Lucrarea 6

INTEGRAREA NUMERICĂ

1. SCOPUL LUCRĂRII

Prezentarea principiului integrării numerice precum şi a unor algoritmi consacraţi de integrare numerică, implementarea acestora în limbaje de nivel înalt (în particular ANSI C*) şi aplicarea lor în rezolvarea diferitelor probleme din electronică.*

2. PREZENTARE TEORETICĂ

Metodele numerice de integrare se clasifică după tipul funcţiei de integrat şi valoarea limitelor de integrare.

I. *Prima grupă* de metode se referă la funcţiile continue şi cu limite finite de integrare. Aceste metode se împart la rândul lor în două subgrupe în funcţie de modul de divizare a intervalului de integrare:

a) Metode ce împart intervalul de integrare în subintervale de aceeaşi lungime, numărul subintervalelor fiind impus de operator. Dintre aceste metode amintim: metoda *dreptunghiului*, metoda *trapezului*, metoda lui *Simpson* şi metoda lui *Richardson*;

b) Metode ce împart intervalul de integrare în aşa fel încât eroarea de calcul să fie minimă. Dintre aceste metode amintim şi studiem *metoda cuadraturii a lui Gauss*.

II. *A doua grupă* de metode se referă la integralele improprii, adică la integrarea funcţiilor cu discontinuităţi de speţa întâi şi a doua pe intervale de integrare finite sau integrarea funcţiilor continue pe intervale de integrare infinite.

III. *A treia grupă* de metode numerice de integrare se ocupă cu integrarea dublă a funcţiilor de două variabile. Amintim în acest sens formulele de *cubatură a trapezului* şi a lui *Simpson*.

2.1. INTEGRAREA FUNCȚIILOR DE O SINGURĂ VARIABILĂ. METODE CU DIVIZARE CONSTANTĂ

2.1.1. METODA DREPTUNGHIULUI

Fie integrala:

$$I=\int_{a}^{b} f(x)\,dx \qquad (6.1)$$

unde $f(x)$ este o funcție continuă pe $[a,b]$, iar a şi b sunt finite.

Calculul numeric al acestei integrale se realizează prin divizarea intervalului $[a,b]$ în *n* subintervale de lungime egală cu:

$$Dx_i=\frac{b-a}{n}=x_{i+1}-x_i=h \text{ , cu } i = 0,1,...,n-1 \qquad (6.2)$$

Se calculează aproximativ *aria* fiecărui dreptunghi:

$$s_i=\frac{b-a}{n}\cdot f(x_i) \qquad (6.3)$$

şi se însumează aceste arii:

$$\sum_{i=0}^{n-1} s_i=h\cdot\sum_{i=0}^{n-1} f(x_i) \qquad (6.4)$$

unde $x_i=a+h\cdot i$

Formula (6.4) reprezintă formula de integrare a *dreptunghiului*.

Precizia metodei este limitată. Această variantă de calcul este de obicei didactică, având ca scop înțelegerea principiului numeric de calcul al integralelor. Eroarea sa de trunchiere este zero doar pentru *funcții constante*, adică polinoame de *grad 0*.

2.1.1. Algoritmul 5.1. Metoda dreptunghiului

```
real I_Dreptunghi
(
   Adr_functie,       // Adresa unei funcții reale de
                      // variabilă reală (pointer la funcție)
   real ls,    // limita stângă a intervalului de integrare
   real ld,    // limita dreaptă a intervalului de integrare
   întreg n    // numărul de subintervale
)
{ // corpul de instrucțiuni al funcției

// declararea şi definirea variabilelor locale
   real h;     // valoarea lungimii unui subinterval
   real sum;   // variabilă care ține valoarea integralei
   întreg i;   // indice pentru ciclul 'for()'
```

```
    // operațiile efective
    sum = 0;    // inițializarea variabilei în care ținem
                // valoarea integralei.
```
$h = \dfrac{ld - ls}{n}$;
```
    pentru i= 1 ÷ n
          sum = sum + h* (*Adr_functie)(ls+i*h);
    returnează  sum;
}
```

Obs.:

Notațiile '*Adr_functie*' și '*(*Adr_functie)(...)*' sunt simbolice. Ele reprezintă, respectiv, adresa unei funcții (numele pointerului la funcție) și apelul indirect al unei funcții, prin intermediul pointerului către acea funcție. Vedeți *Anexa B* - "*Noțiuni de C necesare desfășurării lucrărilor de laborator*" pentru lămuriri.

2.1.2. METODA TRAPEZULUI

Fie integrala:

$$I = \int_a^b f(x)\,dx\ ,$$

unde $f(x)$ este continuă pe [a, b] și *a, b* sunt finite.

