GRADIMIR V. MILOVANOVIĆ

ĐORĐE R. ĐORĐEVIĆ

PROGRAMIRANJE NUMERICKIH METODA NA FORTRAN JEZIKU

PROGRAMIRANJE
NUMERIČKIH METODA
NA FORTRAN JEZIKU

UNIVERZITET U NIŠU

Dr Gradimir V. Milovanović
docent Elektronskog fakulteta u Nišu

Đorđe R. Đorđević
asistent Građevinskog fakulteta u Nišu

PROGRAMIRANJE NUMERIČKIH METODA NA FORTRAN JEZIKU

– DRUGO IZDANJE –

Niš, 1981.

PROGRAMIRANJE NUMERIČKIH METODA
NA FORTRAN JEZIKU

Autori:
Dr Gradimir V. Milovanović
Đorđe R. Đorđević

Recenzenti:
Dr Petar Madić, redovni profesor Pedagoško-tehničkog fakulteta u Zrenjaninu
Dr Milivoje Stanković, vanredni profesor Tehničkog fakulteta u Novom Sadu

Glavni i odgovorni urednik
Momčilo Zdravković

Urednik Olgica Dedić, tehnički urednik Centra Simeon Zinovijev, tehnički urednik Velimir Popović, crteže izradio Veljko Nikolić, dipl. inž., korice Vladimir Krstić, meter Mirjana Gičić.

Izdaje i priprema za štampu: Institut za dokumentaciju zaštite na radu „Edvard Kardelj", Centar za informativno izdavačku delatnost, Birotehnika, Niš, Višegradska 33, tel.: (018) 334-244.

Štampa: Štamparija „Bakar" - Bor

Tiraž: 2000

PREDGOVOR DRUGOM IZDANJU

Prvo izdanje „Programiranja numeričkih metoda na FORTRAN jeziku" objavljeno je 1979. godine i za kratko vreme rasprodato tako da su autori bili prinuđeni da bez većih izmena daju u štampu drugo izdanje ove knjige.

Knjiga je podeljena u pet glava, od kojih je prva uvodna. Druga glava je posvećena optimizacijama FORTRAN programa u smislu tačnosti izračunavanja, racionalnog korišćenja centralne memorije računara i brzine izvršenja programa. U trećoj glavi dat je veći broj kompletno rešenih problema, koji imaju za cilj savlađivanje osnovnih principa programiranja, kao i postepeno uvođenje čitaoca u složenije numeričke procese. U četvrtoj glavi su obrađeni numerički metodi u linearnoj algebri, dok je peta glava posvećena metodima za integraciju funkcija, rešavanju integralnih jednačina, kao i numeričkim metodima za rešavanje običnih i parcijalnih diferencijalnih jednačina.

Knjiga je prvenstveno namenjena studentima tehničkih fakulteta kao udžbenik, ali ona može biti od koristi i svima onima koji u svojoj praksi koriste numeričke metode.

Svi programi koji su dati u knjizi testirani su na računaru PDP–11/40 Građevinskog fakulteta u Nišu.

1 avgust 1981.

Gradimir V. Milovanović
Đorđe R. Đorđević

IZ PREDGOVORA PRVOM IZDANJU

Iz oblasti numeričke matematike ova knjiga je prva na našem jeziku po svojoj koncepciji i sadržaju. Naime, na našem jeziku postoji veći broj knjiga koje se bave programskim jezicima (uglavnom, FORTRAN i COBOL) i relativno mali broj knjiga iz oblasti numeričke analize. Zadatak ove knjige je da uputi čitaoce u programsku realizaciju numeričkih metoda na FORTRAN jeziku.

Rukopis za ovu knjigu proistekao je iz predavanja koje je prvopotpisani autor držao studentima poslediplomskih studija na Građevinskom fakultetu u Nišu u toku školske 1978/79. godine, kao i iz višegodišnje nastave na Elektronskom fakultetu u Nišu iz predmeta Numerička analiza I i Numerička analiza II.

7. aprila 1979.

Gradimir V. Milovanović
Đorđe R. Đorđević

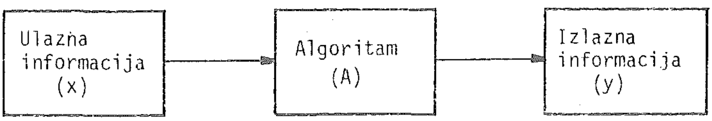
SADRŽAJ

1. UVOD U PROGRAMIRANJE

1.1. PREDSTAVLJANJE ALGORITAMA BLOK ŠEMAMA

Savremena nauka i tehnika postavljaju niz matematičkih problema koji se klasičnim matematičkim metodima ne mogu uvek uspešno rešiti ili bi njihovo rešavanje bilo suviše glomazno, s obzirom da se najčešće zahteva numerički rezultat.

U opštem slučaju, problem koji treba rešavati zvaćemo ulaznom informacijom. Postupak transformacije ulazne informacije (x) u izlaznu informaciju (y) zvaćemo algoritmom (A). Navedena transformacija se može predstaviti blok dijagramom



ili simbolički $x \xrightarrow{A} y$.

Svaki algoritam treba da poseduje sledeće osobine: diskretnost, determinisanost, rezultativnost i masovnost.

Diskretnost je takva osobina algoritma da svakom algoritamskom koraku odgovara diskretni vremenski interval na vremenskoj osi.

Determinisanost je osobina koja obezbeđuje jednoznačnost međurezultata posle svakog algoritamskog koraka u odnosu na ulaznu informaciju.

Osobina rezultativnosti algoritma obezbeđuje da se posle konačnog broja algoritamskih koraka dobije traženi rezultat (izlazna informacija).

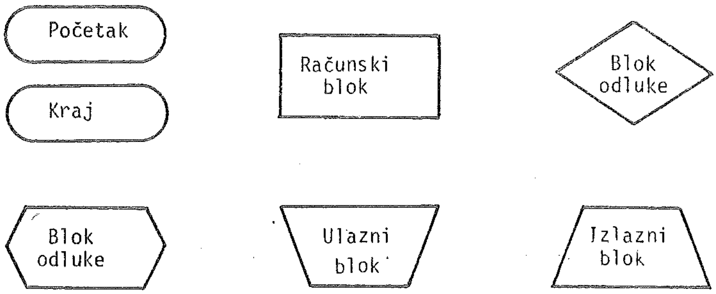
Masovnost algoritma je osobina algoritma koja obezbeđuje rešavanje šire klase problema.

Pri rešavanju nekog problema potrebno je izabrati pogodan algoritam koji najbrže dovodi do željenog rezultata. Ovo je naročito važno kod primene savremenih elektronskih računskih mašina, s obzirom na cenu mašinskog vremena. Izbor najracionalnijeg algoritma u prethodnom smislu, predstavlja vrlo složen problem, koji teorijski u opštem slučaju još uvek nije rešen.

