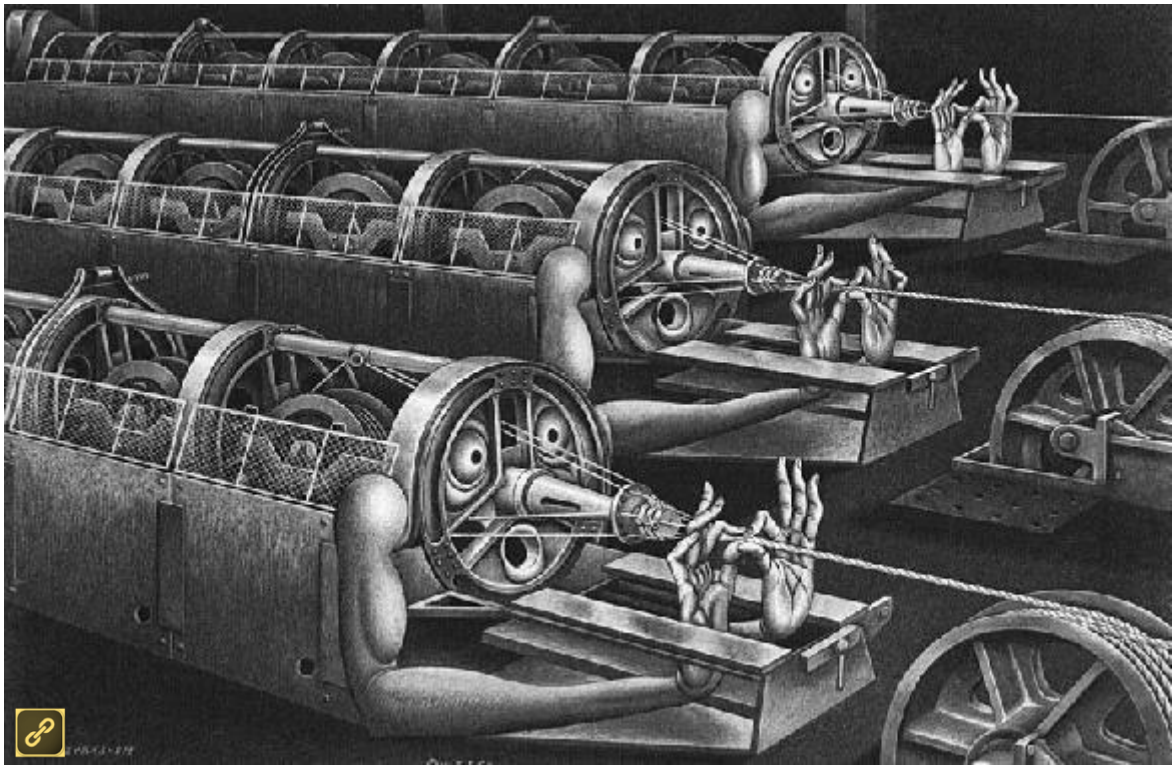
ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

Εισαγωγή στον Προγραμματισμό Η/Υ για Χημικούς Μηχανικούς

Παρουσίαση Διαλέξεων: 12. Εφαρμογές

Καθηγητής Δημήτρης Ματαράς

Copyright © 2014 by Prof. D. S. Mataras (mataras@upatras.gr). This work is made available under the terms of the Creative Commons Attribution-Noncommercial-NoDerivs 3.0 license, http://creativecommons.org/licenses/by-nc-nd/3.0/



"Σωστή παρατήρηση, είπε ο Τρερλ. Θα πρέπει να βρούμε μια λύση...
το κυριότερο όμως, φυσικά, είναι ο αλγόριθμος.

Και ένα μωρό παιδί το ξέρει αυτό! Τι είναι ένα ζώο χωρίς αλγόριθμο;"

'*Κυβεριάδα*', Στανισλάβ Λεμ

Βασική Στατιστική

Μια συλλογή n αριθμητικών δεδομένων χαρακτηρίζεται συνήθως από δύο μεγέθη: τον *αριθμητικό μέσο όρο* και την *τυπική απόκλιση*. Ο πρώτος δίνεται από τον πολύ γνωστό τύπο: $\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n}$

Η τυπική απόκλιση είναι η τετραγωνική ρίζα της διακύμανσης s^2 των τιμών από τον μέσο όρο: $s^2 = \frac{\sum_{i=1}^{n}(x_i - \overline{x})}{n-1}$

ή αλλιώς: $s = \sqrt{\frac{\sum x_i^2 - \frac{(\sum x_i)^2}{n}}{n-1}}$

Η τυπική απόκλιση έχει την ιδιότητα ότι το 68% των τιμών θα βρίσκεται στο διάστημα $\pm s$, το 95% στο $\pm 2s$ και το 98% στο $\pm 3s$.

Βασική Στατιστική

II.

```
module stat_tools
   implicit none

   type exp_data
      real(8),allocatable :: x(:)
      real(8),allocatable :: y(:)
   end type exp_data

   type stats
      real(8)::stdev
      real(8)::x_
      real(8)::variance
   end type stats

contains
```

Βασική Στατιστική

III.

```
pure function normal_prob(x1, x2, step) result(g)
   real,intent(in) :: x1, x2, step
   type(exp_data)  :: g

!  τοπικές μεταβλητές:
   real, parameter :: pi = 4 * atan(1.)
   integer         :: i
   real            :: y0

   allocate(g%x(nint(1 + (x2 - x1) / step)))
   g%y = g%x ! Allocation F03

   y0 = 1 / sqrt(2 * pi)
   do i = 1, size(g%x)
      g%x(i) = x1 + (i - 1) * step
      g%y(i) = y0 * exp(-g%x(i)**2 / 2)
   end do
end function normal_prob
```

$$\varphi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$

Βασική Στατιστική


```
   pure subroutine statistics(sample, test)
      type(exp_data), intent(in) :: sample
      type(stats),    intent(out):: test

   !  τοπικές μεταβλητές:
      real(8) :: sumx, sumx2
      real(8) :: nr

      nr            =  real(size(sample%y),8)
      sumx          =  sum (sample%y)
      sumx2         =  sum (sample%y**2)
      test%x_       =  sumx / nr
      test%variance= (sumx2 - sumx**2 / nr)/(nr - 1)
      test%stdev    =  sqrt(test%variance)

   end subroutine statistics

end module stat_tools
```

Βασική Στατιστική

V.

```
 program average
    use stat_tools
    implicit none

! δηλώσεις:
   type(exp_data):: sample
   type(stats)   :: results
   character(40) :: w_form

! αρχή:
   write(w_form,*)'(a20,1x,g15.8)'
   sample = normal_prob(x1=-3.0, x2=3.0, step=0.3)
   call statistics(sample, results)

   print w_form,'average           =', results%x_
   print w_form,'variance          =', results%variance
   print w_form,'standard deviation=', results%stdev

end program average
```

```
average           =  0.15848129
variance          =  0.20643327E-01
standard deviation=  0.14367786
```

Η μέθοδος των ελαχίστων τετραγώνων

I.

