Laboratorul 1 – Introducere in calcul numeric

Responsabil: As.univ. Drd. Ing. Bogdan Țigănoaia <bogdantiganoaia@gmail.com>

Data publicării: 13-02-2012

Data ultimei modificări: 13-02-2012

Obiective

În urma parcurgerii acestui laborator studentul va fi capabil să:

- Utilizeze programul Octave pentru a rezolva probleme de calcul numeric
- Sa inteleaga diferenta intre solutia numerica a unei probleme si cea matematica.
- Sa foloseasca algoritmi simpli de determinare a unei solutii a ecuatiei f(x) = 0.

Noţiuni teoretice

1. Utilizarea programului Octave

1. Deschideţi terminalul Octave dând dublu-clic pe iconiţa Octave sau scriind octave in consola. Retineti ca fisierele externe ulterioare pe care le veti crea trebuie salvate in directorul curent de lucru (comanda pwd, similara cu cea din unix).

```
> pwd
/home/student/Desktop
```

2. Pentru a crea o matrice de tip coloana, tipăriţi:

```
> v = [ 0; 1; 2 ]
```

3. Pentru a crea o matrice de tip linie, tipăriţi:

```
> l = [ 0  1+5i  2 ]
```

Observati modul de reprezentare al numerelor complexe. Aceasta este unica situatie in care se poate eluda semnul de inmultire *, subintelegandu-se coeficientul numarului imaginar i.

4. Pentru a crea o matrice de 2 linii si 3 coloane, tipăriţi:

```
> m = [ 0 1 2; 3 4 5 ]
```

Se observa deci ca atunci cand este intalnit simbolul ";", programul considera ca incepe o linie noua in matrice. Cazul pentru matrice de m linii si n coloane se generalizeaza usor.

5. Sunt cazuri cand dorim sa construim vectori cu multe elemente, cu termeni in progresie aritmetica. Introducerea lor manuala ar lua prea mult, asa ca exista urmatoarea comanda care simplifica lucrurile:

```
> v = [ initial : pas : final ]
```

Dati diverse valori pasului(chiar si pas negativ, sau care sa nu fie divizor al lui final-initial), valorii initiale si celei finale. Pentru pas 1, comanda poate fi data doar ca [I:F]. Urmariti outputul.

6. Dupa ce ati construit diverse matrice de tip coloana, linie sau matrice generala, introduceti comenzile:

```
> length(v)
> size(m)
```

In timp ce length returneaza lungimea unui vector sau a unei linii, size returneaza un vector de tip [nr, nc], adica numarul de linii si coloane ale matricei m. Pentru mai multe detalii, tastati help length.

7. Ca in C/C++, pare natural sa putem accesa un element al unui vector. Aceasta se face simplu prin v(i), unde "i" este indicele. Analog pentru o matice m, avem m(i,j) pentru elementul de pe linia i, coloana j.

ATENTIE! Indicierea incepe de la 1, si nu de la 0, ca in C/C++!!

8. Pentru a extrage o submatrice dintr-o matrice m, extragand doar liniile l1, l2, l3... si coloanele c1, c2, c3..., construim doi vectori l si c(fie linie, fie coloana, nu are relevanta) si tiparim m(l, c)

```
> m = [0 1 2 ; 3 4 5; -3 -1 10 ]
> m([1 2 3], [2, 3])
> m([1:3], [2, 3])
> v = [1 2 3]
> m(v, [2 3])
```

Am aratat mai multe metode de extragere a unei submatrice. Pentru a nu avea output, folositi ";" dupa comanda care doriti sa nu genereze output.

9. Pentru transpunerea unei matrice, se foloseste operatorul "'". Spre exemplu, m' va returna transpusa (hermitica) a lui m.

10. Operatii aritmetice pe matrice:

```
> m + 3     %se aduna 3 la toate componentele
> 3*m       %se inmultesc toate componentele cu 3
> m*n       %se inmultesc doua matrice, in caz ca dimensiunile sunt
            %compatibile
> m+n       %se aduna doua matrice, ca mai sus
> a .* b    %noua matrice are componentele a(i,j)*b(i,j), si evident
            %aceeasi dimensiune ca a si b
```

11. Functii si constante. Tipariti urmatoarele comenzi si observati efectul.

```
> cos(pi/3)
> sin(pi/4)
> ans
> inf
> eps
> realmax
> realmin
```

12. Bucla for. Sintaxa generala:

for variabila = vector

...

endfor

Exemplu de program ce calculeaza media elementelor unui vector:

```
x = [2 3 4 1 -20];
suma = 0;
for var = x
      suma = suma + var;
endfor
disp('Media este')
disp(suma / length(x))
```

13. Bucla while. Sintaxa generala se poate observa in urmatorul exemplu:

```
x = 1.0;
while x < 1000
      x = x*2;
      disp(x);
endwhile
```

Observati outputul.

