prof. Marinel Şerban
Liceul de Informatică "Gr. C. Moisil" Iaşi

METODE NUMERICE

I. EROARI

1. Introducere

Rezolvarea unei probleme presupune:
1) elaborarea algoritmului
2) transpunerea într-un limbaj de programare
3) verificarea rezultatelor (intermediare, finale)
4) precizia rezultatului

Din 4) şi din faptul că în memorie reprezentarea informațiilor NU este exactă deducem faptul că rezultatele suferă aproximări.

Definiţie. Fie $x \in R$. Notăm $\overline{x}$ valoarea aproximativă a lui x. Atunci:

$e_x = x - \overline{x}$ se numeşte ***eroare absolută***, iar

$\varepsilon_x = \frac{e_x}{\overline{x}}$ se numeşte ***eroare relativă***.

Obs. Se obişnuieşte şi $e_x = \left| x - \overline{x} \right|$

$$\varepsilon_x = \left| \frac{e_x}{\overline{x}} \right|$$

2. Clasificarea erorilor

Erorile se pot clasifica astfel:
- erori **inerente** (ale problemei) – acestea rezultă din simplificarea modelului fizic pentru a putea fi cuprins într-un model matematic
- erori de **metodă** (de trunchiere) – aceste erori provin din metoda matematică utilizată
- erori de **rotunjire** – acestea provin din datele de intrare/ieşire sau la calcule (erori de operație)

Exemple:
Să se calculeze suma:

$$S = \frac{1}{1\cdot 2} + \frac{1}{2\cdot 3} + \mathrm{K} + \frac{1}{n\cdot(n+1)} + \mathrm{K}$$

Evident

$$S_n = \frac{1}{1\cdot 2} + \frac{1}{2\cdot 3} + \mathrm{K} + \frac{1}{n\cdot(n+1)}$$ suma primilor n termeni,

şi evident

$S_n \to S$, adică S_n este şir convergent cu limita S.

Problema este că în programare nu putem determina limita, ca noțiune matematică.
Se poate descrie calculul sumei printr-o formulă de recurență:

$$\begin{cases} S_1 = \frac{1}{2} \\ S_n = S_{n-1} + \frac{1}{n(n+1)}, \quad n \geq 2 \end{cases}$$

Atunci:

$$S_2 = S_1 + \frac{1}{6} = \frac{2}{3}$$

$$S_3 = S_2 + \frac{1}{12} = \frac{3}{4}$$

M

$$S_9 = S_8 + \frac{1}{90} = \frac{9}{10}$$

Atunci se poate lua $S \cong S_9 = 0.9$
Dar

$$S_n = \frac{n}{n+1}$$

şi

$$\lim_{n\to\infty} S_n = 1 = S$$

deci eroarea este de 0.1.
Cea mai utilizată metodă de rotunjire este rotunjirea prin trunchiere (prin tăiere).
Exemplu:

x = 12945.734

Această valoare se poate trunchia la:

$\overline{x}_1 = 12940$

$\overline{x}_2 = 12945$

$\overline{x}_3 = 12945.7$

$\overline{x}_4 = 12945.73$

dar evident şi

$\overline{x}_5 = 12900$

În ceea ce priveşte **propagarea erorilor** se poate demonstra că lucrând cu valori rotunjite (lucru obligatoriu când se lucrează în virgulă mobilă) pot să apară erori semnificative la rezultate.
Exemplu:
Să se calculeze

x = sin(x_0) unde x_0 = 25.7 radiani cu eroare absolută de 10^{-8}

Se poate calcula astfel:

$$\left|x-\overline{x}\right| = \left|\sin x_0 - (x_0 - \frac{x_0^3}{3!} + \frac{x_0^5}{5!} + \mathrm{L} + \frac{(-1)^n x_0^{2n+1}}{(2n+1)!})\right| \le \frac{x_0^{2n+3}}{(2n+3)!} < 10^{-8}$$

Folosindu-se, în calculul în virgulă mobilă, 6 cifre pentru mantisă şi 2 pentru exponent se ajunge la:

sin $x_0 \cong$ 891.789 !!! evident eronat

Explicațiile acestei aberații:

- viteza de convergență a seriei folosite
- acumularea erorilor de rotunjire

prof. Marinel Şerban
Liceul de Informatică "Gr. C. Moisil" Iaşi

II. REZOLVAREA ECUAȚIILOR ALGEBRICE

1. Generalități

Fie ecuația $f(x) = 0$, cu f continuă pe intervalul $(a,b) \subset R$
Un număr real $\alpha \in R$ cu $f(\alpha) = 0$ se numeşte rădăcină a ecuației date.

Definiție:
O ecuație de forma
$P(x) = 0$, cu $P(x)$ polinom se numeşte
ECUAȚIE ALGEBRICĂ

$$P_n(x) = a_0 \cdot x^n + a_1 \cdot x^1 + ... + a_{n-1}x^{n-1} + a_n = 0$$
$$P_n(x) = 0$$

Dacă ecuația nu poate fi pusă sub forma unui polinom se numeşte
ECUAȚIE TRANSCENDENTĂ

Exemple de ecuații transcendente:

$$x - 10\sin x = 0;$$
$$2^x - 2\cos x = 0;$$
$$\lg(x+5) = \cos x;$$
$$x^3 + \cos x - 1 = 0;$$
$$e^x + 2x - 5 = 0;$$

Observații:

1. Este cunoscut faptul că pentru n=1 (ecuație liniară) şi n=2 (ecuație pătratică) există formule exacte de rezolvare. Pentru n=3 şi n=4 există formulele Cardano şi Ferrari de determinare a soluțiilor. Pentru n>4 nu există formule de determinare a soluțiilor (metode exacte), decât numai pentru anumite cazuri particulare. De obicei rezolvarea unei ecuații algebrice pentru n>4 se face prin metode iterative de calcul.
2. Mai este cunoscut faptul, că orice ecuație algebrică are cel puțin o rădăcină reală sau complexă.
3. La scrierea unei ecuații în formă canonică vom avea aceleaşi rădăcini ca şi pentru ecuația inițială, însă în acest caz pot să apărea şi rădăcini suplimentare.

De exemplu, ecuația algebrică $\sqrt{2x^2-1} + x = \sqrt{2x^2+1} - 1$, poate fi adusă la forma canonică:

$$7x^4 + 12x^3 + 2x^2 - 4x - 5 = 0.$$

În general astfel de ecuații nu au rădăcini exacte. Din această cauză se determină o aproximație a rădăcinilor, procedând astfel:

a) se separă rădăcinile astfel încât un interval să conțină o singură rădăcină
b) se îmbunătățeşte aproximația cât de mult se poate (sau cere problema)

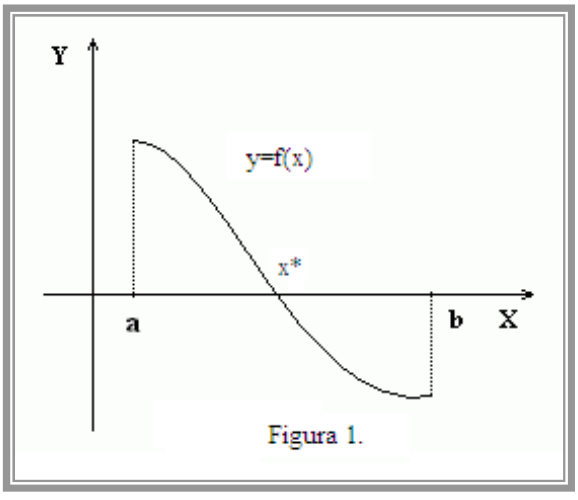


Figura 1.

TEOREMĂ
Dacă
$f:[a,b] \to R$, continuă pe [a, b], are semne contrare la capetele intervalului, adică
$f(a) \cdot f(b) < 0$ atunci
$\exists \alpha \in (a,b)$ cu $f(\alpha) = 0$
adică ecuația $f(x) = 0$ are soluție în acest interval.

prof. Marinel Şerban
Liceul de Informatică "Gr. C. Moisil" Iaşi

Obs.

1. Dacă funcția f este injectivă atunci $\exists!\alpha\in(a,b)$ cu $f(\alpha)=0$ rădăcină unică
2. Separarea rădăcinilor se face, de obicei, astfel:
   a. se împarte *(a, b)* în intervale
      $a_1 < a_2 < ... < a_n$
   b. se calculează $f(a_1), f(a_2), ..., f(a_n)$
   c. în general se utilizează metoda înjumătățirii intervalului

Izolarea soluțiilor. Diverse metode de separare (reper matematic)

Prima etapă de rezolvare numerică a ecuației $f(x)=0$ constă în *separarea rădăcinilor*, (izolarea soluțiilor), adică în stabilirea intervalelor ce conțin o rădăcină unică.
Această etapă poate fi realizată în câteva moduri: separarea grafică, separarea analitică sau separarea prin metoda încercărilor.

Separarea grafică a rădăcinilor

Să analizăm mai întâi modalitatea grafică de separare a soluțiilor.

Luând în considerație faptul că rădăcinile reale ale ecuației $f(x)=0$, nu reprezintă altceva decât abscisele punctelor de intersecție ale graficului funcției $y=f(x)$ cu axa Ox, este destul să construim graficul funcției $f(x)$ şi apoi să separăm intervalele care conțin exact câte o rădăcină. Dacă ecuația nu are soluții foarte apropiate una de alta, atunci prin această metodă rădăcinile se izolează foarte uşor. În practică însă, de cele mai multe ori, întâlnim cazuri când funcția este de o formă destul de complicată pentru a construi graficul ei. În astfel de cazuri, pentru simplificarea construirii graficului, ecuația $f(x)=0$ se înlocuieşte cu ecuația echivalentă ei

$$f_1(x)=f_2(x),$$

unde $f_1(x)$ şi $f_2(x)$ sunt mai simple decât ecuația inițială.

Atunci construind graficele funcțiilor $y=f_1(x)$ şi $y=f_2(x)$, putem foarte uşor izola segmentele sau intervalele care conțin rădăcinile, deoarece ele reprezintă abscisele punctelor de intersecție ale acestor grafice (vezi fig.1).

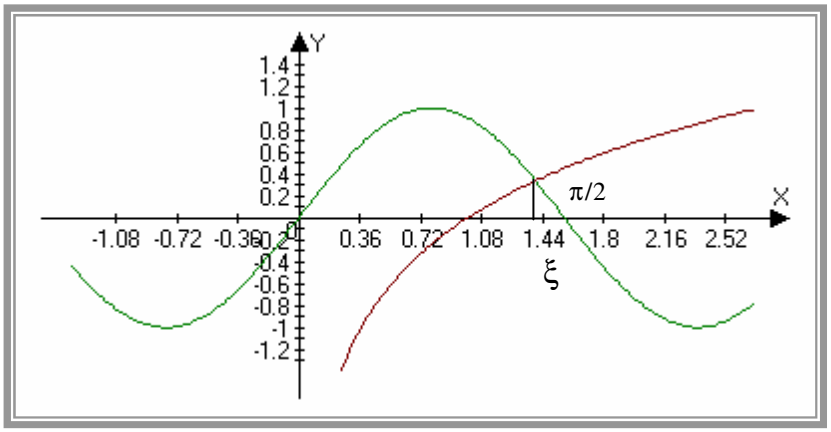


Fig. 1

Exemplu: Să se separe grafic rădăcinile ecuației $\sin 2x - \ln x = 0$. Construim separat graficele funcțiilor $y=\sin(2x)$ şi $y=\ln(x)$. Din grafic rezultă că ecuația are o singură soluție, care aparține intervalului [1; 1.5].

prof. Marinel Şerban
Liceul de Informatică "Gr. C. Moisil" Iaşi

Separarea analitică

În alte cazuri este bine ca separarea grafică să fie însoțită de separarea analitică a rădăcinilor, care poate fi realizată prin calcul direct sau aplicând una dintre metodele cunoscute.

Una dintre metodele de separare analitică a rădăcinilor se bazează pe următoarea *consecință a teoremei lui Rolle*: între două rădăcini reale consecutive ale ecuației $f'(x)=0$ există *cel mult o rădăcină reală* a ecuației $f(x)=0$.

Fie $x_1<x_2<...<x_n$ rădăcinile ecuației $f'(x)=0$. Se formează şirul lui Rolle:
$f(-\infty), f(x_1), f(x_2),..., f(x_n), f(\infty)$. Conform consecinței enunțate, în fiecare interval

$$(-\infty, x_1), (x_1, x_2),..., (x_i, x_{i+1}),..., (x_n, \infty)$$

se află o rădăcină reală a ecuației $f(x)=0$ numai dacă funcția ia valori de semne contrare la capetele intervalului, adică dacă $f(x_i)\cdot f(x_{i+1})<0$.

Prin urmare, ecuația $f(x)=0$ are atâtea rădăcini câte variații de semn prezintă şirul lui Rolle. Din punct de vedere practic, această metodă prezintă un dezavantaj, deoarece necesită rezolvarea ecuației $f'(x)=0$, ceea ce uneori este la fel de dificil ca şi rezolvarea ecuației $f(x)=0$.

Foarte utile pentru separarea soluțiilor sunt următoarele teoreme din analiza matematică, teoreme stabilite de Augustin Cauchy, matematician francez.

Teorema 1
Dacă funcția $f(x)$ de o singură variabilă reală definită pe un segment $[a,b]$, este *continuă* pe acest segment şi admite valori de semn opus la capetele segmentului cercetat, adică $f(a)\cdot f(b)<0$, atunci pe acest interval există *cel puțin* o rădăcină a ecuației $f(x)$, adică se va găsi un număr $\xi\in(a,b)$, astfel încât $f(\xi)=0$ (fig. 2).

Teorema 2
Dacă o funcție este continuă şi strict monotonă pe un segment oarecare $[a,b]$, şi $f(a)\cdot f(b)<0$, atunci pe acest segment există *o rădăcină* şi *numai una singură*.

Deci, rădăcina ξ este unică, dacă există derivata de ordinul întâi şi această derivată îşi păstrează semnul constant în interiorul acestui interval (a,b), adică $f'(x)>0$ (sau $f'(x)<0$) pentru $a<x<b$ (vezi fig. 3). În caz contrar, este posibil ca în intervalul dat să existe mai multe rădăcini sau rădăcini pot să existe chiar dacă $f(a)\cdot f(b)>0$.