Intervalul [a, b] se împarte în *n* subintervale de lungime egală, Δx_i:

$$Dx_i = x_{i+1} - x_i = \frac{b-a}{n} = h \qquad ,\ i=0,1,...,n-1,$$

Aria I_i este aproximată prin aria trapezului având coordonatele vârfurilor $(x_{i,}\ f(x_i))\, respectiv (x_{i+1,}\ f(x_{i+1}))$, adică:

$$I_i » \frac{f(x_i)+f(x_{i+1})}{2} h \tag{6.5}$$

Sumându-se apoi relația (6.5) după *i*, acoperindu-se astfel toate fâșiile de arie posibile, rezultă:

$$I » \frac{h}{2}\sum_{i=0}^{n-1}\left[f(x_i)+f(x_{i+1})\right] = \frac{h}{2}\left[f(x_0)+2\mathrm{f}(x_1)+\ldots+2(x_{n-1})+f(x_n)\right] \tag{6.6}$$

În această relație se ține cont că limitele de integrare sunt de fapt $x_0=a$, și $x_n=b$.

Această expresie reprezintă formula de integrare numerică prin *metoda trapezului* și are eroare de trunchiere nulă pentru funcții polinomiale *până la gradul întâi inclusiv*. Această exprimare se traduce prin aceea că, dacă dorim să comparăm metodele între ele trebuie să facem apel la dezvoltarea în serie de puteri (sau Taylor) a funcției de integrat. În raport de câți termeni preluăm din această dezvoltare, putem să clasificăm metodele din punctul de vedere al preciziei de calcul.

2.1.2.1. Algoritmul 5.2. Metoda trapezului

```
real I_Trapez
(
   Adr_functie,       // Adresa unei funcții reale de
              // variabilă reală (pointer la funcție)
   real ls,    // limita stângă a intervalului de integrare
   real ld,    // limita dreaptă a intervalului de integrare
   întreg n    // numărul de subintervale
)
{
   // declararea şi definirea variabilelor locale
   real h;     // valoarea lungimii unui subinterval
   real sum;   // se reține valoarea integralei
   întreg i;   // indice pentru ciclul 'for()'

   // corpul de instrucțiuni al funcției
```

$$h = \frac{ld - ls}{n}$$;

$$sum = \frac{(*Adr(f))(l_s) + (*Adr(f))(l_d)}{2}$$ * h;

```
   pentru i= 1÷ n-1
         sum = sum + h*(*Adr_functie)(ls + i*h);
   returnează  sum;
}
```

Obs.:

Notațiile '*Adr_functie*' şi '*(*Adr_functie)(...)*' sunt simbolice. Ele reprezintă, respectiv, adresa unei funcții (numele pointerului la funcție) şi apelul indirect al unei funcții, prin intermediul pointerului către acea funcție. Vedeți *Anexa B* - "*Noțiuni de C necesare desfăşurării lucrărilor de laborator*" pentru lămuriri.

2.1.3. METODA LUI RICHARDSON

Această metodă dă o precizie mai bună de calcul a integralei numerice decât metoda trapezului şi s-a obținut prin modificarea metodei trapezului. Este o metodă în care apare *dubla-divizare* a intervalului de integrare.

Se pleacă de la eroarea de trunchiere a metodei trapezului, care pentru o diviziune de lățime $h = (b-a)/n$ este:

$$e_{T,h} = Ch^2 \tag{6.7}$$

Pentru cealaltă diviziune, $k = (b-a)/m$, se obține eroarea de trunchiere:

$$e_{T,k}=Ck^2 \tag{6.7'}$$

Pentru fiecare diviziune se poate scrie:

$$I = I_h + e_{T,h}$$

respectiv (6.8)

$$I = I_k + e_{T,k}$$

Eliminând constanta *C* între relaţiile (6.8) se obţine expresia:

$$I = I_h + \frac{I_h - I_k}{\left(\frac{k}{h}\right)^2 - 1} \tag{6.8'}$$

expresie ce poartă denumirea de *formula lui Richardson*.

Precizia acestei metode este mai bună decât metoda trapezului, oferind rezultate comparabile sau mai bune decât ale metodei trapezului, dar folosind un număr de diviziuni mai mic. Însă gradul maxim al unei evaluări polinomiale a funcţiei de integrat rămâne acelaşi cu cel din metoda trapezului. Astfel, această metodă oferă erori de trunchiere minime sau nule pentru funcţii polinom de până în *grad 1*, inclusiv.

