Laborator 13 – Polinoame şi ecuaţii

1. Probleme cu polinoame

1.1 Se consideră un polinom de gradul n:

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

și se cere să se calculeze valorile acestui polinom în intervalul $[x_i, x_f]$ cu un pas h de variaţie a abscisei, efectuând un număr minim de operaţii.

Pentru a rezolva problema de mai sus realizând un număr minim de operaţii, se poate scrie polinomul utilizând forma lui Horner:

$$P_n(x) = (\cdots(((a_n x + a_{n-1})x + a_{n-2})x + a_{n-3})x + \cdots + a_1)x + a_0$$

Se observă că dacă se cunoaște valoarea unei paranteze interioare, atunci valoarea parantezei exterioare rezultă prin înmnulţirea parantezei interioare cu abscisa curentă și prin adunarea unui coeficient al polinomului. Dacă paranteza la un anumit pas are valoarea P, atunci procedeul de obţinere al unei paranteze exterioare pornind de la valoarea celei interioare este: $P = Px + a_j$, cu P iniţial 0 și $j = \overline{n..0}$.

```
#include <stdio.h>
#include <math.h>
#define MAX 1000
void main(void)
{
double xi,xf,a[MAX],h,x,P=0;
int i,j,np,n;
printf("Introduceti capetele intervalului:\n");
printf("xi=");
scanf("%lf",&xi);
printf("xf=");
scanf("%lf",&xf);
printf("Pasul h=");
scanf("%lf",&h);
printf("Introduceti gradul polinomului n=");
scanf("%d",&n);
while(n<=0||n>MAX){
printf("Dimensiune eronata:%d\n",n);
printf("Introduceti alt n:");
scanf("%d",&n);
}
printf("Introduceti coeficientii polinomului:\n");
for(i=0;i<=n;i++){
	printf("a[%d]=",i);
	scanf("%lf",&a[i]);
}
np=(xf-xi)/h+1;
for(i=1;i<=np;i++){
	x=xi+(i-1)*h;
	for(j=n;j>=0;j--)
```

```
            P=P*x+a[j];
printf("Valoarea polinomului in punctul %.2lf este: %.2lf\n",x,P);
}
}
```

1.2 Fiind dat un polinom de gradul n al cărui coeficienţi sunt cunoscuţi, să se determine câtul c şi restul r ai împărţirii acestuia la polinomul x-b. Câtul va fi un polinom de gradul n-1.

$$a_n x^n + \ldots + a_1 x + a_0 = \left(c_{n-1}x^{n-1} + \ldots + c_1 x + c_0\right)\cdot(x-b) + r$$

Acesta este un caz particular al împărţirii polinoamelor, deoarece împărţitorul este un polinom de gradul I. Probabil cel mai uşor mod de a explica împărţirea a două polinoame este folosind un exemplu:

Fie polinomul deîmpărţit: $P = x^3 - 6x^2 - 15$, iar împărţitorul: $p = x - 2$.

Împărţirea acestor polinoame se realizează în felul următor:

Deîmpărţitul se scrie astfel: $P = x^3 - 6x^2 + 0x - 15$.

1. Se împarte primul termen al deîmpărţitului la termenul cu cel mai înalt rang al împărţitorului, iar rezultatul se scrie deasupra barei, aşa cum este prezentat mai jos:

$$x-2\big)\overline{x^2}\\ x^3 - 6x^2 + 0x - 15$$

2. Se înmnulţeşte împărţitorul cu rezultatul obţinut în acest moment (x^2), iar rezultatul se scrie sub primii doi termeni ai deîmpărţitului:

$$\begin{array}{l} x-2\big)\overline{\quad\quad x^2\quad\quad\quad\quad} \\ \quad\quad\; x^3 - 6x^2 + 0x - 15 \\ \quad\quad\; x^3 - 2x^2 \end{array}$$

3.Se scade produsul obţinut din termenii corespunzători ai deîmpărţitului, iar apoi se ”aduce jos” şi următorul termen al deîmpărţitului (0x în cazul nostru).

$$\begin{array}{l} x-2\big)\overline{\quad\quad x^2\quad\quad\quad\quad} \\ \quad\quad\; x^3 - 6x^2 + 0x - 15 \\ \quad\quad\; \underline{x^3 - 2x^2} \\ \quad\quad\quad -4x^2 \;\; +0x \end{array}$$

4. Se repetă paşii 1-3 cu menţiunea că cei doi noi termeni devin deîmpărţit acum:

$$\begin{array}{l} x-2\big)\overline{\quad\quad x^2 - 4x\quad\quad\quad} \\ \quad\quad\; x^3 - 6x^2 + 0x - 15 \\ \quad\quad\; \underline{x^3 - 2x^2} \\ \quad\quad\quad -4x^2 \;\; +0x \\ \quad\quad\quad \underline{-4x^2 \;\; +8x} \\ \quad\quad\quad\quad\; -8x \quad -15 \end{array}$$