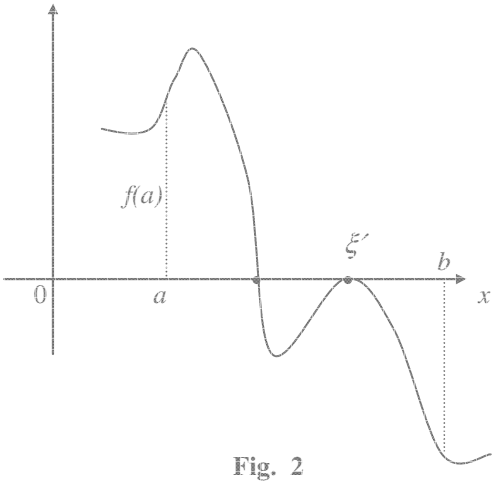


Fig. 2

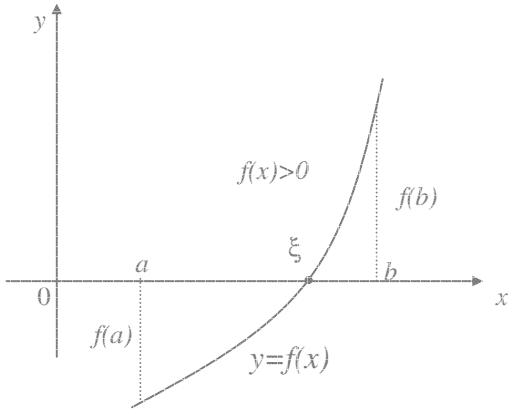


Fig. 3

prof. Marinel Şerban
Liceul de Informatică "Gr. C. Moisil" Iaşi

Separarea prin metoda numerică aproximativă. Algoritmul metodei separării rădăcinilor

Să discutăm în continuare despre modalitatea de separare a intervalelor ce conțin rădăcini, aplicabilă atât pentru rezolvarea ecuațiilor algebrice cât şi pentru cele transcendente.
Problemă: Fie este dată funcția $f(x)=0$ – o funcție continuă, monoton crescătoare sau descrescătoare pe un segment oarecare $[a, b]$ şi care are semne diferite la capete, adică $f(a)\cdot f(b)<0$.
Să se separe rădăcinile ecuației, adică să se determine toate intervalele $[x_1,x_2]\subset[a,b]$ care conțin exact câte o rădăcină.
Vom calcula valorile lui $f(x)$, începând cu punctul $x=a$, deplasându-ne la dreapta cu un anumit pas h (fig. 4). În cazul când se va detecta o pereche de valori vecine ale lui $f(x)$, având cele trei proprietăți (***continuitate, semne diferite la capete şi monotonie strictă***), atunci putem considera că am separat un interval care conține o rădăcină unică a ecuației $f(x)$. Evident că siguranța acestei metode depinde în mare măsură de caracterul funcției $f(x)$ cât şi de mărimea pasului de deplasare h, care trebuie luat de cele mai multe ori destul de mic.

Pseudocod

```
citeşte a,b,h
x1 <- a
x2 <- x1+h
y1 <- f(x1)
cât_timp x2<b execută
  {
      y2 <- f(x2)
      dacă y1·y2<0 atunci
            scrie x1,x2
      x1 <- x2
      x2 <- x1+h
      y1 <- y2
   }
```

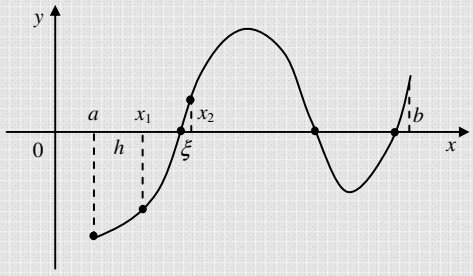


Fig. 4

```
program separarea_solutiilor;
uses crt;
var a,b,h,x1,x2,y1,y2:real;

function f(x:real):real;
begin
  f:=cos(x)-0.1*x;
end;

begin
  clrscr;
  write('dati a='); readln(a);
  write('dati b='); readln(b);
  write('dati pasul h='); readln(h);
  x1 := a; x2 := h+x1; y1 := f(x1);
  while x2<b do
    begin
      y2 := f(x2);
      if y1*y2<0
        then writeln('solutia se afla in intervalul [',x1:0:4,'; ',x2:0:4,']');
      x1 := x2;
      x2 := x2+h;
      y1 := y2;
    end;
end.
```

```
dati a=-10
dati b=10
dati pasul h=0.1
solutia se afla in intervalul [-1.5000; -1.4000]
solutia se afla in intervalul [0.7000; 0.8000]
dati a=-10
dati b=10
dati pasul h=0.01
solutia se afla in intervalul [-1.4900; -1.4800]
solutia se afla in intervalul [0.7300; 0.7400]
dati a=-2
dati b=1
dati pasul h=0.001
solutia se afla in intervalul [-1.4810; -1.4800]
solutia se afla in intervalul [0.7360; 0.7370]
```

Exemple:

a) $3^x+5x-2=0$;
b) $3x^4+4x^3-12x+1=0$;
c) $0{,}5^x+1=(x-2)^2$;
d) $(x+3)\cdot\cos x=1$, $-2\pi\le x\le 2\pi$

prof. Marinel Şerban
Liceul de Informatică "Gr. C. Moisil" Iaşi

2. Rezultate generale privind ecuațiile algebrice

Fie

$$P(x) = a_0x^n + a_1x^{n-1} + ... + a_{n-1}x + a_n = 0 \text{ cu } a_i \in R, \quad a_0 \neq 0$$

Un număr real α se numeşte ***rădăcină simplă*** dacă $P(\alpha)=0$ şi $P'(\alpha)\neq 0$

Un număr real α se numeşte ***rădăcină multipă de ordin k*** dacă

$$P(\alpha)=P'(\alpha)=\text{L}\ =P^{(k-1)}(\alpha)=0 \text{ şi } P^{(k)}(\alpha)\neq 0$$

TEOREMA fundamentală a algebrei.

Orice ecuație algebrică de grad n are exact n rădăcini.

TEOREMĂ

Dacă

$$\alpha=\alpha_1+i\cdot\alpha_2 \ cu \ P(\alpha)=0 \text{ atunci}$$

$$P(\bar{\alpha})=0 \text{ unde } \bar{\alpha}=\alpha_1-i\cdot\alpha_2$$

Concluzii:

1. rădăcinile complexe sunt în număr par
2. o ecuație algebrică de grad impar are cel puțin o rădăcină reală

TEOREMĂ

Toate rădăcinile $x_1, x_2, ..., x_n$ ale unei ecuații algebrice se află în interiorul cercului cu centrul în origine şi de rază

$$R=1+\frac{\alpha}{|a_0|} \text{ unde}$$

$$\alpha=\max\{|a_1|,|a_2|,\text{L}\ ,|a_n|\}$$

Demonstrație

$a_0x^n = P(x) - (a_1x^{n-1} + ... + a_n)$, de unde rezultă

$|a_0x^n| \le |P(x)| + |a_1x^{n-1}| + ... + |a_n|$, deci

$|P(x)| \ge |a_0x^n| - (|a_1|\bullet|x^{n-1}| + ... + |a_n|) \ge |a_0|\bullet|x^n| - \alpha\bullet(|x^{n-1}| + ... + 1) = |a_0||x^n| - \alpha\frac{|x^n|-1}{|x|-1} >$

$$> (|a_0|-\frac{\alpha}{|x|-1})\cdot|x^n|$$

Dacă alegem x astfel ca $|a_0| \ge \frac{\alpha}{|x|-1}$

astfel încât paranteza să devină pozitivă, deducem $|x| > 1+\frac{\alpha}{|a_0|}$

va rezulta că $P(x)>0$ deci x NU mai poate fi rădăcină. De aici rezultă că toate posibilele rădăcini îndeplinesc condiția:

$$|x_j| \le 1+\frac{\alpha}{|a_0|}, \ \forall j=\overline{1,n}$$

În mod similar se obține şi următorul rezultat:

Dacă $a_n \neq 0$ şi $\beta = \max\{|a_0|, |a_1|, ..., |a_{n-1}|\}$, toate rădăcinile ecuației se află în exteriorul cercului cu centrul în origine şi de rază r dată de:

$$\frac{1}{r}=1+\frac{\beta}{|a_n|}$$

Demonstrație

Notăm $x=\frac{1}{y}$ şi prin înlocuire se obține:

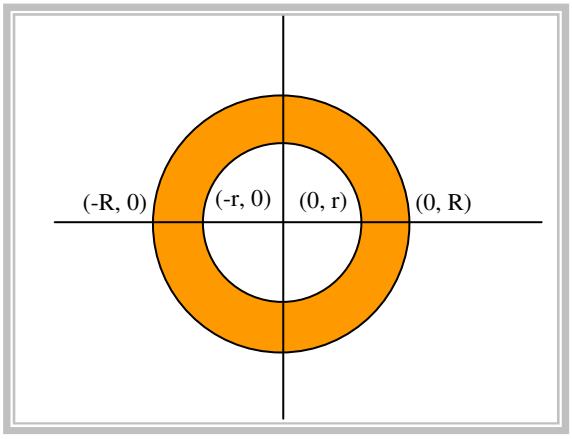
$$P(x)=a_0\frac{1}{y^n}+a_1\frac{1}{y^{n-1}}+\mathrm{L}\ +a_{n-1}\frac{1}{y}+a_n=\frac{a_0+a_1y+\mathrm{L}\ +a_{n-1}y^{n-1}+a_ny^n}{y^n}=\frac{Q(y)}{y^n}$$

Rădacinile ecuației

$Q(y)=0$ sunt $y_j=\frac{1}{x_j},\ j=1,n$ iar din teorema anterioară ştim că $|y_j|<1+\frac{\beta}{|a_n|},\ \forall j=1,n$

adică:

$\frac{1}{|x_j|}<\frac{1}{r}$, de unde $|x_j|<r$



Exemplu

Fie ecuația algebrică

$$x^5-5x^4+15x^3-25x^2+26x-12=0$$

Se calculează

$\alpha = \max(5, 15, 25, 26, 12) = 26$

$$R=1+\frac{\alpha}{|a_0|}=1+\frac{26}{1}=27$$

$\beta = \max(1, 5, 15, 25, 26) = 26$

$$\frac{1}{r}=1+\frac{\beta}{|a_n|}=1+\frac{26}{12}=\frac{19}{6},\text{deci } r=\frac{6}{19}$$

Rezolvând ecuația cu metodele clasice rezultă:

$$x_1=1, x_2=1+i\sqrt{2}, x_3=1-i\sqrt{2}, x_4=1+i\sqrt{3}, x_5=1-i\sqrt{3}$$

vom observa că toate rădăcinile sunt în intervalul calculat.

III. METODE DE APROXIMARE A RĂDĂCINILOR UNEI ECUAȚII ALGEBRICE SAU TRANSCENDENTE

1. Generalități

Fie

$f(x) = 0, f:(a, b) \to \mathbf{R}$, continuă pe intervalul (a, b)

Vom presupune că pe intervalul respectiv există o unică rădăcină $\alpha\in(a,b)$

În ceea ce priveşte aprecierea erorii de determinare a rădăcinii α se poate demonstra următoarea

TEOREMĂ

Fie $\alpha\in(a,b)$ cu $f(\alpha)=0$ şi fie $\alpha_0\in(a,b)$

Dacă f este derivabilă pe (a, b) şi $\exists\mu>0$ cu $|f'(x)|>\mu,\forall x\in(a,b)$ rezultă

$$|\alpha-\alpha_0|\le\frac{|f(\alpha_0)|}{\mu}$$

Demonstrație

Fie $\alpha<\alpha_0$. Din teorema lui Lagrange rezultă

$\exists c \in (\alpha, \alpha_0)$ cu $f(\alpha_0) - f(\alpha) = (\alpha_0 - \alpha) \cdot f'(c)$

Dar $f(\alpha) = 0$, deci

$|f(\alpha_0)| = |\alpha - \alpha_0| \cdot |f'(c)|$ de unde

$|\alpha - \alpha_0| = \frac{|f(\alpha_0)|}{|f'(c)|} \le \frac{|f(\alpha_0)|}{\mu}$, deoarece $\mu \le |f'(x)|, \forall x \in (a,b)$ ceea ce arată că eroarea de apreciere a soluției poate fi oricât de mică.

2. Metoda înjumătățirii intervalului (**metoda bisecției**)

Fie $f:[a, b] \to \mathbf{R}$, continuă pe intervalul (a, b), cu $f(a) \cdot f(b) < 0$, deci are semne opuse la capetele intervalului.

Împărțim intervalul [a, b] în două părți egale

$$\left[a, \frac{a+b}{2}\right]\left[\frac{a+b}{2}, b\right]$$

avem două cazuri:

dacă $f(\frac{a+b}{2}) = 0 \Rightarrow \alpha = \frac{a+b}{2}$

dacă $f(\frac{a+b}{2}) \neq 0$ atunci vom alege acel interval $[a_1, b_1]$ dintre cele două pentru care $f(a_1) \cdot f(b_1) < 0$. Vom obține astfel un şir de intervale

$[a_1, b_1]$ cu $f(a_1) \cdot f(b_1) < 0$
$[a_2, b_2]$ cu $f(a_2) \cdot f(b_2) < 0$
...
$[a_n, b_n]$ cu $f(a_n) \cdot f(b_n) < 0$ (*)

unde, evident $b_n - a_n = \frac{b-a}{2^n}$

Deoarece şirul (a_n) este crescător şi mărginit, el are o limită.
Analog, deoarece şirul (b_n) este descrescător şi mărginit, el are o limită.
Se observă, de asemenea că

$$\lim_{n\to\infty}(b_n - a_n) = \lim_{n\to\infty}\frac{b-a}{2^n} = 0$$

ceea ce înseamnă că

$$\lim_{n\to\infty}(a_n) = \lim_{n\to\infty}(b_n) = \alpha$$

Atunci din (*) se obține

$\lim_{n\to\infty}(f(a_n) \cdot f(b_n)) = (f(\alpha))^2 \le 0$ şi din (evident) $(f(\alpha))^2 \ge 0$ va rezulta

$f(\alpha) = 0$, adică α este rădăcină.