2.1.3.1. Algoritmul 5.3. Metoda lui Richardson

```
real I_Richardson
(
   Adr_functie,      // Adresa unei funcţii reale de
                     // variabilă reală (pointer la funcţie)
   real ls,    // limita stângă a intervalului de integrare
   real ld,    // limita dreaptă a intervalului de integrare
   întreg n,   // numărul de subintervale_1
   întreg m    // numărul de subintervale_2
)
{
// declararea şi definirea variabilelor locale
   real h;     // valoarea lungimii unui subinterval divizat
               // in n sub-diviziuni
   real k;     // valoarea lungimii unui subinterval divizat
               // im m sub-diviziuni
   real sumh;  // valoarea integralei cu diviziunea h
   real sumk;  // valoarea integralei cu diviziunea k
   real sum;   // valoarea integralei
   întreg i;   // indice întreg pentru ciclul 'for()'

// instrucţiunile de calcul
```

$h = \frac{ld - ls}{n}$; $k = \frac{ld - ls}{m}$;

$sumh = \frac{(*Adr(f))(l_s) + (*Adr(f))(l_d)}{2}$ `* h;`

$$sumk = \frac{(*Adr(f))(l_s) + (*Adr(f))(l_d)}{2}$$ `* k;`

```
pentru i= 1 ÷ n-1
      sumh = sumh + h*(*Adr_functie)(ls + i*h);
pentru i= 1 ÷ m-1
      sumk = sumk + k*(*Adr_functie)(ls + i*k);
sum=sumh + (sumh-sumk)/( (k/h)*(k/h)-1 );

returnează sum;
}
```

Obs.:
Notațiile '*Adr_functie*' şi '*(*Adr_functie)(...)*' sunt simbolice. Ele reprezintă, respectiv, adresa unei funcții (deci un pointer la o funcție) şi apelul indirect al unei funcții, prin intermediul pointerului către acea funcție. Vedeți *Anexa B* - "*Noțiuni de C necesare desfăşurării lucrărilor de laborator*" pentru lămuriri.

2.1.4. METODA LUI SIMPSON

Formula de calcul a metodei lui *Simpson* se poate deduce uşor utilizând formula lui Richardson.

Şi această metodă utilizează două diviziuni (dublă-divizare, ca în metoda Richardson), între care avem dependența:

$$k = 2\mathrm{h} \tag{6.9}$$

Dacă cele două diviziuni sunt $k = \frac{b-a}{m}$, şi $h = \frac{b-a}{n}$, şi scriem formula metodei trapezului pentru fiecare diviziune în parte:

$$I_h \approx \frac{h}{2} \sum_{i=0}^{n-1} \left[f(x_i) + f(x_{i+1}) \right] = \frac{h}{2} \left[f(x_0) + 2\mathrm{f}(x_1) + \ldots + 2(x_{n-1}) + f(x_n) \right] ,$$

respectiv:

$$I_k \approx \frac{k}{2} \sum_{i=0}^{m-1} \left[f(x_i) + f(x_{i+1}) \right] = \frac{k}{2} \left[f(x_0) + 2\mathrm{f}(x_1) + \ldots + 2(x_{n-1}) + f(x_n) \right]$$

atunci folosind formula (6.8'), în care înlocuim relațiile (6.10) deducem relația metodei Simpson:

$$I = \frac{h}{3} \left[f(x_0) + 4\mathrm{f}(x_1) + 2\mathrm{f}(x_2) + 4\mathrm{f}(x_3) + \ldots + 4\mathrm{f}(x_{n-1}) + f(x_n) \right] \tag{6.10}$$

Expresia (6.10) reprezintă formula de calcul numeric al integralei pentru *metoda lui Simpson*. Atragem atenția că *n* este un *număr par*.

Precizia de calcul a acestei metode este mai bună decât ale metodelor precedente. Se arată (v. [*Dorn, McCracken*]) că eroarea de trunchiere este nulă pentru polinoame cu *grad cel mult 3*.

Nu trebuie interpretat că este o metodă dedicată funcțiilor tip polinom. Ci doar că, pentru erori minime sau nule, funcția de integrat trebuie ori să fie un polinom de grad cel mult 3, ori o funcție nepolinomială, care de fapt se trunchiază la primii 4 termeni ai dezvoltării în serie de puteri, sau Taylor, asociată.

2.1.4.1. Algoritmul 5.4. Metoda lui Simpson

```
real I_Simpson
(
   Adr_functie,       // Adresa unei funcţii reale de
               // variabilă reală (pointer la o funcţie)
   real ls,    // limita stângă a intervalului de integrare
   real ld,    // limita dreaptă a intervalului de integrare
   întreg n    // numărul de subintervale (par)
)
{
   // declararea şi definirea variabilelor locale
   real h;     // valoarea lungimii unui subinterval
   real suma;  // în această variabila se reţine valoarea
               // integralei
   întreg i;   // contor al ciclului for()

   // instrucţiunile efective de calcul
```

$h = \dfrac{ld - ls}{n}$;

$sum = h * \dfrac{(*Adr(f))(l_s) + (*Adr(f))(l_d)}{3}$;

```
   pentru i= 1 ÷ n-1
      dacă (i este par)
         sum = sum + 2*h*(*Adr_functie)(ls+i*h)/3;
      altfel  sum = sum + 4*h*(*Adr_functie)(ls+i*h)/3;

   returnează  sum;
}
```

Obs.:

1) Vedeți *Anexa B* - “*Noțiuni de C necesare desfăşurării lucrărilor de laborator*”, secțiunea “*Operatori*”, pentru testul de paritate (folosirea operatorului % ‘*modulus*’). O altă variantă (de asemenea discutată în *Anexa B*) este folosirea în tandem a unei structuri de tipul ‘*div_t*’ şi a funcției ‘*div()*’.

