Μάθημα 4^0

Εύρεση ριζών

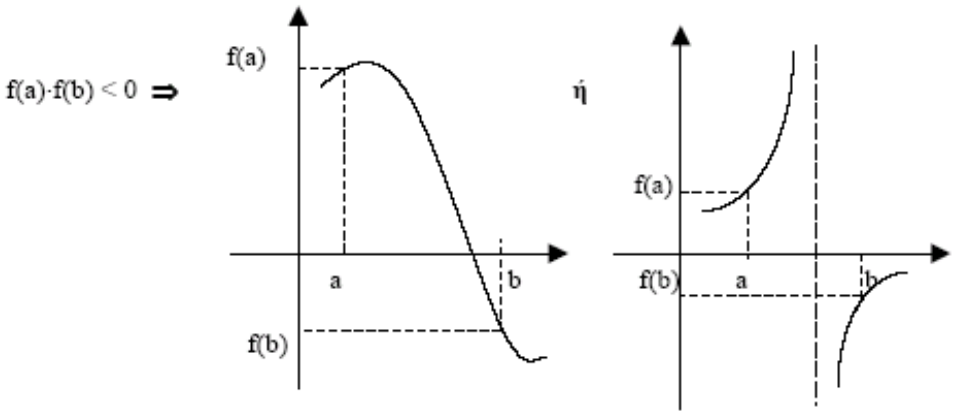
Μας ενδιαφέρουν μέθοδοι επίλυσης μη γραμμικών εξισώσεων με έναν άγνωστο: δοθείσης μιας συνεχούς συνάρτησης f: $\boldsymbol{R} \rightarrow \boldsymbol{R}$, αναζητούμε ένα σημείο $x^* \in \boldsymbol{R}$ τέτοιο ώστε $f(x^*) = 0$. Λέμε ότι x^* είναι μια *λύση* της εξίσωσης $f(x) = 0$ ή μια *ρίζα* της συνάρτησης f.

Στην αριθμητική πεπερασμένης ακρίβειας, όπως είναι η αριθμητική των αριθμών κινητής υποδιαστολής, μπορεί να μην υπάρχει αριθμός μηχανής x^* τέτοιο ώστε $f(x^*)$ να είναι ακριβώς μηδέν. Ένας τρόπος προσέγγισής του είναι να αναζητήσουμε ένα (πολύ) μικρό διάστημα $[a, b]$ στο οποίο η f αλλάζει πρόσημο (το διάστημα αυτό στη διεθνή βιβλιογραφία λέγεται *bracket*). Ένας τρόπος να δουλέψει κανείς είναι:

- Κάνε γραφική παράσταση για μια αρχική εκτίμηση της (των) ρίζας (ριζών) της f
- Με επαναληπτική διαδικασία βελτίωσε την αρχική εκτίμηση (αλγόριθμος υπολογισμού ρίζας)

Το θεώρημα στο οποίο στηριζόμαστε είναι το

Θεώρημα Bolzano ⇒ προσδιορισμός διαστήματος (αρχική εκτίμηση ρίζας) στο οποίο υπάρχει ρίζα.



Το θεώρημα αναφέρεται στην ύπαρξη ρίζας σε διάστημα όταν η συνάρτηση είναι συνεχής σ' αυτό.

Μπορούμε να χρησιμοποιήσουμε τον ακόλουθο αλγόριθμο ο οποίος θα εντοπίζει τα διαστήματα που περιέχουν μια ρίζα της συνάρτησης:

Αλγόριθμος (Bracketing)

Δοσμένα: f(x), x_{min}, x_{max}, n
dx = (x_{max} − x_{min})/n (μήκος διαστήματος)
$x_{left} = x_{min}$
i = 0
while i<n
  i ← i+1
  $x_{right} = x_{left}$ +dx
  if f(x) αλλάζει πρόσημο στο [x_{left}, x_{right}]
    αποθήκευσε [x_{left}, x_{right}] για παραπέρα αναζήτηση
  end
  $x_{left} = x_{right}$
end