14. Functii in Octave. O functie face acelasi lucru ca in C/C++. Primeste parametri,returneaza un rezultat. Fiecare functie trebuie definita intr-un fisier separat, iar numele functiei trebuie sa coincida cu numele fisierului(exceptand, bineinteles, extensia, care este .m). Exemplu de functie, care face suma a doua numere:

```
function s = suma( a, b )
s = a + b;
endfunction
```

Aceasta trebuie salvata intr-un fisier cu numele suma.m

Un apel al functiei arata astfel: > *suma(3,2)*.

Putem avea si functii void, de tipul *function functie(parametri)*. Se pot returna si mai multi parametri (un vector de parametri), in felul urmator:

function [x y z] = functie(parametri)

2. Gasirea unei solutii a ecuatiilor de tip f(x) = 0

Dacă ecuaţia **f(x)=0** are separată o rădăcină în **[a,b]**, (metodele a) şi b) de jos), localizarea ei în limitele unei toleranţe **tol**, înseamnă determinarea unui subinterval **[a,b]**, astfel ca:

(1) **b - a < tol** sau **|f(x)| < tol** pentru $x \in [a, b]$

În caz că nu este dat un interval de separare (metodele c) şi d)), rădăcina se consideră localizată cu toleranţa **tol**, dacă:

(2) $\mathbf{|x^{(k+1)}-x^{(k)}| < tol*|x^{(k)}|}$

Întrucât nu avem asigurată convergenţa (pentru metodele c) şi d)), vom impune un număr maxim de iteraţii **max**. Pentru aceste metode intrările vor fi: aproximaţia iniţială, toleranţa şi numărul impus de iteraţii, iar ieşirile: rădăcina localizată şi un indicator (1/0), care stabileşte dacă relaţia (2) este sau nu verificată după **max** iteraţii.

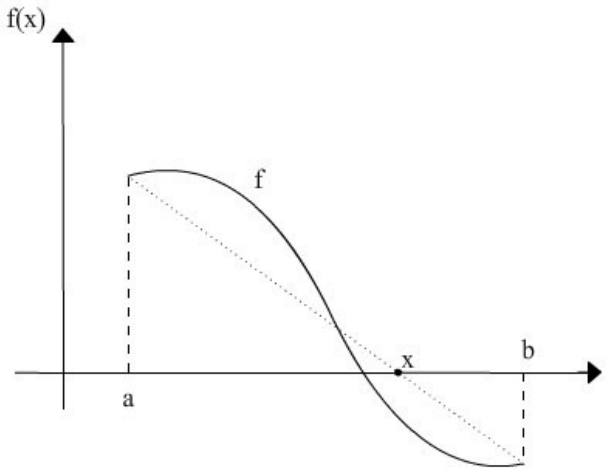
a) **Metoda bisecţiei** - înjumătăţeşte mereu intervalul de căutare, până când este îndeplinită condiţia (1). Aproximarea rădăcinii se ia la mijlocul intervalului: **x=(a+b)/2** şi se alege subintervalul la capetele căruia funcţia schimbă semnul. Se întorc:

- rădăcina localizată = mijlocul ultimului subinterval,

- valoarea funcţiei în rădăcină şi estimarea erorii = lungimea ultimului subinterval.

După **m** iteraţii intervalul de căutare a rădăcinii devine $\mathbf{(b-a)/2^m}$. Numărul de iteraţii va fi dat de: $\mathbf{(b-a)/2^m < tol}$ de unde:

$$\mathbf{m > \log_2((b-a)/tol)} \textit{ (conditie oprire)}$$



b) **Regula secantei** împarte intervalul în două părţi prin intersecţia lui cu dreapta ce trece prin **(a,f(a))** şi **(b,f(b))**.

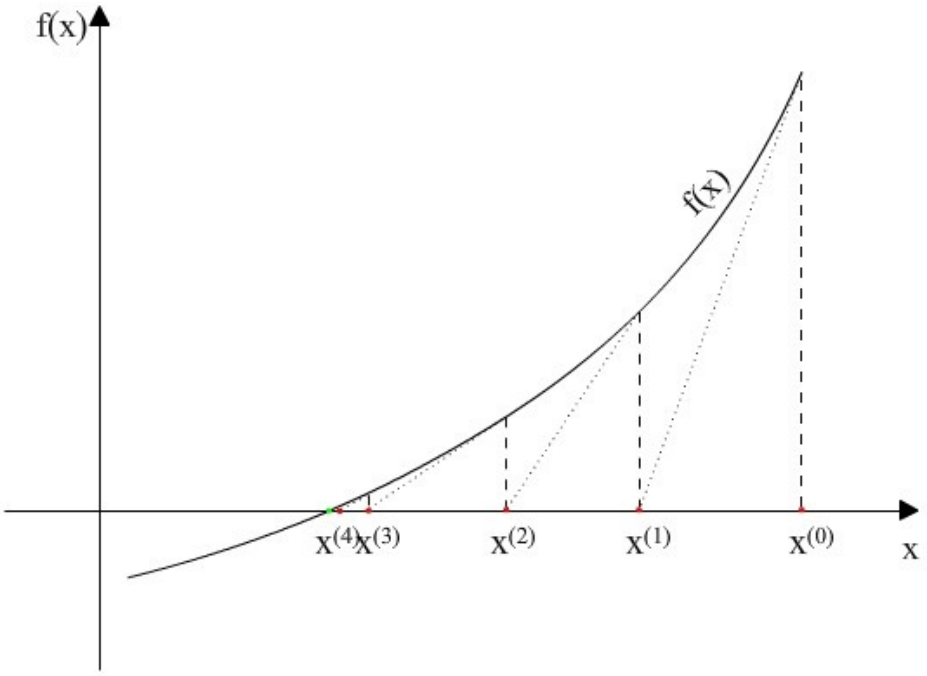
(3) $$x=\frac{af(b)-bf(a)}{f(b)-f(a)}$$

Se alege subintervalul la capetele căruia funcţia schimbă semnul. Se procedează la fel ca în metoda bisecţiei, cu deosebirea că x se calculează cu relaţia (3).