Έστω n ζεύγη πειραματικών τιμών $(x,\ y)$ που υποθέτουμε ότι συνδέονται με μια γραμμική σχέση: $y = a \cdot x + b$
Μπορούμε να υπολογίσουμε την βέλτιστη ευθεία με την βοήθεια της μεθόδου των ελαχίστων τετραγώνων (μέθοδος γραμμικής παλινδρόμησης).

Η *κλίση* της βέλτιστης ευθείας δίνεται από την σχέση: $a = \frac{\sum x_i y_i - \bar{y}\sum x_i}{\sum x_i^2 - \bar{x}\sum x_i}$ και η *τομή* της από τη σχέση: $b = \bar{y} - a \cdot \bar{x}$

Το *τυπικό σφάλμα* υπολογίζεται από την σχέση: $s_{y,x}^2 = \frac{\sum y_i^2 - b\sum y_i - a\sum x_i y_i}{n-2}$ και έχει ιδιότητες παρόμοιες με την τυπική απόκλιση.
Δηλαδή, το 68%, 95% και 99.7% των μετρήσεων θα περιέχονται στα διαστήματα $\pm s_{y,x}$, $\pm 2s_{y,x}$ και $\pm 3s_{y,x}$ αντίστοιχα. Τέλος ένα ποσοτικό μέτρο της εφαρμοσιμότητας του γραμμικού μοντέλου στις μετρήσεις μας δίνεται από τον *συντελεστή συσχέτισης*:

$$R = \frac{n\sum x_i - \sum x_i \sum y_i}{\sqrt{\left[n\sum x_i^2 - (\sum x_i)^2\right]\left[n\sum y_i^2 - (\sum y_i)^2\right]}}$$

Όταν το $R = 1$ όλα τα σημεία είναι επάνω στην ευθεία, ενώ όταν το $R = 0$ τα σημεία δεν συσχετίζονται με την ευθεία.

Η μέθοδος των ελαχίστων τετραγώνων

II.

```
module least_squares
   private

   type lsquared
      real,allocatable::x(:)     ! τιμές ανεξάρτητης μεταβλητής
      real,allocatable::y(:)     ! τιμές εξαρτημένης μεταβλητής
      real              ::a      ! κλίση της ευθείας
      real              ::b      ! τομή της ευθείας
      real              ::s_err  ! τυπικό σφάλμα
      real              ::r      ! συντελεστής συσχέτισης
      character(40)     ::filename ! όνομα αρχείου δεδομένων
   end type lsquared

   public:: lsquared, read_in, l_squares

contains
```

Τα Πάντα Όλα

Η μέθοδος των ελαχίστων τετραγώνων

III.

```
subroutine read_in(line) ! Διαβάζει τις τιμές από το αρχείο
   type(lsquared):: line
   integer        :: ioerr, i, n = 0
   real           :: a; character(80):: msg
  open  (10, file = line%filename)
     do
        read(10, *, iostat = ioerr, iomsg = msg) a
        if (ioerr /= 0)    then
           print *, msg;   exit
        else
           n = n + 1  ! Προσδιορίζω το πλήθος των τιμών
        end if
     end do
  rewind (10)
     allocate(line%x(n), line%y(n))
     do i = 1, n      ! Διαβάζω τις τιμές
        read(10,*) line%x(i), line%y(i)
     end do
  close (10)
end subroutine read_in
```

Η μέθοδος των ελαχίστων τετραγώνων

IV.

```
subroutine l_squares(line)
!  εικονικές μεταβλητές:
   type(lsquared):: line

!  τοπικές μεταβλητές:
   real:: ni, sx, sy, x_, y_, sx2, sy2, sxy

!  υπολογισμοί όρων
   ni    = real(size(line%x))
   sx    = sum(line%x)              !άθροισμα x(i)
   sy    = sum(line%y)              !άθροισμα y(i)
   x_    = sx/ni                    !μέσος όρος x(i)
   y_    = sy/ni                    !μέσος όρος y(i)
   sx2   = sum(line%x * line%x)     !άθροισμα x(i)**2
   sy2   = sum(line%y * line%y)     !άθροισμα y(i)**2
   sxy   = sum(line%x * line%y)     !άθροισμα x(i)*y(i)
```

Η μέθοδος των ελαχίστων τετραγώνων

V.

```
   !  υπολογισμός y=a*x+b
      line%a = (sxy - sx * y_) / (sx2 - sx * x_)  !κλίση
      line%b =  y_- line%a * x_                   !τομή

   !  υπολογισμός τυπικού σφάλματος
      line%s_err = sqrt( (sy2 - line%b * sy  &
                          &  - line%a * sxy) / (ni-2.) )

   !  υπολογισμός συντελεστή συσχέτισης
      line%r =   (ni * sxy - sx * sy) /  &
         & sqrt( (ni * sx2 - sx**2) *    &
         &       (ni * sy2 - sy**2) )

   end subroutine l_squares

end module least_squares
```