Razradom i realizacijom algoritama i analizom greške u izlaznoj informaciji bavi se posebna oblast matematike, tzv. numerička matematika.

Centralni deo numeričke matematike čine numerički metodi. Oni moraju biti takvi da su pogodni sa stanovišta primene savremenih elektronskih računskih mašina. Problemima praktične realizacije algoritama bavi se oblast teorija programiranja koja se u poslednje vreme uspešno razvija. Jedan od njenih osnovnih zadataka je priprema i sastavljanje programa za računsku mašinu prema izabranom algoritmu.

Da bi se olakšalo sastavljanje programa najčešće se algoritmi predstavljaju blok šemama, a ponekad i tekstualnim opisom. U daljem tekstu ukazaćemo na grafičko predstavljanje algoritama pomoću blok šema, koristeći sledeće blokove:



Dodeljivanje vrednosti a promenljivoj x označavaćemo sa $x := a$. Blokovi za ulaz i izlaz informacija se često izostavljaju.

Primer 1.1. Blok šema algoritma za izračunavanje sume

$$S = \sum_{i=1}^{n} a_i b_i$$

data je na sl. 1.1.

Primer 1.2. Kod određivanja vrednosti

$$u = \min (x, \max (y,z))$$

treba, najpre, odrediti $v = \max (y,z)$. Odgovarajuća blok šema algoritma data je na sl. 1.2, pri čemu su blokovi za ulaz veličina x, y, z i izlaz veličine u izostavljeni.

Primer 1.3. Posmatrajmo konvergentni iterativni proces

$$z_0 = a, \qquad z_{n+1} = f(z_n) \qquad (n = 0,1, \ldots).$$

Ako se kao kriterijum za prekidanje ovog procesa izabere uslov $|z_{n+1} - z_n| \leqslant \epsilon$, gde je ϵ zadata tačnost, blok šema algoritma za određivanje granične vrednosti niza $\{z_n\}$ (sa tačnošću ϵ) izgleda kao na sl. 1.3. Primetimo da u ovom algoritmu koristimo samo dve sukcesivne vrednosti niza $\{z_n\}$. Naime, na osnovu z_0 izračunavamo z_1, a zatim z_1 proglašavamo za z_0 i ponavljamo proces do ispunjenja usvojenog kriterijuma za prekid procesa.

Na primer, jedan iterativni proces za nalaženje m–tog korena iz broja A (> 0) ima oblik

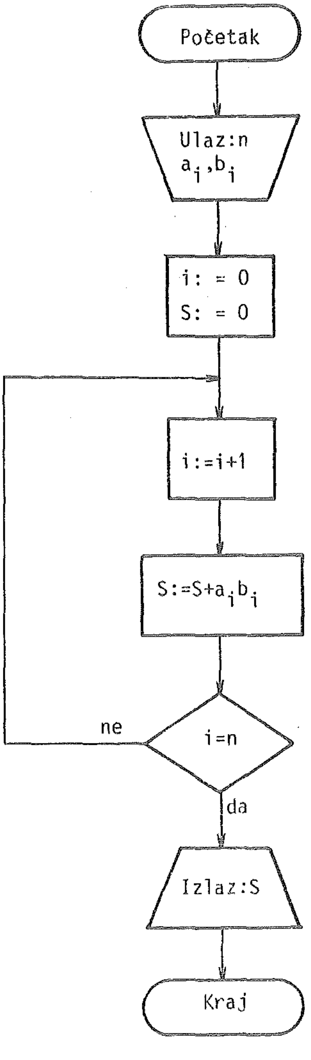
$$z_{n+1} = z_n + \frac{1}{m} \left(\frac{A}{z_n^{m-1}} - z_n \right) \qquad (n = 0,1, \ldots)$$

Primer 1.4. Za izračunavanje vrednosti polinoma

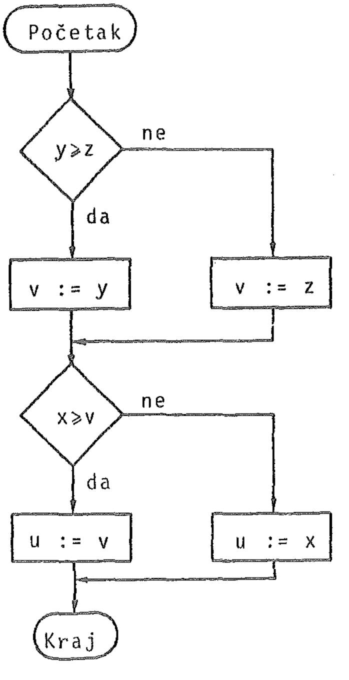
$$F(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$$

koristi se Hornerova šema, po kojoj se polinom F predstavlja u obliku

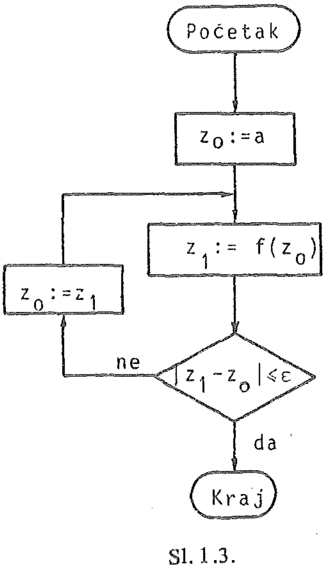
$$F(x) = a_o + x(a_1 + x(a_2 + \ldots + x(a_{n-1} + xa_n) \ldots)).$$



Sl. 1.1

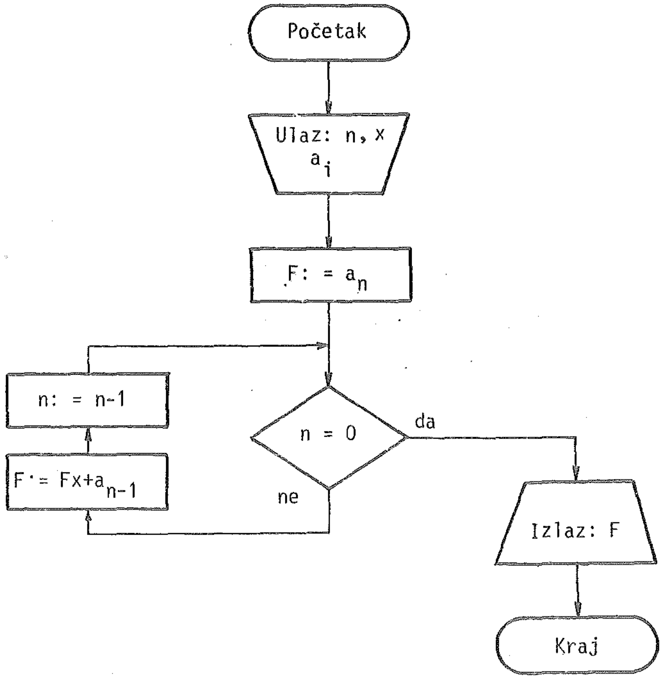


Sl. 1.2



Sl. 1.3.