c) **Metoda tangentei (Newton)** - generează şirul de aproximaţii succesive cu relaţia de recurenţă:

(4) $$x^{(k+1)}=x^{(k)}-\frac{f(x^{(k)})}{f'(x^{(k)})}$$

Este necesară o aproximaţie iniţială a rădăcinii ecuaţiei şi cunoaşterea derivatei funcţiei asociate.



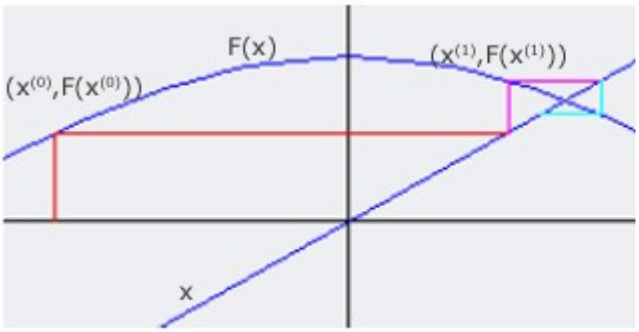
Demonstratie video - http://vimeo.com/3311920

d) **Metoda aproximaţiilor succesive (contracţiei)** - rescrie ecuaţia **f(x)=0** sub forma **x=F(x)** şi se utilizează relaţia de recurenţă:

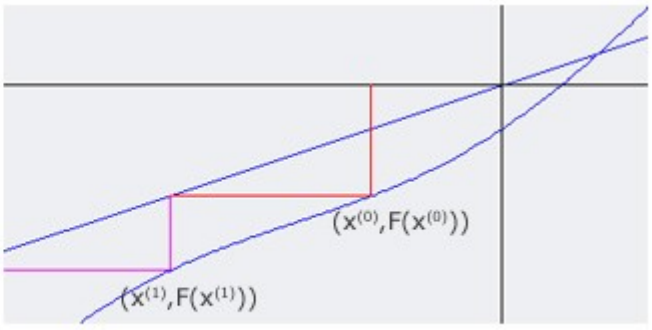
(5) $x^{(n+1)}=F(x^{(n)})$ pornind cu $\mathbf{x^{(0)}}$ dat

care generează un şir $\mathbf{\{x^{(n)}\}}$ convergent către rădăcină, dacă **F(x)** este o contracţie. Studiul contractiilor depaseste cadrul acestui curs, insa este cunoscut faptul ca daca **sup F(x)<1,** functia este contractie**.** In particular, daca exista un eps oricat de mic astfel ca:

(6) **|F'(x)| < 1-eps** avem o contractie.



Caz convergent



Caz divergent

Aplicaţii

[2p] 1. Construiţi o funcţie care sa calculeze suma numerelor pare mai mici ca n utilizând bucla for. Faceţi acelaşi lucru utilizând bucla 'while'. Citirea unui numar de la tastatura se face utilizand comanda *var = input('Introduceti variabila: ')*

[4p] 2. Consideram ecuaţia f(x)=0 unde $f(x)=tg(x)-2x$ pe care vrem să o rezolvam pe intervalul [0.1,1.5] utilizând metoda bisecţiei.

a) Creaţi un m-file numit f.m care să calculeze funcţia f(x) într-un punct x dat ca parametru. Creaţi apoi un fisier f_sample.m care întoarce un vector cu valori ale funcţiei calculate în suficient de multe puncte.

b) Trasaţi un grafic pe [0.1, 1.5] utilizând funcţia plot. (pentru detalii, *help plot*)

c) Creaţi un fişier numit bisect.m cu o funcţie care sa identifice o radacina prin metoda bisectiei.

[4p] 3. Fie ecuaţia: $x^3+2x^2-5x-6=0$.

a) Să se rezolve prin metoda bisecţiei şi metoda secantei, ştiind că ecuaţia are separată o rădăcină în intervalul **[0,3]**.

b) Să se rezolve prin metoda Newton cu **x0=1**, **max=10.**

c) Să se rezolve prin metoda aproximării succesive cu **x0=1**, **max=10**, folosind pe rând explicitările:

$$x=\frac{x^3+2x^2-6}{5}$$

$$x=\sqrt{\frac{6+5x-x^3}{2}}$$

$$x=\sqrt[3]{6+5x-2x^2}$$

Pentru metodele prezentate mai sus se vor compara rezultatele obţinute.