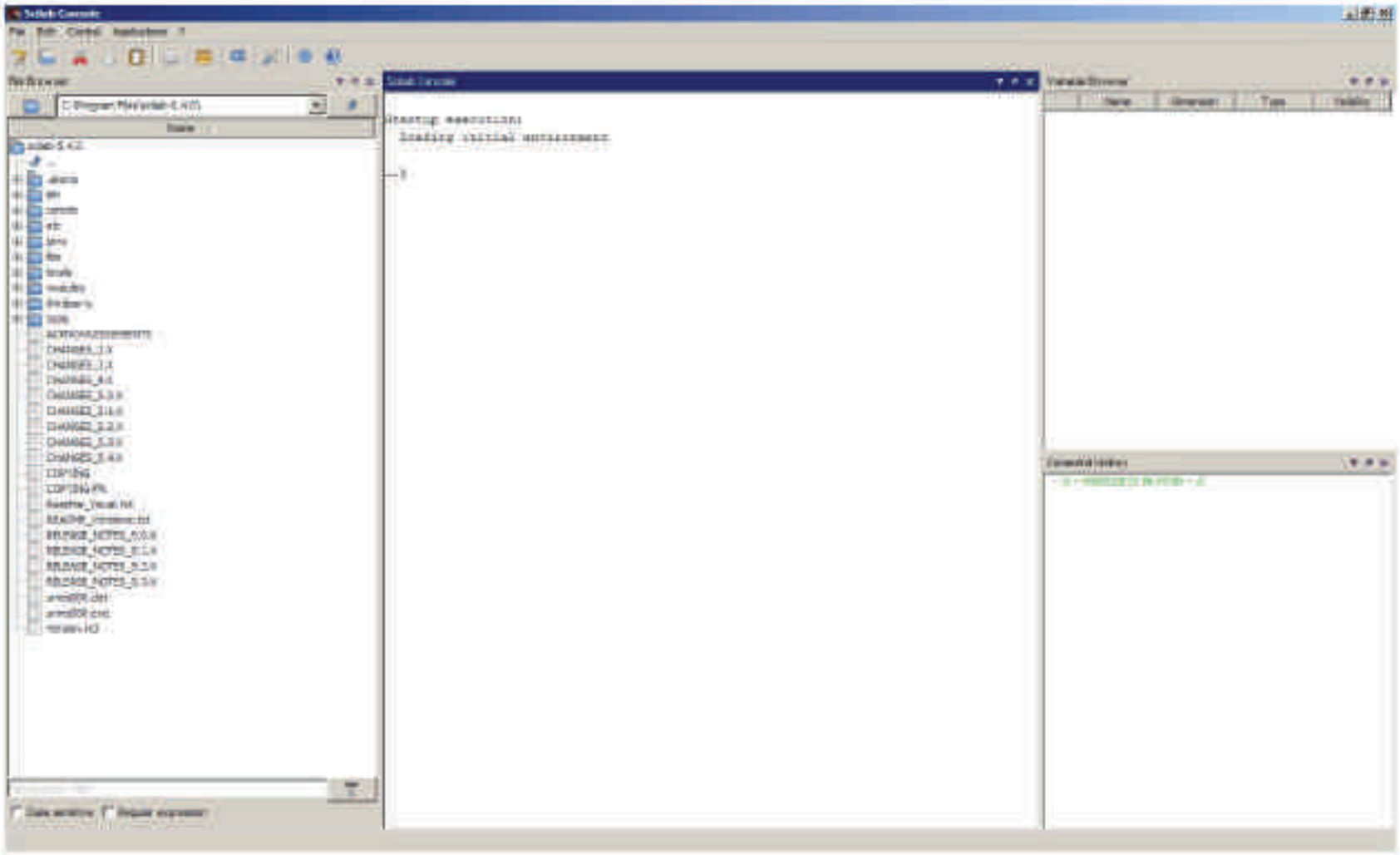
Εισαγωγή στο Scilab

Το γενικό περιβάλλον



Απλοί αριθμητικοί υπολογισμοί

- Ο συνήθεις αριθμητικές πράξεις πραγματοποιούνται με τα σύμβολα
- πρόσθεση ‘ + ’
- αφαίρεση ‘ - ’
- πολλαπλασιασμός ‘ * ‘
- Διαίρεση ‘ / ‘
- Ύψωση σε δύναμη ‘ ^ ‘

Παραδείγματα:

```
-->2+3.4
 ans  =
    5.4
```

```
-->sqrt(9)
 ans  =
    3.
```

```
-->SQRT(9)
          !--error 4
Variable non définie: SQ
```

```
--> %e
 %e  =
    2.7182818
```

```
--> %pi
 %pi  =
    3.1415927
```

```
--> 2+3*%i
 ans  =
    2. + 3.i
```

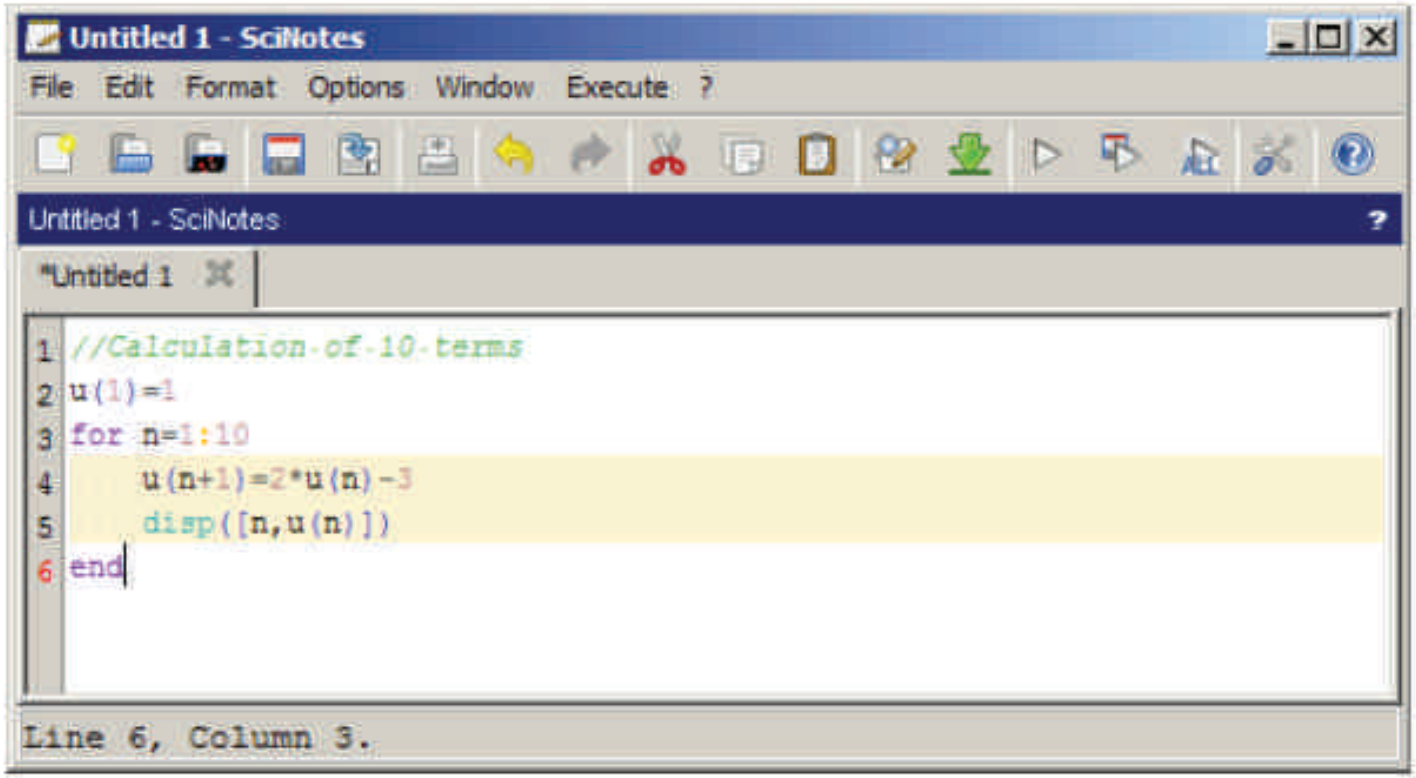
Εάν δεν θέλουμε να εμφανίζεται το αποτέλεσμα, τότε στο τέλος της εντολής βάζουμε ένα ερωτηματικό ‘ ; ’

```
-->(1+sqrt(5))/2;
```

```
--> (1+sqrt(5))/2
 ans  =
    1.618034
```

Ο ενσωματωμένος επεξεργαστής κειμένου SciNotes

Συνήθως για την συγγραφή προγραμμάτων στο Scilab χρησιμοποιούμε τον ενσωματωμένο επεξεργαστή κειμένου, οποίος μας διευκολύνει σημαντικά στο να έχουμε καλύτερη εποπτεία του κώδικα

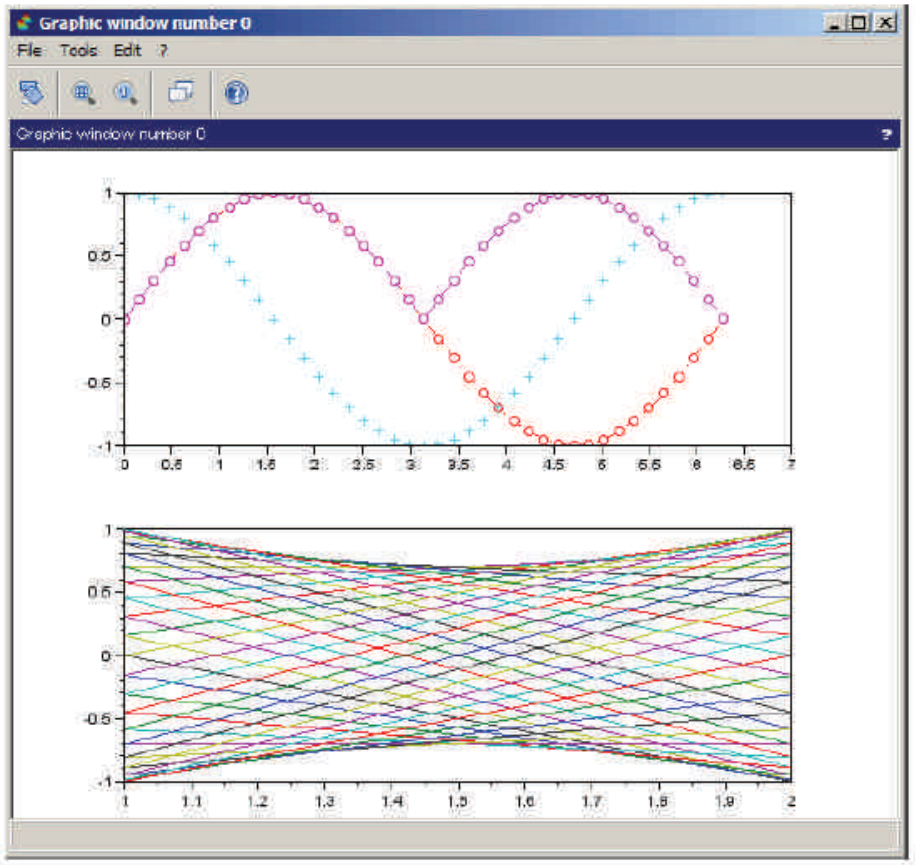


Τα προγράμματα αποθηκεύονται με την προέκταση .sce

Εισαγωγή στο Scilab

Το Scilab μας δίνει την δυνατότητα να απεικονίζουμε γραφικά τα αποτελέσματα των υπολογισμών μας