2) Notațiile ‘*Adr_functie*’ şi ‘*(*Adr_functie)(...)*’ sunt simbolice. Ele reprezintă, respectiv, adresa unei funcții (numele pointerului la funcție) şi apelul indirect al unei

funcţii, prin intermediul pointerului către acea funcţie. Vedeţi Anexa 2 - "*Noţiuni de C necesare desfăşurării lucrărilor de laborator*" pentru lămuriri.

2.2. INTEGRAREA FUNCŢIILOR DE O SINGURĂ VARIABILĂ PRIN METODE CU DIVIZARE VARIABILĂ

Dintre aceste metode se prezintă *metoda cuadraturii lui Gauss*. Această metodă determină punctele de divizare ale intervalului de integrare astfel ca eroarea de calcul a integralei să fie minimă.

2.2.1. METODA CUADRATURII GAUSSIENE CU DOUĂ PUNCTE

Această metodă reduce orice interval de integrare [a,b] la intervalul [-1,1] cu ajutorul formulei de *substituţie*:

$$y=\frac{2x-(b+a)}{b-a} \tag{6.11}$$

Pentru *x=1* rezultă *y=-1*, iar pentru *x=b* rezultă *y=1*. Din relaţia (5.11) se poate scrie:

$$x=\frac{1}{2}(b-a)y+\frac{1}{2}(b+a) \tag{6.12}$$

şi de aici:

$$dx=\frac{1}{2}(b-a)dy \tag{6.13}$$

Ca urmare, integrala iniţială în forma $I=\int_a^b f(x)dx$ se transformă în integrala:

$$I=\int_{-1}^{+1} f\left[\frac{1}{2}(b-a)y+\frac{1}{2}(b+a)\right]\frac{1}{2}(b-a)dy=\int_{-1}^{+1}\phi(y)dy \tag{6.14}$$

în care noua variabilă este *y*. S-au folosit în determinarea relaţiei (6.14) formulele (6.12) şi (6.13).

Integrala din (6.14) este în final aproximată printr-o sumă, aşa cum s-a făcut de fiecare dată (din punct de vedere matematic integrala devine o sumă). Mai exact:

$$\int_{-1}^{+1}\phi(y)dy = \sum_{i=1}^{n} A_i\cdot\phi(U_i) \tag{6.14'}$$

Precizia metodei este cea mai bună între metodele de integrare de *tip cuadratură* (pentru funcţii de o variabilă). Această putere de calcul este de fapt justificată, din moment ce metoda foloseşte în rezolvare *polinoamele Legendre*. Echivalentul matematic este acela de aproximare a funcţiei concrete de integrat printr-un polinom

Legendre echivalent, pentru un anumit grad dorit de către utilizator. Cu cât gradul preluat în calcule este mai mare, cu atât precizia metodei evident creşte.

Polinoamele Legendre sunt definite pe intervalul [-1, 1]. De aceea se impune schimbarea limitelor de integrare inițiale. Pe acest interval polinoamele prezintă un număr de rădăcini reale, distincte (gradul de multiplicitate al fiecărei soluții este 1). Din cauză că aceste rădăcini nu sunt egal depărtate între ele, şi nici față de capetele intervalului de integrat, această metodă este clasificată drept metodă *cu divizare variabilă*.

Această precizie de calcul se poate impune în mod direct, prin stabilirea gradului polinomului *Legendre* dorit în calcul.

Există valori tabelate ale rădăcinilor acestor polinoame (notate aici cu U_i), ca şi a ponderilor (A_i), valori ce intervin în relația (6.14'). S-au dat aceste valori pentru toate gradele polinomului Legendre până la 16.

2.2.1.1. **Algoritmul 5.5. Metoda cuadraturii lui Gauss**

```
real I_Gauss
(
   Adr_functie,       // Adresa unei funcţii reale de o
                   // variabilă reală (pointer la o funcţie)
   real ls,    // limita stângă a intervalului de integrare
   real ld,    // limita dreaptă a intervalului de integrare
   întreg n    // gradul polinomului lui Legendre
)
{
// declararea şi definirea variabilelor locale
   real A[N][N];      // matricea ponderilor
   real U[N][N];      // matricea soluțiilor polinoamelor lui
                      // Legendre
   real sum;          // valoarea integralei
   întreg i;          // contor într-un ciclu for()

   // instrucțiunile de calcul ale metodei
   Construieşte vectorul/matricea A[N][N];   // v. mai jos
   Construieşte vectorul/matricea U[N][N];   // v. mai jos

   suma = 0;
   pentru  i= 0 ÷ n-1
      sum = sum +
            0.5 * A[n][i]*(ld-ls)*
            (*Adr_functie)(0.5*(ld- ls)*U[n][i] + 0.5*( ls+ld));

   returnează sum;
}
```

Obs.:

Notațiile '*Adr_functie*' şi '*(*Adr_functie)(...)*' sunt simbolice. Ele reprezintă, respectiv, adresa unei funcții (numele pointerului la funcție) şi apelul indirect al unei funcții, prin intermediul pointerului către acea funcție. Vedeți *Anexa B* - "*Noțiuni de C necesare desfăşurării lucrărilor de laborator*" pentru lămuriri.