➢ Πώς υλοποιείται το "**if** f(x) αλλάζει πρόσημο;"
Μια απλή υλοποίηση είναι

$$f(a)\cdot f(b) < 0$$

ή στο MATLAB

```
fa = …

fb = …
if   fa*fb < 0
       save bracket
end
```

αλλά αυτό το τεστ είναι επιδεικτικό σε underflow.(όταν οι τιμές που αντιστοιχούν στα άκρα του διαστήματος είναι κοντά στο 0, δηλαδή είμαστε κοντά στη ρίζα, ⇒ |fa*fb| ≅0)

Για παράδειγμα,

```
EDU» format long e
EDU» fa=1e-120;
EDU» fb=-2e-300;
EDU» fa*fb

ans =

      0
```

Επειδή fa*fb < realmin ⇒ το αποτέλεσμα underflows.

Ένα πιο αξιόπιστο τεστ χρησιμοποιεί την built-in συνάρτηση **sign** (→ επιστρέφει true αν τα πρόσημα είναι αντίθετα, ανεξάρτητα από το μέγεθος των fa, fb)

```
fa = …
fb = …
if  sign(fa) ~= sign(fb)
   save bracket
end
```

βλ. υλοποίηση στη συνάρτηση **brackPlot**, η οποία:

- ψάχνει για διαστήματα μιας συνάρτησης (ορισμένης από το χρήστη)
- κάνει γραφική παράσταση των διαστημάτων και της f(x)
- επιστρέφει τα άκρα διαστημάτων σε διάστημα 2 στηλών

```
function Xb = brackPlot(fun,xmin,xmax,nx)
% brackPlot  Find subintervals on x that contain sign changes of f(x)
%
% Synopsis:  Xb = brackPlot(fun,xmin,xmax)
%            Xb = brackPlot(fun,xmin,xmax,nx)
%
% Input:     fun = (string) name of mfile function that evaluates f(x)
%            xmin, xmax = endpoints of interval to subdivide into
%            brackets.
%            nx = (optional) number of samples along x axis used to
%                 test for brackets.  The interval xmin <= x <= xmax is
%                 divided into nx-1 subintervals.  Default:  nx = 20.
%
% Output:   Xb = two column matrix of bracket limits. Xb(k,1) is the
%                left (lower x value) bracket and Xb(k,2) is the right
%                bracket for the k^th potential root. If no brackets
%                are found, Xb = [].
```

```
if nargin<4, nx=20; end

% --- Create data for a plot of f(x) on interval xmin <= x <= xmax
xp = linspace(xmin,xmax);  yp = feval(fun,xp);
% --- Save data used to draw boxes that indicate brackets
ytop = max(yp);  ybot = min(yp);          %  y coordinates of the box
ybox = 0.05*[ybot ytop ytop ybot ybot];  %  around a bracket
c = [0.7 0.7 0.7];                    %  RGB color used to fill the box

% --- Begin search for brackets
x = linspace(xmin,xmax,nx);     %  Vector of potential bracket limits
f = feval(fun,x);          %  Vector of f(x) values at potential brackets
nb = 0;  Xb = [];                %  Xb is null unless brackets are found
for k = 1:length(f)-1
  if sign(f(k))~=sign(f(k+1))    %  True if sign of f(x) changes in the
                                 % interval
    nb = nb + 1;
    Xb(nb,:) = [x(k) x(k+1)];    % Save left and right ends of the
                                 %   bracket
    hold on;  fill([x(k) x(k) x(k+1) x(k+1) x(k)],ybox,c);
                                                    %  Add  filled box
  end
end
if isempty(Xb)    %  Free advice
 warning('No brackets found.  Check [xmin,xmax] or increase nx');
 return;          % return without drawing a plot
end
% --- Add plot here so that curve is on top of boxes used to indicate
% brackets
plot(xp,yp,[xmin xmax],[0 0]);
grid on;  xlabel('x');
if isa(fun,'inline')
  ylabel(sprintf('Roots of f(x) = %s',formula(fun)));
                                                %  label is formul in f(x)
else
  ylabel(sprintf('Roots of f(x) defined in %s',fun));
                                                %  label is name of m-file
end
hold off
```