Ένα παράδειγμα γραφικής παράστασης μπορούμε να έχουμε εάν στην κονσόλα γράψουμε --> plot οπότε και εμφανίζεται ένα ενδεικτικό παράθυρο γραφικών



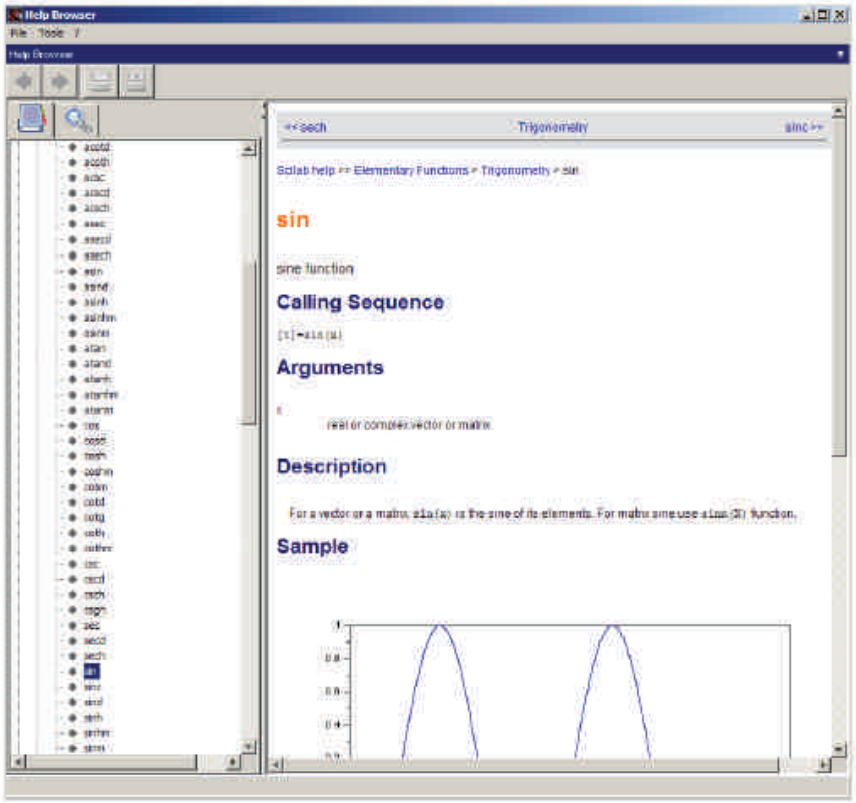
Παρατήρησεις

- Για να σβήσουμε ένα παλιό γράφημα δίνουμε την εντολή **clf** (clear figure)
- Για να ανοίξουμε ένα νέο παράθυρο γραφικών δίνουμε την εντολή **scf** (set current window)
- Εάν είναι ανοικτά πολλά παράθυρα γραφικών, μπορούμε να επιλέξουμε σε ποιο θα γίνει η γραφική μας παράσταση με την εντολή **scf(n)** όπου n είναι ο αριθμός του παραθύρου.

Βοήθεια στο Scilab

Το Scilab περιέχει έναν εκτεταμένο οδηγό χρήσης που καλύπτει όλες τις βασικές του λειτουργίες με πολλά και κατατοπιστικά παραδείγματα.

Πρόσβαση στην βοήθεια μπορούμε να έχουμε είτε από **menu bar > Scilab help** είτε απλά γράφοντας στην κονσόλα --> help



Εάν χρειαζόμαστε βοήθεια για το πώς χρησιμοποιείται μία εντολή ή κάποια συνάρτηση του Scilab, τότε μπορούμε να ζητήσουμε την βοήθεια από την κονσόλα
π.χ.

```
-->help sin
```

Εισαγωγή στο Scilab

Ορισμός μεταβλητών και εμφάνιση

- Στο Scilab δεν χρειάζεται να δηλώσουμε το είδος των μεταβλητών
- Τα πάντα είναι πίνακες.
- Οι μεταβλητές πρέπει να έχουν ονόματα τα οποία δεν είναι δεσμευμένα από το Scilab.
- Το αποτέλεσμα ενός υπολογισμού εάν δεν αποδοθεί σε μία μεταβλητή, τότε αυτόματα από δίδεται σε μία μεταβλητή που ονομάζεται **ans**

```
-->a
  !--error 4
Undefined variable : a
```

```
--> a=%pi/4
 a  =
    0.7853982
--> a
 a  =
    0.7853982
```

```
--> Piby2=%pi/2
 Piby2  =
    1.5707963
```

```
-->3*(4-2)
 ans  =
    6.

-->ans
 ans  =
    6.
```

Συναρτήσεις

Εάν θέλουμε να ορίσουμε μία συνάρτηση f(x) αυτό γίνεται στον Editor ως εξής

```
function y=f(x)
    y=36/(8+exp(-x));
endfunction
```

ή σε μία μόνο γραμμή στην κονσόλα

```
function y=f(x); y=36/(8+exp(-x)); endfunction
```

Εάν μετά τον ορισμό της συνάρτησης θέλουμε την τιμή της π.χ. για x=10 τότε απλά γράφουμε

```
--> f(10)

 ans  =

    4.4999745
```

Ορισμός πινάκων

Εάν θέλουμε να ορίσουμε έναν πίνακα στήλης

```
-->v=[3;-2;5]
 v  =
    3.
  - 2.
    5.
```

Εάν θέλουμε να ορίσουμε έναν πίνακα γραμμής

```
-->v=[3,-2,5]
 v  =
    3.   - 2.    5.
```

Εάν θέλουμε να ορίσουμε έναν πίνακα nxm

```
-->m=[1 2 3;4 5 6;7 8 9]
 m  =
    1.    2.    3.
    4.    5.    6.
    7.    8.    9.
```

Ο τελεστής ‘ : ’

Εάν γράψουμε στην κονσόλα την εντολή --> 3:10 τότε θα πάρουμε την ακολουθία των αριθμών από το 3 έως το 10

```
-->3:10

 ans  =

    3.    4.    5.    6.    7.    8.    9.    10.
```

Εάν γράψουμε την εντολή --> 1:2:10 τότε θα πάρουμε την ακολουθία των αριθμών

```
-->1:2:10

 ans  =

    1.    3.    5.    7.    9.
```

Δηλαδή από το 1 έως το 10 με βήμα 2

Εάν για παράδειγμα θέλουμε να φτιάξουμε έναν πίνακα γραμμής που να ονομάζεται x και να έχει ως στοιχεία τους περιττούς αριθμούς από το 1 έως το 10, τότε μπορούμε να γράψουμε:

```
-->x=1:2:10
 x  =

    1.    3.    5.    7.    9.
```

και να επιβεβαιώσουμε ότι πρόκειται για πίνακα γραμμής χρησιμοποιώντας την εντολή size

```
-->size(x)
 ans  =

    1.    5.
```

η οποία μας λέει ότι ο x είναι ένας πίνακας με 1 γραμμή και 5 στήλες.

Χειρισμός πινάκων

- Δημιουργία πινάκων με τις συναρτήσεις **ones(), zeros(),eye()**

```
-->x=ones(1,4)
 x  =

    1.    1.    1.    1.

-->size(x)
 ans  =

    1.    4.
```

```
-->x=ones(3,3)
 x  =

    1.    1.    1.
    1.    1.    1.
    1.    1.    1.

-->size(x)
 ans  =

    3.    3.
```

```
-->y=1:3
 y  =

    1.    2.    3.

-->ones(y)
 ans  =

    1.    1.    1.
```

Με παρόμοιο τρόπο λειτουργεί και η συνάρτηση **zeros()**

Ορισμός μοναδιαίου πίνακα (3x3)

```
-->A=eye(3,3)
 A  =

    1.    0.    0.
    0.    1.    0.
    0.    0.    1.
```

```
-->B=[1 2 3 ; 3 2 1; 4 5 6]
 B  =

    1.    2.    3.
    3.    2.    1.
    4.    5.    6.

-->eye(B)
 ans  =

    1.    0.    0.
    0.    1.    0.
    0.    0.    1.
```

Ορισμός μοναδιαίου πίνακα με τις ίδιες διαστάσει όπως ο πίνακας B

Χειρισμός πινάκων

- Συνδυασμοί δύο οι περισσότερων πινάκων

```
-->A=[1 2 3]
 A  =

    1.    2.    3.

-->B=[4 5 6]
 B  =

    4.    5.    6.

-->C=[A,B]
 C  =

    1.    2.    3.    4.    5.    6.
```

```
-->C=[A;B]
 C  =

    1.    2.    3.
    4.    5.    6.
```

```
-->A=[]
 A  =

     []

-->A=[A,1:3]
 A  =

    1.    2.    3.
```

```
-->A=[1 2 ; 3 4]
 A  =

    1.    2.
    3.    4.

-->B=[5 6 ; 7 8]
 B  =

    5.    6.
    7.    8.

-->C=[A;B]
 C  =

    1.    2.
    3.    4.
    5.    6.
    7.    8.
```

```
-->A=[1 2 ; 3 4]
 A  =

    1.    2.
    3.    4.

-->B=[5 6 ; 7 8]
 B  =

    5.    6.
    7.    8.

-->C=[A,B]
 C  =

    1.    2.    5.    6.
    3.    4.    7.    8.
```