Na ovaj način se broj množenja od $2n-1$, koliko je potrebno kod direktnog izračunavanja, svodi na samo n množenja. Blok šema algoritma, u ovom slučaju, izgleda kao na sl. 1.4.



Sl. 1.4.

Primer 1.5. Blok šema algoritma za izračunavanje vrednosti funkcije f, date pomoću verižnog razlomka

$$f(x)=a_1+\cfrac{x}{a_2+\cfrac{x}{a_3+\cfrac{x}{a_4+\ddots+\cfrac{x}{a_{n-1}+\cfrac{x}{a_n}}}}}$$

data je na sl. 1.5.

Primer 1.6. Posmatrajmo konvergentni red $f(x)=\sum_{n=0}^{+\infty} u_n(x)$. Opšti član i parcijalna suma ovog reda mogu se predstaviti u obliku

$$u_n(x)=\Phi(n,x)\,u_{n-1}(x) \qquad \text{i} \qquad s_n=s_{n-1}+u_n(x),$$

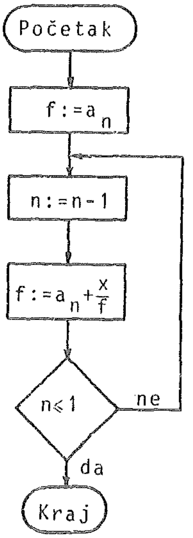
gde je $s_0=u_0(x)$.

U konkretnom slučaju bismo imali:

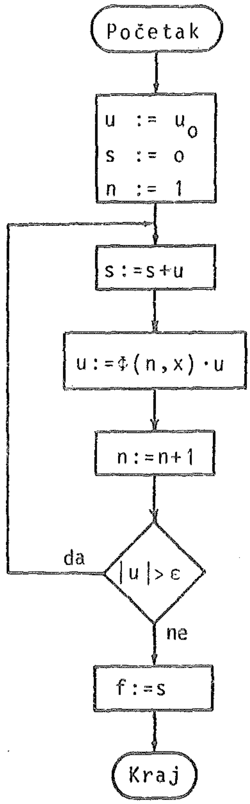
a) Za $f(x) = e^x$: $\quad u_0 = 1, \ \Phi(n, x) = \dfrac{x}{n}$;

b) Za $f(x) = shx$: $\quad u_0 = x, \ \Phi(n, x) = \dfrac{x^2}{(2n+1)\,2n}$.

Ako je algoritam za približno određivanje sume $f(x)$ takav da se sumiranje prekida kada je opšti član reda po apsolutnoj vrednosti manji od unapred zadatog pozitivnog broja ϵ, njegova blok šema izgleda kao na sl. 1.6.



Sl. 1.5

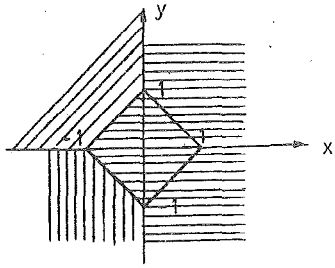


Sl. 1.6

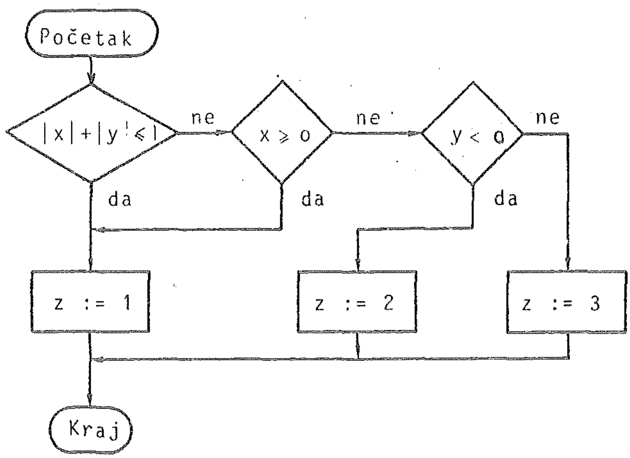
Primer 1.7. Neka je

$$z(x, y) = \begin{cases} 1 & (|x| + |y| \leqslant 1 \ \vee x \geqslant 0), \\ 2 & (|x| + |y| > 1 \wedge x < 0 \wedge y < 0), \\ 3 & (|x| + |y| > 1 \wedge x < 0 \wedge y \geqslant 0). \end{cases}$$

Na osnovu sl.1.7., na kojoj je ravan xOy razdeljena na tri oblasti, obrazovan je algoritam čija je blok šema data na sl. 1.8.



Sl. 1.7



Sl. 1.8.

Primer 1.8. Blok šema algoritma za izračunavanje vrednosti funkcije

$$F(y)=\begin{cases} y^3-1 & (|y|\leqslant 1),\\ 2y & (1<|y|\leqslant 2),\\ y/8-1 & (|y|>2)\end{cases}$$

za n vrednosti promenljive x, pri čemu je $y=x^2+x-1$, data je na sl. 1.9.