// Valorile coeficienților *A[n][i]*

```
n=0
       {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
n=1
       {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
n=2
       {1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
n=3
       {5.0/9.0,8.0/9.0,5.0/9.0,0,0,0,0,0,0,0,0,0,0,0,0,0},

n=4
        {0.347854854137454.        0.652145145862564
        0.652145145862564,         0.347854854137454
        0                          0
        0                          0
        0                          0
        0                          0
        0                          0
        0                          0}
 n=5
        {0.236926885056189,        0.478628670499366
        0.568888888888889,         0.478628670499366,
        0.236926885056189,         0,
        0,                         0,
        0,                         0,
        0,                         0,
        0,                         0,
        0,                         0}
n=6
        { 0.17132449237917,        0.360761573048139,
        0.467913934572691,         0.467913934572691,
        0.360761573048139,         0.17132449237917,
        0,                         0,
        0,                         0,
        0,                         0,
        0.                         0,
        0,                         0}
n=7
        {0.12948496616887,         0.279705391489277,
        0.381830050505119,         0.417959183673469,
        0.381830050505119,         0.279705391489277,
        0.12948496616887,          0,
        0,                         0,
        0,                         0,
        0,                         0,
        0,                         0}
```

```
n=8
        {0.101228536290376,        0.222381034453374,
        0.313706645877887,         0.362684783378362,
        0.362684783378362,         0.313706645877887,
        0.222381034453374,         0.101228536290376,
        0,                         0,
        0,                         0,
        0,                         0,
        0,                         0},

n=9
        { 0.08127438361574,        0.180648160694857
        0.260610696402935          0.312347077040003,
        0.33023935500126           0.312347077040003,
        0.260610696402935,         0.180648160694857,
        0.08127438361574,          0,
        0,                         0,
        0,                         0,
        0,                         0},
n=10
        {0.066671344308688,        0.14945134915058
        0.219086362515982,         0.269266719309996,
        0.295524224714753,         0.295524224714753,
        0.269266719309996,         0.219086362515982,
        0.14945134915058,          0.066671344308688,
        0,                         0,
        0,                         0,
        0,                         0},
n=11
        {0.055668567116,           0.125580369465,
        0.186290210928,            0.233193764592,
        0.26280454451,             0.272925086778,
        0.26280454451,             0.233193764592,
        0.186290210928,            0.125580369465,
        0.055668567116,            0,
        0,                         0,
        0,                         0},
n=12
        {0.047175336387,           0.106939325995,
        0.160078328542,            0.203167426723,
        0.233492536538,            0.249147045813,
        0.249147045813,            0.233492536538,
        0.203167426723,            0.160078328542,
        0.106939325995,            0.047175336387,
        0,                         0,
        0,                         0},
n=13
        {0.040484004765,           0.092121499838,
        0.13887351022,             0.178145980762,
        0.207816047537,            0.226283180263
        0.232551553231,            0.226283180263,
        0.207816047537             0.178145980762,
        0.13887351022,             0.092121499838
        0.040484004765,            0,
        0,                         0},
        0,
```

```
n=14
        {0.035119460332,          0.08015808716,
        0.121518570688,           0.157203167158,
        0.185538397478,           0.205198463721,
        0.215263853463,           0.215263853463,
        0.205198463721,           0.185538397478,
        0.157203167158,           0.121518570688,
        0.08015808716,            0.035119460332,
        0,                        0},
n=15
        {0.030753241996,          0.070366047488,
        0.107159220467,           0.139570677926,
        0.166269205817,           0.186161000016,
        0.198431485327,           0.202578241926,
        0.198431485327,           0.186161000016,
        0.166269205817,           0.139570677926,
        0.107159220467,           0.070366047488,
        0.030753241996,           0},
n=16
        {0.027152459412,          0.062253523939,
        0.095158511682,           0.124628971256,
        0.149595988817,           0.169156519395,
        0.182603415045,           0.189450610455,
        0.189450610455,           0.182603415045,
        0.169156519395,           0.149595988817,
        0.124628971256,           0.095158511682,
        0.062253523939,           0.027152459412};
```