Πολλαπλασιασμός πινάκων : Έστω οι (2x2) πίνακες $\mathrm{A}=\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$, $\mathrm{B}=\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$

Ο πολλαπλασιασμός πινάκων ορίζεται ως:

$$\mathrm{A}\times\mathrm{B}=\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}\times\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}=\begin{bmatrix} a_{11}b_{11}+a_{12}b_{21} & a_{11}b_{12}+a_{12}b_{22} \\ a_{21}b_{11}+a_{22}b_{21} & a_{21}b_{12}+a_{22}b_{22} \end{bmatrix}$$

Γενικά, για να μπορούμε να πολλαπλασιάσουμε τους πίνακες Α και Β θα πρέπει ο Α να είναι (nxm) και ο Β (mxk). Το αποτέλεσμα της πράξης θα είναι πίνακας (nxk)

$$(n\times m)(m\times k)=(n\times k)$$

Γενικά, για τους πίνακες Α(nxm) και Β (mxk) μπορούμε να πούμε ότι το ij στοιχείο του γινομένου των δύο πινάκων (AxB) γράφεται ως:

$$(\mathrm{A}\times\mathrm{B})_{ij}=\sum_{q=1}^{m}a_{iq}b_{qj}$$

Εισαγωγή στο Scilab

Παράδειγμα πολλαπλασιασμού πινάκων

$$A=\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix},\qquad B=\begin{bmatrix}5 & 6\\ 7 & 8\end{bmatrix}$$

$$A\times B=\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\times\begin{bmatrix}5 & 6\\ 7 & 8\end{bmatrix}=\begin{bmatrix}1\cdot5+2\cdot7 & 1\cdot6+2\cdot8\\ 3\cdot5+4\cdot7 & 3\cdot6+4\cdot8\end{bmatrix}=\begin{bmatrix}19 & 22\\ 43 & 50\end{bmatrix}$$

Η ίδια πράξη στο Scilab

```
-->A=[1 2 ; 3 4]
 A  =

    1.    2.
    3.    4.

-->B=[5 6 ; 7 8]
 B  =

    5.    6.
    7.    8.

-->A*B
 ans  =

    19.   22.
    43.   50.
```

Δηλαδή στο Scilab εάν έχω τους πίνακες A και B και γράψω A*B τότε **θα πραγματοποιήσει πολλαπλασιασμό Πινάκων.!!!**

ΠΡΟΣΟΧΗ

Εάν θέλουμε να πολλαπλασιάσουμε τα στοιχεία των πινάκων 1 προς 1 τότε θα πρέπει να χρησιμοποιήσουμε την πράξη ‘ .* ’

```
-->A.*B
 ans  =

    5.    12.
    21.   32.
```

Ο ανάστροφος ενός πίνακα (Transpose)

Έστω ο πίνακας A(nxm). Εάν σε αυτόν τον πίνακα κάνουμε τις γραμμές στήλες και τις στήλες γραμμές, τότε λέμε ότι έχουμε κατασκευάσει τον ανάστροφο του A, ο οποίος συμβολίζεται με A^T και είναι πίνακας (mxn).

$$A=\begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{bmatrix} \qquad A^T=\begin{bmatrix}1 & 3 & 5\\ 2 & 4 & 6\end{bmatrix}$$

Στο **Scilab** αυτό επιτυγχάνεται ως εξής:

```
-->A=[1 2 ; 3 4; 5 6]
 A  =

    1.    2.
    3.    4.
    5.    6.
```

ανάστροφος

```
-->A'
 ans  =

    1.    3.    5.
    2.    4.    6.
```

Ο αντίστροφος ενός πίνακα (Inverse)

Παράδειγμα: Έστω ότι έχουμε το σύστημα των εξισώσεων

$$\begin{aligned} x_1 - 2x_2 &= 1 \\ -3x_1 + 4x_2 &= 3 \end{aligned}$$

Το σύστημα αυτό μπορεί να γραφεί με την χρήση πινάκων ως εξής

$$A \times X = B \quad \text{όπου} \quad A = \begin{bmatrix} 1 & -2 \\ -3 & 4 \end{bmatrix}, \; X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

μπορεί να λυθεί με την χρήση του αντίστροφου πίνακα A^{-1} που έχει την ιδιότητα $A^{-1}A = \mathbf{1}$

$$AX = B \Leftrightarrow A^{-1}AX = A^{-1}B \Leftrightarrow \left(A^{-1}A\right)X = A^{-1}B \Leftrightarrow \mathbf{1}X = A^{-1}B \Leftrightarrow X = A^{-1}B$$

Στο Scilab, ο αντίστροφος ενός πίνακα υπολογίζεται με την συνάρτηση inv(A)

Ο αντίστροφος ενός πίνακα (Inverse)

Παράδειγμα: Για το σύστημα των εξισώσεων

$$x_1 - 2x_2 = 1$$
$$-3x_1 + 4x_2 = 3$$

$$A = \begin{bmatrix} 1 & -2 \\ -3 & 4 \end{bmatrix}, \quad X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

Η λύση του συστήματος είναι $X = A^{-1}B$ και στο Scilab υλοποιείται ως εξής

```
-->A=[1 -2 ; -3 4]
 A  =

    1.   - 2.
  - 3.     4.

-->B=[1 ;3]
 B  =

    1.
    3.

-->X=inv(A)*B
 X  =

  - 5.
  - 3.
```

Δηλαδή $x_1 = -5$ και $x_2 = -3$

Βρόγχοι (loops)

Ο συνήθης τρόπος να εκτελούμε μία επαναλαμβανόμενη διαδικασία είναι η χρησιμοποίηση βρόγχων.

Βρόγχος for … end

Έστω πάλι ότι θέλουμε να φτιάξουμε έναν πίνακα που να ονομάζεται x και να έχει ως στοιχεία τους περιττούς αριθμούς από το 1 έως το 10, τότε μπορούμε να γράψουμε:

```
for n=1:5
    x(n)=2*n-1;
end
```

```
-->x
 x  =

    1.
    3.
    5.
    7.
    9.
```

Εάν στην κονσόλα γράψουμε απλά x θα πάρουμε:

Βλέπουμε ότι ο πίνακας αυτή την φορά είναι πίνακας στήλης, κάτι που επιβεβαιώνεται και με την εντολή size. Εάν θέλουμε οπωσδήποτε ο πίνακάς μας να είναι πίνακας γραμμή τότε μπορούμε να χρησιμοποιήσουμε την εντολή για ανάστροφο πίνακα

```
-->size(x)
 ans  =

    5.    1.
```

```
-->x=x'
 x  =

    1.    3.    5.    7.    9.
```

```
-->size(x)
 ans  =

    1.    5.
```

Παράδειγμα με **for … end**

- Να υπολογιστούν οι πρώτοι 20 όροι της σειράς που ορίζεται από την αναδρομική σχέση

$$\begin{cases} u_1 = 4 \\ u_{n+1} = u_n + 2n + 3 \end{cases}$$

Πρόγραμμα:

```
u(1)=4;
for n=1:20
    u(n+1)= u(n)+2*n+3;
    disp([n,u(n)])
end
```

Εμφάνιση αποτελεσμάτων

- Να υπολογιστεί το παραγοντικό (n!) ενός αριθμού

```
n=3;
fact=1;
for i=1:n
    fact=fact*i;
end

disp(fact)
```

Σε αυτή την περίπτωση Θα μπορούσαμε την εσωτερική συνάρτηση factorial() του Scilab

```
-->factorial(3)
 ans  =

    6.
```

Παράδειγμα με **for … end**

• Να υπολογιστεί το άθροισμα $\sum_{i=1}^{10} \frac{1}{n}$

```
n=10;
s=0;
for i=1:n
    s=s + 1.0/double(i) ;
disp([s,1.0/double(i)])
end
```

Διαφορετικός τρόπος

```
x=1:10;
y=1.0 ./ x;
sum(y)
```

Τελεστής ./
Διαίρεση στοιχείο
Προς στοιχείο

Άθροισμα στοιχείων πίνακα

• Να υπολογιστεί η μέση τιμή (μ), η τυπική απόκλιση (σ) και η τυπική απόκλιση της μέσης τιμής της ακόλουθης σειράς μετρήσεων: 5,2 5,4 4,9 5,3 5,5 4,8 4,7

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad \sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \mu)^2}{N-1}} \qquad \sigma_\mu = \frac{\sigma}{\sqrt{N}}$$

Εισαγωγή στο Scilab

```
clear

x=[5.2;5.4;4.9;5.3;5.5;4.8;4.7];
n=7
s=0;
for i=1:n
    s=s+x(i);
end
mt=s/n;
//
s=0;
for i=1:n
    s=s+(x(i)-mt)^2;
end
ta=sqrt(s/((n-1)));

mprintf("mesi timi = %f",mt);
mprintf("typiki apoklisi = %f",ta);
mprintf("typiki apoklisi mesis timis = %f",ta/sqrt(n));
```

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i-\mu)^2}{N-1}} \qquad \sigma_\mu = \frac{\sigma}{\sqrt{N}}$$

Κλασική δομή προγράμματος όπου χρησιμοποιούντα βρόγχοι για τον υπολογισμό των αθροισμάτων

```
clear
x=[5.2;5.4;4.9;5.3;5.5;4.8;4.7];
mt=mean(x);
ta=stdev(x);
mprintf("mesi timi = %f",mt);
mprintf("typiki apoklisi = %f",ta);
mprintf("typiki apoklisi mesis timis = %f",ta/sqrt(n));
```

Υπολογισμός με την βοήθεια των εσωτερικών συναρτήσεων **mean()** και **stdev()**

Διαμέριση διαστήματος σε διακριτά σημεία

Έστω ότι θέλουμε να δώσουμε στην μεταβλητή x, N ισαπέχοντες τιμές στο διάστημα [α,β]. Ποιες θα είναι αυτές οι τιμές και πόσο θα απέχουν μεταξύ τους;

Ισχύουν οι σχέσεις: $dx = \frac{\beta-\alpha}{N-1}$ $N = \frac{\beta-\alpha+dx}{dx}$

Καθορισμένο N
(παραδοσιακός τρόπος)

```
clear
a=0;
b=1;
N=8;
dx=(b-a)/(N-1);
for i=1:N
    x(i)=a+(i-1)*dx;
end

disp(x)
```

Καθορισμένο dx
(παραδοσιακός τρόπος)

```
clear
a=0;
b=1;
dx=0.15;
N=(b-a+dx)/dx;
for i=1:N
    x(i)=a+(i-1)*dx;
end

disp(x)
```