Obs.

Din $\frac{b-a}{2^n} < \varepsilon$ rezultă $n = \frac{\ln(\frac{b-a}{\varepsilon})}{\ln 2} + 1$, adică numărul de paşi pentru a obține precizia ε.

Exemplu:

Să se determine soluția ecuației

$x^4 - x^3 + x^2 + 2x - 6 = 0$ din intervalul [1, 2]

Se observă, din calcul, faptul că

f(1) = -3 şi f(2) = 10, deci funcția are semne diferite la capetele intervalului.

Pseudocod

```
citeste a, b, ε, f    //expresia funcţiei f o dăm explicit
cât_timp (f((a+b)/2)<>0 şi b-a>ε) execută
   {
      dacă f(a)•f((a+b)/2)<0 atunci
            b <- (a+b)/2
      altfel
            a <- (a+b)/2
   }
y <- (a+b)/2
scrie y
```

```
program injumatatire;
uses crt;
var a,b,eps,y:real;
    functia:string;
    nr_pasi:integer;

function f(x:real):real;
begin
     f := x*x*x-10*ln(x)
end;

function f1(x:real):real;
begin
      f1:=x*x*x*x-x*x*x+x*x+2*x-6
end;

function f2(x:real):real;
begin
      f2:=x*ln(x)-1       {în [1, 4] are soluţia y=1.7632228434}
end;

function f3(x:real):real;
begin
      f3:=x*x+ln(x)-4     {în [0.5, 2] are soluţia y=1.84109706}
end;

function f4(x:real):real;
begin
      f4:=2/3*sqrt(x*x+9)-x    {în [1, 3] are soluţia y=2.68328157440}
end;

function f5(x:real):real;
begin
      f5:=2*x*x*x-6*x*x-x+3
end;

begin
    write('limita inferioara a=');readln(a);
    write('limita superioara b=');readln(b);
    write('precizia E=');readln(eps);
    clrscr;
    nr_pasi:=0;
    writeln('[a,b]=[',a:5:1,',',b:5:1,']    E=',eps:10:8);
    writeln('         METODA INJUMATATIRII INTERVALULUI');
    writeln('    a     ':12,'               b');
    writeln('=============================================================');
    while (f3((a+b)/2)<>0) and (b-a>eps) do
      begin
            nr_pasi:=nr_pasi+1;
            if f3(a)*f3((a+b)/2)<0 then
               b:=(a+b)/2
            else
               a:=(a+b)/2;
           writeln(a:12:8,b:20:8,'b-a=':8,b-a:12:8)
       end;
    y:=(a+b)/2;
    writeln('-------------------------------------------------------------');
    writeln('SOLUTIA APROXIMATIVA DUPA ',nr_pasi,' PASI ESTE y=',y:12:8);
end.
```

$$\frac{2}{3}\sqrt{x^2-9}-\ln x=0,$$

$[1,3]\ y = 2.99999999630$

$[0.1,1]\ y = 0.99999999665$

prof. Marinel Şerban
Liceul de Informatică "Gr. C. Moisil" Iaşi

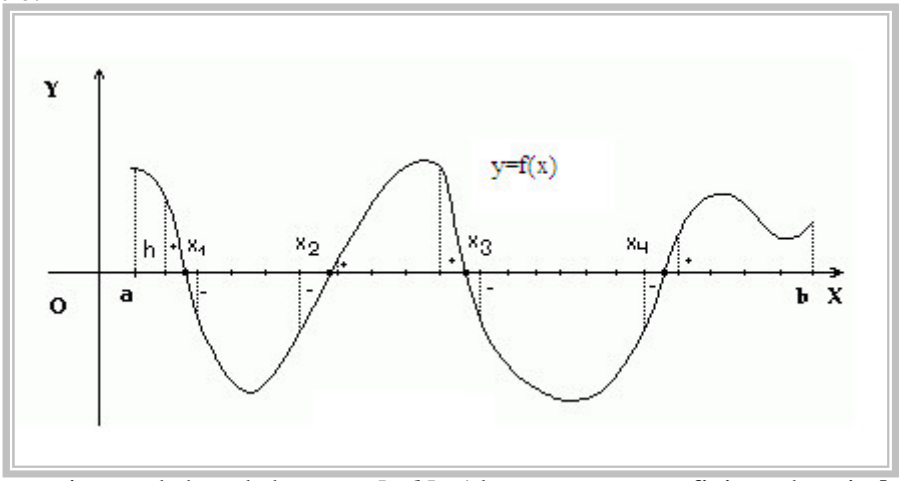
APLICAREA METODEI BISECȚIEI LA AFLAREA TUTUROR SOLUȚIILOR UNEI ECUAȚII NELINIARE

Rezolvarea ecuațiilor neliniare cu ajutorul calculatorului prevede următorii paşi:

- Selectarea intervalelor ce conțin soluții separate.
- Aplicarea metodei respective de găsire a soluției în fiecare interval, ce conține o soluție seperată

Algoritmul separării soluțiilor poate fi următorul:

Fie dată ecuația $f(x) = 0$. Presupunem că funcția $f(x)$ este continuă şi graficul ei este reprezentat în figura următoare:

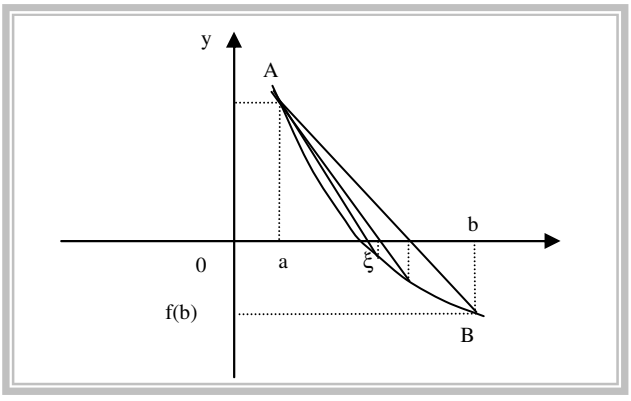


Vom începe cu un interval destul de mare $[a,b]$. Alegem un pas suficient de mic **h** şi începând din capătul **a** vom calcula valorile funcției la capetele tuturor segmentelor de lungime **h**. Dacă funcția $f(x)$ capătă valori de semn opus la capetele segmentului, atunci înseamnă că acest interval conține cel puțin o soluție a ecuației $f(x)=0$. Din grafic observăm că soluțiile x_1, x_2, x_3 şi x_4 aparțin unor astfel de intervale de lungime **h**, iar la capetele fiecărui interval, funcția $f(x)$ are valori de semn opus. Capetele acestor intervale le vom memora într-un tablou.
În continuare vom aplica metoda **bisecției** pe fiecare interval ce conține o soluție separată a ecuației $f(x) = 0$.

Exerciții: aplicați metoda bisecției pentru ecuațiile $e^{-x} + x^2 - 2 = 0$
$\ln x - x + 1.8 = 0$
$x \cos(x) - \sin(x) = 0$

3. Metoda coardei

Fie $f:[a, b] \to \mathbf{R}$, continuă pe intervalul (a, b), cu $f(a)\cdot f(b)<0$, deci are semne opuse la capetele intervalului.
Pentru început vom presupune f(a)>0 şi f(b)<0, adică:



când f(a)>0

$f''(x) > 0, \forall x \in (a,b)$
"adună apa"

Ducem coarda AB.

Ştiind că ecuația dreptei care trece prin două puncte $A(x_1, y_1)$ şi $B(x_2, y_2)$ este

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$ şi că, în cazul nostru punctele sunt A(a, f(a)) şi B(b, f(b)), obținem

$$\frac{x-a}{b-a} = \frac{y-f(a)}{f(b)-f(a)}$$

Intersectând această dreaptă cu dreapta $y=0$ vom obține

$$x_1 = a - \frac{f(a)}{f(b)-f(a)}(b-a)$$ sau, notând $h_1 = -\frac{f(a)}{f(b)-f(a)}(b-a)$

$x_1 = a + h_1$

Pseudocod

```
citeste a, b, ε, f    //expresia funcției f o dăm explicit
                f(a)
x1 <- a - ---------- (b-a)
           f(b)-f(a)
cât_timp |f(x1|> ε execută
   {
      dacă f(a)f(x1)<0 atunci
         b <- x1
      altfel
         a <- x1
                   f(a)
      x1 <- a - ---------- (b-a)
                f(b)-f(a)
   }
y <- x1
scrie y
```

```
program coarda1;
var a,b,eps,y, x1:real;
    nr_pasi:integer;
    functia:string;

function f5(x:real):real;
begin
      f5:=x*x*x*x-x*x*x+x*x+2*x-6
end;

function f2(x:real):real;
begin
      f2:=x*ln(x)-1
end;

function f3(x:real):real;
begin
      f3:=x*x+ln(x)-4
end;

function f4(x:real):real;
begin
      f4:=2/3*sqrt(x*x+9)-x
end;

function f1(x:real):real;
begin
      f1:=2*x*x*x-6*x*x-x+3
end;

begin
    write('limita inferioara a=');readln(a);
    write('limita superioara b=');readln(b);
    write('precizia E=');readln(eps);
    writeln('[a,b]=[',a:5:1,',',b:5:1,']    E=',eps:10:8);
```

```
    writeln('                    METODA COARDEI');
    writeln('    a     ':12,'              b');
    writeln('=======================================================');
    nr_pasi:=0;
    x1:=a-f1(a)/(f1(b)-f1(a))*(b-a);
    while f1(x1)>eps do
      begin
            nr_pasi:=nr_pasi+1;
            if f1(a)*f1(x1)<0
            then b:=x1
               else a:=x1;
            x1:=a-f1(a)/(f1(b)-f1(a))*(b-a);
           writeln(a:12:8,b:20:8,'b-a=':8,b-a:12:8)
       end;
    y:=x1;
    writeln('-------------------------------------------------------');
    writeln('SOLUTIA APROXIMATIVA DUPA ',nr_pasi,' PASI ESTE y=',y:12:8);
end.
```

Făcând în relația de mai sus $+b$ $-b$ se obține:

$$x_1 = b - \frac{f(b)}{f(b)-f(a)}(b-a)$$

Procedând analog se va obține, pe rând

$$x_2 = x_1 - \frac{f(a)}{f(x_1)-f(a)}(x_1-a)$$

$$x_3 = x_2 - \frac{f(a)}{f(x_2)-f(a)}(x_2-a)$$

...

$$x_n = x_{n-1} - \frac{f(a)}{f(x_{n-1})-f(a)}(x_{n-1}-a)$$

deci pentru $f(a)>0$ şi $f''(\frac{a+b}{2}) > 0$ a fix

dacă $f(a)\cdot f''(\frac{a+b}{2}) < 0 \Rightarrow b$ fix

Exemplu:

Să se determine soluția ecuației

$2x^3-6x^2-x+3 = 0$ din intervalul [0, 1]

Se observă, din calcul, faptul că

f(0) = 3 şi f(1) = -2, deci funcția are semne diferite la capetele intervalului.

Pseudocod

```
citeste a, b, ε, f    //expresia funcției f o dăm explicit
dacă f(a)f''((a+b)/2)>0 atunci
        {   fix <- a
            x0 <- b}
altfel  {   fix <- b
            x0 <- a}
               f(x0)
x1 <- x0 - --------------(x0-fix)
            f(x0)-f(fix)
cât_timp |x1-x0|>= ε execută
   {
      x0 <- x1
                        f(x0)
      x1 <- x0 - --------------(x0-fix)
                  f(x0)-f(fix)
   }
y <- x1
scrie y
```

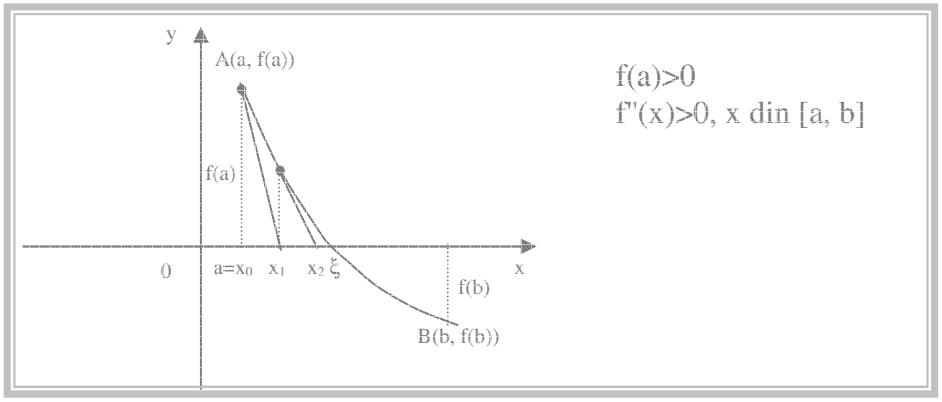
```
Program coarda2;
var a,b,eps,y,x0,x1,fix:real;
    nr_pasi:integer;

function f5(x:real):real;
begin
      f5:=x*x*x*x-x*x*x+x*x+2*x-6
end;

function f5secund(x:real):real;
begin
      f5secund:=12*x*x-6*x+2
end;

function f2(x:real):real;
begin
      f2:=x*ln(x)-1
end;

function f2secund(x:real):real;
begin
      f2secund:=1/x
end;

function f3(x:real):real;
begin
      f3:=x*x+ln(x)-4
end;

function f3secund(x:real):real;
begin
      f3secund:=2-1/(x*x)
end;

function f4(x:real):real;
begin
      f4:=2/3*sqrt(x*x+9)-x
end;

function f4secund(x:real):real;
begin
      f4secund:=6/((x*x+9)*sqrt(x*x+9))
end;

function f1(x:real):real;
begin
      f1:=2*x*x*x-6*x*x-x+3
end;

function f1secund(x:real):real;
begin
      f1secund:=12*x-12
end;

begin
    write('limita inferioara a=');readln(a);
    write('limita superioara b=');readln(b);
    write('precizia E=');readln(eps);
    write('f(x)=');readln(functia);
    clrscr;
    writeln('[a,b]=[',a:5:1,',',b:5:1,']    E=',eps:10:8);
    writeln('                    METODA COARDEI');
    writeln('    x0    ':12,'       x1                              fix');
    writeln('=============================================================');
    nr_pasi:=0;
       if f1(a)*f1secund((a+b)/2)>0
       then
         begin
            fix:=a; x0:=b
         end
       else
         begin
            fix:=b; x0:=a
         end;
    x1:=x0-f1(x0)/(f1(x0)-f1(fix))*(x0-fix);
    while abs(x1-x0)>eps do
       begin
          nr_pasi:=nr_pasi+1;
          x0:=x1;
          x1:=x0-f1(x0)/(f1(x0)-f1(fix))*(x0-fix);
          writeln(x0:12:8,x1:12:8,'x0-x1=':8,abs(x0-x1):12:8,'  ',fix:4:2)
       end;
    y:=x1;
    writeln('-------------------------------------------------------');
    writeln('SOLUTIA APROXIMATIVA DUPA ',nr_pasi,' PASI ESTE y=',y:12:8);
    repeat until keypressed
end.
```