// Soluțiile polinoamelor lui *Legendre*, U[n][i]

```
n=0
       {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
n=1
       {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
n=2
        {0.5773502692,             -0.5773502692,
        0,                         0,
        0,                         0,
        0,                         0,
        0,                         0,
        0,                         0,
        0,                         0,
        0,                         0},
n=3
        {0.7745966692,             0,
        -0.7745966692              0,
        0,                         0,
        0,                         0,
        0,                         0,
        0,                         0,
        0,                         0,
        0,                         0},
n=4
        {0.861136311594053,        0.339981043584856,
        -0.339981043584856,        -0.861136311594053,
        0,                         0,
        0,                         0,
        0,                         0,
```

```
        0,                        0,
        0,                        0,
        0,                        0}
n=5
        {0.906179845938664,       0.538469310105683,
        0,                        -0.538469310105683,
        -0.906179845938664,       0,
        0,                        0,
        0,                        0,
        0,                        0,
        0,                        0,
        0,                        0},
n=6
        {0.932469514203152,       0.661209386466265,
        0.238619186083197,        -0.238619186083197,
        -0.661209386466265,       -0.932469514203152,
        0,                        0,
        0,                        0,
        0,                        0,
        0,                        0,
        0,                        0},
n=7
        {0.949107912342759,       0.741531185599394,
        0.405845151377397,        0,
        -.405845151377397,        -0.741531185599394,
        -0.949107912342759,       0,
        0,                        0,
        0,                        0,
        0,                        0,
        0,                        0},
n=8
        {0.96269856497336,        0.796666477413627,
        0.525532409916329,        0.18343464249565,
        -0.18343464249565,        -0.525532409916329,
        -0.796666477413627,       -0.96269856497336,
        0,                        0,
        0,                        0,
        0,                        0,
        0,                        0},
n=9
        {0.968160239507626,       0.836031107326636,
        0.613371432700591,         0.324253423403809,
        0,                        -0.324253423403809,
        -0.613371432700591,       -0.836031107326636,
        -0.968160239507626,       0,
        0,                        0,
        0,                        0,
        0,                        0},
n=10
        {0.973906528517172,       0.865063366688985,
        0.679409568299024,        0.433395394129247,
        0.148874338981631,        -0.148874338981631,
        -0.433395394129247,       -0.679409568299024,
        -0.865063366688985,       -0.973906528517172,
        0,                        0,
        0,                        0,
        0,                        0},
```

```
n=11
        {0.978228658146,            0.887062599768,
        0.730152005574,             0.519096129207,
        0.269543155952,             0,
        -0.269543155952,            -0.519096129207,
        -0.730152005574,            -0.887062599768,
        -0.978228658146,            0,
        0,                          0,
        0,                          0},

n=12
        {0.981560634247,            0.90411725637,
        0.769902671494,             0.587317954287,
        0.367831498998,             0.125233408511,
        -0.125233408511,            -0.367831498998,
        -0.587317954287,            -0.769902671494,
        -0.90411725637,             -0.981560634247,
        0,                          0,
        0,                          0},
n=13
        {0.984183054719,            0.917598399223,
        0.801578090733,             0.64234933944,
        0.448492751036,             0.230458315955,
        0,                          -0.230458315955,
        -0.448492751036,            -0.64234933944,
        -0.801578090733,            -0.917598399223,
        -0.984183054719,            0,
        0,                          0},
n=14
        {0.986283808697,            0.928434883664,
        0.82720131507,              0.687292904812,
        0.515248636358,             0.319112368928,
        0.108054948707,             -0.108054948707,
        -0.319112368928,            -0.515248636358,
        -0.687292904812,            -0.82720131507,
        -0.928434883664,            -0.986283808697,
        0,                          0},
n=15
        { 0.98799251802,            0.937273392401,
         0.84820658341,             0.72441773136,
         0.570972172609,            0.394151347078,
         0.201194093997,            0,
        -0.201194093997,            -0.394151347078,
        -0.570972172609,            -0.72441773136,
        -0.84820658341,             -0.937273392401,
        -0.98799251802,             0},
n=16
        {0.989400934992,            0.944575023075,
        0.865631202388,             0.755404408355,
        0.617876244403,             0.458016777657,
        0.281603550779,             0.095012509838,
        -0.095012509838,            -0.281603550779,
        -0.458016777657,            -0.617876244403,
        -0.755404408355,            -0.865631202388,
        -0.944575023075,            -0.989400934992}
                                    };
```

2.3. CALCULUL NUMERIC AL INTEGRALELOR DUBLE

Pentru simplitate vom considera domeniul de integrare al funcției de două variabile un dreptunghi

$$\iint_D f(x,y)\,dxdy = \int_a^b \int_c^d f(x,y)\,dxdy \tag{6.15}$$

reprezintă integrala dublă dintr-o funcție de două variabile. Pentru calculul valorii acestei integrale vom utiliza *formula de cubatură a trapezului* şi *formula de cubatură a lui Simpson*, prezentate în continuare.