Καθορισμένο N
(εσωτερική συνάρτηση Scilab)

```
x=linspace(0,1,8)
```

Βρόγχος while … end

Με την δομή αυτή, κατασκευάζουμε βρόγχους οι οποίοι εκτελούνται επαναλαμβανόμενα, για όσο διάστημα ικανοποιείται η «συνθήκη»

```
while συνθηκη
      .......
end
```

Παραδείγμα

```
i = 0
while i<5
      disp([i i^2]);
      i = i + 1;
end
```

Προσπαθήστε να καταλάβετε τι κάνει το παραπάνω παράδειγμα

Οι τελεστές που μας χρειάζονται για την σωστή έκφραση της οποιασδήποτε συνθήκης είναι: (τελεστές σύγκρισης)

ίσο	**==**	**αληθές**	**%T**
διάφορο	**<>**	**Ψευδές**	**%F**
μικρότερο	**<**	**Και**	**&**
μεγαλύτερο	**>**	**ή**	**|**
Μικρότερο ή ίσο	**<=**	**Όχι**	**~**
Μεγαλύτερο ή ίσο	**=>**		

Παράδειγμα

Ποια από τις δύο εκδοχές του προγράμματος θα μας εμφανίσει κάτι στην οθόνη και τι;

```
i = 0
while i>5 & i^2<200
    disp([i i^2]);
    i = i + 1;
end
```

?

```
i = 6
while i>5 & i^2<200
    disp([i i^2]);
    i = i + 1;
end
```

Τι θα μας δώσουν οι δύο παρακάτω εκδοχές του προγράμματος;

```
i = 0
while 1==1 & i^2<200
    disp([i i^2]);
    i = i + 1;
end
```

?

```
i = 0
while 1==1
    disp([i i^2]);
    i = i + 1;
end
```

Η έννοια του ατέρμονος βρόγχου

Έξοδος από βρόγχο → `break`

Εισαγωγή στο Scilab

Συνθήκη if … then …. Else …… end

Με την δομή αυτή, κατασκευάζουμε συνθήκες και ελέγχουμε την ροή ενός προγράμματος

```
if x==10 then
    ektelese aytes tis entoles
else
    ektelese aytes tis entoles
end
```

Παράδειγμα

```
clear
x=grand(1,1,"unf",1,10);
if x<=5 then
    a="mikro";
else
    a="megalo";
end

disp(a+" "+string(x))
```

Εσωτερική συνάρτηση του Scilab για την δημιουργία τυχαίων αριθμών

```
x=grand(1,1,"unf",1,10);
```

Διαστάσεις του τυχαίου πίνακα

Είδος κατανομής

Οι τυχαίοι αριθμοί να είναι από … εως…

Η συνάρτηση disp() εμφανίζει στη οθόνη
Το σύνολο των χατακτήρων που βρίσκονται
Εντός των ()

Συνθήκη if … then …. elseif … else … end

Με τρόπο αυτό, συνδυάζουμε περισσότερες από μία συνθήκες στην ίδια δομή και ελέγχουμε την ροή ενός προγράμματος

```
clear
x=grand(1,1,"unf",1,10);
if x<=2.5 then
    a="mikrote tou 2.5";
elseif x>2.5 & x<=5.0 then
    a="megalytero tou 2.5 kai mikrotero i iso tou 5";
elseif x>5.0 & x<=7.5 then
    a="megalytero tou 5.0 kai mikrotero i iso tou 7.5";
else
    a="megalytero tou 7.5";
end

disp(a+": i timi tou x einai "+string(x))
```

Εισαγωγή στο Scilab

Παράδειγμα Χρήση της συνάρτησης grand() για την εισαγωγή ‘θορύβου’ σε μία συνάρτηση

```
clear
function y=f(t)
    y=sin(t);
endfunction

x=linspace(0,2*%pi,100);
clf;
plot(x,f)

y=feval(x,f);
noise=grand(1,100,"unf",-0.2,0.2);
y=y+noise;
plot(x,y,'r')
```

Ορισμός συνάρτησης

Διαμέριση του διαστήματος [0,2π] σε 100 ισαπέχοντα σημεία

Δημιουργία γραφικής παράστασης

Δημιουργία του πίνακα y που περιέχει τις τιμές της συνάρτησης για όλα τα x

Δημιουργία τυχαίου θορύβου

Πρόσθεση του θορύβου στις τιμές της συνάρτησης

Δημιουργία της νέας γραφικής παράστασης

Εισαγωγή στο Scilab

Εγγραφή/ανάγνωση δεδομένων σε/από αρχείο

Στο παράδειγμα της προηγούμενη διαφάνειας θα προσθέσουμε μερικές γραμμές ώστε να αποθηκευτούν τα δεδομένα μας σε ένα αρχείο και στην συνέχεια να τα διαβάσουμε από εκεί

```
clear
function y=f(t)
    y=sin(t);
endfunction

x=linspace(0,2*%pi,100);
clf;
plot(x,f)

y=feval(x,f);
noise=grand(1,100,"unf",-0.2,0.2);
y=y+noise;
plot(x,y,'r')

tmpdir="C:\temp";
chdir(tmpdir);
fid=file('open','test2.txt','unknown');
W=[x',y'];
write(fid,W);
file('close',fid);

fid=file('open','test2.txt','unknown');
Q=read(fid,-1,2);
file('close',fid);
```

Ορισμός του directory στο οποίο Θα γραφτεί το αρχείο

Αλλαγή directory

Άνοιγμα του αρχείου

Μορφοποίηση των δεδομένων Ώστε να γραφτούν σε 2 στήλες

Εγγραφή στο αρχείο

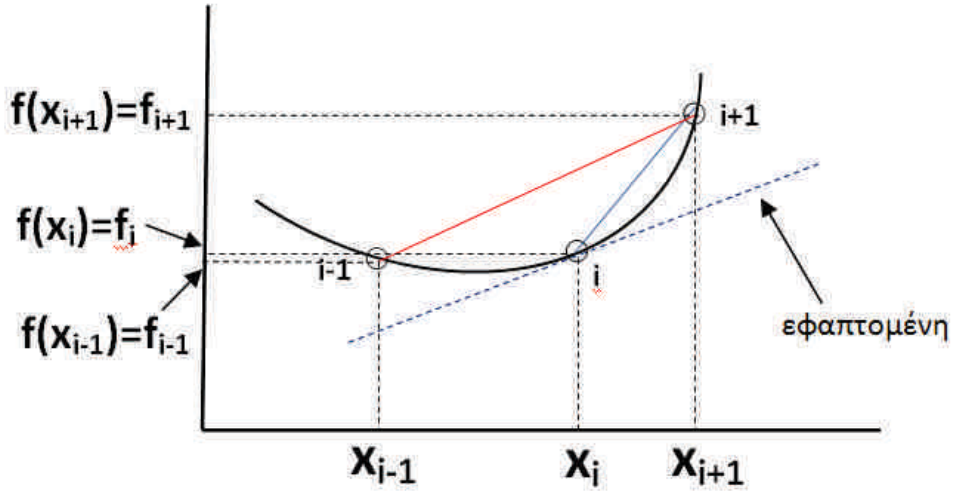
Κλείσιμο αρχείου

Άνοιγμα του αρχείου

Ανάγνωση των τιμών στον πίνακα Q (ΠΡΟΣΟΧΗ στο -1)

Αριθμητική παραγώγιση

Η παράγωγος της συνάρτησης στο σημείο x_i μπορεί να προσεγγιστεί αριθμητικά με την εξής σχέση



$$\left.\frac{df}{dx}\right|_{x=x_i} \simeq \frac{f\left(x_{i+1}\right)-f\left(x_i\right)}{x_{i+1}-x_i} = \frac{f_{i+1}-f_i}{x_{i+1}-x_i}$$

Εάν $x_{i+1}-x_i=h$ τότε μπορούμε να γράψουμε την παραπάνω σχέση ως:

$$\left.\frac{df}{dx}\right|_{x=x_i} \simeq \frac{f\left(x_i+h\right)-f\left(x_i\right)}{h} = \frac{f_{i+1}-f_i}{h}$$

Forward differences

Ένας διαφορετικός, και εν γένει προτιμητέος τρόπος, είναι οι λεγόμενες κεντρικές διαφορές (**central differences**):

$$\left.\frac{df}{dx}\right|_{x=x_i} \simeq \frac{f\left(x_{i+1}\right)-f\left(x_{i-1}\right)}{x_{i+1}-x_{i-1}} = \frac{f_{i+1}-f_{i-1}}{x_{i+1}-x_{i-1}}$$

Εάν οι αποστάσεις μεταξύ των x_i είναι ίσες με h, τότε

$$\left.\frac{df}{dx}\right|_{x=x_i} \simeq \frac{f\left(x_i+h\right)-f\left(x_i-h\right)}{2h} = \frac{f_{i+1}-f_{i-1}}{2h}$$

Central differences

Παράδειγμα αλγόριθμου αριθμητικής παραγώγισης

```
clear
function y=f(x)
    y=sin(x);
endfunction

n=50;
x=linspace(0,3*%pi,n);
h=x(2)-x(1);
y=feval(x,f);
clf;
plot(x,y);

for i=1:n-1
    df(i)=(y(i+1)-y(i))/h;
end
plot(x(1:n-1),df,'--r')
```

Ορισμός της συνάρτησης που θέλουμε να παραγωγίσουμε

Αριθμός σημείων για την διαμέριση

διαμέριση

Απόσταση μεταξύ2 διαδοχικών σημείων στον άξονα x

Κατασκευή του πίνακα y που περιέχει τις τιμές $f(x_i)$

Βρόγχος για τον υπολογισμό της παραγώγου σε όλα τα σημεία πλην του τελευταίου

Forward differentiation

Γραφική παράσταση της παραγώγου

Εάν θέλουμε να χρησιμοποιήσουμε **κεντρικές διαφορές**, τότε στον προηγούμενο κώδικα μπορούμε κάνουμε την ακόλουθη αντικατάσταση

```
clear
function y=f(x)
    y=sin(x);
endfunction

n=50;
x=linspace(0,3*%pi,n);
h=x(2)-x(1);
y=feval(x,f);
clf;
plot(x,y);

for i=1:n-1
    df(i)=(y(i+1)-y(i))/h;
end
plot(x(1:n-1),df,'--r')
```

Προσπαθήστε να εξηγήσετε τις αλλαγές
Που περιλαμβάνει το νέο κομμάτι κώδικα

```
for i=2:n-1
    df(i)=(y(i+1)-y(i-1))/(2*h);
end
plot(x(2:n-1),df(2:n-1),'--r')
```

-- Κάνοντας χρήση του HELP του SciLab να προσπαθήσετε να κατανοήσετε την λειτουργία των εσωτερικών συναρτήσεων **derivative()** και **diff()**

-- Να χρησιμοποιήσετε τις συναρτήσεις αυτές για να πραγματοποιήσετε την παραγώγιση του προηγούμενου παραδείγματος.