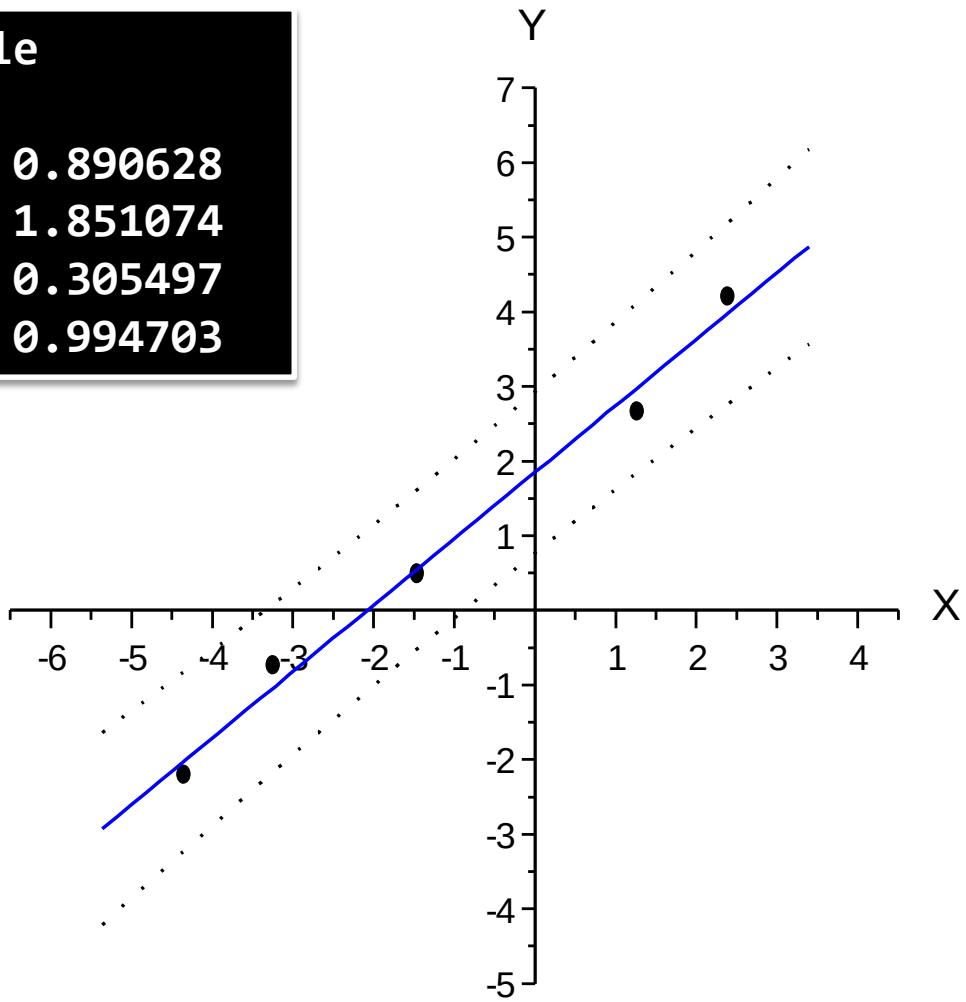
Η μέθοδος των ελαχίστων τετραγώνων

VI.

```
program linear  !ελάχιστα τετράγωνα
   use least_squares
   implicit none

! δηλώσεις:
   type(lsquared):: line; character(40) :: w_form
! αρχή:
   write (w_form,*) '(a10,x,f8.6)'
   line%filename = 'data.dat'

   call read_in(line)          !διαβάζω τα x, y
   call l_squares(line)        !υπολογίζω

   print w_form,'slope    =', line%a
   print w_form,'intercept=', line%b
   print w_form,'st_error =', line%s_err
   print w_form,'r        =', line%r

end program linear
```

Η μέθοδος των ελαχίστων τετραγώνων

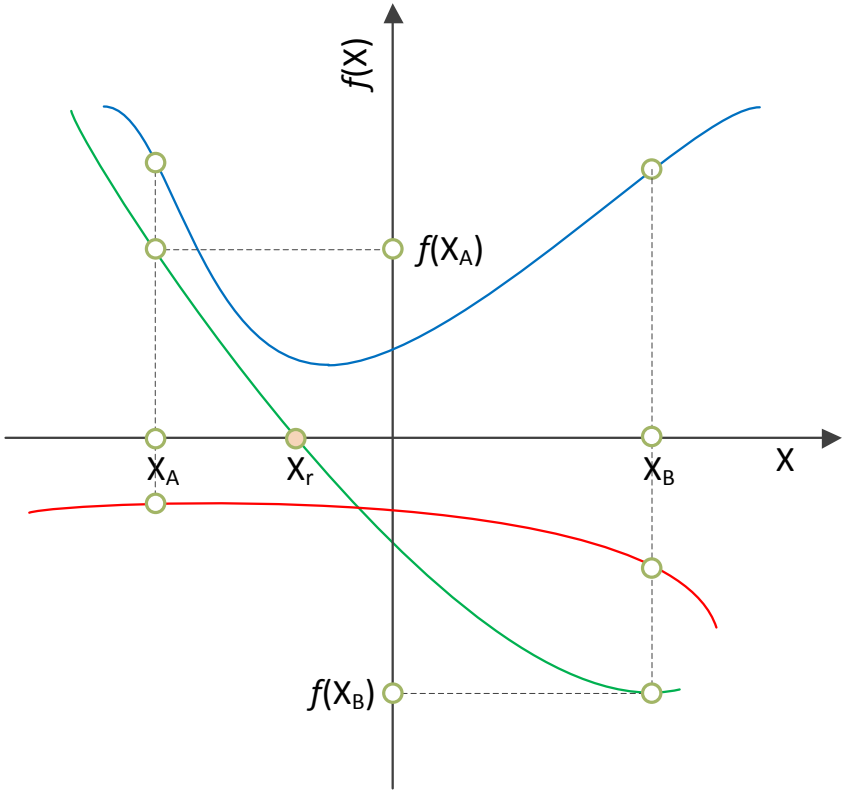

```
 End of file

slope    = 0.890628
intercept= 1.851074
st_error = 0.305497
r        = 0.994703
```



Μη γραμμικές εξισώσεις

Γραμμική εξίσωση: $f(x) = ax + b$

- Οτιδήποτε άλλο (π.χ. πολυώνυμο, τριγωνομετρική, λογαριθμική, εκθετική κ.λ.) είναι *μη γραμμική εξίσωση*
- Ρίζα της εξίσωσης είναι η τιμή του x για την οποία $f(x) = 0$
- Στην πλειοψηφία των περιπτώσεων η επίλυση είναι πρακτικά εφικτή μόνο με τη χρήση επαναληπτικών μεθόδων ξεκινώντας από μια καλή αρχική πρόβλεψη της λύσης
- Σε αυτή την περίπτωση το αποτέλεσμα εξαρτάται από την αρχική πρόβλεψη
- Επικουρικά μπορεί κανείς να χρησιμοποιήσει την γραφική αναπαράσταση της συνάρτησης για να προσδιορίσει χονδρικά τη ρίζα.