Primer 1.9. Blok šema algoritma za određivanje sume proizvoda

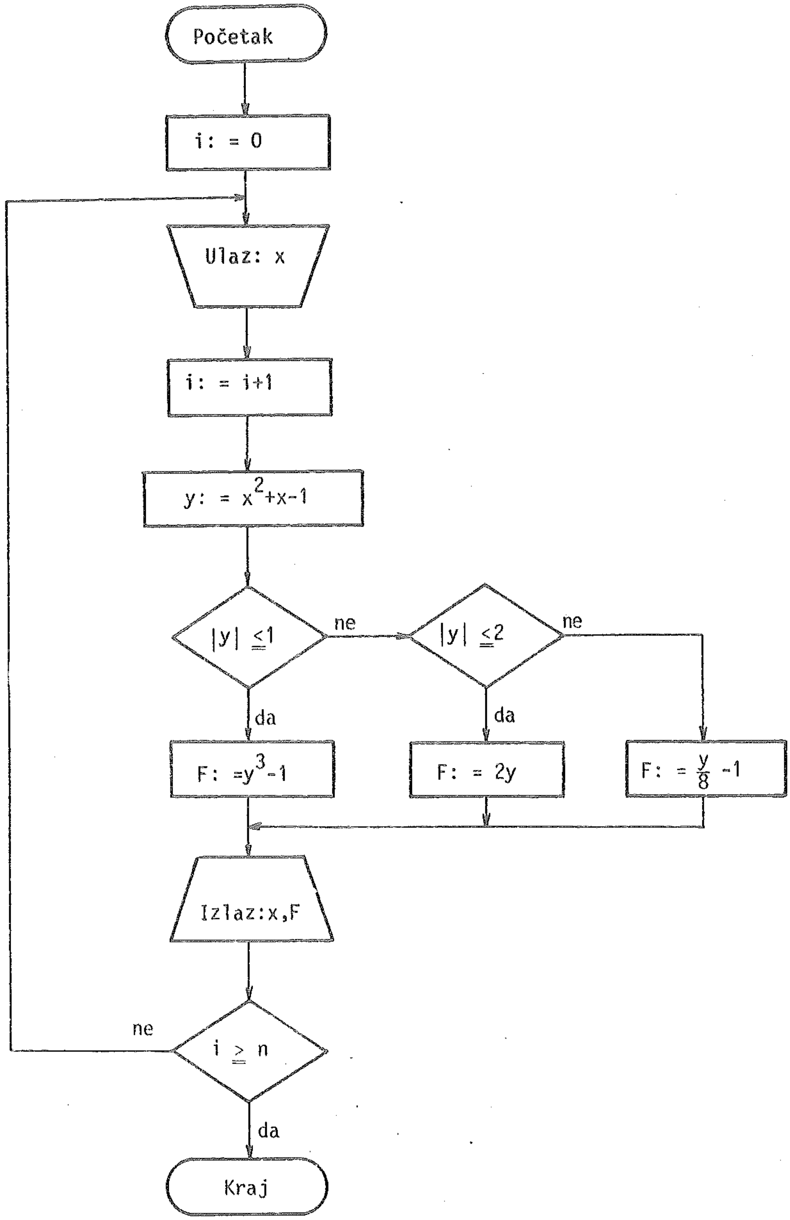
$$s_{mn}=\sum_{i=1}^{m}\prod_{\substack{j=1\\ j\neq i}}^{n}(x_i-x_j)$$

predstavljena je na sl. 1.10.

Primer 1.10. Blok šema algoritma za rešavanje kvadratne jednačine

$$ax^2+bx+c=0 \qquad (a\neq 0)$$

prikazana je na sl. 1.11.



Sl. 1.9.

Primer 1.11. Konstruišimo jedan algoritam za određivanje svih prostih brojeva do M (M prirodan neparan broj) i njihovo prebrojavanje.

Iz razmatranja, najpre, isključimo sve parne brojeve. Neka je traženi skup prostih brojeva $A = \{a_1, \ldots, a_n\}$. Pri ispitivanju da li neparan broj i pripada skupu A, dovoljno je ispitati deljivost ovog broja samo brojevima iz podskupa $\{a_1, \ldots,$

i \ j	1	2	....	m
1	1 a_{11}	n+1 a_{12}		(m-1)n+1 a_{1m}
2	2 a_{21}	n+2 a_{22}		(m-1)n+2 a_{2m}
⋮				
n	n a_{n1}	2n a_{n2}		nm a_{nm}

Tabela 2.1

Neka su

$$A = \begin{bmatrix} 25.123 & 18.444 & -11.111 & 6.102 \\ -1.532 & 0.456 & 3.470 & -2.412 \\ 4.210 & 0.345 & 0. & 3.153 \end{bmatrix}$$

i

$$B = \begin{bmatrix} -1.456 & 13.555 & 21.526 & -24.654 \\ 1.234 & 7.254 & 32.744 & 0.542 \\ -7.654 & 21.687 & 22.687 & 15.198 \end{bmatrix}$$

Slučaj 1. Neka su elementi matrice A i matrice B dati na karticama, u formatu 8F10.3, po vrstama

I kartica: $a_{11}, a_{12}, a_{13}, a_{14}, a_{21}, a_{22}, a_{23}, a_{24}$,

II kartica: $a_{31}, a_{32}, a_{33}, a_{34}, b_{11}, b_{12}, b_{13}, b_{14}$,

III kartica: $b_{21}, b_{22}, b_{23}, b_{24}, b_{31}, b_{32}, b_{33}, b_{34}$.

Programom

```
      DIMENSION A(3,4),B(3,4),C(3,4)
      READ(8,10) ((A(I,J),J=1,4),I=1,3),((B(I,J),J=1,4),I=1,3)
   10 FORMAT(8F10.3)
      DO 15 I=1,3
      DO 15 J=1,4
   15 C(I,J)=A(I,J)+B(I,J)
      WRITE(5,25) ((C(I,J),J=1,4),I=1,3)
   25 FORMAT(1H1,5X,'MATRICA C'//(4F12.3))
      CALL EXIT
      END
```

realizuje se određivanje elemenata matrice $C = A + B$. Izlazna lista ima oblik

```
     MATRICA C

     23.667      31.999      10.415     -18.552
     -0.298       7.710      36.214      -1.870
     -3.444      22.032      22.687      18.351
```

Elementi matrice A (takođe i elementi matrica B i C) su memorisani prema tabeli 2.2. Primetimo da se elementi prve vrste matrice A memorišu u registrima sa relativnim adresama 1, 4, 7, 10, elementi druge vrste u registrima sa relativnim adresama 2, 5, 8, 11, i najzad, elementi treće vrste u registrima sa adresama 3, 6, 9, 12.

i \ j	1	2	3	4
1	$\boxed{1}$ a_{11}	$\boxed{4}$ a_{12}	$\boxed{7}$ a_{13}	$\boxed{10}$ a_{14}
2	$\boxed{2}$ a_{21}	$\boxed{5}$ a_{22}	$\boxed{8}$ a_{23}	$\boxed{11}$ a_{24}
3	$\boxed{3}$ a_{31}	$\boxed{6}$ a_{32}	$\boxed{9}$ a_{33}	$\boxed{12}$ a_{34}

Tabela 2.2

i \ j	1	2	3	4
1	$\boxed{1}$ a_{11}	$\boxed{4}$ a_{14}	$\boxed{7}$ a_{23}	$\boxed{10}$ a_{32}
2	$\boxed{2}$ a_{12}	$\boxed{5}$ a_{21}	$\boxed{8}$ a_{24}	$\boxed{11}$ a_{33}
3	$\boxed{3}$ a_{13}	$\boxed{6}$ a_{22}	$\boxed{9}$ a_{31}	$\boxed{12}$ a_{34}

Tabela 2.3

Prethodni program smo mogli realizovati i korišćenjem jednodimenzionalnih nizova, sa istim ulaznim podacima:

```
      DIMENSION A(12),B(12),C(12)
      READ(8,10) A,B
   10 FORMAT(8F10.3)
      DO 15 I=1,12
   15 C(I)=A(I)+B(I)
      WRITE(5,25) C
   25 FORMAT(1H1,5X,'MATRICA C'//(4F12.3))
      CALL EXIT
      END
```

Način memorisanja elemenata matrice A dat je u tabeli 2.3. Primetimo da se, u ovom slučaju, elementi prve vrste matrice A memorišu u registrima sa relativnim adresama 1, 2, 3, 4.