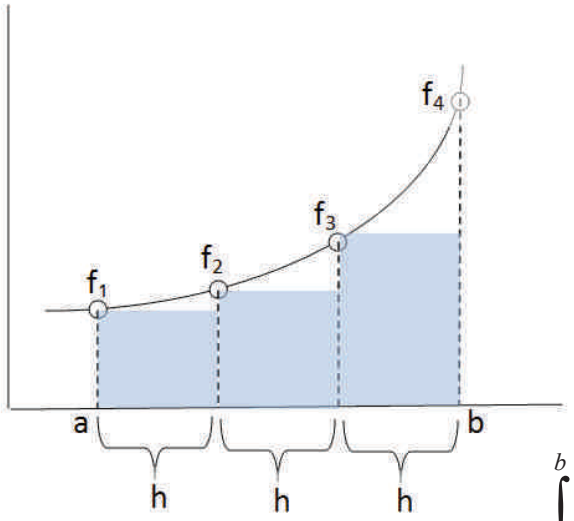
Δεύτερη παράγωγος

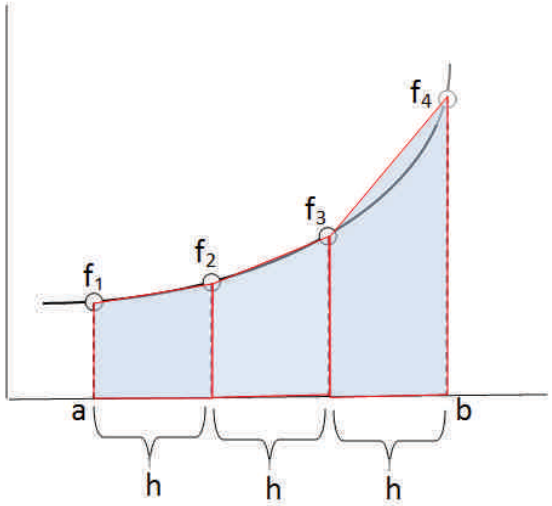
Κεντρικές διαφορές

$$\left.\frac{d^2 f}{dx^2}\right|_{x=x_i} \simeq \frac{f\left(x_i+h\right)-2f\left(x_i\right)+f\left(x_i-h\right)}{h^2} = \frac{f_{i+1}-2f_i+f_{i-1}}{h^2}$$

Να τροποποιηθεί ο προηγούμενος κώδικας ώστε να υπολογίζει την δεύτερη παράγωγο

Αριθμητική ολοκλήρωση





$$\int_a^b f(x)\,dx = \text{εμβαδό}$$

Το εμβαδόν προσεγγίζεται με ένα άθροισμα

Παραλληλογράμμων ⟵ ⟶ **Τραπεζίων**

$$\int_a^b f(x)\,dx = f_1 \cdot h + f_2 \cdot h + f_3 \cdot h$$

$$\int_a^b f(x)\,dx = \sum_{i=1}^{3} f_i \cdot h$$

$$\int_a^b f(x)\,dx = \frac{f_1+f_2}{2}\cdot h + \frac{f_2+f_3}{2}\cdot h + \frac{f_3+f_4}{2}\cdot h$$

$$\int_a^b f(x)\,dx = \frac{h}{2}\left[f_1 + 2f_2 + 2f_3 + f_4\right] = \frac{h}{2}\left[f_1+f_4\right] + \sum_{i=2}^{4-1} f_i \cdot h$$

Στην γενικής περίπτωση που το διάστημα [a,b] διαμερίζεται σε N ισαπέχοντα σημεία

$$\int_a^b f(x)\,dx = \sum_{i=1}^{N-1} f_i \cdot h$$

$$\int_a^b f(x)\,dx = \frac{h}{2}\left[f_1 + f_N\right] + \sum_{i=2}^{N-1} f_i \cdot h$$

Παρατήρηση : Οι δύο προηγούμενες μέθοδοι αριθμητικής ολοκλήρωσης είναι οι απλούστερες δυνατές. Στην πράξη, πολλές φορές χρησιμοποιούνται πολύ πιο αναπτυγμένες τεχνικές και μέθοδοι για τον ακριβή αριθμητικό υπολογισμό ολοκληρωμάτων.

Μέθοδος παραλληλογράμμου

```
clear
function y=f(x)
    y=x.^2;
endfunction

n=10;
x=linspace(0,10,n);
h=x(2)-x(1);
y=feval(x,f);

integral=0;
for i=1:n-1
   integral=integral+y(i)*h;
end

disp("OLOKLHRWMA= "+string(integral))
```

Ακριβής τιμή = 333,333

Μέθοδος τραπεζίου

```
clear
function y=f(x)
    y=x.^2;
endfunction

n=10;
x=linspace(0,10,n);
h=x(2)-x(1);
y=feval(x,f);

integral=0;
for i=2:n-1
   integral=integral+y(i)*h;
end
integral=integral+y(1)*h/2 + y(n)*h/2;
disp("OLOKLHRWMA= "+string(integral))
```

Μεταβάλλοντας τον αριθμό της διαμέρισης **n**, πειραματιστείτε με τους 2 κώδικες ώστε να διαπιστώσετε ποια μέθοδος δίνει ακριβέστερα αποτελέσματα και ποια είναι η κατάλληλη τιμή του **n** ώστε το αποτέλεσμα να έχει ικανοποιητική ακρίβεια.

Αριθμητική επίλυση της Συνήθους Διαφορικής Εξίσωσης (ΣΔΕ) πρώτης τάξης

$$\frac{dy}{dt} = f\left(y,t\right) \quad \text{Με την αρχική συνθήκη} \quad y\left(t_0\right) = y_0$$

Μέθοδος Euler

$$\frac{dy}{dt} = f\left(y,t\right) \xrightarrow{\delta\iota\alpha\kappa\rho\iota\tau o\pi o\acute{\iota}\eta\sigma\eta} \frac{y\left(t+h\right)-y\left(t\right)}{h} = f\left(y,t\right) \Leftrightarrow y\left(t+h\right) = y\left(t\right)+h\cdot f\left(y,t\right)$$

Βλέπουμε ότι εάν γνωρίζουμε την τιμή της συνάρτησης y κατά την χρονική στιγμή t,
Τότε μπορούμε την τιμή της κατά την χρονική στιγμή t+h.
ή
Με διαφορετικό συμβολισμό

$$\frac{dy}{dt} = f\left(y,t\right) \xrightarrow{\delta\iota\alpha\kappa\rho\iota\tau o\pi o\acute{\iota}\eta\sigma\eta} \frac{y_{i+1}-y_i}{h} = f\left(y_i,t_i\right) \Leftrightarrow y_{i+1} = y_i + h\cdot f\left(y_i,t_i\right)$$

Μέθοδος Euler - εφαρμογή

Διακριτοποιούμε την ανεξάρτητη μεταβλητή t

Ανεξάρτητη μεταβλητή

t_0	t_0
$t_1=t_0+h$	$t_1=t_0+h$
$t_2=t_1+h$	$t_2=t_0+2h$
$t_3=t_2+h$	$t_3=t_0+3h$
$t_4=t_3+h$	$t_4=t_0+4h$
.	.
.	.
.	.
$t_n=t_{n-1}+h$	$t_n==t_0+nh$

Υπολογισμός των τιμών Της συνάρτησης y_i στα διακριτά σημεία t_i

Η άγνωστη συνάρτηση

$y_0=y(t_0)$

$y_1=y_0+hf(y_0,t_0)$

$y_2=y_1+hf(y_1,t_1)$

$y_3=y_2+hf(y_2,t_2)$

$y_4=y_3+hf(y_3,t_3)$

.

.

.

$y_n=y_{n-1}+hf(y_{n-1},t_{n-1})$

Μέθοδος Euler - εφαρμογή στο Scilab

Έστω ότι θέλουμε να υπολογίσουμε την χρονική μεταβολή ενός πληθυσμού χρησιμοποιώντας την ΔΕ του λογιστικού μοντέλου.

$$\frac{dN}{dt} = r(M - N)N$$

Όπου r μία αριθμητική σταθερά και M η φέρουσα ικανότητα της περιοχής

Έστω ότι για το πρόβλημα μας ισχύει r=0.01, M=100 και για t=0, N(0)=10

```
clear
function dN=f(N)
    r=0.01;
    M=100;
    dN=r*(M-N)*N;
endfunction

n=100;
t=linspace(0,10,n);
h=t(2)-t(1);

N(1)=10;
for i=1:n-1
    N(i+1)=N(i)+h*f(N(i));
end
clf;
plot(t,N)
```

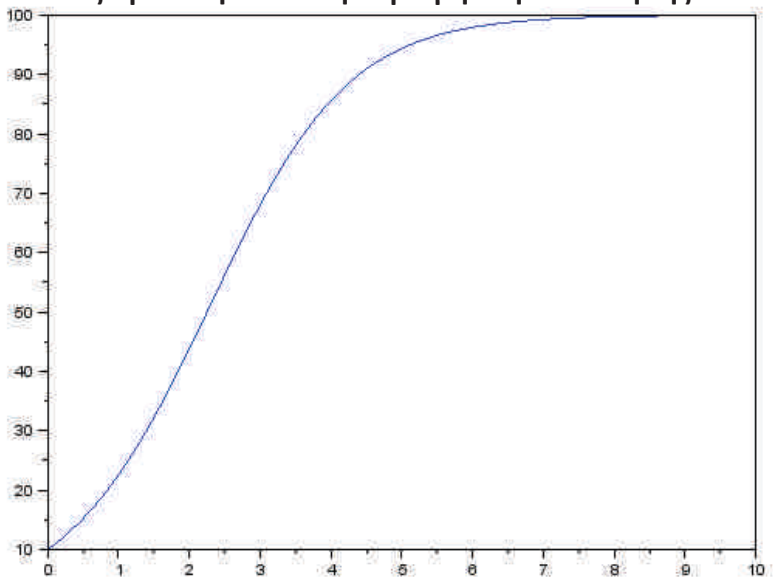
Ορισμός συνάρτησης Και παραμέτρων

Διακριτοποίηση Του χρόνου

Αρχική συνθήκη

Μέθοδος Euler

Γραφική παράσταση της συνάρτησης N(t)
Όπως προέκυψε από την αριθμητική επίλυση της ΔΕ