- Παρατηρείτε από το σχήμα ότι μόνο η πράσινη συνάρτηση έχει ρίζα. Δηλαδή μια συνάρτηση έχει ρίζα στο διάστημα X_A, X_B όταν:

$$f(X_A) \cdot f(X_B) < 0$$

Βηματική Προσέγγιση

σε ψευδοκώδικα 

1. Ελέγχουμε αν υπάρχει ρίζα στα όρια του διαστήματος **A, B** και αν δεν υπάρχει ζητάμε καινούρια A,B
2. Σαρώνουμε το **X** αρχίζοντας από το **A** μέχρι το **B** με βήμα **ΔX**
3. Υπολογίζουμε το **Y <- Φ(X)** για κάθε **X**
4. Αν φτάσαμε στο άνω όριο του διαστήματος **(X=B)** τότε θέτουμε το **X <- A-ΔX** και το **ΔX <- ΔX/2**. Δηλαδή υποδιπλασιάζουμε το βήμα και ξαναρχίζουμε από την αρχή του διαστήματος
5. Επαναλαμβάνουμε τα βήματα 2 μέχρι 5 εως ότου η απόλυτη τιμή του **Y** να γίνει μικρότερη από την επιθυμητή ακρίβεια

Αν θέλετε να το δοκιμάσετε κατεβάζετε το διερμηνευτή της γλώσσας από τη διεύθυνση:

http://alkisg.mysch.gr/steki/index.php?action=dlattach;topic=4061.0;attach=2539

Βηματική Προσέγγιση

σε ψευδοκώδικα 

```
ΠΡΟΓΡΑΜΜΑ βηματική_προσέγγιση

ΜΕΤΑΒΛΗΤΕΣ
   ΠΡΑΓΜΑΤΙΚΕΣ: Χ, Υ, ΔΧ, Α, Β

ΑΡΧΗ
   Α <- 0
   Β <- 0
   ΟΣΟ Φ(Α)*Φ(Β) > 0 ΕΠΑΝΑΛΑΒΕ     !1. έλεγχος ορίων
     ΓΡΑΨΕ 'δώσε τα όρια του διαστήματος'
     ΔΙΑΒΑΣΕ Α, Β
   ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

   Υ  <- 0
   ΔΧ <- 0.1                        !αρχικό βήμα
   Χ  <- Α - ΔΧ                     !αρχή του διαστήματος
```

Βηματική Προσέγγιση

σε ψευδοκώδικα 

```
   ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
      Χ <- Χ + ΔΧ              !2. σάρωση του διαστήματος
      Υ <- Φ(Χ)                !3. υπολογισμός Υ
      ΑΝ Χ >= Β ΤΟΤΕ            !4. τέλος του διαστήματος
        ΔΧ <- ΔΧ/2
        Χ <- Α - ΔΧ
        ΓΡΑΨΕ 'Το βήμα έγινε: ', ΔΧ
      ΤΕΛΟΣ_ΑΝ
   ΜΕΧΡΙΣ_ΟΤΟΥ Α_Τ(Υ) < 0.0001 !5. συνθήκη εξόδου

   ΓΡΑΨΕ 'στο Χ= ', Χ, '  το Υ είναι:', Υ

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ βηματική_προσέγγιση

ΣΥΝΑΡΤΗΣΗ Φ(Χ): ΠΡΑΓΜΑΤΙΚΗ
ΜΕΤΑΒΛΗΤΕΣ
   ΠΡΑΓΜΑΤΙΚΕΣ: Χ
ΑΡΧΗ
   Φ <- 2*Χ^4 - 3*Χ^3 - 2*Χ - 1
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

Βηματική Προσέγγιση

σε ψευδοκώδικα 

```
δώσε τα όρια του διαστήματος
2
3
δώσε τα όρια του διαστήματος
-1
1
Το βήμα έγινε: 0.0500000
Το βήμα έγινε: 0.0250000
Το βήμα έγινε: 0.0125000
Το βήμα έγινε: 0.0062500
Το βήμα έγινε: 0.0031250
Το βήμα έγινε: 0.0015625
Το βήμα έγινε: 0.0007813
Το βήμα έγινε: 0.0003906
Το βήμα έγινε: 0.0001953
Το βήμα έγινε: 0.0000977
Το βήμα έγινε: 0.0000488
στο x= -0.3888672  το y είναι:0.0000658
```

Ενθετικές μέθοδοι: διχοτόμηση

- Στις ενθετικές μεθόδους χρειαζόμαστε 2 τιμές X_A, X_B ανάμεσα στις οποίες προβλέπουμε ότι βρίσκεται η ρίζα
- Η γνωστότερη από τις ενθετικές μεθόδους είναι η μέθοδος υποδιπλασιασμού του διαστήματος ή μέθοδος διχοτόμησης (διχοτομική αναζήτηση).
- Στη μέθοδο αυτή θεωρούμε ότι η ρίζα βρίσκεται στη μέση του διαστήματος και κατόπιν ελέγχουμε σε ποιο από τα δύο υποδιαστήματα που δημιουργούνται αλλάζει πρόσημο η συνάρτηση. Συνεχίζουμε τον υποδιπλασιασμό μέχρι να προσδιορίσουμε τη ρίζα με την επιθυμητή ακρίβεια

1. Ελέγχουμε αν υπάρχει ρίζα στα όρια του διαστήματος **A, B** και αν δεν υπάρχει ζητάμε καινούρια A,B
2. Θέτουμε $X_r \leftarrow (X_A + X_B)/2$
3. Αν $f(X_A) \cdot f(X_r) < 0$ τότε θέτουμε $X_B \leftarrow X_r$
4. Αλλιώς αν $f(X_A) \cdot f(X_r) > 0$ τότε θέτουμε $X_A \leftarrow X_r$
5. Επαναλαμβάνουμε τα βήματα 2 μέχρι 5 εως ότου ικανοποιηθεί το κριτήριο σύγκλισης

Ενθετικές μέθοδοι: διχοτόμηση

I.