5. Se repetă pasul 4 până când nu mai rămâne niciun termen de adus jos:

$$
\begin{array}{r|l}
 & x^2 - 4x - 8 \\
\cline{2-2}
x-2) & x^3 - 6x^2 + 0x - 15 \\
 & x^3 - 2x^2 \\
\cline{2-2}
 & \quad -4x^2 \quad + 0x \\
 & \quad -4x^2 \quad + 8x \\
\cline{2-2}
 & \qquad\quad -8x \quad -15 \\
 & \qquad\quad -8x \quad +16 \\
\cline{2-2}
 & \qquad\qquad -31
\end{array}
$$

```
#include <stdio.h>
#include <math.h>
#define MAX 1000
void main(void)
{
double a[MAX],q[MAX],r,b;
int i,n;
printf("Introduceti gradul polinomului n=");
scanf("%d",&n);
while(n<=0||n>MAX){
printf("Dimensiune eronata:%d\n",n);
printf("Introduceti alt n:");
scanf("%d",&n);
}
printf("Introduceti coeficientii polinomului:\n");
for(i=0;i<=n;i++){
	printf("a[%d]=",i);
	scanf("%lf",&a[i]);
}
printf("Introduceti valoarea lui b=");
scanf("%lf",&b);
for(i=n;i>0;i--){
	q[i-1]=a[i];
	r=a[i-1]-b*q[i-1];
	a[i-1]=r;
}
printf("Coeficientii polinomului cat sunt:");
for(i=n-1;i>=0;i--)
		printf("%lf ",q[i]);
printf("\nRestul impartirii polinoamelor este: %lf\n",r);
}
```

1.3 Se consideră un polinom de gradul n cu coeficienţi cunoscuţi:

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

Să se determine elementele unei matrice cu n+1 coloane şi n+1 linii care conţine (pe linii) coeficienţii polinomului dat precum şi coeficienţii polinoamelor care rezultă în urma derivărilor succesive ale polinomului iniţial până când se ajunge la un polinom de gradul zero.

```
#include <stdio.h>
#include <math.h>
#define MAX 100
void main(void)
{
double a[MAX],b[MAX][MAX];
int i,j=0,n,k=0;
printf("Introduceti gradul polinomului n=");
scanf("%d",&n);
while(n<=0||n>MAX){
printf("Dimensiune eronata:%d\n",n);
printf("Introduceti alt n:");
scanf("%d",&n);
}
printf("Introduceti coeficientii polinomului:\n");
for(i=0;i<=n;i++){
	printf("a[%d]=",i);
	scanf("%lf",&a[i]);
}
printf("Matricea ceruta este:\n");
for(i=0;i<=n;i++){
	b[j][i]=a[i];
	printf("%lf\t",b[j][i]);
}
printf("\n");
for(j=1;j<=n;j++){
	for(i=0;i<=n;i++){
		a[i]=a[i]*(i-k);
		b[j][i]=a[i];
		printf("%lf\t",b[j][i]);
		}
	printf("\n");
	k++;
}
}
```

2. Rezolvarea ecuaţiilor utilizând metode numerice

2.1 Să se scrie un program cu ajutorul căruia să se rezolve ecuaţia: $f(x) \equiv x \cdot \sin\left(\frac{50}{x}\right) - 10 = 0$ folosind metoda înjumătăţirii intervalelor, știind că are o rădăcină în intervalul [10,25].

O ecuaţie are o rădăcină în intervalul [a,b], dacă funcţia corespunzătoare ecuaţiei admite o schimbare de semn ceea ce matematic se transpune în relaţia: $f(a) \cdot f(b) < 0$. Reprezentarea grafică a funcţiei este în figura 1.