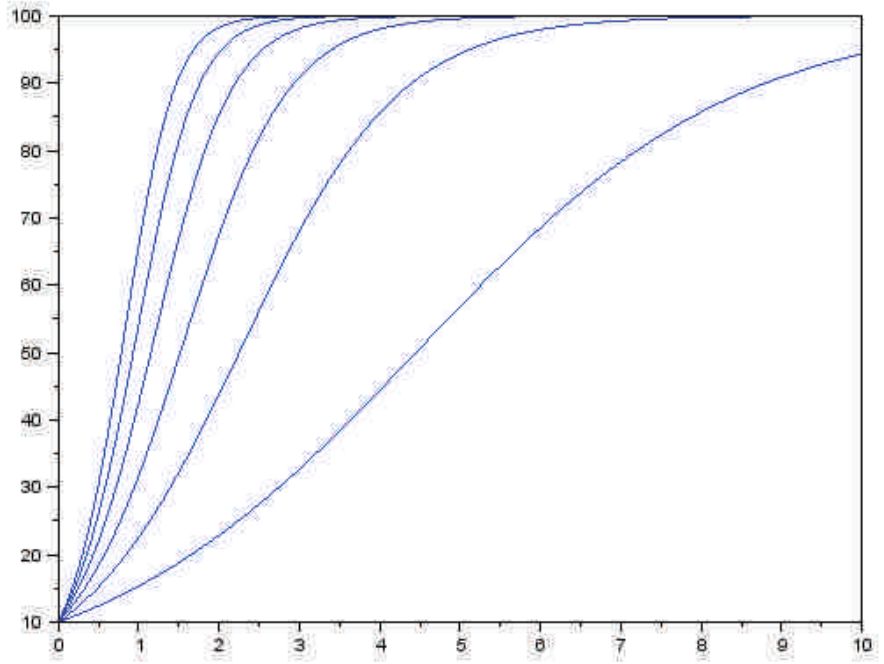


Μέθοδος Euler - εφαρμογή στο Scilab

Έστω ότι θέλουμε να μελετήσουμε την συμπεριφορά της λύσης συναρτήσει της παραμέτρου r, δηλαδή να μεταβάλουμε το r και να βλέπουμε πως μεταβάλλεται η N(t). Σε αυτή την περίπτωση μπορούμε να τροποποιήσουμε λίγο τον προηγούμενο κώδικα ώστε να πάρουμε όλες τις N(t;r) στην ίδια γραφική παράσταση.

Προσπαθήστε να κατανοήσετε την λογική των αλλαγών που κάναμε

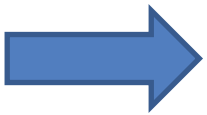
```
clear
function dN=f(N,r)
    M=100;
    dN=r*(M-N)*N;
endfunction

n=100;
t=linspace(0,10,n);
h=t(2)-t(1);

clf;
for r=0.005:0.005:0.03
 N(1)=10;
 for i=1:n-1
     N(i+1)=N(i)+h*f(N(i),r);
 end
 plot(t,N)
end
```

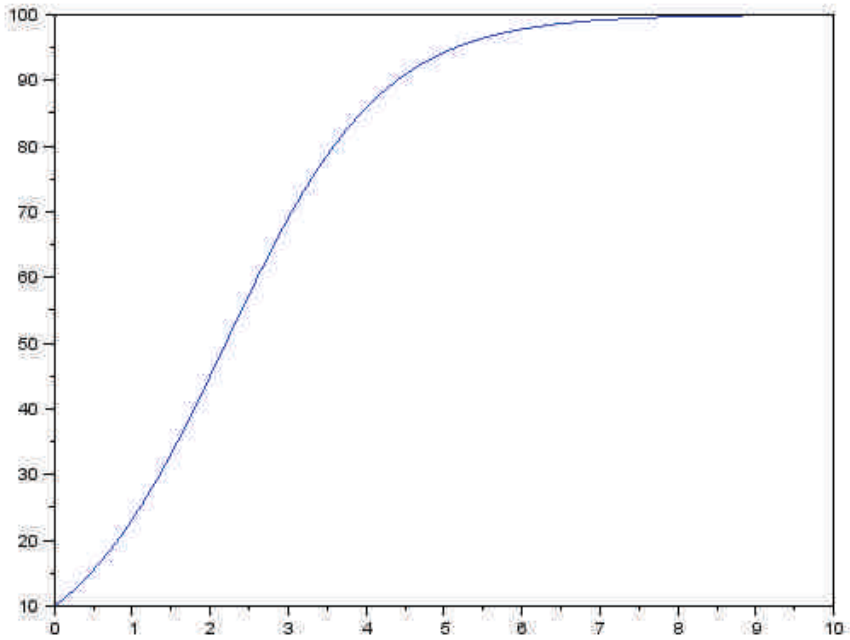


Επίλυση ΣΔΕ πρώτης τάξης με χρήση της ρουτίνας ode() του Scilab

Η επίλυση των προηγούμενων προβλημάτων μπορεί να πραγματοποιηθεί εύκολα, κάνοντας χρήση της εσωτερικής ρουτίνας **ode()** του Scilab. Χρησιμοποιώντας το help του Scilab και το παρακάτω παράδειγμα, προσπαθήστε να κατανοήσετε την χρήση της ode().

```
clear
function dN=f(t,N)
    M=100;
    r=0.01;
    dN=r*(M-N)*N
endfunction
N0=10;
t0=0;
t=0:0.1:10;
N=ode(N0,t0,t,f);
clf;
plot(t,N)
```

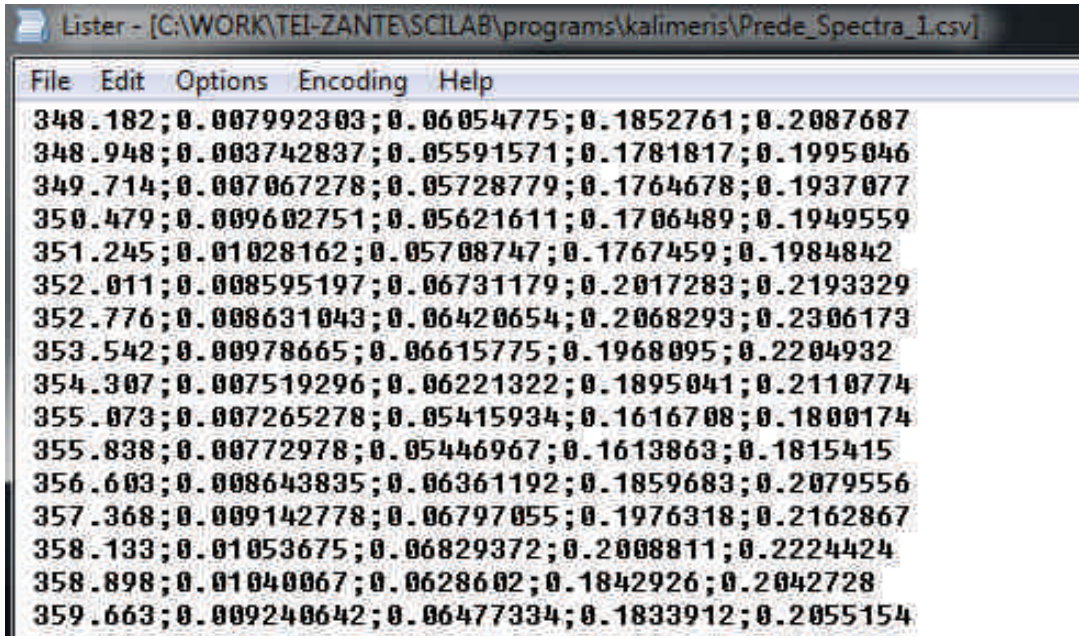




Ανάγνωση και επεξεργασία πειραματικών δεδομένων (αρχεία *.csv)

--Πολλές φορές υπάρχει ανάγκη να επεξεργαστούμε πειραματικά δεδομένα, τα οποία περιλαμβάνουν αρκετές χιλιάδες τιμών. Ένας από τους συνήθης τύπους αρχείων που χρησιμοποιούνται για την αποθήκευση τέτοιων δεδομένων είναι τα αρχεία csv (comma separated values)

-- Εάν ανοίξουμε με το σημειωματάριο (notepad) ένα τέτοιο αρχείο θα δούμε ότι η δομή του μοιάζει με αυτή που φαίνεται στην εικόνα



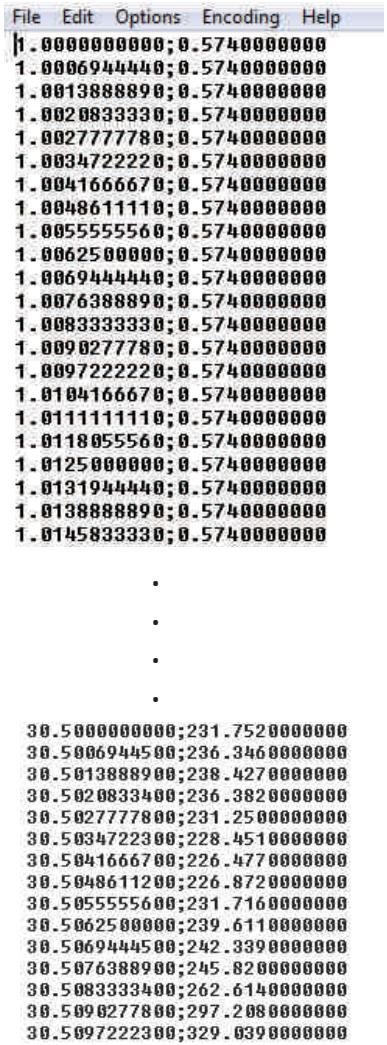
• Κοιτώντας προσεκτικά την πρώτη γραμμή του αρχείου, παρατηρούμε ότι περιέχει πραγματικούς αριθμούς οι οποίοι διαχωρίζονται μεταξύ τους με “ **;** ”. Το ίδιο ισχύει και για τις υπόλοιπες γραμμές. Με άλλα λόγια το αρχείο περιέχει 5 στήλες δεδομένων.