Σύνταξη της brackPlot συνάρτησης:

```
brackPlot('myFun', xmin, xmax)
brackPlot('myFun', xmin, xmax, nx)
```

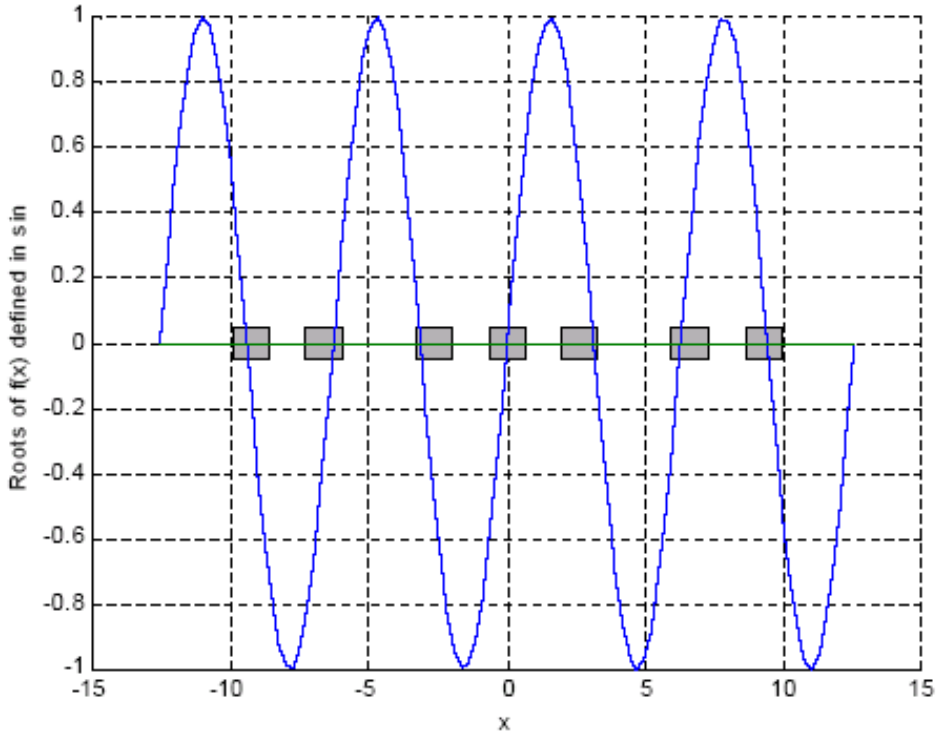
όπου,

myFun	είναι το όνομα ενός m-file που υπολογίζει το f(x)
xmin, **xmax**	ορίζουν το διάστημα αναζήτησης στον άξονα x
nx	είναι ο αριθμός των υποδιαστημάτων που χρησιμοποιούνται για την υποδιαίρεση του $[x_{max}, x_{min}]$. Εξ ορισμού nx = 20.

* Ο αλγόριθμός μας (bracketing) εγγυάται μόνον ότι η συνάρτηση αλλάζει πρόσημο στο διάστημα. Δεν δίνει πληροφορία για το πόσο κοντά σ' ένα άκρο (αριστερό ή δεξιό) είναι η ρίζα. Παρατηρήστε, στα παραδείγματα που ακολουθούν, ότι τα διαστήματα δεν είναι συμμετρικά ως προς τη ρίζα.