```
program bisection
   implicit none

! δηλώσεις:
   real    :: old_xr, error, xa, xb, check
   real    :: xr = 0, errmax = 1e-4
   integer :: iter = 0, maxiter = 40

! αρχή:
   do
      print *, 'give limits xa and xb:'
      read *, xa, xb

      if(f(xa) * f(xb) < 0) exit !1. έλεγχος ορίων

   end do
```

Ενθετικές μέθοδοι: διχοτόμηση


```
do
   iter = iter + 1 !αριθμητής πλήθους επαναλήψεων
   old_xr = xr

   xr = (xa + xb) / 2.      !2. διχοτόμηση διαστήματος
   if(xr /= 0) error = errf(xr,old_xr)

   check = f(xa) * f(xr)
   if(check < 0)       then !3.
      xb = xr
   elseif(check > 0)   then !4.
      xa = xr
   else
      error = 0
   endif

   print*, iter, xr, f(xr), error
   if((error < errmax).or.(iter >= maxiter))exit
end do                          !5.
```

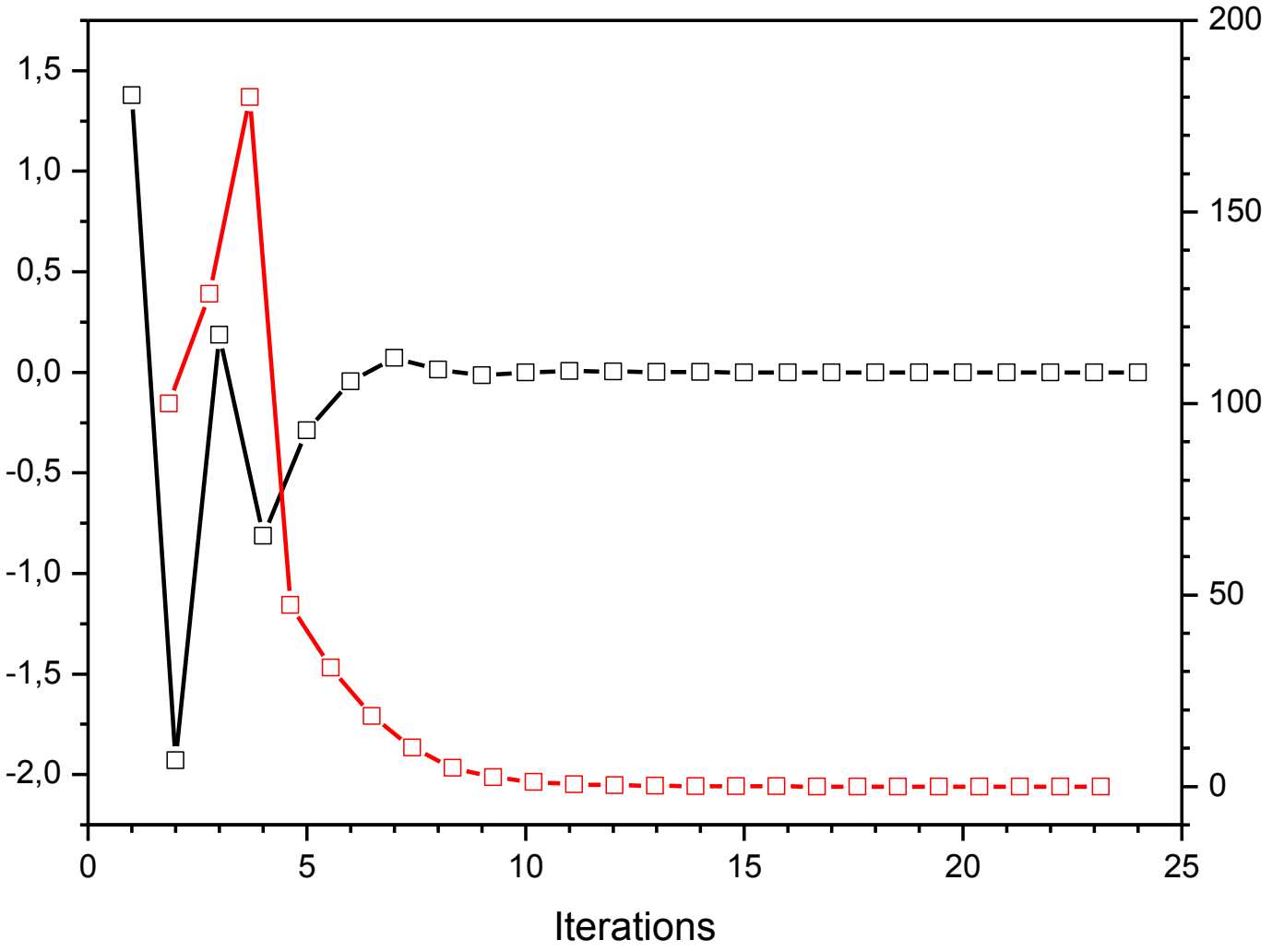
Ενθετικές μέθοδοι: διχοτόμηση

III.