2.3.1. FORMULA DE CUBATURĂ A TRAPEZULUI

Intervalele de integrat [a,b] şi [c,d] se împart în subintervale de lungimi egale

$$h=\frac{b-a}{n} \text{ şi, respectiv } k=\frac{d-c}{m} \tag{6.16}$$

şi se consideră dreptunghiul cu vârfurile $\left[x_i, y_i\right]$, $\left[x_{i+1}, y_i\right]$, $\left[x_{i+1}, y_{i+1}\right]$, $\left[x_i, y_{i+1}\right]$

Pentru dreptunghiul dat se calculează integrala $I_{i,j}$ aplicând *formula trapezului*.

Integrala pe întreg domeniul cu două dimensiuni din plan este:

$$I=\sum_{i=0}^{n-1}\sum_{j=0}^{m-1} I_{ij}=\frac{kh}{4}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}\left[f(x_i,y_j)+f(x_i,y_{j+1})+f(x_{i+1},y_j)+f(x_{i+1},y_{j+1})\right] \tag{6.17}$$

expresie cunoscută sub numele de *formula de* ***cubatură*** *a trapezului*.

2.3.1.1. Algoritmul 5.6. Metoda cubaturii trapezului

```
real I_Cubatura_Trapez
(
   Adr_functie, // Adresa unei funcții reale de două
                // variabile reale (pointer la funcție(x,y))
   real a,      // limita stângă a intervalului de integrare
                // pe axa Ox
   real b       // limita dreaptă a intervalului de integrare
                // pe axa Ox
   real c,      // limita stângă a intervalului de integrare
                // pe axa Oy
   real d,      // limita dreaptă a intervalului de integrare
                // pe axa Oy
```

```
   întreg n,   // numărul de subintervale pe axa Ox
   întreg m    // numărul de subintervale pe axa Oy
)
{
// declararea şi definirea variabilelor locale
întreg i, j;        // indici pentru buclele 'for()'
real h;  // valoarea lungimii unui subinterval diviziunea n
real k;  // valoarea lungimii unui subinterval diviziunea m
real sum;      // valoarea integralei

// instrucţiunile de calcul ale metodei
```

$h=\frac{b-a}{n}$; $k=\frac{d-c}{m}$; // calculul valorii fiecărei diviziuni

```
sum = 0;
pentru i= 0 ÷  n-1
  pentru j= 0 ÷  m-1
   sum = sum + ((h*k)/4) *
   (
      (*Adr_functie)(a+i*h,      c+j*k) +
      (*Adr_functie)(a+i*h,      c+(j+1)*k) +
      (*Adr_functie)(a+(i+1)h,   c+j*k) +
      (*Adr_functie)(a+(i+1)h,   c+(j+1)*k)
   );

returnează   sum;
}
```

Obs.:

Notaţiile '*Adr_functie*' şi '*(*Adr_functie)(...)*' sunt simbolice. Ele reprezintă, respectiv, adresa unei funcţii (numele pointerului la funcţie) şi apelul indirect al unei funcţii, prin intermediul pointerului către acea funcţie. Vedeţi *Anexa B* - "*Noţiuni de C necesare desfăşurării lucrărilor de laborator*" pentru lămuriri.

2.3.2. FORMULA LUI SIMPSON DE CUBATURĂ

Pentru acelaşi dreptunghi $[a,b,c,d]$ vom aplica formula lui *Simpson de integrare*.

Vom considera dreptunghiul de integrare cu vârfurile

$$[x_i, y_j],\ [x_{i+1}, y_j],\ [x_{i+1}, y_{j+1}],\ [x_i, y_{j+1}]$$

şi cu punctul central $[x_i, y_j]$.

Integrala pe dreptunghiul $[a,b,c,d]$ este dată de *formula de cubatură a lui Simpson*:

$$I=\sum_{i=0}^{n-1}\sum_{j=0}^{m-1} I_{ij}=\frac{kh}{9}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}\left\{f\left(x_{i-1},y_{j-1}\right)+f\left(x_{i+1},y_{j-1}\right)+f\left(x_{i-1},y_{j+1}\right)+f\left(x_{i+1},y_{j+1}\right)+\right.$$
$$\left.+4\left[f\left(x_i,y_{j+1}\right)+f\left(x_i,y_{j-1}\right)+f\left(x_{i-1},y_j\right)+f\left(x_{i+1},y_j\right)\right]+16 f\left(x_i,y_j\right)\right\} \tag{6.18}$$