Pretpostavimo sada da su elementi matrica A i B dati na karticama u sledećem redosledu (po kolonama):

I kartica: $a_{11}, a_{21}, a_{31}, a_{12}, a_{22}, a_{32}, a_{13}, a_{23}$,

II kartica: $a_{33}, a_{14}, a_{24}, a_{34}, b_{11}, b_{21}, b_{31}, b_{12}$,

III kartica: $b_{22}, b_{32}, b_{13}, b_{23}, b_{33}, b_{14}, b_{24}, b_{34}$.

Tada odgovarajući program ima oblik

```
      DIMENSION A(12),B(12),C(12)
      READ(8,10) A,B
   10 FORMAT(8F10.3)
      DO 15 I=1,12
   15 C(I)=A(I)+B(I)
      WRITE(5,20)
   20 FORMAT(1H1,5X,'MATRICA C'/)
      DO 30 I=1,3
```

```
30 WRITE(5,25) (C(J),J=I,12,3)
25 FORMAT(' ',4F12.3)
   CALL EXIT
   END
```

Registrovanje elemenata matrice A je kao u tabeli 2.2.

Slučaj 2. Neka matrice $A_1 = [a_{ij}]_{n \times m}$ i $B_2 = [b_{ij}]_{n \times m}$ imaju dimenzije manje od dimenzija definisanih u naredbi DIMENSION (3 x 4 ili 12). Na primer, neka je, u konkretnom slučaju, $n = 2$, $m = 3$ i neka su A_1 i B_1 glavne submatrice tipa 2 x 3 prethodno korišćenih matrica A i B respektivno.

Pretpostavimo da su elementi matrica A_1 i B_1 (u programu A i B) dati na karticama, u formatu 8F10.3, na sledeći način:

I kartica: $a_{11}, a_{12}, a_{13}, a_{21}, a_{22}, a_{23}, b_{11}, b_{12}$,

II kartica: $b_{13}, b_{21}, b_{22}, b_{23}$.

Program i izlazna lista, u ovom slučaju, su

```
   DIMENSION A(3,4),B(3,4),C(3,4)
   READ(8,5) N,M
 5 FORMAT(2I1)
   READ(8,10) ((A(I,J),J=1,M),I=1,N),((B(I,J),J=1,M),I=1,N)
10 FORMAT(8F10.3)
   DO 15 I=1,N
   DO 15 J=1,M
15 C(I,J)=A(I,J)+B(I,J)
    WRITE(5,20)
20 FORMAT(1H1,5X,'MATRICA C'/)
   DO 30 I=1,N
30 WRITE(5,25)(C(I,J),J=1,M)
25 FORMAT(' ',4F12.3)
   CALL EXIT
   END
```

```
MATRICA C

 23.667       31.999       10.415
 -0.298        7.710       36.214
```

Deo programa koji se odnosi na štampanje matrice C može se uprostiti korišćenjem naredbe FORMAT, koja je definisana promenljivim celobrojnim aritmetičkim izrazom. Tada prethodni program postaje

```
   DIMENSION A(3,4),B(3,4),C(3,4)
   READ(8,5) N,M
 5 FORMAT(2I1)
   READ(8,10) ((A(I,J),J=1,M),I=1,N),((B(I,J),J=1,M),I=1,N)
10 FORMAT(8F10.3)
   DO 15 I=1,N
   DO 15 J=1,M
15 C(I,J)=A(I,J)+B(I,J)
    WRITE(5,20)
20 FORMAT(1H1,5X,'MATRICA C'/)
   WRITE(5,25)((C(I,J),J=1,M),I=1,N)
25 FORMAT(' ',<M>F12.3)
   CALL EXIT
   END
```

Način memorisanja elemenata matrice A_1 dat je u tabeli 2.4.

i \ j	1	2	3	4
1	a_{11} [1]	a_{12} [4]	a_{13} [7]	[10]
2	a_{21} [2]	a_{22} [5]	a_{23} [8]	[11]
3	[3]	[6]	[9]	[12]

Tabela 2.4

i \ j	1	2	3	4
1	a_{11} [1]	a_{21} [4]	[7]	[10]
2	a_{12} [2]	a_{22} [5]	[8]	[11]
3	a_{13} [3]	a_{23} [6]	[9]	[12]

Tabela 2.5

Prethodna dva programa, korišćenjem jednodimenzionalnih nizova (sa istim ulaznim podacima), postaju

```
      DIMENSION A(12),B(12),C(12)
      READ(8,5) N,M
    5 FORMAT(2I1)
      NM=N*M
      READ(8,10)(A(I),I=1,NM),(B(I),I=1,NM)
   10 FORMAT(8F10.3)
      DO 15 I=1,NM
   15 C(I)=A(I)+B(I)
      WRITE(5,20)
   20 FORMAT(1H1,5X,'MATRICA C'/)
      DO 25 I=1,NM,M
      L=M-1+I
   25 WRITE(5,30) (C(J),J=I,L)
   30 FORMAT(' ',4F12.3)
      CALL EXIT
      END
```

```
      DIMENSION A(12),B(12),C(12)
      READ(8,5) N,M
    5 FORMAT(2I1)
      NM=N*M
      READ(8,10)(A(I),I=1,NM),(B(I),I=1,NM)
   10 FORMAT(8F10.3)
      DO 15 I=1,NM
   15 C(I)=A(I)+B(I)
      WRITE(5,20)
   20 FORMAT(1H1,5X,'MATRICA C'/)
      WRITE(5,30)(C(J),J=1,NM)
   30 FORMAT(' ',<M>F12.3)
      CALL EXIT
      END
```

U ovom slučaju elementi matrice A_1 su registrovani kao što je prikazano u tabeli 2.5.

Na kraju, pretpostavimo da su elementi matrica A_1 i B_1 dati na karticama, u formatu 8F10.3, na sledeći način (po kolonama):

I kartica: $a_{11}, a_{21}, a_{12}, a_{22}, a_{13}, a_{23}, b_{11}, b_{21}$,

II kartica: $b_{12}, b_{22}, b_{13}, b_{23}$.

U slučaju korišćenja jednodimenzionalnih nizova, imamo sledeći program:

```
      DIMENSION A(12),B(12),C(12)
      READ(8,5) N,M
    5 FORMAT(2I1)
      NM=N*M
      READ(8,10) (A(I),I=1,NM),(B(I),I=1,NM)
   10 FORMAT(8F10.3)
       DO 15 I=1,NM
   15 C(I)=A(I)+B(I)
       WRITE(5,20)
   20 FORMAT(1H1,5X,'MATRICA C'/)
       DO 25 I=1,N
   25 WRITE(5,30) (C(J),J=I,NM,N)
   30 FORMAT(' ',4F12.3)
      CALL EXIT
      END
```

Smeštanje elemenata matrice A_1 je dato u tabeli 2.6.

i \ j	1	2	3	4
1	1; a_{11}	4; a_{22}	7	10
2	2; a_{21}	5; a_{13}	8	11
3	3; a_{12}	6; a_{23}	9	12

Tabela 2.6

1.3. NIZOVI I MATRICE U POTPROGRAMIMA

U ovom poglavlju razmotrićemo slučajeve kada su nizovi i matrice argumenti potprograma. U protivnom, ukoliko to nisu, imamo jednostavan slučaj. Naime, tada je potrebno, pomoću naredbe DIMENSION, definisati maksimalne vrednosti indeksa u cilju obezbeđenja potrebnog memorijskog prostora.