Pentru exemplul de mai sus se obțin

y ≈ 0.70710678	27 paşi	înjumătățire
y ≈ 0.70710678	6 paşi	coarda1
y ≈ 0.70710678	7 paşi	coarda2
y ≈ 0.70710678	4 paşi	tangenta

4. Metoda tangentei (NEWTON - RAPHSON)

Fie $f:[a, b] \to \mathbf{R}$, continuă pe intervalul (a, b), cu f', f'' semn constant pe $[a, b]$



şi fie $x_0 = a$ astfel încât

$$f(x_0) \bullet f''(x_0) > 0$$

atunci, scriind ecuația tangentei la curba în punctul x_0, vom obține

$$y - y_0 = f'(x_0)(x - x_0)$$

şi intersectând această dreapta cu dreapta

$$y = 0$$

vom avea

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

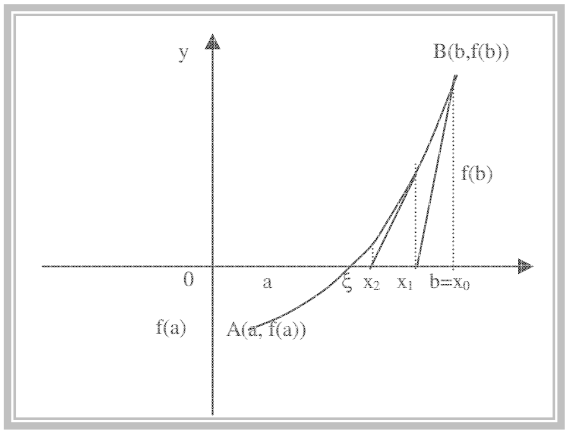
şi procedând analog pentru punctul x_1 obținut, apoi în continuare:

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

$$x_3 = x_2 - \frac{f(x_2)}{f'(x_2)}$$

...

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})} \quad (**)$$



În mod analog se deduce modul de calcul şi dacă graficul funcției este

caz în care $x_0 = b$.

Exemplu:

Să se determine soluția ecuației

$x^3+x^2-2x-2 = 0$ din intervalul (1, 2)

Se observă, din calcul, faptul că

$f(1) = -2$ şi $f(2) = 6$, deci funcția are semne diferite la capetele intervalului.

Pseudocod

```
citeste a, b, ε, f    //expresia funcţiei f o dăm explicit
                a+b
dacă f(a)f(---)>0 atunci
                 2
     x0 <- a
altfel
     x- <- b
repetă
              f(x0)
     y <- ------
              f'(x0)
     x0 <- x0 - y
până_când |y| <= ε
scrie x0
```

Obs. Deoarece de multe ori este dificil de calculat $f'(x)$ se poate aproxima aceasta prin

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

```
program tangenta;
var a,b,eps,y,x0:real;
     nr_pasi:integer;

function f5(x:real):real;
begin
  f5:=x*x*x*x-x*x*x+x*x+2*x-6
end;

function f5prim(x:real):real;
begin
  f5prim:=4*x*x*x-3*x*x+2*x+2
end;

function f5secund(x:real):real;
begin
  f5secund:=12*x*x-6*x+2
end;

function f2(x:real):real;
begin
  f2:=x*ln(x)-1
end;

function f2prim(x:real):real;
begin
  f2prim:=ln(x)+1
end;

function f2secund(x:real):real;
begin
  f2secund:=1/x
end;

function f3(x:real):real;
begin
  f3:=x*x+ln(x)-4
end;

function f3prim(x:real):real;
begin
  f3prim:=2*x+1/x
end;

function f3secund(x:real):real;
begin
  f3secund:=2-1/(x*x)
end;
```

```
function f4(x:real):real;
begin
  f4:=2/3*sqrt(x*x+9)-x
end;

function f4prim(x:real):real;
begin
  f4prim:=2/3*x/sqrt(x*x+9)-1
end;

function f4secund(x:real):real;
begin
  f4secund:=6/((x*x+9)*sqrt(x*x+9))
end;

function f1(x:real):real;
begin
  f1:=2*x*x*x-6*x*x-x+3
end;

function f1prim(x:real):real;
begin
  f1prim:=6*x*x-12*x-1
end;

function f1secund(x:real):real;
begin
  f1secund:=12*x-12
end;

function f6(x:real):real;
begin
  f6:=x*x*x+x*x-2*x-2
end;

function f6prim(x:real):real;
begin
  f6prim:=3*x*x+2*x-2
end;

function f6secund(x:real):real;
begin
  f6secund:=6*x+2
end;
```

```
begin
    write('limita inferioara a=');readln(a);
    write('limita superioara b=');readln(b);
    write('precizia E=');readln(eps);
    writeln('[a,b]=[',a:5:1,',',b:5:1,']    E=',eps:10:8);
    writeln('                    METODA TANGENTEI');
    writeln('    x0    ':12,'       y');
    writeln('=======================================================');
    nr_pasi:=0;
    if f1(a)*f1secund((a+b)/2)>0
    then x0:=a else x0:=b;
    repeat
       nr_pasi:=nr_pasi+1;
       y:=f1(x0)/f1prim(x0);
       x0:=x0-y;
       writeln(x0:12:8,abs(y):12:8)
    until abs(y)<eps;
    y:=x0;
    writeln('-------------------------------------------------------');
    writeln('SOLUTIA APROXIMATIVA DUPA ',nr_pasi,' PASI ESTE y=',y:12:8);
end.
```

Exerciții:

1)	$e^x+x-2=0$	[-1, 1]
2)	$x^5-4x^4+x-9=0$	[0, 10]
3)	$x^3-\sin(x)-9=0$	[0, 4]
4)	$6\ln(x)-\cos(x)-\sqrt{x}=0$	[1/e, e^2]
5)	$x^2\sqrt{x}-\sqrt{(x^2+11)}-8=0$	[0, 9]
6)	$\sqrt{(2+\ln(x))}+4x^2-x-5=0$	[1, 3]
7)	$x\sin(x^2)+x^3-1=0$	[0, 2]
8)	$e^{x^2}-x-\cos x-1=0$	[0, 3]
9)	$e^{2x}-x\sin(x)-2=0$	[0, 1]
10)	$\ln(1+2x)+e^x-x-2=0$	[0, 2]

5. Aplicație la metoda NEWTON-RAPHSON (algoritmul lui HERO)

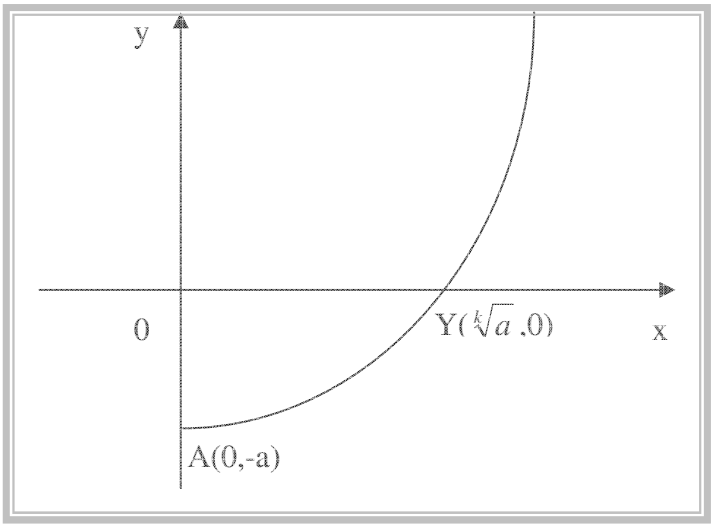
Fie $a>0$ => $\sqrt[k]{a}$ este rădăcina ecuației $x^k-a=0$

Conform metodei Newton-Raphson avem

$$f'(x)=k\cdot x^{k-1}>0$$
$$f''(x)=k\cdot(k-1)\cdot x^{k-2}>0,\ \forall x\in(0,\infty)$$

Graficul funcției

$f(x)=x^k-a$ este



deci ecuația are soluție unică
$\alpha\in R,\ \alpha>0$

Aplicând formula tangentelor (**) se obține

$$x_n = x_{n-1} - \frac{x_{n-1}^k - a}{k \cdot x_{n-1}^{k-1}},\ n = 1,2,\text{K}$$

sau transformând

$$\boxed{x_n = \frac{1}{k}\left[(k-1)x_{n-1} + \frac{a}{x_{n-1}^{k-1}}\right],\ n = 1,2,\text{K}}$$

Pentru k=2 se obține rădăcina de ordinul 2

$$x_n = \frac{1}{2}\left[x_{n-1} + \frac{a}{x_{n-1}}\right],\ n = 1,2,\text{K}$$

numită şi formula lui HERO

iar pentru k=3

$$x_n = \frac{1}{3}\left[2x_{n-1} + \frac{a}{x_{n-1}^2}\right],\ n = 1,2,\text{K}$$

Obs. Se poate demonstra că doar pentru $\mathbf{x_0 > 0}$ procesul este convergent.

Metoda mixtă (aplicarea simultană a metodei coardelor şi tangentelor). Reper matematic.
Metoda corzilor şi metoda tangentelor ne dau valorile aproximative ale rădăcinii ξ din ambele părți (prin adaos şi prin lipsă). De aceea ar fi mai rațional să le folosim simultan, caz în care precizarea rădăcinii are loc mai rapid.
Fie dată ecuația $f(x)=0$, rădăcina ξ care este deja separată şi se află în segmentul $[a,b]$. În plus mai sunt îndeplinite condițiile următoare: $f(a)\cdot f(b)<0$, iar $f'(x)$ şi $f''(x)$ îşi păstrează semnul pe segmentul cercetat.
Aplicăm simultan metoda coardelor şi secantelor, luând în considerație de asemenea şi tipul graficului funcției $f(x)$.
Teoretic sunt posibile 4 cazuri:

1) $f'(x)>0; f''(x)>0$ (fig.4)
2) $f'(x)<0; f''(x)<0$ (fig.5)
3) $f'(x)<0; f''(x)>0$ (fig.6)
4) $f'(x)>0; f''(x)<0$ (fig.7)

Concretizăm:
Dacă $f'(x)\cdot f''(x)>0$, atunci ***metoda coardelor*** ne dă soluția aproximativă ***prin lipsă***, iar ***metoda tangentelor – prin adaos***.

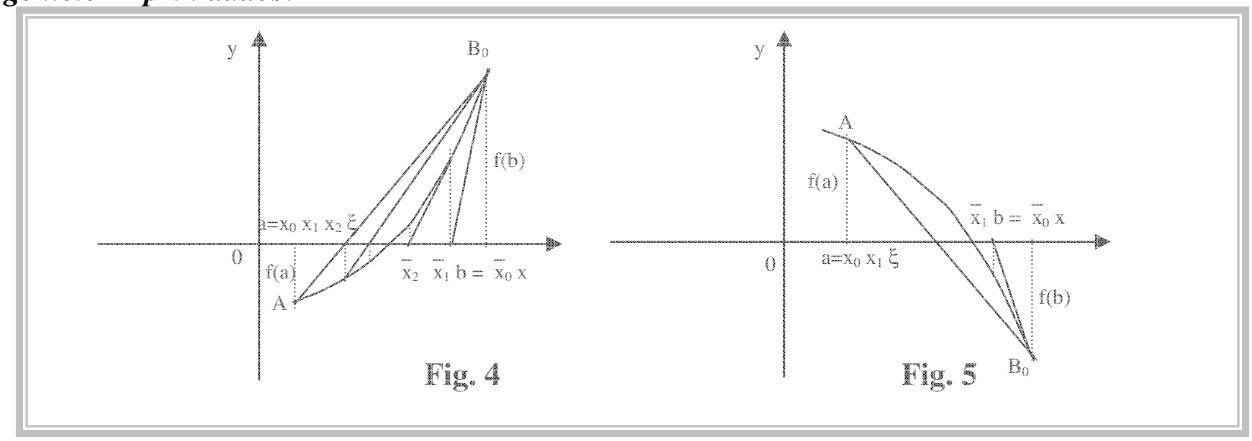


Fig. 4 Fig. 5

Dacă $f'(x)\cdot f''(x)<0$, atunci ***metoda coardelor*** ne dă valoarea rădăcinii ***prin adaos***, iar ***metoda tangentelor – prin lipsă***.