```
contains

   real function f(x)                    !ορισμός συνάρτησης
      real:: x

      f = x + cos(x)

   end function f

   real function errf(xnew,xold)    !υπολογισμός σφάλματος
      real:: xold, xnew

      errf = abs( (xnew - xold) / xnew) * 100
   end function errf

end program bisection
```

```
Give limits Xa and Xb:
-4  5
          1  0.50000000       1.3775826       100.00000
          2  -1.7500000      -1.9282460       128.57143
          3 -0.62500000       0.18596312      180.00000
          4  -1.1875000      -0.81352037      47.368420
          5 -0.90625000      -0.28954792      31.034483
          6 -0.76562500      -4.46756184E-02  18.367348
          7 -0.69531250       7.25410432E-02  10.112360
          8 -0.73046875       1.43929757E-02  4.8128343
          9 -0.74804688      -1.50280772E-02  2.3498695
         10 -0.73925781      -2.89009156E-04  1.1889036
         11 -0.73486328       7.05914712E-03  0.59800667
         12 -0.73706055       3.38685629E-03  0.29811195
         13 -0.73815918       1.54937000E-03  0.14883414
         14 -0.73870850       6.30291994E-04  7.43617266E-02
         15 -0.73898315       1.70669300E-04  3.71670462E-02
         16 -0.73912048      -5.91629541E-05  1.85800698E-02
         17 -0.73905182       5.57549174E-05  9.29089822E-03
         18 -0.73908615      -1.70358328E-06  4.64523304E-03
         19 -0.73906898       2.70257751E-05  2.32267054E-03
         20 -0.73907757       1.26611230E-05  1.16132176E-03
         21 -0.73908186       5.47877698E-06  5.80657506E-04
         22 -0.73908401       1.88759850E-06  2.90327909E-04
         23 -0.73908508       9.20080225E-08  1.45163751E-04
         24 -0.73908561      -8.05787522E-07  7.25818245E-05
```

Ενθετικές μέθοδοι: διχοτόμηση



Ενθετικές μέθοδοι: διχοτόμηση

σε C++

```
#include <iostream>
#include <cstdlib>
#include <cmath>
using namespace std;

int main() {
  system("chcp 1253");
  double f(double), errf(double, double);
  double Xa, Xb, Xr, error, check,
         oldXr = 0, errmax = 1e-5;
  int iter = 0;

  do {

     cout << "δώσε τα όρια Xa και Xb:" << endl;
     cin  >> Xa >> Xb;

  } while (f(Xa) * f(Xb) > 0); //1. έλεγχος ορίων
```

Ενθετικές μέθοδοι: διχοτόμηση

σε C++

```
do {
   iter++; //αριθμητής πλήθους επαναλήψεων
   oldXr = Xr;
   Xr = (Xa + Xb) / 2;      //2. διχοτόμηση διαστήματος
   if (Xr != 0) error = errf(Xr, oldXr);

   check = f(Xa) * f(Xr);
   if (check < 0)           //3.
     Xb = Xr;
   else if (check > 0)      //4.
     Xa = Xr;
   else
     error = 0;

   cout << iter << "\t" << Xr << "\t" << f(Xr) <<
           "\t" << error << endl;
} while (error > errmax); //5.
  return 0;
}
```

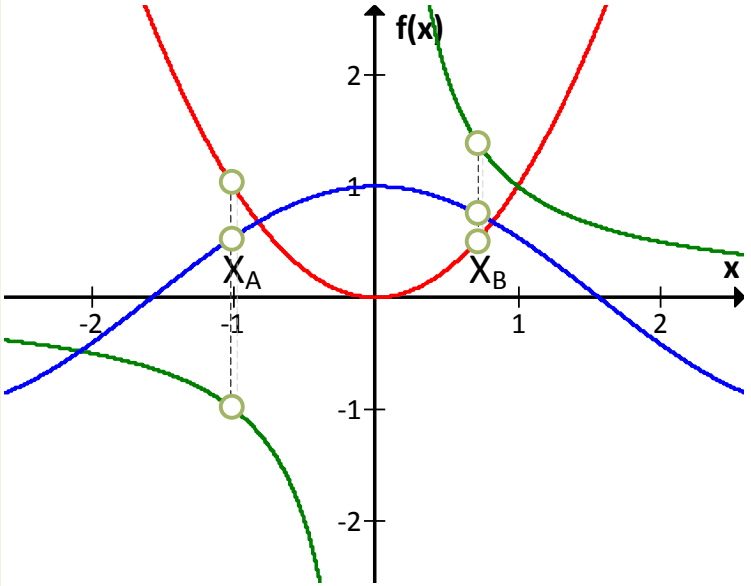
Ενθετικές μέθοδοι: διχοτόμηση

σε C++

```
double f(double x){                    //ορισμός συνάρτησης
  double y;
  y = x + cos(x);
  return y;
}

double errf(double x1,double x0){ //υπολογισμός σφάλματος
  double y;
  y = abs( (x1 - x0) / x1) * 100;
return y;
}
```

Ενθετικές μέθοδοι: διχοτόμηση

- Η διχοτόμηση συγκλίνει πάντα αλλά συγκλίνει αργά, ιδιαίτερα αν ένα από τα δύο όρια βρίσκεται πολύ κοντά στη ρίζα.
- Η διχοτόμηση θα βρει μια ρίζα ανεξάρτητα από το πόσες υπάρχουν στο διάστημα που ερευνούμε.
- Στο σχήμα δεξιά έχουμε τη γραφική παράσταση τριών απλών συναρτήσεων:
  - $f(x) = x^2$, $cos(x)$ και $\frac{1}{x}$
- Η σχέση $f(X_A) \cdot f(X_B) < 0$ δεν ισχύει για την πρώτη συνάρτηση σε καμιά περίπτωση ενώ υπάρχει ρίζα.
- Η ισχύς της σχέσης για τη δεύτερη συνάρτηση εξαρτάται από τα όρια που θα δώσουμε. π.χ. για X_A=-2, X_B=2 δεν ισχύει ενώ υπάρχουν ρίζες.
- Η σχέση ισχύει για την τρίτη συνάρτηση ενώ προφανώς δεν υπάρχουν ρίζες!



Ενθετικές μέθοδοι: αναδρομική διχοτόμηση

I.

```
program recursive_bisection
   implicit none
   interface
      real function f(x)
         real, intent (in) :: x
      end function
   end interface
   real:: xa = 1, xb = 1, x = 1
   real:: errmax = 1.E-6

   call execute_command_line('chcp 1253')
   do while (f(xa) * f(xb) > 0)
      print *,'Δώσε τα όρια xa, xb'
      read  *, xa, xb
   end do

   x = bisect(f, xa, xb, errmax)
   print   '(a17, f10.7, a6, e14.7)',&
         &  'η λύση είναι: x=', x, 'f(x)=', f(x)
```

Ρητή Διεπιφάνεια Εξωτερικής Διαδικασίας

Ενθετικές μέθοδοι: αναδρομική διχοτόμηση ιδιότητα SAVE II.