Kao što je poznato (videti [1]), za nizove koji su argumenti potprograma, obezbeđenje memorijskog prostora u okviru potprograma nije potrebno, s obzirom da je to učinjeno u glavnom programu. Međutim, da bi se u potprogramu znalo da se radi o nizu, potrebno je koristiti naredbu DIMENSION. Ilustrujmo ovo prostim primerom. Neka je niz A argument u potprogramu SIGMA (A, N, F). Tada je dovoljno u naredbi DIMENSION staviti samo A (1), tako da početni deo potprograma izgleda

```
SUBROUTINE SIGMA (A, N, F)
DIMENSION A (1)
```

Fiktivni argument N ukazuje na konkretan broj elemenata niza A. Potpuno isti efekat je ako se stavi A (N). Poslednji oblik je dozvoljen samo u potprogramima, za nizove koji su argumenti potprograma.

Nešto složeniji slučaj javlja se kod matrica kada se nađu kao argumenti potprograma. Ilustrovaćemo ovo na primeru matrice

$$K = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}.$$

U sledećem programu, matrica K se generiše, a zatim se pozivaju potprogrami PP1, PP2, PP3 i PP4, u kojima je redom, u naredbi DIMENSION, uzeto K (N,M), K (1,1), K (3,1) i K (1).

Iz odgovarajućih potprograma štampa se matrica K za sve moguće vrednosti N i M ($N = 1, 2, 3$; $M = 1, 2, 3, 4$).

```
      DIMENSION K(3,4)
      DO 20 I=1,3
      DO 20 J=1,4
   20 K(I,J)=(I-1)*4+J
      DO 10 IJ=1,4
      WRITE(5,11)IJ
   11 FORMAT('1',5X,'MATRICA K U PP',I1//6X,'N',1X,'M'/)
      DO 10 N=1,3
      DO 10 M=1,4
      GO TO(1,2,3,4),IJ
    1 CALL PP1(K,N,M)
      GO TO 10
    2 CALL PP2(K,N,M)
      GO TO 10
    3 CALL PP3(K,N,M)
      GO TO 10
    4 CALL PP4(K,N,M)
   10 CONTINUE
      CALL EXIT
      END
```

```
      SUBROUTINE PP1(K,N,M)
      DIMENSION K(N,M)
      WRITE(5,20)N,M,((K(I,J),J=1,M),I=1,N)
   20 FORMAT(/6X,I1,1X,I1,<M>I4/((9X,<M>I4))/)
      RETURN
      END
```

```
      SUBROUTINE PP2(K,N,M)
      DIMENSION K(1,1)
      WRITE(5,20)N,M,((K(I,J),J=1,M),I=1,N)
   20 FORMAT(/6X,I1,1X,I1,<M>I4/((9X,<M>I4))/)
      RETURN
      END
```

```
      SUBROUTINE PP3(K,N,M)
      DIMENSION K(3,1)
      WRITE(5,20)N,M,((K(I,J),J=1,M),I=1,N)
   20 FORMAT(/6X,I1,1X,I1,<M>I4/((9X,<M>I4))/)
      RETURN
      END

      SUBROUTINE PP4(K,N,M)
      DIMENSION K(1)
      WRITE(5,10)N,M
   10 FORMAT(/5X,2I2)
      NM=N*M
      DO 15 I=1,N
   15 WRITE(5,20)(K(J),J=I,NM,N)
   20 FORMAT('+',   <M>I4/(9X ,<M>I4))
      RETURN
      END
```

Elementi matrice K smešteni su u redosledu (po kolonama)

Elementi matrice K	1	5	9	2	6	10	3	7	11	4	8	12
Relativna adresa	1	2	3	4	5	6	7	8	9	10	11	12

Pri prenošenju matrice K u potprogram može se, pri nepravilnoj definiciji u naredbi DIMENSION, doći do pogrešnog rezultata, što se lepo vidi iz ovog primera.

Posmatrajmo, najpre, potprogram PP1, gde je u naredbi DIMENSION stavljeno K (N, M). Pri ovome, za fiksirano N i M ($1 \leqslant N \leqslant 3$; $1 \leqslant M \leqslant 4$) iz niza elemenata matrice K uzima se prvih $N \cdot M$ elemenata, na osnovu kojih se formira nova matrica sa N vrsta i M kolona. Na primer, za $N = 2$ i $M = 3$ uzima se prvih šest elemenata: 1, 5, 9, 2, 6, 10, na osnovu kojih se formira matrica tipa 2 x 3

$$\begin{bmatrix} 1 & 9 & 6 \\ 5 & 2 & 10 \end{bmatrix}.$$

Primetimo da se tačan rezultat dobija kada je $N = 3$, tj. kada N ima vrednost maksimalnog prvog indeksa u naredbi DIMENSION u glavnom programu. Naravno, tačan rezultat se dobija i u slučaju kada je $M = 1$.

Potpuno isti rezultat se dobija ako se u potprogramu umesto matrice koristi niz (potprogram PP4), što znači da su elementi niza jednaki elementima matrice, koji su poređani u niz kolona po kolona.

Ako je u naredbi DIMENSION u potprogramu prvi indeks matrice isti kao u naredbi DIMENSION u glavnom programu (videti PP3), na osnovu prethodnog može se zaključiti da će elementi matrice u potprogramu imati pravilan tretman.