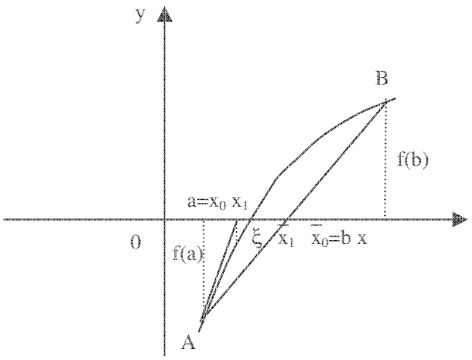


Fig. 6

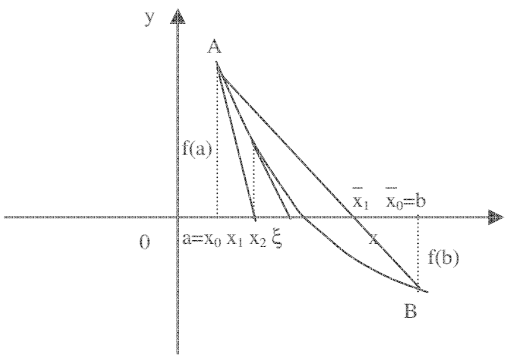


Fig. 7

Însă în toate cazurile, rădăcina reală ξ a funcției $f(x)$ este cuprinsă între valorile aproximative ale rădăcinilor aflate în urma aplicării simultane a ambelor metode. Mai precis, se respectă inegalitatea:

$$a < \overline{x_i} < \xi < \overline{\overline{x_i}} < b,$$

unde $\overline{x_i}$ – aproximația prin lipsă, iar $\overline{\overline{x_i}}$ – aproximația prin adaos.

Calculele pot fi efectuate în următoarea ordine.

Cazul I:

Dacă $f'(x)\cdot f''(x)>0$ şi $x\in[a, b]$, atunci la extremitatea stângă a se aplică metoda coardelor, iar la extremitatea dreaptă b – metoda tangentelor şi atunci vom avea:

$$a_1 = a - \frac{f(a)\cdot(b-a)}{f(b)-f(a)}, \quad b_1 = b - \frac{f(b)}{f'(b)} \qquad (1)$$

Acum rădăcina adevărată se găseşte pe intervalul $[a_1, b_1]$. Aplicând pe acest interval metoda mixtă vom căpăta aproximațiile următoare:

$$a_2 = a_1 - \frac{f(a_1)\cdot(b_1-a_1)}{f(b_1)-f(a_1)}, \quad b_2 = b_1 - \frac{f(b_1)}{f'(b_1)}$$

iar în caz general:

$$a_{i+1} = a_i - \frac{f(a_i)\cdot(b_i-a_i)}{f(b_i)-f(a_i)}, \quad b_{i+1} = b_i - \frac{f(b_i)}{f'(b_i)}. \qquad (2)$$

Cazul II:

Dacă $f'(x)\cdot f''(x)<0$, atunci de partea extremității stângi a, vor fi situate aproximațiile căpătate prin aplicarea metodei tangentelor, iar de partea extremității drepte b – valorile căpătate prin metoda corzilor. Atunci:

$$a_1 = a - \frac{f(a)}{f'(a)}, \quad b_1 = b - \frac{f(b)\cdot(b-a)}{f(b)-f(a)} \qquad (3)$$

Aplicând pe segmentul $[a_1, b_1]$ metoda mixtă, vom căpăta:

$$a_2 = a_1 - \frac{f(a_1)}{f'(a_1)}, \quad b_2 = b_1 - \frac{f(b_1)\cdot(b_1-a_1)}{f(b_1)-f(a_1)}$$

În caz general:

$$a_{i+1} = a_i - \frac{f(a_i)}{f'(a_i)}, \quad b_{i+1} = b_i - \frac{f(b_i)\cdot(b_i-a_i)}{f(b_i)-f(a_i)} \qquad (4)$$

Metoda mixtă este foarte comodă pentru estimarea erorii de calcul. Procesul de calcul se încheie atunci când se îndeplineşte inegalitatea: $\left|\overline{\overline{x_i}} - \overline{x_i}\right| < \varepsilon$. În acest caz în calitate de valoarea aproximativă a rădăcinii se poate lua valoarea: $\xi = \frac{1}{2}\left(\overline{x_i} - \overline{\overline{x_i}}\right)$, unde valorile $\overline{x_i}$ şi $\overline{\overline{x_i}}$ – aproximaţiile prin lipsă şi corespunzător prin adaos.

Algoritmul realizării metodei mixte

```
1. Definirea funcţiei f(x), a intervalului de lucru [a, b], a preciziei ε şi a
   numărului maxim de iteraţii nmax.
2. Proces de calcul:
      a) iniţializarea procesului de calcul:
         fa:=f(a); fb:=f(b); niter:=0;
         x:=a-(b-a)*fa/(fb-fa);     { calcul soluţie iniţială }
         fx:=f(x);
      b) Verificare semnelor la capetele intervalului cercetat cu alegerea
         aproximaţiei iniţiale:
      dacă f(a)*f(x)<=0 atunci
          {
            xt:=a;                  { aproximaţia iniţială este capătul a }
            repetă
               x:=x-(x-a)*fx/(fx-fa) { calcul după metoda coardelor }
               fx:=f(x);
               xt:=xt-f(xt)/fd(xt)   { calcul după metoda tangentelor }
               niter:=niter+1;
            până_când |x-xt|<ε       { s-a atins precizia dată }
          }
         altfel                      { f(a)*f(x)>0 }
           {
             xt:=b;                  { aproximaţia iniţială este capătul b }
             repetă
                x:=x-(b-x)*fx/(fb-fx)      {calcul după metoda coardelor }
                fx:=f(x);
                xt:=xt-f(xt)/fd(xt)        {calcul după metoda tangentelor }
                niter:=niter+1;
             până_când |x-xt|<ε            { s-a atins precizia dată }
           }
      c) soluţia este x:=(x+xt)/2
```

Implementarea programului de determinare a rădăcinilor prin metoda mixtă.
Problemă. Utilizând metoda mixtă, să se precizeze rădăcina ecuaţiei
$y=\cos x - 0{,}1x$ pe segmentul [1; 2] cu precizia $\varepsilon=10^{-3}$.

```
program metoda_mixta_initial;
var x,xt,fx,fa,fb,a,b,eps:real;
    i,niter:integer;

function f(x:real):real;
begin
  f:=cos(x)-0.1*x;
end;

function fd(x:real):real;
begin
  fd:=-sin(x)-0.1;
end;

begin { program de baza }
  a:=1; b:=2; eps:=0.001;
  fa:=f(a); fb:=f(b); niter:=0;
  x:=a-(b-a)*fa/(fb-fa);     { calcul solutie initiala }
  fx:=f(x);
```

```
  if f(a)*f(x)<=0
    then begin
           xt:=a;            { aproximatia initiala este capatul a }
           repeat
             x:=x-(x-a)*fx/(fx-fa);{ calcul dupa metoda coardelor }
             fx:=f(x);
             xt:=xt-f(xt)/fd(xt);  { calcul dupa metoda tangentelor }
             niter:=niter+1;
           until abs(x-xt)<eps     { s-a atins precizia data }
         end
    else begin { f(a)*f(x)>0 }
           xt:=b;                  { aproximatia initiala este capatul b }
           repeat
             x:=x-(b-x)*fx/(fb-fx);{ calcul dupa metoda coardelor }
             fx:=f(x);
             xt:=xt-f(xt)/fd(xt);  { calcul dupa metoda tangentelor }
             niter:=niter+1;
           until abs(x-xt)<eps;    { s-a atins precizia data }
         end;
  x:=(x+xt)/2;
  writeln('solutia ecuatiei date este x=',x:0:5);
  writeln('dupa ',niter,' iteratii');
end.
```

Aplicarea metodei mixte la determinarea tuturor soluțiilor unei ecuații neliniare

Problemă. Să se elaboreze programul de determinare a tuturor soluțiilor ecuații neliniare

$y=\cos x-0{,}1x$ cu precizia $\varepsilon=10^{-3}$.

Determinarea rădăcinilor ecuațiilor neliniare la calculator presupune următoarele:

- Separarea intervalelor ce conțin soluții separate (vezi modulul corespunzător);
- Aplicarea metodei în studiu pentru precizarea tuturor soluțiilor cu precizia dată.

Implementarea programului este dată în continuare:

```
program metoda_mixta;
var a,b,c,eps,x,x1,x2,h,y1,y2,fa,fb,fx,xt:Real;
    i,nit:Integer;

function f(x:real):real;
begin
  f:=cos(x)-0.1*x;
end;

function fd(x:real):real;
begin
  fd:=-sin(x)-0.1;
end;

function prec(eps:real):integer; { calcularea numarului de zecimale dupa virgula }
var k:integer;
begin
  k:=-1;
  repeat
    eps:=eps*10;
    k:=k+1
  until eps>1;
  prec:=k
end;

begin
  write('dati valoarea lui a=>'); readln(a);
  write('dati valoarea lui b=>'); readln(b);
  write('dati valoarea pasului h=>'); readln(h);
  write('dati valoarea lui eps='); readln(eps);
  x1:=a; x2:=x1+h; y1:=f(x1);
  while x2<b do
  begin
    y2:=f(x2);
    if f(x1)*f(x2)<0
      then begin
```

```
            writeln('pe intervalul [',x1:0:prec(eps),'; ',x2:0:prec(eps),']');
            fa:=f(x1); fb:=f(x2); nit:=0;
            x:=x1-(x2-x1)*fa/(fb-fa);     { calcul solutie initiala }
            fx:=f(x);
            if f(a)*f(x)<=0
              then begin
                     xt:=x1;   { aproximatia initiala este capatul a }
                     repeat
                       x:=x-(x-x1)*fx/(fx-fa);{ calcul dupa metoda coardelor }
                       fx:=f(x);
                       xt:=xt-f(xt)/fd(xt); { calcul dupa metoda tangentelor }
                       nit:=nit+1;
                     until abs(x-xt)<eps    { s-a atins precizia data }
                   end
              else begin
                     xt:=x2;  { aproximatia initiala este capatul b }
                     repeat
                       x:=x-(x2-x)*fx/(fb-fx);{ calcul dupa metoda coardelor }
                       fx:=f(x);
                       xt:=xt-f(xt)/fd(xt); { calcul dupa metoda tangentelor }
                       nit:=nit+1;
                     until abs(x-xt)<eps;   { s-a atins precizia data }
                   end;
            x:=(x+xt)/2;
            write('solutia ecuatiei este x=',x:0:prec(eps));
            writeln(' dupa ',nit,' iteratii');
          end;
          x1:=x2; x2:=x2+h;
          y1:=y2;
  end;
end.
```

Exemple

1. $2x^2-\cos x=0$ $(0;1)$, 10^{-3}
2. $x^3-\sin x=0$ 10^{-4}
3. $2e^x+3x+1=0$
4. $3x^4+4x^3-12x^2-5=0$
5. $0{,}5^x-1=(x+2)^2$
6. $(x+3)\cdot\cos x=1$ $-2\pi\le x\le 2\pi$

6. Metoda aproximațiilor succesive

Fie $f:[a, b] \to \mathbf{R}$, continuă pe intervalul (a, b), cu $f(a)\cdot f(b)<0$, deci are semne opuse la capetele intervalului, adică $\exists!\alpha\in(a,b), f(\alpha)=0$

Punem ecuația

$$f(x)=0$$

sub forma

$$x=g(x)$$

Deci trebuie determinat $\alpha\in(a,b)$ cu $\alpha=g(\alpha)$. O astfel de valoare se numeşte **punct fix** pentru funcția g.

Fie $x_0\in(a,b)$ o primă aproximație a lui α.

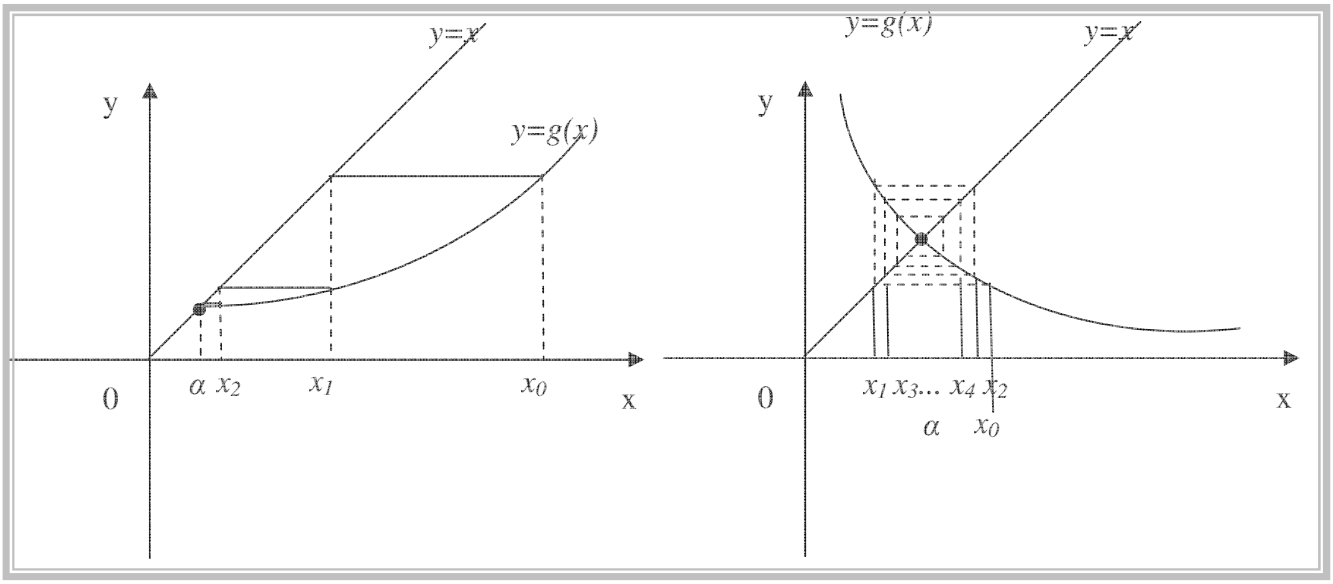
Construim pe rând şirul

$$x_1=g(x_0)$$
$$x_2=g(x_1)$$
$$\ldots$$
$$x_{k+1}=g(x_k) \quad (***)$$

Dacă şirul (x_k) este convergent, adică $\exists \alpha = \lim_{k\to\infty} x_k$, atunci, trecând la limită în (***) vom obține

$\alpha = g(\alpha)$ adică α este rădăcina căutată.