• Το σύμβολο “ **;** ” διαχωρίζει του αριθμούς μεταξύ τους, για αυτό και ονομάζεται **separator**. Τον ρόλο του separator μπορούν να τον παίξουν και άλλα σύμβολα π.χ. **TAB, “,” “ ” “:”** κ.λ.π.

• Τα αρχεία *.csv διαβάζονται από όλα τα προγράμματα υπολογιστικών φύλλων (π.χ. Excel, Calc(Libre Office), Gnumeric, Origin Grapher κ.λ.π.)

Ανάγνωση και επεξεργασία πειραματικών δεδομένων (αρχεία *.csv)

Έστω ότι θέλουμε να επεξεργαστούμε ένα αρχείο της μορφής:



--Το αρχείο περιέχει την πυκνότητα ισχύος για έναν συγκεκριμένο μήνα του χρόνου, η οποία καταγράφεται με την βοήθεια ενός πυρανόμετρου και ενός data logger το οποίο παίρνει μία μέτρηση ανά λεπτό.

--Δηλαδή ένα πλήρες αρχείο για έναν μήνα περιέχει
60min x 24hours x 30days=43200 ζεύγη τιμών

--Η πρώτη στήλη περιέχει έναν αριθμό που χαρακτηρίζει την μέρα και ώρα της μέτρησης π.χ. όλες οι μετρήσεις της πρώτης μέρας έχουν αριθμό 1.* ενώ οι μετρήσεις της 7^{ης} μέρας έχουν έναν αριθμό 7.* Οι αριθμοί αυτοί (π.χ. οι 1.*) προκύπτουν εύκολα διαμερίζοντας το διάστημα [1,2] σε 1441 σημεία linspace(1,2,1441). Σε αυτό το σημείο δεν μας ενδιαφέρουν περισσότερες λεπτομέρειες για τα στοιχεία της πρώτης στήλης.

-- Η δεύτερη στήλη περιέχει την μετρούμενη πυκνότητα ισχύος για την συγκεκριμένη χρονική στιγμή. (Στην πράξη το data logger καταγράφει μία διαφορά δυναμικού η οποία στην συνέχεια μεταφράζεται σε πυκνότητα ισχύος)

Ανάγνωση και επεξεργασία πειραματικών δεδομένων (αρχεία *.csv)

Άνοιγμα πολλαπλών αρχείων

```
clear
clc
stacksize("max");
chdir("C:\temp");
filelist=uigetfile(["*.csv"],"C:\temp", "Choose files", %t)
bRes = csvDefault("eol","linux")

for name=filelist
  B = csvRead(name,';');
  x=B(:,1);
  y=B(:,2);
  [n1,n2]=size(x);
```

Τεχνική λεπτομέρεια σχετικά με την μνήμη Που επιτρέπεται να χρησιμοποιήσει το πρόγραμμα

Τεχνική λεπτομέρεια σχετικά Με το line termination

Εντολή με την οποία ανοίγει ένα παράθυρο για την επιλογή των αρχείων που θέλουμε να διαβαστούν(βλ. Scilab help). Τα ονόματα των αρχείων αποθηκεύονται στον πίνακα filelist.

Με το for ανοίγουμε ένα ένα τα αρχεία τα οποία επιλέχτηκαν από την εντολή uigetfile(). Ο πίνακας B τις 2 στήλες του αρχείου της προηγούμενης διαφάνειας.

Ο πίνακας x είναι πίνακας στήλης και περιέχει την πρώτη στήλη των δεδομένων

Ο πίνακας y είναι πίνακας στήλης και περιέχει την δεύτερη στήλη των δεδομένων

n1 είναι το πλήθος των δεδομένων που διαβάστηκαν.

ΠΕΡΙΟΧΗ ΕΝΤΟΛΩΝ
Εδώ θα μπουν οι εντολές
Και τα κομμάτια του κώδικα για την επεξεργασία των δεδομένων που διαβάστηκαν παραπάνω

```
end
```

Ανάγνωση και επεξεργασία πειραματικών δεδομένων (αρχεία *.csv)

“Σπάσιμο” του μεγάλου μηνιαίου αρχείου σε ημερήσια

```
clear
clc
stacksize("max");
chdir("C:\temp");
filelist=uigetfile(["*.csv"],"C:\temp", "Choose files", %t)
bRes = csvDefault("eol","linux")

for name=filelist
  B = csvRead(name,';');
  x=B(:,1);
  y=B(:,2);
  [n1,n2]=size(x);

end
```

ΠΕΡΙΟΧΗ ΕΝΤΟΛΩΝ
Εδώ θα μπουν οι εντολές
Και τα κομμάτια του κώδικα για την επεξεργασία των δεδομένων που διαβάστηκαν παραπάνω

Εάν εισάγουμε το παρακάτω κομμάτι κώδικα στην **ΠΕΡΙΟΧΗ ΕΝΤΟΛΩΝ** τότε μπορούμε να “σπάσουμε” το μεγάλο μηνιαίο αρχείο σε μικρότερα ημερήσια, τα οποία θα ονομάζονται day_1.csv day_2.csv κ.λ.π

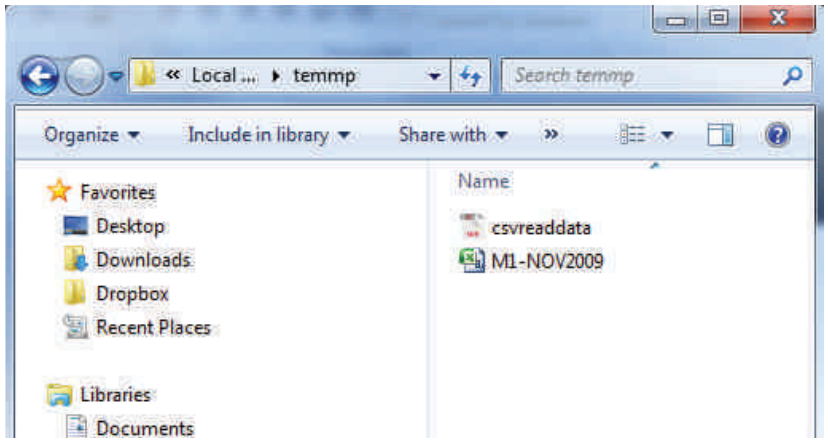
Σημαντικό

Προσπαθήστε να κατανοήσετε την χρήση της εντολή find()

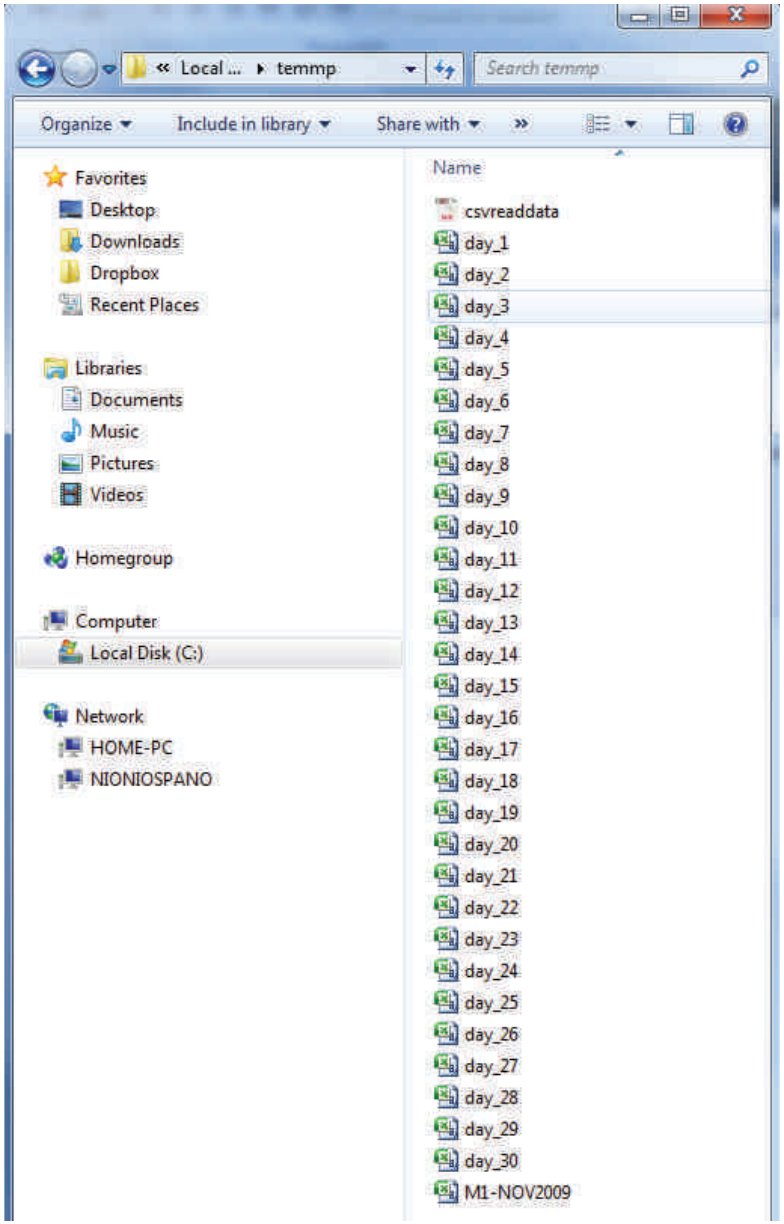
```
//split the month file to days
for i=1:30
    [row,column]=find(x>=real(i) & x<real(i+1));
    dayx=x(row);
    dayy=y(row);
    day=[dayx,dayy];
    fname="day_"+string(i)+".csv";
    csvWrite(day,fname,';');
end
```

Εισαγωγή στο Scilab

Το working directory πριν την εκτέλεση του προγράμματος



Το working directory μετά την εκτέλεση του προγράμματος



Ανάγνωση και επεξεργασία πειραματικών δεδομένων (αρχεία *.csv)

Γραφική Παράσταση των ημερήσιων δεδομένων

Το κομμάτι κώδικα που εισάγαμε στην προηγούμενη διαφάνεια δεν το χρειαζόμαστε πλέον και μπορούμε να το σβήσουμε ή να το κάνουμε comment

```
clear
clc
stacksize("max");
chdir("C:\temmp");
filelist=uigetfile(["*.csv"],"C:\temmp", "Choose files", %t)
bRes = csvDefault("eol","linux")

k=0;
for name=filelist
  k=k+1;
  B = csvRead(name,';');
  x=B(:,1);
  y=B(:,2);
  [n1,n2]=size(x);
```

Μετρητής (counter)

Εάν εισάγουμε το παρακάτω κομμάτι κώδικα στην **ΠΕΡΙΟΧΗ ΕΝΤΟΛΩΝ** τότε μπορούμε να κάνουμε με το πάτημα ενός κουμπιού τις γραφικές παραστάσεις για όλα τα ημερήσια αρχεία.