```
contains
   real recursive function bisect(f,xa,xb,errmax) result(x)
      interface
         real function f(x)
            real, intent (in) :: x
         end function
      end interface
      real, intent(in) :: errmax
      real             :: xa, xb, xr, errors

      integer,save :: i
      real,   save :: s

      character(50):: ff = '(i4,a3,f10.7,a6,e14.7,a7,e14.7)'

      i  = i + 1              !αρίθμηση επαναλήψεων
      xr = (xa + xb) / 2.
      errors = abs( (xr - s) / xr) * 100
      print ff, i, 'x=', xr, 'f(x)=', f(xr), 'error=', errors
```

Ρητή Διεπιφάνεια Εξωτερικής Διαδικασίας

SAVE
οι τιμές διατηρούνται

Ενθετικές μέθοδοι: αναδρομική διχοτόμηση

III.

```
      if (errors > errmax)        then
         s = xr  !κρατάμε την προηγούμενη τιμή
         if (f(xa) * f(xr) < 0)  then
            xb = xr
         else     !(f(xr) * f(xb) < 0)
            xa = xr
         end if
         x = bisect(f, xa, xb, errmax)!αναδρομική κλήση
      else
         x = xr
      end if
   end function bisect
end program recursive_bisection
```

```
real function f(x)
    real, intent(in):: x
    f = x + cos(x)
end function f
```

Εξωτερική Διαδικασία

```
 Δώσε τα όρια xa, xb
-2 1
   1 x=-0.5000000 f(x)= 0.3775826E+00 error= 0.1000000E+03
   2 x=-1.2500000 f(x)=-0.9346777E+00 error= 0.6000000E+02
   3 x=-0.8750000 f(x)=-0.2340031E+00 error= 0.4285714E+02
   4 x=-0.6875000 f(x)= 0.8533495E-01 error= 0.2727273E+02
   5 x=-0.7812500 f(x)=-0.7121611E-01 error= 0.1200000E+02
   6 x=-0.7343750 f(x)= 0.7874725E-02 error= 0.6382979E+01
   7 x=-0.7578125 f(x)=-0.3147121E-01 error= 0.3092783E+01
   8 x=-0.7460938 f(x)=-0.1174782E-01 error= 0.1570681E+01
   9 x=-0.7402344 f(x)=-0.1923873E-02 error= 0.7915567E+00
  10 x=-0.7373047 f(x)= 0.2978603E-02 error= 0.3973510E+00
  11 x=-0.7387695 f(x)= 0.5281584E-03 error= 0.1982816E+00
  12 x=-0.7395020 f(x)=-0.6976590E-03 error= 0.9904259E-01
  13 x=-0.7391357 f(x)=-0.8470073E-04 error= 0.4954583E-01
  14 x=-0.7389526 f(x)= 0.2217412E-03 error= 0.2477905E-01
  15 x=-0.7390442 f(x)= 0.6852335E-04 error= 0.1238799E-01
  16 x=-0.7390900 f(x)=-0.8087914E-05 error= 0.6193612E-02
  17 x=-0.7390671 f(x)= 0.3021791E-04 error= 0.3096902E-02
  18 x=-0.7390785 f(x)= 0.1106505E-04 error= 0.1548427E-02
  19 x=-0.7390842 f(x)= 0.1488578E-05 error= 0.7742075E-03
  20 x=-0.7390871 f(x)=-0.3299665E-05 error= 0.3871023E-03
  21 x=-0.7390857 f(x)=-0.9055426E-06 error= 0.1935515E-03
  22 x=-0.7390850 f(x)= 0.2915181E-06 error= 0.9677585E-04
  23 x=-0.7390853 f(x)=-0.3070122E-06 error= 0.4838790E-04
  24 x=-0.7390851 f(x)=-0.7747024E-08 error= 0.2419396E-04
  25 x=-0.7390851 f(x)= 0.9200802E-07 error= 0.8064652E-05
  26 x=-0.7390851 f(x)= 0.9200802E-07 error= 0.0000000E+00
 η λύση είναι: x=-0.7390851 f(x)= 0.9200802E-07
```

Διευκρίνιση
ιδιότητα optional εγγενής συνάρτηση present

```
program sum_of_series
    implicit none
    real           :: pi = 4 * atan(1.)
    character(15)  :: frm = '(t5, f8.2)'

    print frm, zumserie(0, 99, 4*atan(1.), 0.1)
    print frm, zumserie(0, 99, d=0.1, s=pi)
    print frm, zumserie(n=99, d=0.1, s=4*atan(1.))
    print frm, zumserie(d=0.1, s=4*atan(1.), n=99)
    print frm, zumserie(m=0, n=99, d=0.1, s=4*atan(1.))

contains

    real function zumserie(m, n, s, d) result(r)
        integer, optional, intent(in):: m
        integer, intent(in)          :: n
        real,    intent(in)          :: s, d
        integer                      :: i, m_t
```

$$\sum_{i=m}^{n} (s + d \times i)$$

Διευκρίνιση
ιδιότητα optional εγγενής συνάρτηση present

```
        if (present(m)) then
            m_t = m
        else
            m_t = 0
        end if