Geometric se poate reprezenta astfel:



Pseudocod

```
citeste x0, nr_it, ε, f    //expresia funcției f o dăm explicit
x(1) <- x0
i <- 1
execută
  i <- i+1
  x(i) <- f(x(i-1))
până_când (abs(x(i)-x(i-1)< ε)) sau (i>nrit)
dacă i<nrit atunci
  scrie i, ε, x(i)
altfel
  scrie „NU s-a obținut precizia dorită"
```

```
program aprox;
var eps, x0: real;
    i, nrit: integer;
    x: array[1..100] of real;
function g(x: real): real;
begin
     g := sqrt(4-ln(x))     { pentru ecuatia x^2+ln(x)-4=0 }
end;
begin
     write('aproximatia initiala:');readln(x0);
     write('precizia:');readln(eps);
     write('numar maxim de pasi:');readln(nrit);
     x[1] := x0;
     i := 1;
     writeln('in ', i, ' iteratii s-a obtinut solutia ', x[i]:13:8);
     repeat
       i := i+1;
       x[i] := g(x[i-1]);
       writeln('in ', i, ' iteratii s-a obtinut solutia ', x[i]:13:8)
     until (abs(x[i]-x[i-1])<eps) or (i>nrit);
     if i<nrit then
       writeln('in ', i, ' iteratii s-a obtinut solutia ', x[i]:13:8)
     else
       begin
         writeln(' in ',nrit,' iteratii NU s-a atins precizia ',eps:13:8);
         writeln(' ultima valoare obtinuta este ',x[i]:13:8)
       end;
end.
```

Exemple:

cox(x)-x = 0	(0, PI/2)	$\varepsilon=10^{-8}$	y=0.73908514	47 paşi
2sin(x)-x = 0	(PI/2, PI)	$\varepsilon=10^{-8}$	y=1.89549427	44 paşi
x^2+ln(x)-4 = 0	(0.5, 2)	$\varepsilon=10^{-8}$	y=1.84109706	12 paşi

Obs. Această ultimă ecuație pusă sub forma

$$x = \sqrt{4 - \ln x}$$

este convergentă către soluție, dar pusă sub forma

$x = e^{4-x^2}$, ceea ce se poate scrie (în Pascal) x=exp(4-x^2)

NU este convergentă.

Apare deci întrebarea: „cum alegem funcția *g* astfel încât să fie convergentă"?

TEOREMĂ

Fie ecuația $f(x) = 0$ (funcția $f(x)$ este continuă pe intervalul (a, b) şi $f(a)f(b)<0$). Punem ecuația sub forma $x=g(x)$. Procesul descris de metoda aproximțiilor succesive este convergent către soluția α a ecuției dacă

|g'(x)|<1, pentru orice x din intervalul (a, b)

Cum procedăm practic?

Ecuația

$$f(x) = 0$$

se înlocuieşte cu ecuația

$x = x - q \cdot f(x)$, unde $q > 0$,

astfel ca

|g'(x)| = | 1 – $q \cdot f'(x)$|<1

IV. REZOLVAREA SISTEMELOR DE ECUAȚII LINIARE

Rezolvarea sistemelor cu un mare număr de ecuații liniare reprezintă unul din domeniile în care calculatoarele numerice şi-au dovedit din plin eficiența. Problema rezolvării sistemelor de ecuații liniare este foarte des întâlnită în simularea numerică. Enumerăm câteva situații: interpolare cu funcții spline cubice, rezolvarea sistemelor de ecuații neliniare cu ajutorul metodelor iterative care au la bază liniarizarea ecuațiilor, discretizarea ecuațiilor diferențiale ordinare cu condiții la limită, discretizarea ecuațiilor cu derivate parțiale. În mod corespunzător, a trebuit să fie puse la punct procedee numerice adecvate, atât pentru reducerea numărului mare de operații, cât şi pentru reducerea erorilor de calcul care cresc cu dimensiunile sistemului de ecuații. În cazul general, problema care trebuie rezolvată poate fi scrisă sub forma

$$\sum_{j=1}^{n} a_{ij} x_j = b_i,\ i = \overline{1,m}$$

unde $a_{ij} \in$ R sunt coeficienți, x_j , j = 1,..., n sunt necunoscutele sistemului, iar b_i sunt termenii liberi. Vom distinge trei situații.

a. Pentru m < n sistemul este *subdeterminat*, avem mai puține ecuații decât necunoscute. În general, vor trebui aleşi n−m parametrii pentru a obține soluție.
b. Pentru m = n şi detA ≠ 0 sistemul este *compatibil determinat.* Sistemul are o soluție unică. Este cazul cel mai des întâlnit. Pentru m = n şi detA = 0 sistemul poate fi *compatibil nedeterminat*, cu o infinitate de soluții posibile, sau *incompatibil*, cu nici o soluție.
c. Pentru m > n sistemul este *supradeterminat*, caz în care se caută o soluție care să verifice "cel mai bine" ecuațiile în sensul minimizării reziduului $R = \sum_{i=1}^{n} (b_i - \sum_{j=1}^{n} a_{ij} \cdot x_j)^2$.

Nu trebuie uitat că la reprezentarea numerelor în calculator nu putem reține decât un număr finit de cifre, iar erorile de rotunjire se propagă. Se poate ajunge în situația ca din cauza acestor erori, determinantul să devină egal cu zero sau să aibă o valoare foarte mică în modul. Complexitatea implementărilor prezente în bibliotecile matematice se datorează în mare parte tehnicilor de evitare a acestei situații, dar totodată şi eforturilor de minimizare a memoriei utilizate şi a timpului de calcul.

Metodele de rezolvare a sistemelor de ecuații liniare sunt de două tipuri:

- *metode directe* (sau *metode de eliminare* sau *metode exacte*), în care soluția este obținută în urma unui număr de operații dinainte cunoscut;
- *metode iterative*, care se bazează pe folosirea unei aproximații inițiale ce se îmbunătățeşte de la o etapă la alta.

1. Metode exacte

a) Metoda lui CRAMER

Metoda este indicată pentru $n \le 4$.
Sistemul este

$A \cdot X = B$ cu $\det(A) \neq 0$, caz în care soluția este unică – sistemul este *compatibil determinat*.

Dacă $\det(A) \neq 0$, există A^{-1} (inversa matricii), care are proprietatea

$A^{-1} \cdot A = A \cdot A^{-1} = I$ (matricea unitate)

Atunci, înmulțind ecuația la stânga cu A^{-1} se obține, pe rând

$A^{-1} | A \cdot X = B$
$A^{-1} \cdot A \cdot X = A^{-1} \cdot B$
$I \cdot X = A^{-1} \cdot B$

Adică

$X = A^{-1} \cdot B$ soluția sistemului

Dar

$A^{-1} = 1/\Delta \cdot A^{*}$ unde A^{*} matricea cu complementi algebrici

rezultă

$X = 1/\Delta \cdot A^{*} \cdot B$

adică

$$\begin{Vmatrix} x_1 \\ x_2 \\ M \\ x_n \end{Vmatrix} = \frac{1}{\Delta} \cdot \begin{Vmatrix} \Delta_1 \\ \Delta_2 \\ M \\ \Delta_n \end{Vmatrix}$$

unde Δ_i sunt determinanții obținuți prin înlocuirea coloanei i cu coloana termenilor liberi.
Deci

$$x_i = \frac{\Delta_i}{\Delta}, \ \forall i = \overline{1,n}$$

Exemplu:

$$\begin{cases} 2x_1 + x_2 - x_3 = 3 \\ x_1 + x_2 + x_3 = 6 \\ -2x_1 + x_2 + 3x_3 = 7 \end{cases}$$

$\Delta = -4 \neq 0$
(1, 3, 2) – soluția

```
uses crt;
const n=3;
type matrice=array[1..n,1..n] of real;
var a, asav:array[1..n,1..n] of real;
    b, x:array[1..n] of real;
```

```
    perm:array[1..n] of integer;
    i,j:integer;
    det,deta:real;

procedure change(var a,b:integer);
var aux:integer;
begin
     aux:=a; a:=b; b:=aux;
end;

procedure suma;
var p:real;
    sign,i,j:integer;
begin
     p:=1;
     sign:=0;
     for i:=1 to n do
         for j:=i to n do
             if perm[i]>perm[j] then
                sign:=sign+1;
     for i:=1 to n do
         p:=p*a[i,perm[i]];
     if sign mod 2=0 then
        det:=det+p
     else
        det:=det-p;
end;

procedure permuta(n:integer);
var i:integer;
begin
     if n=1 then
        suma
     else
        for i:=1 to n do
            begin
                 change(perm[i],perm[n]);
                 permuta(n-1);
                 change(perm[i],perm[n]);
            end;
end;

procedure refacere;
var i,j:integer;
begin
     for i:=1 to n do
         for j:=1 to n do
             a[i,j]:=asav[i,j];
end;

begin
     clrscr;
     writeln('Intoduceti matricea A');
     for i:=1 to n do
         begin
              gotoxy(3,i+3); write(#186);
              gotoxy(6*n+4,i+3); write(#186);
         end;
     for i:=1 to n do
         for j:=1 to n do
             begin
                  gotoxy(6*j-1,i+3); readln(a[i,j]);
                  asav[i,j]:=a[i,j];
             end;
     gotoxy(6*n+10,1); writeln('Introduceti matricea B');
     for i:=1 to n do
         begin
              gotoxy(6*n+10,i+3); write(#186);
              gotoxy(6*n+17,i+3); write(#186);
         end;
     for i:=1 to n do
```

```
        begin
             gotoxy(6*n+11,i+3); readln(b[i]);
        end;
    for i:=1 to n do
        perm[i]:=i;
    gotoxy(10,n+5); writeln('Rezultat:');
    det:=0;
    permuta(n);
    deta:=det;
    if deta=0 then
       writeln('Eroare-det(A)=0')
    else
      for i:=1 to n do
        begin
             refacere;
             for j:=1 to n do
                 a[j,i]:=b[j];
             det:=0;
             permuta(n);
             x[i]:=det/deta;
             writeln('          x[',i,']=',x[i]:5:2);
        end;
end.
```

b) Metoda (de eliminare) lui GAUSS

Considerăm sistemul:

$A \cdot X = B$ cu $\det(A) \neq 0$

Pentru a descrie mai uşor metoda vom considera cazul n = 4. Deci sistemul nostru este:

$$\begin{cases} a_{11}x_1+a_{12}x_2+a_{13}x_3+a_{14}x_4 = b_1 \\ a_{21}x_1+a_{22}x_2+a_{23}x_3+a_{24}x_4 = b_2 \\ a_{31}x_1+a_{32}x_2+a_{33}x_3+a_{34}x_4 = b_3 \\ a_{41}x_1+a_{42}x_2+a_{43}x_3+a_{44}x_4 = b_4 \end{cases}$$

Presupunem $a_{11} \neq 0$, în caz contrar se schimbă între ele ecuațiile – soluțiile nu se schimbă.
(a) Împărțim prima ecuație cu a_{11} şi se obține:

$$x_1 + \frac{a_{12}}{a_{11}}x_2 + \frac{a_{13}}{a_{11}}x_3 + \frac{a_{14}}{a_{11}}x_4 = \frac{b_1}{a_{11}} \text{ adică}$$

$$x_1 + a_{12}^{(1)}x_2 + a_{13}^{(1)}x_3 + a_{14}^{(1)}x_4 = b_1^{(1)} \quad (*) \text{ unde } a_{1j}^{(1)} = \frac{a_{1j}}{a_{11}},\ b_1^{(1)} = \frac{b_1}{a_{11}},\ j>1$$

(b) Se elimină x_1 din ecuațiile 2, 3, 4 înmulțind pe rând ecuația (*) cu a_{21}, a_{31}, a_{41} şi scăzând-o din ecuațiile 2, 3, 4.

Vom obține sistemul:

$$\begin{cases} a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + a_{24}^{(1)}x_4 = b_2^{(1)} \\ a_{32}^{(1)}x_2 + a_{33}^{(1)}x_3 + a_{34}^{(1)}x_4 = b_3^{(1)} \\ a_{42}^{(1)}x_2 + a_{43}^{(1)}x_3 + a_{44}^{(1)}x_4 = b_4^{(1)} \end{cases} \text{ unde } a_{ij}^{(1)} = a_{ij} - a_{i1}\cdot a_{1j}^{(1)},\ b_i^{(1)} = b_i - a_{i1}\cdot b_1^{(1)} \quad i,j \geq 2$$

Presupunem $a_{22}^{(1)} \neq 0$. Se repetă etapele (a) şi (b) pentru sistemul obținut. Se obține:

$$x_2 + a_{23}^{(2)}x_3 + a_{24}^{(2)}x_4 = b_2^{(2)} \quad \text{unde } a_{2j}^{(2)} = \frac{a_{2j}^{(1)}}{a_{22}^{(1)}},\ b_2^{(2)} = \frac{b_2^{(1)}}{a_{22}^{(1)}},\ j>2$$

după pasul (b) se obține sistemul

$$\begin{cases} a_{33}^{(2)}x_3 + a_{34}^{(2)}x_4 = b_3^{(2)} \\ a_{43}^{(2)}x_3 + a_{44}^{(2)}x_4 = b_4^{(2)} \end{cases} \text{ unde } a_{ij}^{(2)} = a_{ij}^{(1)} - a_{i2}^{(1)}\cdot a_{2j}^{(2)},\ b_i^{(2)} = b_i^{(1)} - a_{i2}^{(1)}\cdot b_2^{(2)} \quad i,j \geq 3$$

Analog se va obține:

$$x_3 + a_{34}^{(3)} x_4 = b_3^{(3)} \quad \text{unde } a_{3j}^{(3)} = \frac{a_{3j}^{(2)}}{a_{33}^{(2)}},\ b_3^{(3)} = \frac{b_3^{(2)}}{a_{33}^{(2)}},\ j > 3$$

apoi, eliminând x_3

$$a_{44}^{(3)} x_4 = b_4^{(3)} \text{ unde } a_{ij}^{(3)} = a_{ij}^{(2)} - a_{i3}^{(2)} \cdot a_{3j}^{(3)} \quad i, j \geq 4$$