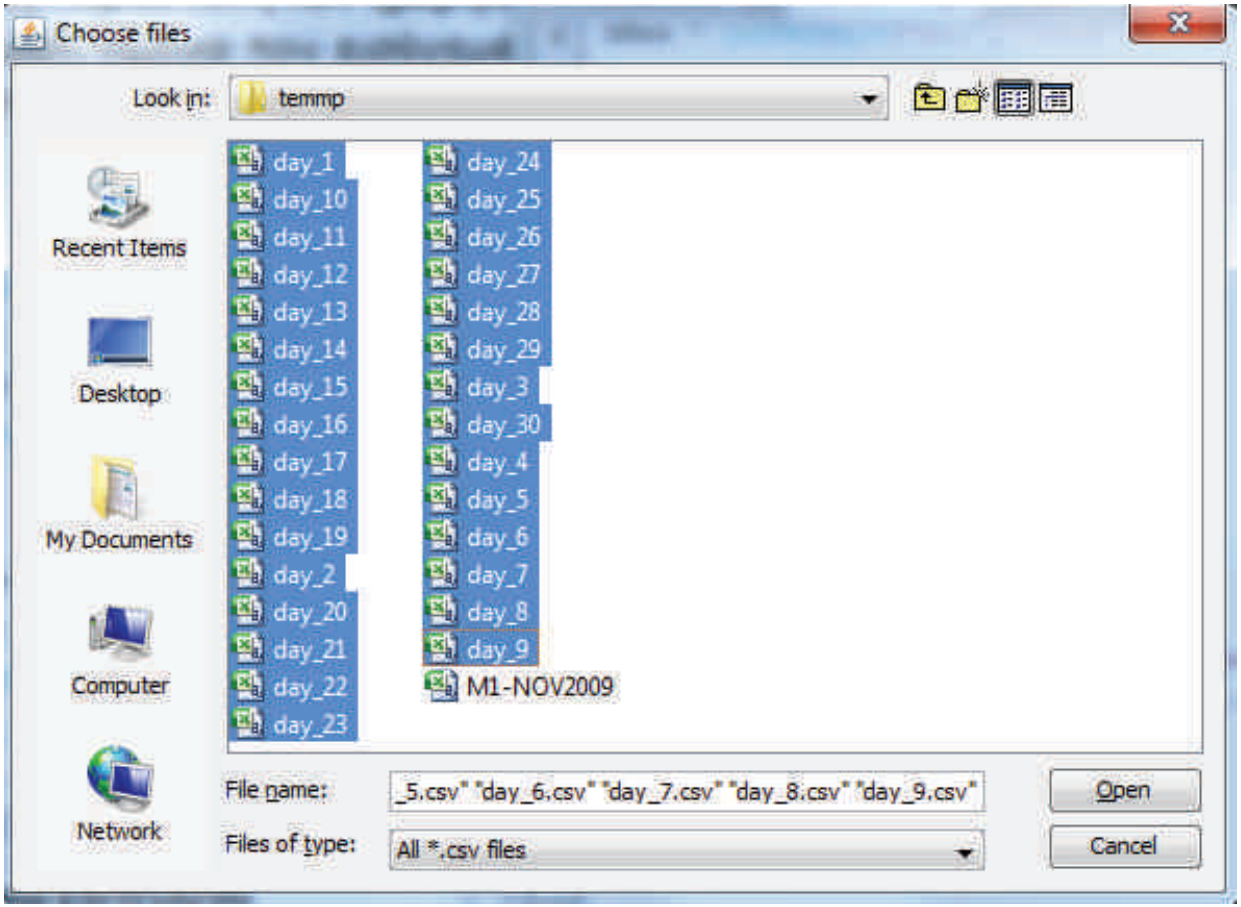
ΠΕΡΙΟΧΗ ΕΝΤΟΛΩΝ
Εδώ θα μπουν οι εντολές
Και τα κομμάτια του κώδικα για την επεξεργασία των δεδομένων που διαβάστηκαν παραπάνω

```
scf(k);
plot(x,y,'-b');
xtitle(name)
xset("font size",3)
ylabel("$Solar \quad Irradiance \quad ( W m^{-2} )$", "fontsize", 4)
xlabel("$Day \quad number$", "fontsize", 4)
hl=legend(['Irradiance']);
```

```
end
```

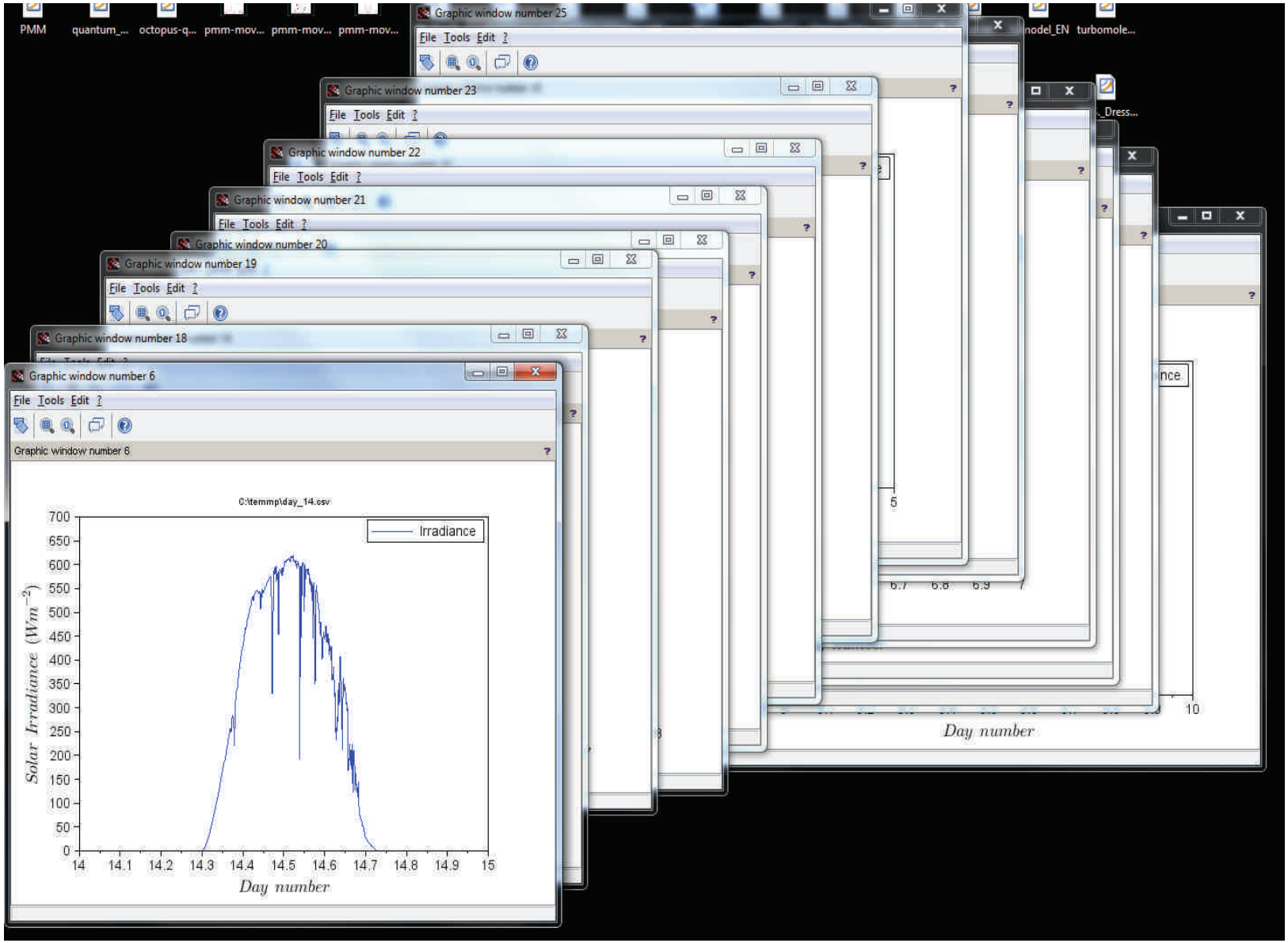
Εκτέλεση του προγράμματος για τις γραφικές παραστάσεις

Επιλέγουμε από το παράθυρο που ανοίγει όλα τα αρχεία που θέλουμε να επεξεργαστούμε (στην προκειμένη περίπτωση να κάνουμε την γραφική τους παράσταση)



Εισαγωγή στο Scilab

Όλες οι γραφικές παραστάσεις εμφανίζονται αυτόματα η μία μετά την άλλη



Ανάγνωση και επεξεργασία πειραματικών δεδομένων (αρχεία *.csv)

Υπολογισμός της ημερήσιας απολαβής ενέργειας

Εάν στο κομμάτι κώδικα που εισάγαμε στην **ΠΕΡΙΟΧΗ ΕΝΤΟΛΩΝ** στην προηγούμενη διαφάνεια τροποποιηθεί ελάχιστα, τότε μπορούμε να υπολογίσουμε εύκολα και την ημερήσια απολαβή ενέργειας. Για να το πετύχουμε αυτό θα πρέπει ο οριζόντιος άξονας να μετράει τον χρόνο σε sec και να υπολογίσουμε αριθμητικά το εμβαδόν της περιοχής που βρίσκεται κάτω από την καμπύλη της πυκνότητας ισχύος.

```
scf(k);
plot(x,y,'-b');
xtitle(name)
xset("font size",3)
ylabel("$Solar \quad Irradiance \quad ( W m^{-2} )$", "fontsize", 4)
xlabel("$Day \quad number$", "fontsize", 4)
hl=legend(['Irradiance']);
t=1:nl;
t=t*60;
total_energy(k)=inttrap(t,y)
```

Υπολογισμός των χρονικών στιγμών κατά τις οποίες πραγματοποιείται μέτρηση σε sec

Αριθμητικός υπολογισμός του εμβαδού, με την μέθοδο του τραπεζίου

Εάν θέλουμε να κάνουμε plot την μεταβολή της ημερήσιας απολαβής ενέργειας, τότε **μπορούμε μετά το end του βασικού βρόγχου** να γράψουμε :

```
xx=1:k;
plot(xx,total_energy)
```