2.3.2.1. **Algoritmul 5.7. Metoda lui Simpson de cubatură**

```
real I_Cubatura_Simpson
( Adr_functie, // Adresa unei funcții reale de două
               // variabile reale (pointer la funcție(x,y))
   real a,     // limita stângă a intervalului de integrare
               // pe axa Ox
   real b,     // limita dreaptă a intervalului de integrare
               // pe axa Ox
   real c,     // limita stângă a intervalului de integrare
               // pe axa Oy
   real d,     // limita dreaptă a intervalului de integrare
               // pe axa Oy
   întreg n,   // numărul de subintervale pe axa Ox
   întreg m    // numărul de subintervale pe axa Oy
)
{
// declararea şi definirea variabilelor locale
real h; // valoarea lungimii unui subinterval divizat in n
real k; // valoarea lungimii unui subinterval divizat in m
real sum;      // valoarea integralei
întreg i, j;   // indici pentru buclele 'for()'
// instrucțiunile de calcul ale funcției
h=(b-a)/n ; k=(d-c)/m ;   //calculul valorilor fiecărei diviziuni
sum = 0;
i = 1;
   execută
   {
      j = 1;
      execută
      {
          sum = sum + ((h*k)/9)*
           (
                  (*Adr_functie)(sx+(i-1)*h, sy+(j-1)*k) +
             +    (*Adr_functie)(sx+(i+1)*h, sy+(j-1)*k) +
             +    (*Adr_functie)(sx+(i-1)*h, sy+(j+1)*k) +
             +    (*Adr_functie)(sx+(i+1)*h, sy+(j+1)*k) +
             +  4*(*Adr_functie)(sx+i*h,     sy+(j+1)*k) +
             +  4*(*Adr_functie)(sx+i*h,     sy+(j-1)*k) +
             +  4*(*Adr_functie)(sx+(i-1)*h, sy+j*k) +
             +  4*(*Adr_functie)(sx+(i+1)*h, sy+j*k) +
             + 16*(*Adr_functie)(sx+i*h,     sy+j*k)
            ); /* vedeți formula (6.18) */
```

```
            j = j+2;     // sau: j+=2;
         } cât timp(j<= m-1);
         i = i+2;        // sau:  i+=2;
      } cât timp(i<= n-1);
      returnează sum;
   }
```

Obs.:

Notațiile '*Adr_functie*' şi '*(*Adr_functie)(...)*' sunt simbolice. Ele reprezintă, respectiv, adresa unei funcții (numele pointerului la funcție) şi apelul indirect al unei funcții, prin intermediul pointerului către acea funcție. Vedeți *Anexa B* - "*Noțiuni de C necesare desfăşurării lucrărilor de laborator*" pentru lămuriri.

3. DESFĂŞURAREA LUCRĂRII

3.1. ÎNTREBĂRI

1. Enumerați în ordinea crescătoare a preciziei metodele de integrare.
2. Cum se justifică eroarea de calcul a metodei SIMPSON, în situația în care numărul de diviziuni este impar?
3. Ce ordonate au valoarea dublată în calculul sumei pentru metoda trapezului?
4. Care este gradul maxim al unei funcții în exprimare polinomială pentru care metoda RICHARDSON oferă erori minime de calcul?
5. Cine impune schimbarea limitelor de integrare în metoda cuadraturii Gauss?
6. Se pot compara metodele de integrare din punctul de vedere al numărului de diviziuni?

3.2. PROBLEME LABORATOR

1. Să se implementeze toți algoritmii prezentați sub formă de pseudo-cod, în *limbaj* ANSI C.
2. Se dă funcția $f(x)=\frac{x^3}{1+\cos(1+x)}.e^{x^2}.(1+\sin x^2)$ pentru care se cere integrala de la *0* la *3*.
3. Se consideră funcția de două variabile:
$$f(x,y)=\frac{x^2+y^2}{1+2.x.y}.\exp(1+x).\sin(x+y+2)$$
Se cere valoarea integralei din funcția dată pe domeniul:
$x \in [0,2]$; $y \in [0,2]$.

3.3. TEME DE CASĂ

1. În cazul determinării valorii integralei:

$$I = \int_{-2}^{3.5} x^2 \cdot \exp\left(x^2\right) dx$$

prin metoda *Simpson*, în ce situație rezultatul obținut este mai precis: când `N=202` sau când `N=2021`?

2. Să se calculeze valoarea medie a tensiunii in cazul unui redresor comandat prin tiristor, ştiind unghiul de comandă $\alpha=70^0$ iar amplitudinea maximă a tensiunii de intrare, $U_m=5V$. Folosiți o metoda numerica luând `N=10`. Calcul manual.

Indicație:

$$U_0 = \frac{1}{T}\int_0^T \text{¿}\, f(t)\,dt; f(t) = U_m \sin(\omega \cdot t)$$

$$\text{¿}$$

BIBLIOGRAFIE

1. I. Rusu – "*Metode numerice în electronică. Aplicații în limbaj C*", Editura Tehnică, Bucureşti, 1997.
2. W.S. Dorn, D.D. McCracken - "*Metode numerice cu programe Fortran IV*", Editura Tehnică, Bucureşti, 1976.