Acum:

$$x_4 = \frac{b_4^{(3)}}{a_{44}^{(3)}} = b_4^{(4)} \text{ apoi se calculează pe rând:}$$

$$x_3 = b_3^{(3)} - a_{34}^{(2)} \cdot x_4$$

$$x_2 = b_2^{(2)} - a_{24}^{(2)} \cdot x_4 - a_{23}^{(2)} \cdot x_3$$

$$x_1 = b_1^{(1)} - a_{14}^{(1)} \cdot x_4 - a_{13}^{(1)} \cdot x_3 - a_{12}^{(1)} \cdot x_2$$

În general, pentru un sistem

$A \cdot X = B$ cu $\det(A) \neq 0$

după k transformărise obține un sistem echivalent:

$A^{(k)} \cdot X = B^{(k)}$, cu $A^{(1)}=A$, $B^{(1)}=B$

iar pentru $k \geq 2$ elementele se calculează astfel:

$$a_{ij}^{(k)} = \begin{cases} a_{ij}^{(k-1)} & i \leq k-1 \\ 0 & i \geq k,\ j \leq k-1 \\ a_{ij}^{(k-1)} - \dfrac{a_{i,k-1}^{(k-1)}}{a_{k-1,k-1}^{(k-1)}} \cdot a_{k-1,j}^{(k-1)} & i, j \geq k \end{cases}$$

$$b_i^{(k)} = \begin{cases} b_i^{(k-1)} & i \leq k-1 \\ b_i^{(k-1)} - \dfrac{a_{i,k-1}^{(k-1)}}{a_{k-1,k-1}^{(k-1)}} \cdot b_{k-1}^{(k-1)} & i \geq k \end{cases}$$

Exemplu:

$$\begin{cases} x_1+2x_2-x_3+2x_4=2 \\ 2x_1-x_2+3x_3-x_4=-1 \\ x_1+3x_2+x_3+x_4=4 \\ -x_1+2x_2+4x_3+x_4=-3 \end{cases}$$ care are soluția (1, 2, -1, -2)

Metoda eliminării a lui Gauss constă în a obține zerouri succesiv, întâi pe prima coloană (sub coeficientul a_{11}), apoi pe a doua coloană (sub coeficientul a_{22})¸ s.a.m.d., pe ultima linie a matricei A rămânând doar coeficientul a_{nn} (evident modificat de operațiile de eliminare anterioare). Aceasta revine la a reduce matricea A la o *matrice superior triunghiulară*, iar sistemul la forma

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & & a_{1n} \\ 0 & a^{(1)}_{22} & \cdots & & a^{(1)}_{2n} \\ 0 & 0 & \cdots & & \\ 0 & 0 & 0 \cdots & a^{(i-1)}_{ij} \cdots & a^{(i-1)}_{in} \\ 0 & 0 & 0 & 0 & \\ \vdots & & & & \\ 0 & 0 & 0 & 0 \cdots & a^{(n-1)}_{nn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \\ x_i \\ \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b^{(1)}_2 \\ \\ b^{(i-1)}_i \\ \\ b^{(n-1)}_n \end{bmatrix}$$

Indicii superiori indică etapa în care a fost obținut elementul. Pentru a obține zerourile de sub diagonala principală, se folosesc operații simple de înmulțire a unei linii cu un *multiplicator* şi de

scădere din altă linie. Spre exemplu, pentru a obține zerouri pe prima coloanăa, din linia i se scade prima linie înmulțită cu multiplicatorul m_{i1}, obținându-se

$a_{i1}^{(1)} = a_{i1} - m_{i1}a_{11} = 0$ de unde se deduce $m_{i1} = a_{i1}/a_{11}$

Repetând procedeul pentru i = 2, n, se obțin elemente nule pe coloana întâi în această primă etapă. Evident, pentru a putea opera, trebuie ca $a_{11} \neq 0$. Mai mult, se recomandă ca a_{11} să fie în modul cât mai mare posibil, deoarece în acest mod, erorile de rotunjire sunt reduse. Elementul a_{ii} plasat pe diagonala principală se numeşte *pivot*. Obținerea unui pivot cu modul cât mai mare este posibilă prin schimbări de linii şi coloane care nu afectează zerourile deja produse, adică pivotul se poate alege dintre elementele de sub şi/sau la dreapta poziției de pe diagonala principală a pivotului. În funcție de numărul de elemente dintre care este selectat, pivotul poate fi *parțial*, când se alege cel mai mare în modul dintre elementele de pe coloana pivotului, sub diagonala principală, sau *total*, când se alege cel mai mare în modul dintre toate elementele conținute în linii şi coloane care, interschimbate, *nu modifică zerourile deja obținute*.

Pseudocod

```
citeşte n, a(i,j), i=1,n, j=1,n+1
pentru i=1, n execută
   {
      schimb(i, cod)
      dacă cod<>0 atunci scrie „sistem incompatibil sau nedetrminat”; exit
      redus(i)
   }
rezolv
scrie x(i), i=1,n

procedura maxim(i, k0)
//k0 - furnizează rangul ecuației cu coeficientul lui x_i cel mai mare în
//     valoare absolută
{
  m <- |a_ii|
  k= <- i
  pentru k <- i,n execută
     dacă |a_ki|>m atunci
        {
          m <- |a_ki|
          k0 <- k
        }
}

procedura schimb(i, cod)
//face ca ecuația de rang i să aibă coeficientul lui x_i mai mare în valoare
//absolută decât coeficienții lui x_i din ecuațiile de rang i+1, i+2, …, n
{
  maxim(i, k0)
  dacă a(k0, i)=0 atunci cod=1
  altfel
      {
        cod=0
        //schimbă linia i cu linia k0
        pentru j <- i,n+1 execută
            {t <- a(k0, j); a(k0, j) <- a(i, j); a(i, j) <- t}
      }
}

procedura redus(i)
//imparte ecuația de rang i cu a(i, i)
{
  pentru j <- i+1, n+1 execută
     a(i, j) <- a(i, j)/a(i, i)
```

```
        a(i, i) <- 1
        pentru k <- i+1, n execută
          {
            pentru j <- i+1, n+1 execută
              a(k, j) <- a(k, j)- a(i, j)a(k, i)
            a(k, i) <- 0
          }
      }

      procedura rezolv
      {
        x(n) <- a(n, n+1)
        pentru i <- n-1, 1 execută
            {
              s <- 0
              pentru j <- i+1, n execută
                  s <- s+a(i, j)x(j)
              x(i) <- a(i, n+1)-s
            }
      }
```

```
{ rezolvarea sistemelor prin metoda  de ELIMINARE a lui GAUSS  }
Program gauss;
Uses WinCrt;

Const Nmax=10;
{ Maxim 10 ecuatii cu 10 necunoscute - se poate modifica       }
Type Matrice=Array[1..Nmax,1..Nmax+1] Of Real;
{    a[i,n+1] este termenul liber b[i] al ecuatiei i           }
     Sir=Array[1..Nmax] Of Real;

Var a:Matrice;
{   x matricea coloana a solutiilor                            }
    x:Sir;
    i,j,Cod,n:Integer;
    Ch:Char;

Procedure ScrieMat;
Var i,j:0..Nmax+1;
Begin
       For i:=1 To n Do
       Begin
            WriteLn;
            For j:=1 To n Do Write(a[i,j]:8:3);
            Write(a[i,n+1]:12:3)
        End;
    WriteLn;WriteLn
End;

Procedure Maxim(i:Integer;Var Rang:Integer);
{ Rang - furnizeaza rangul ecuatiei cu coeficientul lui Xi     }
{        cel mai mare in valoare absoluta                      }
Var m:Real;
      k:Integer;
Begin
    m:=Abs(a[i,i]);
    Rang:=i;
    For k:=i To n Do
       If Abs(a[k,i])>m
          Then
              Begin
                m:=Abs(a[k,i]);
                Rang:=k
              End
End;

Procedure Schimb(i:Integer;Var Cod:Integer);
{ face ca ecuatia de Rang i sa aiba coeficientul lui Xi mai    }
{ mare in valoare absoluta decit coeficientii lui Xi din       }
```

```
{ ecuatiile de Rang i+1,i+2,...,n                                  }
Var t:Real;
    Rang,j:Integer;
Begin
    Maxim(i,Rang);
    If a[Rang,i]=0
              Then Cod:=1
{       daca coeficientul Maxim este 0=>toti sint =0 deci          }
{       sistemul nu se poate rezolva (pivotul = 0)                 }
    else
{           se face schimbarea ecuatiilor i cu Rang                }
       Begin
          Cod:=0;
          For j:=i To n+1 Do
            Begin
              t:=a[Rang,j];
              a[Rang,j]:=a[i,j];
              a[i,j]:=t
            End
      End;
End;

Procedure Redus(i:Integer);
Var j,k:Integer;
Begin
{   imparte ecuatia i cu a[i,i]                                    }
    For j:=i+1 To n+1 Do a[i,j]:=a[i,j]/a[i,i];
    a[i,i]:=1;
{   transforma toti coeficientii de deasupra diagonalei            }
    For k:=i+1 To n Do
       Begin
         For j:=i+1 To n+1 Do a[k,j]:=a[k,j]-a[i,j]*a[k,i];
{           si ii anuleaza pe cei de sub diagonala                 }
         a[k,i]:=0
      End
End;

Procedure Rezolv;
{ calculeaza solutiile prin eliminare inversa                      }
Var i,j:Integer;
      s:Real;
Begin
    x[n]:=a[n,n+1];
    For i:=n-1 DownTo 1 Do
       Begin
          s:=0;
          For j:=i+1 To n Do s:=s+a[i,j]*x[j];
          x[i]:=a[i,n+1]-s
      End
End;

Begin
  ClrScr;
  Write('dimensiune sistem:');ReadLn(n);
  For i:=1 To n Do
    Begin
      For j:=1 To n Do
        Begin
          Write('a[',i,',',j,']='); ReadLn(a[i,j])
        End;
      Write('b[',i,']=');ReadLn(a[i,n+1])
    End;
ClrScr;
WriteLn('Sistemul initial');
ScrieMat;
Ch:=ReadKey;
DetA:=1;
For i:=1 To n Do
  Begin
    Schimb(i,Cod);
      If Cod<>0
```

```
        Then
          Begin
            WriteLn('!!!!!!');
            WriteLn('sistem INCOMPATIBIL sau NEDERMINAT');
            Halt(1)
          End
      Else Redus(i);
    WriteLn('!!! PASUL:',i);
    ScrieMat;
    Ch:=ReadKey
  End;
  WriteLn;
  ClrScr;
  WriteLn('*** SOLUTIA ***');WriteLn;
  Rezolv;
  For i:=1 To n Do WriteLn('x[',i:1,']=',x[i]:8:3); WriteLn;
End.
```

EXEMPLU. Folosind metoda eliminării gaussiene, să se rezolve sistemul de ecuaţii

$$\left\{\begin{array}{rcrcrcr} x_1 & + & \frac{2}{9}x_2 & + & x_3 & = & 2 \\ 8x_1 & + & 2x_2 & - & 3x_3 & = & -1 \\ x_1 & + & 2x_2 & - & 5x_3 & = & 1 \end{array}\right. . \tag{5.11}$$

$$\left[\begin{array}{ccc|c} 1 & 0.222 & 1 & 2 \\ 8 & 2 & -3 & -1 \\ 1 & 2 & -5 & 1 \end{array}\right] \overset{l_1 \leftrightarrow l_2}{\rightarrow} \left[\begin{array}{ccc|c} \boxed{8} & 2 & -3 & -1 \\ 1 & 0.222 & 1 & 2 \\ 1 & 2 & -5 & 1 \end{array}\right] \overset{s=1}{\rightarrow}$$

$$\overset{s=1}{\rightarrow} \left[\begin{array}{ccc|c} 8 & 2 & -3 & -1 \\ 0 & -0.028 & 1.38 & 2.13 \\ 0 & 1.75 & -4.63 & 1.13 \end{array}\right] \overset{l_2 \leftrightarrow l_3}{\rightarrow} \left[\begin{array}{ccc|c} 8 & 2 & -3 & -1 \\ 0 & \boxed{1.75} & -4.63 & 1.13 \\ 0 & -0.028 & -1.38 & 2.13 \end{array}\right] \overset{s=2}{\rightarrow}$$

$$\overset{s=2}{\rightarrow} \left[\begin{array}{ccc|c} 8 & 2 & -3 & -1 \\ 0 & 1.75 & -4.63 & 1.13 \\ 0 & 0 & 1.31 & 2.15 \end{array}\right].$$

Rezultă

$$\bar{x}_3 = 0.164 \cdot 10^1,\ \bar{x}_2 = 0.498 \cdot 10^1,\ \bar{x}_1 = -0.755 \cdot 10^0\ . \tag{5.13}$$

Pentru a observa influenţa alegerii pivotului asupra preciziei rezultatelor, refacem calculele de mai sus fără a mai schimba liniile 2 cu 3 pentru a aduce valoarea 1.75 pe poziţia pivotului în etapa a doua. Se va obţine :

$$\left[\begin{array}{ccc|c} 8 & 2 & -3 & -1 \\ 0 & \boxed{-0.028} & 1.38 & 2.13 \\ 0 & 1.75 & -4.63 & 1.13 \end{array}\right] \overset{s=2'}{\rightarrow} \left[\begin{array}{ccc|c} 8 & 2 & -3 & -1 \\ 0 & -0.028 & 1.38 & 2.13 \\ 0 & 0 & 81.6 & 1.13 \end{array}\right]$$

şi soluţiile

$$\bar{x}'_3 = 0.164 \cdot 10^1,\ \bar{x}'_2 = 0.476 \cdot 10^1,\ \bar{x}'_1 = -0.7 \cdot 10^0$$

Se observă apariția unor valori diferite, comparativ cu valorile obținute în cazul anterior.