Παράδειγμα: (με ενσωματωμένη συνάρτηση)

```
EDU» brackPlot('sin', -4*pi, 4*pi)

ans =

   -9.920818906073031e+000   -8.598043051929960e+000
   -7.275267197786890e+000   -5.952491343643819e+000
   -3.306939635357677e+000   -1.984163781214607e+000
   -6.613879270715355e-001    6.613879270715355e-001
    1.984163781214605e+000    3.306939635357677e+000
    5.952491343643818e+000    7.275267197786889e+000
    8.598043051929960e+000    9.920818906073034e+000
```



Ασκηση

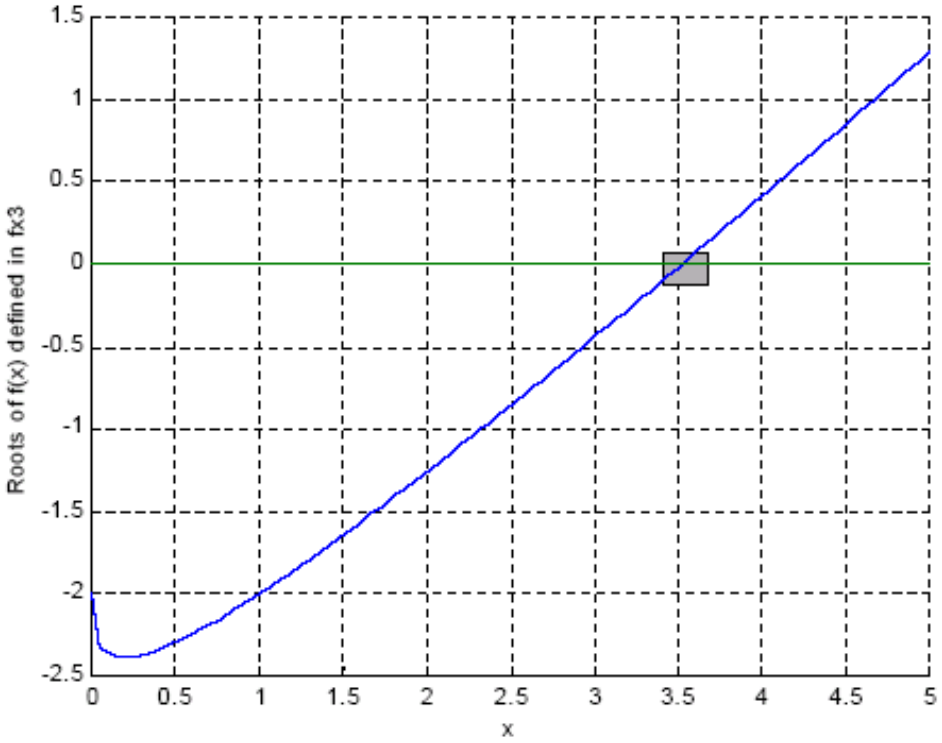
$$f(x) = x - x^{1/3} - 2 = 0$$

Λύση
Δημιουργούμε ένα m-fille για τον υπολογισμό της f(x)

```
function f = fx3(x)
% fx3   Evaluates f(x) = x - x^(1/3) - 2

 f = x - x.^(1/3) - 2;
```

και τρέχουμε τη συνάρτηση **brackPlot**

```
EDU» brackPlot('fx3', 0, 5)

ans =

    3.421052631578947e+000    3.684210526315789e+000
```



- Αντί να δημιουργήσουμε ξεχωριστό m-file μπορούμε να χρησιμοποιήσουμε ένα in – line αντικείμενο:

```
EDU» f=inline('x-x.^(1/3)-2')

f =

     Inline function:
     f(x) = x-x.^(1/3)-2

EDU» brackPlot(f,0,5)

ans =

    3.421052631578947e+000    3.684210526315789e+000
```

* Όταν μια in – line συνάρτηση αντικείμενο περνάει στην **brackPlot**, δεν περικλείεται σε '' δηλ. η σύνταξη είναι **brackPlot(f,0,5)** αντί της **brackPlot('fun',0,5)**.