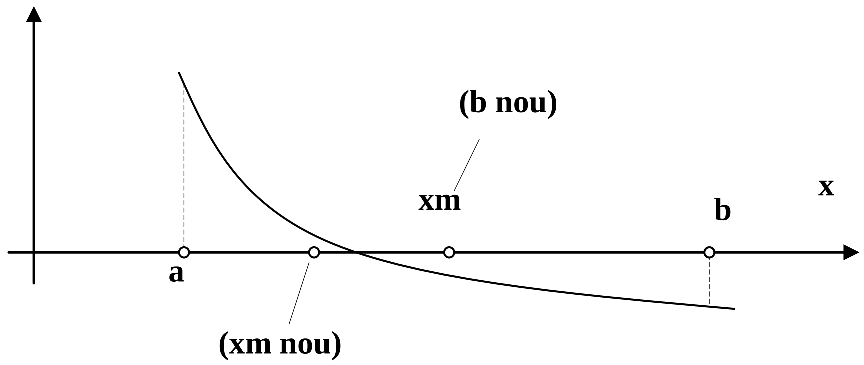

Fig. 1 Metoda înjumătăţirii intervalelor

Metoda înjumătăţirii intervalului este o metodă numerică de determinare a soluţiei care se bazează pe următoarele: se calculează mijlocul intervalului [a,b] notat cu xm după care se stabileşte unde se găseşte soluţia faţă de xm: la stânga lui xm în intervalul [a,xm], la dreapta lui xm în intervalul [xm,b] sau este chiar xm, situaţie în care soluţia este determinată exact şi eroarea ε devine 0.

```
#include<stdio.h>
#include<math.h>
#define F x*sin(50/x)-10
void main()
{
double a,b,eps,vs,vd,x,xm,vm;
printf("Introduceti capatul stanga al intervalului a=");
scanf("%lf",&a);
printf("Introduceti capatul dreapta al intervalului b=");
scanf("%lf",&b);
printf("Introduceti valoarea erorii eps=");
scanf("%lf",&eps);
x=a;
vs=F;
x=b;
vd=F;
if(vs*vd==0)
        if(vs==0)
                printf("Solutia ecuatiei este: %lf\n",a);
        else
                printf("Solutia ecuatiei este: %lf\n",b);
else
        if(vs*vd>0)
        printf("Ecuatia nu are solutii in intervalul [%lf,%lf]\n",a,b);
        else
        while(vs*vd<0){
                xm=(a+b)/2;
                x=xm;
                vm=F;
                if(fabs(a-b)<eps){
```

```
                    printf("Solutia ecuatiei este: %lf, cu o eroare de: %lf\n",xm,eps);
                    break;
            }
            else
                    if(vs*vm<0)
                            b=xm;
                    else
                            if(vs*vm==0){
                                    eps=0;
                                    printf("Solutia ecuatiei este: %lf, cu o eroare de:
%lf\n",xm,eps);
                                    break;
                            }
                            else{
                                    a=xm;
                                    x=a;
                                    vs=F;
                            }
        }

}
```

Algoritmul de calcul este următorul:

1. Se citesc a,b, ε.
2. Se calculează valorile funcţiei vs şi vd în respectiv .
3. Se verifică dacă produsul . Dacă este adevărat se continuă cu pasul 6 iar în caz contrar cu pasul 4.
4. Dacă produsul se trece la pasul 5 iar în caz contrar se afişează mesajul "Ecuaţia nu are soluţii in intervalul [a,b]".
5. Dacă soluţia ecuaţiei este a, se afişează şi STOP. În caz contrar soluţia este b, se afişează şi STOP.
6. Se calculează xm.
7. Se calculează valoarea funcţiei vm în punctul xm.
8. Grupul următor de calcule se realizează în mod repetat până când mărimea intervalului [a,b] devine mai mică decât ε.

8.a. Se verifică dacă . Dacă este adevărat (se lucrează cu jumătatea din stânga a intervalului). În caz contrar se trece la pasul 8.b.
8.b. Se verifică dacă . Dacă este adevărat ε devine 0 iar în caz contrar a=xm (se lucrează cu jumătatea din dreapta a intervalului) şi se calculează valoarea funcţiei vs.

Se tipăresc în final valorile xm şi ε.

2.2 Să se determine cel mai mic interval în care se găseşte o soluţie a ecuaţiei $f(x) \equiv x - \tan(x) = 0$, fiind dat intervalul iniţial [a,b] și pasul de parcurgere h.

Se va utiliza metoda parcurgerii. Prin această metodă se doreşte determinarea intervalului de lăţime „h" în care să se găsească o singură rădăcină reală. Pentru aplicarea acestei metode limitele inferioară şi superioară a rădăcinilor trebuie cunoscute, (acestea fiind totodată şi date de intrare în cadrul algoritmului), acestea definind de fapt domeniul în care se caută intervalele.

```
#include<stdio.h>
#include<math.h>
#define F x-tan(x)
void main()
{
double a,b,m,n,h,vs,vd,sol,x;
printf("Introduceti limita stanga a intervalului a=");
scanf("%lf",&a);
printf("Introduceti limita dreapta a intervalului b=");
scanf("%lf",&b);
printf("Introduceti pasul de parcurgere a intervalului h=");
scanf("%lf",&h);
m=a;
n=a+h;
sol=0;
while(n<=b){
	x=m;
	vs=F;
	x=n;
	vd=F;
	if(vs*vd>0){
		m=n;
		n=m+h;
	}
	else{
		if(vs*vd==0)
			if(vs==0){
				printf("Solutia ecuatiei este x[%.0lf]=%lf\n",a);
				sol++;
			}
			else{
				printf("Solutia ecuatiei este x[%.0lf]=%lf\n",b);
				sol++;
			}
		else{
			printf("Solutia ecuatiei este x[%.0lf] in intervalul
[%lf,%lf]\n",sol,m,n);
			sol++;
		}
		m=n;
		n=m+h;
	}
}
}
```