Numărul de operații necesare pentru obținerea soluției prin procedeul de eliminare gaussiană se calculează uşor. Evident, nu luăm în considerare şi operațiile legate de permutarea elementelor în vederea găsirii pivotului parțial sau total, deoarece acestea depind de fiecare matrice în parte. De regulă, se consideră că împărțirea reprezintă o singură "operație", în timp ce o adunare şi o înmulțire formează împreună tot o "operație" (aproximativ acelaşi timp de calcul pe calculator). Amintindu-ne că operațiile se efectuează cu elementele matricei extinse A0, de dimensiuni n · (n + 1), vom avea de efectuat într-o etapă s, n-s împărțiri pentru a obține multiplicatorii $m_{i,s}$. Numărul de adunări şi înmulțiri este egal cu produsul numărului de valori luate de indicii i, j, adică (n-s)(n+1-s). Rezultă numărul de operații la eliminarea Gauss

$$n_G = \sum_{s=1}^{n-1}[(n-s)^2 + 2(n-s)] = \frac{n^3}{3} + \frac{n^2}{2} - \frac{5n}{6}$$

Acest număr este foarte mic (începând cu n >= 3) față de $n \cdot n!$ operații cerute de regula lui Cramer (cu determinanții calculați după minori). La aceste operații se adaugă încă aproximativ

$1+(n-1)+n(n-1)/2 = n^2/2+n/2$

operații pentru retrosubstituire. Rezultă numărul total de operații pentru eliminarea gaussiană

$$n_G = \frac{n^3}{3} + n^2 - \frac{n}{3} \approx \frac{n^3}{3} + n^2$$

2. Metode iterative

Principiul general al metodelor iterative poate fi prezentat prin analogie cu metoda iterației simple de rezolvare a ecuației

$F(x) = 0$

în care ecuația originală este transcrisă ca

$x = f(x)$

ce conduce la procedeul iterativ

$x_{k+1} = f(x_k)$

În cazul sistemelor liniare

$A \cdot X = B$

vom forța o descompunere a matricei A

a) Metoda lui JACOBI (metoda iterațiilor simultane)

Vom considera, ca şi până acum, un sistem de forma

$$\sum_{j=1}^{n} a_{ij} \cdot x_j = b_j, \ i = 1, n$$

adică A·X = B, cu det(A) ≠ 0 – matrice nesingulară

Vom presupune că

$a_{ii} \neq 0$, i=1, n

Forțăm descompunerea matricei A sub forma

A = D + (-T) + (-S)

adică

$$A = \begin{pmatrix} a_{11} & a_{12} & \text{L} & a_{1n} \\ a_{21} & a_{22} & \text{L} & a_{2n} \\ \text{M} & & & \\ a_{n1} & a_{n2} & \text{L} & a_{nn} \end{pmatrix} = \begin{pmatrix} a_{11} & 0 & \text{L} & 0 \\ 0 & a_{22} & \text{L} & 0 \\ \text{M} & & & \\ 0 & 0 & \text{L} & a_{nn} \end{pmatrix} + \begin{pmatrix} 0 & 0 & \text{L} & 0 \\ a_{21} & 0 & \text{L} & 0 \\ \text{M} & & & \\ a_{n1} & a_{n2} & \text{L} & 0 \end{pmatrix} + \begin{pmatrix} 0 & a_{12} & \text{L} & a_{1n} \\ 0 & 0 & \text{L} & a_{2n} \\ \text{M} & & & \\ 0 & 0 & \text{L} & 0 \end{pmatrix}$$

adică, sub formă matricială

$(D-T-S)\cdot X = B$

deci

$D\cdot X=(T+S)\cdot X + B$

În acest fel, pentru orice element $a_{ii} \neq 0$, i=1, n avem o metodă iterativă

$$a_{ii}\cdot x_i^{(k+1)} = -\sum_{\substack{j=1\\ j\neq i}}^{n} a_{ij}\cdot x_j^{(k)} + b_i, \quad i=1,2,\text{L}\ ,n \qquad k\geq 0$$

cu $x_i^{(0)}$, $i=1,n$ aproximațiile inițiale

Se obține metoda iterativă JACOBI

$$\boxed{x_i^{(k+1)} = -\sum_{\substack{j=1\\ j\neq i}}^{n} (\frac{a_{ij}}{a_{ii}})\cdot x_j^{(k)} + \frac{b_i}{a_{ii}}, \quad i=1,2,\text{L}\ ,n \qquad k\geq 0}$$

Pseudocod

```
citește n, a(i,j), i=1,n, j=1,n+1
citește x(i), i=1,n
pentru k=1, 7 execută  //pentru 7 iterații
   {
      pentru i=1,n execută
        {
            y(i) <- a(i, n+1)/a(i,i)
            pentru j=1,n execută
              dacă j≠i
                  y(i) <- y(i)- a(i,j)/a(i,i)·x(j)
        }
      pentru j <- 1, n execută
        x(j) <- y(j)
   }
pentru i <- 1,n execută
   scrie x(i)
```

```
{ rezolvarea sistemelor prin metoda ITERATIVA  J A C O B I     }
Program Jacobi;
Const NMax=10;
      Nr_Iteratii=7;
Type Matrice=Array[1..NMax,1..NMax+1] Of Real;
     Sir=Array[1..NMax] Of Real;
Var a:Matrice;
{   x[i] - aproximatia veche                                   }
{   y[i] - aproximatia noua care se calculeaza                 }
    x,y: Sir;
    i,j,k,n: Integer;
    c: Char;

Procedure ScrieMat;
Var i,j: 0..NMax+1;
Begin
       For i:=1 To n Do
         Begin
           WriteLn;
           For j:=1 To n Do Write(a[i,j]:8:3);
           Write(a[i,n+1]:10:3);
           Write(x[i]:13:5,y[i]:10:5)
         End;
    WriteLn; WriteLn
End;

Begin
  Write('dimensiune sistem:');ReadLn(n);
  For i:=1 To n Do
    Begin
```

```
      For j:=1 To n Do
        Begin
          Write('a[',i,',',j,']=');ReadLn(a[i,j]);
        End;
     Write('b[',i,']=');ReadLn(a[i,n+1])
   End;
WriteLn('aproximatia initiala');
For i:=1 To n Do
  Begin
    Write('x[',i,']=');ReadLn(x[i]);
    y[i]:=0
  End;
WriteLn('SISTEMUL INITIAL SI APROXIMATIA INITIALA');
ScrieMat;
{   vom face 7 iteratii - se poate modifica                    }
    For k:=1 To Nr_Iteratii Do
       Begin
{       pentru fiecare ecuatie se face calculul aproximatiei   }
        For i:=1 To n Do
          Begin
            y[i]:=a[i,n+1]/a[i,i];
            For j:=1 To n Do
              If j<>i Then y[i]:=y[i]-a[i,j]/a[i,i]*x[j];
          End;
        WriteLn('!!!   ITERATIA:',k);ScrieMat;
{           noua aproximatie devine cea curenta                }
        For i:=1 To n Do x[i]:=y[i];
       End;
    For i:=1 To n Do Write(x[i]:9:5);
    WriteLn;
End.
```

b) Metoda lui GAUSS-SEIDEL (metoda aproximațiilor succesive)

Metoda reprezintă o îmbunătățire a metodei JACOBI în sensul că utilizează nu doar aproximațiile $x_i^{(k)}$ pentru calculul lui $x_i^{(k+1)}$, ci şi componentele $x_j^{(k+1)}$ cu i>j, adică acele componente care au fost deja calculate în aproximația curentă.
În aceste condiții se poate scrie

$$a_{ii}\cdot x_i^{(k+1)} = -\sum_{j=1}^{i-1} a_{ij}\cdot x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}\cdot x_j^{(k)} + b_i, \quad i=1,2,\text{L}\ ,n \quad k=1,2,...,n$$

deci metoda iterativă GAUSS-SEIDEL se poate descrie prin

$$x_i^{(k+1)} = -\sum_{j=1}^{i-1} (\frac{a_{ij}}{a_{ii}})\cdot x_j^{(k+1)} - \sum_{j=i+1}^{n} (\frac{a_{ij}}{a_{ii}})\cdot x_j^{(k)} + \frac{b_i}{a_{ii}}, \qquad i=1,2,\text{L}\ ,n \quad k=1,2,...,n$$

Pseudocod

```
citeşte n, a(i,j), i=1,n, j=1,n+1
citeşte x(i), i=1,n
pentru k=1, 7 execută  //pentru 7 iteraţii
   {
      pentru i=1,n execută
        {
            y(i) <- a(i, n+1)/a(i,i)
            pentru j=1,i-1 execută
              y(i) <- y(i)- a(i,j)/a(i,i)·y(j)
            pentru j=i+1,n execută
              y(i) <- y(i)- a(i,j)/a(i,i)·x(j)
            x(i) <- y(i)//noua aproximaţie devine cea curentă
        }
   }
pentru i <- 1,n execută
   scrie x(i)
```

```
Program GAUSS_SEIDEL;
Const NMax=10;
      Nr_Iteratii=10;
Type Matrice=Array[1..NMax,1..NMax+1] Of Real;
     Sir=Array[1..NMax] Of Real;
Var a:Matrice;
{   x[i] - aproximatia veche                                  }
{   y[i] - aproximatia noua care se calculeaza                }
    x,y:Sir;
    i,j,k,n:Integer;

Procedure ScrieMat;
Var i,j:0..NMax+1;
Begin
  For i:=1 To n Do
    Begin
       WriteLn;
       For j:=1 To n Do Write(a[i,j]:8:3);
       Write(a[i,n+1]:10:3);
       Write(x[i]:13:5,y[i]:10:5)
    End;
    WriteLn;WriteLn
End;

Begin
   Write('dimensiune sistem:');ReadLn(n);
   For i:=1 To n Do
     Begin
       For j:=1 To n Do
         Begin
           Write('a[',i,',',j,']=');ReadLn(a[i,j]);
         End;
      Write('b[',i,']=');ReadLn(a[i,n+1])
    End;
  WriteLn('aproximatia initiala');
  For i:=1 To n Do
    Begin
      Write('x[',i,']=');ReadLn(x[i]);
      y[i]:=0
    End;
 WriteLn('SISTEMUL INITIAL SI APROXIMATIA INITIALA');
 ScrieMat;
{   vom face 7 iteratii - se poate modifica                   }
 For k:=1 To Nr_Iteratii Do
   Begin
{       pentru fiecare ecuatie se face calculul aproximatiei  }
     For i:=1 To n Do
       Begin
         y[i]:=a[i,n+1]/a[i,i];
         For j:=1 To i-1 Do
           y[i]:=y[i]-a[i,j]/a[i,i]*y[j];
         For j:=i+1 To n Do
           y[i]:=y[i]-a[i,j]/a[i,i]*x[j];
       End;
     WriteLn('!!!   ITERATIA:',k);ScrieMat;
{           noua aproximatie devine cea curenta               }
      For j:=1 To n Do x[j]:=y[j];
  End;
  For i:=1 To n Do Write(x[i]:9:5); WriteLn;
End.
```

Obs. Cele două metode iterative dau rezultate (sunt convergente) doar pentru sisteme în care matricea coeficienților (a_{ij}) este de un tip special şi anume coeficienții diagonali sunt dominanți.

Exemplu:

$$\begin{cases} 10x_1 + x_2 + 2x_3 = 13 \\ 5x_2 + 3x_3 = 8 \\ -x_1 + x_2 + 7x_3 = 7 \end{cases}$$

cu aproximația inițială $x_1^{(0)} = 2, x_2^{(0)} = 3, x_3^{(0)} = 5$ furnizează

```
SISTEMUL INITIAL SI APROXIMATIA INITIALA
  10.000   1.000   2.000    13.000      2.00000   0.00000
   0.000   5.000   3.000     8.000      3.00000   0.00000
  -1.000   1.000   7.000     7.000      5.00000   0.00000

!!!   ITERATIA:1
  10.000   1.000   2.000    13.000      2.00000  -0.00000
   0.000   5.000   3.000     8.000      3.00000  -1.40000
  -1.000   1.000   7.000     7.000      5.00000   1.20000

!!!   ITERATIA:2
  10.000   1.000   2.000    13.000     -0.00000   1.20000
   0.000   5.000   3.000     8.000     -1.40000   0.88000
  -1.000   1.000   7.000     7.000      1.20000   1.04571

!!!   ITERATIA:3
  10.000   1.000   2.000    13.000      1.20000   1.00286
   0.000   5.000   3.000     8.000      0.88000   0.97257
  -1.000   1.000   7.000     7.000      1.04571   1.00433

!!!   ITERATIA:4
  10.000   1.000   2.000    13.000      1.00286   1.00188
   0.000   5.000   3.000     8.000      0.97257   0.99740
  -1.000   1.000   7.000     7.000      1.00433   1.00064

!!!   ITERATIA:5
  10.000   1.000   2.000    13.000      1.00188   1.00013
   0.000   5.000   3.000     8.000      0.99740   0.99962
  -1.000   1.000   7.000     7.000      1.00064   1.00007

!!!   ITERATIA:6
  10.000   1.000   2.000    13.000      1.00013   1.00002
   0.000   5.000   3.000     8.000      0.99962   0.99996
  -1.000   1.000   7.000     7.000      1.00007   1.00001

!!!   ITERATIA:7
  10.000   1.000   2.000    13.000      1.00002   1.00000
   0.000   5.000   3.000     8.000      0.99996   0.99999
  -1.000   1.000   7.000     7.000      1.00001   1.00000

!!!   ITERATIA:8
  10.000   1.000   2.000    13.000      1.00000   1.00000
   0.000   5.000   3.000     8.000      0.99999   1.00000
  -1.000   1.000   7.000     7.000      1.00000   1.00000

!!!   ITERATIA:9
  10.000   1.000   2.000    13.000      1.00000   1.00000
   0.000   5.000   3.000     8.000      1.00000   1.00000
  -1.000   1.000   7.000     7.000      1.00000   1.00000

!!!   ITERATIA:10
  10.000   1.000   2.000    13.000      1.00000   1.00000
   0.000   5.000   3.000     8.000      1.00000   1.00000
  -1.000   1.000   7.000     7.000      1.00000   1.00000

  1.00000  1.00000  1.00000
```

Bibliografie

1. MATEESCU G-D, MATEESCU I-C, **Analiză numerică**, Editura Petrion, 1995
2. COMAN, FRENTIU **– Analiză numerică**, 1992
3. BERBENTE C, MITRAN S, ZANCU S – **Metode numerice**, Editura Tehnică, 1997
4. *** - Materiale preluate de pe Internet