        r = 0

        do  i = m_t, n
            r = r + s + d * i
        end do

    end function zumserie

end program sum_of_series
```

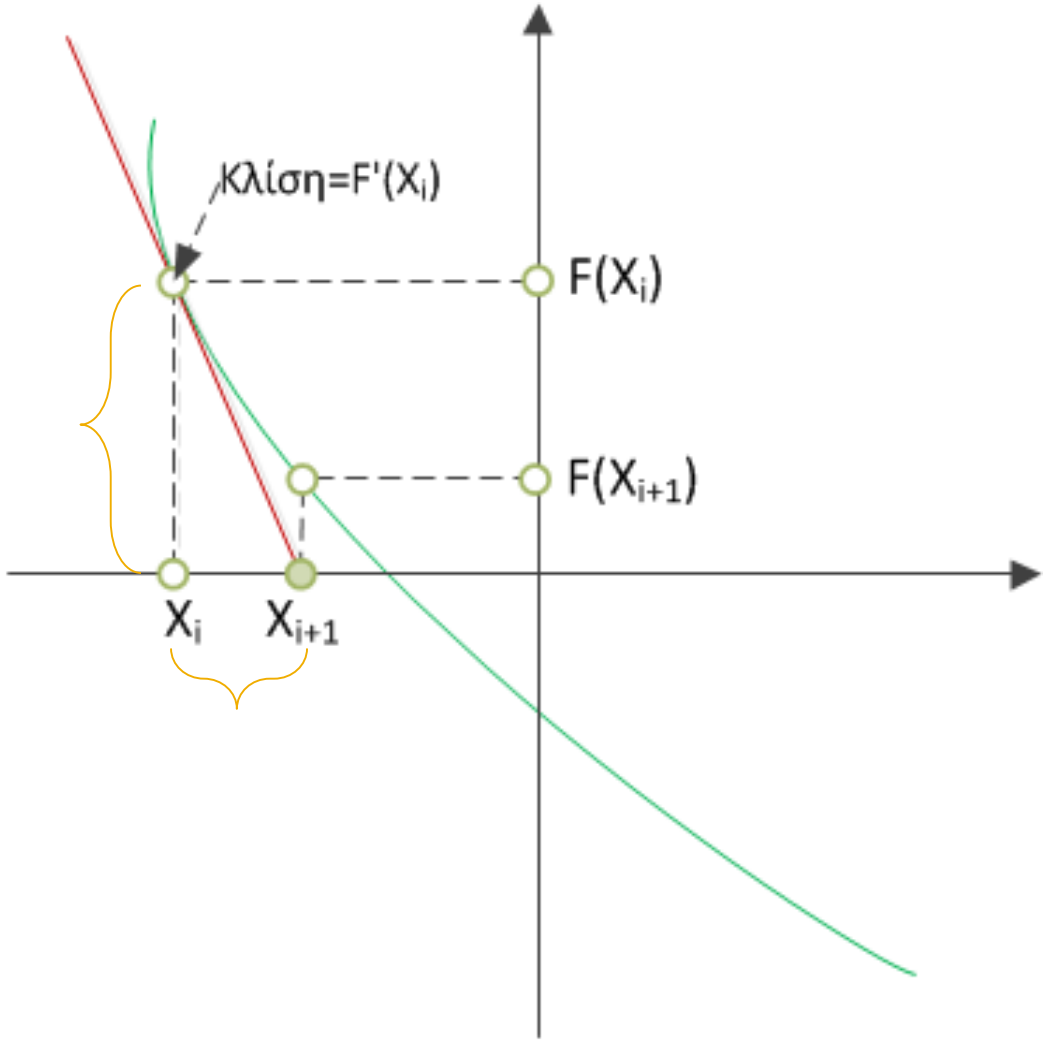
$$\sum_{i=m}^{n} (s + d \times i)$$

Μέθοδος Newton Raphson

$$f'(x_i) = \frac{f(x_i) - 0}{x_i - x_{i+1}}$$

$$\Downarrow$$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$



Μέθοδος Newton Raphson


```
program newton
   implicit none

!  δηλώσεις:
   real    :: oldroot, root, error, errmax = 1e-4
   integer :: iternr = 0, maxiter = 40

!  αρχή:
   print *, 'root estimation?'; read *, root

   do
      iternr  = iternr + 1
      oldroot = root
      root    = f(oldroot)
      if(root /= 0) error = errf(root, oldroot)
      print *, iternr, root, f(root), error

      if( (error < errmax) .or. (iternr >= maxiter))exit
   end do
```

Μέθοδος Newton Raphson

II.

```
contains

   real function f(x)          !ορισμός συνάρτησης
      real:: x

   !  f = x - (f(x) / f'(x))
      f = x - (x + cos(x)) / (1 - sin(x))

   end function f

   real function errf(x1, x0) !υπολογισμός σφάλματος
      real:: x0, x1

      errf = abs(( x1 - x0) / x1) * 100

   end function errf

end program newton
```

```
root estimation?
-0.5
           1 -0.75522238     -0.73914164      33.794334
           2 -0.73914164     -0.73908514      2.1755962
           3 -0.73908514     -0.73908514     7.64529034E-03
           4 -0.73908514     -0.73908514      0.0000000
```

```
root estimation?
8
           1  -730.08325      -354.64548      101.09576
           2  -354.64548      -90.686249      105.86284
           3  -90.686249      -25.616201      291.06863
           4  -25.616201      -8.7333221      254.01912
           5  -8.7333221      -2.9301364      193.31566
           6  -2.9301364      0.29981205      198.05173
           7   0.29981205     -1.4814800      1077.3245
           8  -1.4814800     -0.78394860      120.23734
           9 -0.78394860     -0.73950899      88.976677
          10 -0.73950899     -0.73908520      6.0093408
          11 -0.73908520     -0.73908514     5.73396720E-02
          12 -0.73908514     -0.73908514     8.06465232E-06
```

// MICROSONIC LANDSCAPES

Σημείωμα Αναφοράς

Copyright Πανεπιστήμιο Πατρών, Όνομα μέλους ή μελών ΔΕΠ 2014: Δημήτριος Ματαράς. «Εισαγωγή στον Προγραμματισμό Η/Υ». Έκδοση: 1.0. Πάτρα 2014. Διαθέσιμο από τη δικτυακή διεύθυνση: https://eclass.upatras.gr/courses/CMNG2178.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» έχει χρηματοδοτήσει μόνο την αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σημείωμα Αδειοδότησης

Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά, Μη Εμπορική Χρήση Παρόμοια Διανομή 4.0 [1] ή μεταγενέστερη, Διεθνής Έκδοση. Εξαιρούνται τα αυτοτελή έργα τρίτων π.χ. φωτογραφίες, διαγράμματα κ.λ.π., τα οποία εμπεριέχονται σε αυτό και τα οποία αναφέρονται μαζί με τους όρους χρήσης τους στο «Σημείωμα Χρήσης Έργων Τρίτων».

[1] http://creativecommons.org/licenses/by-nc-sa/4.0/

Ως **Μη Εμπορική** ορίζεται η χρήση:

- που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο
- που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
- που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο

Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.