2.3 Să se rezolve ecuaţia $f(x) \equiv 2x^3 - 6x^2 - x + 3 = 0$ folosind metoda secantei pentru intervalul [0,1].

Metoda secantei, numită şi metoda părţilor proporţionale, restrânge intervalul [a,b] ce încadrează soluţia exactă în iteraţia curentă prin raportarea la valorile funcţiei la extremităţile intervalului. Astfel, dacă noua aproximaţie x se alege astfel încât soluţia exactă să se găsească în intervalul [a,x], valoarea lui x rezultă din egalitatea proporţiilor:

$$\frac{x-a}{b-a} = \frac{0-f(a)}{f(b)-f(a)}$$

Interpretarea geometrică a acestei relaţii corespunde construcţiei din figura 2.

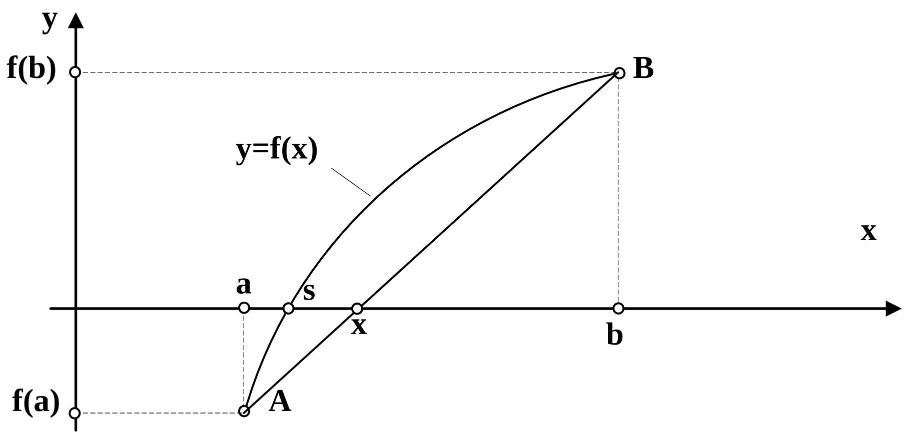

Fig. 2 Metoda secantei

Astfel, pe intervalul [a,b] curba y=f(x) este aproximată prin dreapta ce trece prin cele doua puncte A şi B de la extremităţile intervalului. În aceste condiţii, soluţia exactă (s) urmează a fi aproximată prin abscisa punctului de intersecţie a dreptei AB cu axa OX, notată cu x. Noua aproximaţie x se determină impunând condiţia ca în ecuaţia dreptei AB :

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

punctul (x,y) să se găsească pe axa OX, adică y=0:

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

În continuare, dintre intervalele [a,x] şi [x,b] se reţine acela ce încadrează soluţia s - acel interval la extremităţile căruia funcţia f(x) ia valori de semne contrare. Aplicând o relaţie asemănătoare noului interval se obţine o nouă aproximaţie a soluţiei exacte şi procesul continuă până la satisfacerea unui criteriu de oprire.

```
#include<stdio.h>
#include<math.h>
#define F 2*pow(x,3)-6*pow(x,2)-x+3
void main()
{
double a,b,vs,vd,x,eps,y;
int iter,i=0;
printf("Introduceti limita stanga a intervalului a=");
scanf("%lf",&a);
printf("Introduceti limita dreapta a intervalului b=");
```

```
scanf("%lf",&b);
printf("Introduceti eroarea eps=");
scanf("%lf",&eps);
printf("Introduceti numarul maxim de iteratii iter=");
scanf("%d",&iter);
while(fabs(b-a)<eps||i<iter){
        x=a;
        vs=F;
        x=b;
        vd=F;
        x=a-vs/(vd-vs)*(b-a);
        if(F*vd<0)
                a=x;
        else
                if(F*vd>0)
                        b=x;
                else{
                        a=x;
                        b=x;
                }
                i++;
}
printf("Solutia aproximativa a ecuatiei este: %lf\n",x);
}
```

Algoritmul programului este următorul:

1. Definirea funcţiei f(x), a intervalului de lucru [a,b], a preciziei eps și a numărului maxim de iteratii iter.
2. Calcularea expresiei: $x = a - \frac{b-a}{f(b) - f(a)} f(a)$
    2.a Dacă $f(x) \cdot f(b) < 0$, a=x
    2.b Dacă $f(x) \cdot f(b) > 0$, b=x
    2.c Dacă $f(x) \cdot f(b) = 0$, a=x, b=x
3. Dacă $|b-a| < eps$ sau s-a atins numărul maxim de iteraţii, se scrie soluţia aproximativă a ecuaţiei: x, altfel se revine la pasul 2.