U potprogramu PP2 u naredbi DIMENSION je uzeto K (1,1). U ovom slučaju, niz elemenata matrice K, u potprogramu se tretira kao da matrica ima samo jednu vrstu. Na primer, za $N = 2$ i $M = 3$ dobijamo u prvoj vrsti redom elemente 1, 5, 9, dok se elementi kolona, normalno, dobijaju po pravilnom redosledu niza, polazeći od odgovarajućeg elementa prve vrste. Dakle, na ovaj način dobijamo matricu

$$\begin{bmatrix} 1 & 5 & 9 \\ 5 & 9 & 2 \end{bmatrix}.$$

Pri radu sa matricama u potprogramu, treba postupati dosta obazrivo i uvek imati na umu prethodno razmatranje. Bolje je, međutim, u potprogramima raditi samo sa nizovima, o čemu će biti reči u drugoj glavi. Sledeći deo programa

```
        :
        K = − N
        DO 10 J = 1, M
        K = K + N
        DO 10 I = 1, N
        IK = K + I
10      B (IK) = A (IJ)
        :
```

vrši konverziju matrice A tipa N x M u niz B dužine N · M.

```
MATRICA K U PP1                     MATRICA K U PP2
N M                                 N M

1 1    1                            1 1    1

1 2    1   5                        1 2    1   5

1 3    1   5   9                    1 3    1   5   9

1 4    1   5   9   2                1 4    1   5   9   2

2 1    1                            2 1    1
       5                                   5

2 2    1   9                        2 2    1   5
       5   2                               5   9

2 3    1   9   6                    2 3    1   5   9
       5   2  10                           5   9   2

2 4    1   9   6   3                2 4    1   5   9   2
       5   2  10   7                       5   9   2   6

3 1    1                            3 1    1
       5                                   5

       9                                   9

3 2    1   2                        3 2    1   5
       5   6                               5   9

       9  10                               9   2

3 3    1   2   3                    3 3    1   5   9
       5   6   7                           5   9   2

       9  10  11                           9   2   6

3 4    1   2   3   4                3 4    1   5   9   2
       5   6   7   8                       5   9   2   6

       9  10  11  12                       9   2   6  10
```

```
MATRICA K U PP3

N M

1 1   1

1 2   1  2

1 3   1  2  3

1 4   1  2  3  4

2 1   1
      5

2 2   1  2
      5  6

2 3   1  2  3
      5  6  7

2 4   1  2  3  4
      5  6  7  8

3 1   1
      5
      o

3 2   1  2
      5  6
      9  10

3 3   1  2  3
      5  6  7
      9  10 11

3 4   1  2  3  4
      5  6  7  8
      9  10 11 12
```

```
MATRICA K U PP4

N M

1 1   1

1 2   1  5

1 3   1  5  9

1 4   1  5  9  2

2 1   1
      5

2 2   1  9
      5  2

2 3   1  9  6
      5  2  10

2 4   1  9  6  3
      5  2  10 7

3 1   1
      5
      9

3 2   1  2
      5  6
      9  10

3 3   1  2  3
      5  6  7
      9  10 11

3 4   1  2  3  4
      5  6  7  8
      9  10 11 12
```

1.4. PREGLED BIBLIOTEČKIH FUNKCIJSKIH POTPROGRAMA ZA RAČUNAR *PDP*–11/40

U ovom poglavlju dajemo pregled bibliotečkih funkcijskih potprograma koji se mogu koristiti u svakom programu, navođenjem njihovih imena. U priloženoj tabeli, za tip argumenta i tip rezultata, korišćene su sledeće oznake:

I celobrojna veličina (INTEGER)
R realna veličina obične tačnosti (REAL)
D realna veličina dvostruke tačnosti (DOUBLE PRECISION)
C kompleksna veličina (COMPLEX)

$z = x + iy$, $\mathrm{Re}\, z = x$, $\mathrm{Im}\, z = y$

$|x|$ apsolutna vrednost

$[x]$ celi deo od x

$sgn\, x$ znak od x.

OBLIK		TIP		OPIS
U FORTRANu	u matematici	arg.	rez.	
ABS (X)	$\|x\|$	R	R	Apsolutna vrednost
IABS (I)	$\|i\|$	I	I	
DABS (X)	$\|x\|$	D	D	
CABS (Z)	$\|z\| = \sqrt{x^2+y^2}$	C	R	
FLOAT (I)		I	R	Konverzija veličina
IFIX (X)		R	I	
SNGL (X)		D	R	
DBLE (X)		R	D	
REAL (Z)	$\mathrm{Re}\, z = x$	C	R	
AIMAG (Z)	$\mathrm{Im}\, z = y$	C	R	
CMPLX (X,Y)	$z = z + i\, y$	R	C	
AINT (X)	$[\,\|x\|\,]\, \mathrm{sgn}\, x$	R	R	Odbacivanje decimala datom broju
INT (X)		R	I	
IDINT (X)		D	I	
MOD (I,J)	$i \,(\mathrm{mod}\, j)$	I	I	Modularna aritmetika
AMOD (X,Y)	$x - y\, [\frac{x}{y}]$	R	R	
DMOD (X,Y)		D	D	
AMAX0 (I,J, . . .)	$\max\,(i, j, \ldots)$	I	R	Određivanje maksimalnog argumenta
AMAX1 (X,Y, . . .)		R	R	
MAX0 (I, J, . . .)		I	I	
MAX1 (X,Y, . . .)		R	I	
DMAX1 (X,Y, . . .)		D	D	
AMIN0 (I,J, . . .)	$\min\,(i, j, \ldots)$	I	R	Određivanje minimalnog argumenta
AMIN1 (X,Y, . . .)		R	R	
MIN0 (I,J, . . .)		I	I	
MIN1 (X,Y, . . .)		R	I	
DMIN1 (X,Y, . . .)		D	D	

SIGN (X,Y) ISIGN (I,J) DSIGN (X,Y)	$\lvert y \rvert \operatorname{sgn} x$	R I D	R I D	Znak prvog argumenta dodeljuje se modulu drugog argumenta
DIM (X,Y) IDIM (I,J)	$x-\min(x, y)$ $i-\min(i, j)$	R I	R I	Pozitivna razlika
EXP (X) DEXP (X) CEXP (Z)	e^x e^z	R D C	R D C	Eksponencijalna funkcija
ALOG (X) ALOG10 (X) DLOG (X) DLOG10 (X) CLOG (Z)	$\log_e x$ $\log_{10} x$ $\log_e z$	R R D D C	R R D D C	Logaritam
SQRT (X) DSQRT (X) CSQRT (Z)	$\sqrt{x}$ $\sqrt{z}$	R D C	R D C	Kvadratni koren
SIN (X) DSIN (X) CSIN (Z) COS (X) DCOS (X) CCOS (Z)	$\sin x$ $\sin z$ $\cos x$ $\cos z$	R D C R D C	R D C R D C	Trigonometrijske funkcije
TANH (X)	$\operatorname{th} x$	R	R	Hiperbolički tangens
ATAN (X) DATAN (X) ATAN2 (X,Y) DATAN2 (X,Y)	$\operatorname{arctg} x$ $\operatorname{arctg}\frac{x}{y}$	R D R D	R D R D	Inverzna funkcija od tangens
CONJG (Z)	$x-iy$	C	C	Konjugovani broj

2. METODI OPTIMIZACIJE FORTRAN PROGRAMA

2.1. UVOD

Algoritamski rešiv problem može se programirati na više različitih načina. Programi koji ispravno rešavaju jedan isti problem mogu se prema svojim performansama prilično razlikovati. Na primer, prema zauzeću memorije i brzini izvršenja, razlike dostižu i odnos 1:50. Novije verzije kompajlera, podržane medernijim procesorima i operativnim sistemima, imaju u sebi ugrađene rutine za izvesnu optimizaciju programa. Razume se, logička optimizacija programa je bila i ostala primarni zadatak koji se rešava u dijalogu čovek – mašina. Optimizacija koja se odnosi na optimalnu hardversku podršku programa, odnosno optimalno korišćenje postojeće računarske konfiguracije, obuhvata:

1. Tačnost izračunavanja;
2. Racionalno korišćenje unutrašnje memorije računara;
3. Brzinu izvršenja.

Naredbe i podaci potrebni za izvršenje programa smeštaju se u unutrašnju– operativnu memoriju. Različite operacije nad podacima odvijaju se u komunikaciji između memorije i komandnog organa – procesora. Kako je memorija ograničenog kapaciteta, u nju realno može da stane ograničen broj naredbi i podataka što znači da je veličina programa ograničena kapacitetom centralne memorije. Problem racionalnog korišćenja memorije rešava se na više načina:

– Izborom optimalne dužine podataka;
– Višestrukim korišćenjem istih memorijskih prostora i to:
  a) na nivou promenljivih,
  b) na nivou programskih celina.

2.2. TAČNOST IZRAČUNAVANJA

U skoro svim numeričkim problemima, izlazna informacija, ili kako se češće kaže numeričko rešenje, praćeno je greškama čiji izvori mogu biti različiti. Međutim, u opštem slučaju, može se reći da greška izlazne informacije potiče od

1. greške ulazne informacije;
2. greške algoritma,

i može se predstaviti sledećim blok dijagramom:

U ovom poglavlju ukazaćemo samo na neke važnije pojmove koji se sreću pri analizi grešaka.

Približan broj $\bar{x}$ je broj koji zamenjuje tačan broj x u izračunavanjima i neznatno se razlikuje od njega. Odgovarajuća greška* je

$$e = \bar{x} - x.$$

Svaki broj x može se predstaviti u normalizovanom obliku

$$x = (\pm\, 0.a_1 a_2 \ldots a_n a_{n+1} \ldots)\, b^k \qquad (a_1 \neq 0), \tag{2.1}$$

gde je b osnova brojnog sistema, a a_i $(i = 1,2,\ldots)$ cifra brojnog sistema $(0 \leqslant a_i < b)$.

Broj $\pm\, 0.\, a_1 a_2 \ldots a_n a_{n+1} \ldots$ zvaćemo mantisom i označavati sa x^*. Broj k zvaćemo karakteristikom. Dakle, možemo pisati

$$x = x^* b^k.$$

Najčešće je u upotrebi binarni ili decimalni brojni sistem, tj. $b = 2$ ili $b = 10$.

Za $\bar{x}$ se kaže da aproksimira broj x sa m značajnih cifara ako je m najveći broj cifara mantise, za koji $|\bar{x}^* - x^*|$ ne prelazi jedinicu m–tog mesta.

U praktičnim izračunavanjima, primenom elektronskih računara, prinuđeni smo da radimo sa jednim vrlo uskim skupom brojeva. Naime, svaki realni broj oblika (2.1) koji se dobija kao rezultat određenih računskih operacija, zamenjuje se približnim brojem oblika

$$\bar{x} = (\pm 0.\, a_1 a_2 \ldots a_n)\, b^k \qquad (a_1 \neq 0).$$

U ovom slučaju kažemo da imamo mantisu sa n razreda.

Proces odbacivanja cifara mantise u broju x, počev od cifre a_{n+1} naziva se prosto odsecanje. Apsolutna greška pri ovome je

$$|e| \leqslant b^{k-n}.$$

Apsolutna greška, pri zameni broja x brojem $\bar{x}$, može se smanjiti ako se koristi tzv. postupak zaokrugljivanja brojeva. Taj postupak se sastoji u sledećem:

1° Ako je

$$a_{n+1} + a_{n+2}\, b^{-1} + \ldots < \frac{1}{2} b$$

koristi se prosto odsecanje;

---

*) Često se ova greška naziva apsolutnom greškom. Međutim, ovo je pogrešno jer je apsolutna greška $|e| = |\bar{x} - x|$.

2° Ako je

$$a_{n+1} + a_{n+2}b^{-1} + \ldots > \frac{1}{2}b$$

cifra a_n se povećava za jedinicu, a cifre $a_{n+1}, a_{n+2}, \ldots$ se odbacuju;

3° Ako je

$$a_{n+1} + a_{n+2}b^{-1} + \ldots = \frac{1}{2}b$$

ravnopravno se mogu koristiti pravila 1° i 2°.

Na računskim mašinama zaokrugljivanje se najčešće izvodi dodavanjem broja $\frac{1}{2}\, b^{k-n}$ rezultatu x, a zatim se vrši prosto odsecanje. Ovo znači da se u nerešenom slučaju 3° uvek a_n zamenjuje sa $a_n + 1$ (pravilo 2°).

Napomenimo da se kod ručnog zaokrugljivanja brojeva u dekadnom sistemu ($b = 10$) u nerešenom slučaju 3° preporučuje sledeće pravilo:

Ako je cifra a_n paran broj koristiti pravilo 1°, a ako je neparan broj koristiti pravilo 2°.

Na primer, sukcesivno zaokrugljivanje broja $\pi = 3.1415926535897932$ daje redom brojeve:

```
3.1415926535897932
3.141592653589793
3.14159265358979
3.1415926535898
3.141592653590
3.14159265359
3.1415926536
3.141592654
3.14159265
3.1415927
3.141593
3.14159
3.1416
3.142
3.14
3.1
3.
```

Apsolutna greška kod zaokrugljenog broja je

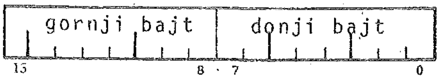
$$|e| \leqslant \frac{1}{2} b^{-n+k}.$$

Detaljnija analiza grešaka može se naći u [2].

2.3. RACIONALNO KORIŠĆENJE UNUTRAŠNJE MEMORIJE RAČUNARA

2.3.1. Smeštanje podataka

Najmanja adresibilna jedinica memorije je bajt (byte). To je skup od osam osnovnih jedinica memorije — bitova. Dva susedna bajta čine šesnaesto bitnu reč, koja se šematski može prikazati kao na sl. 2.1.

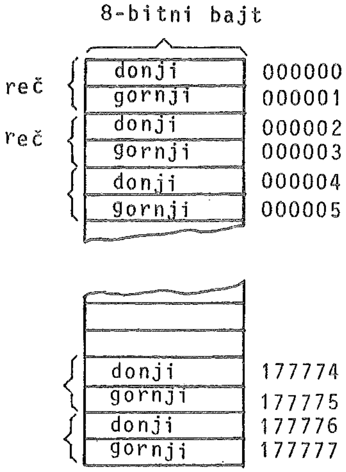


Sl. 2.1

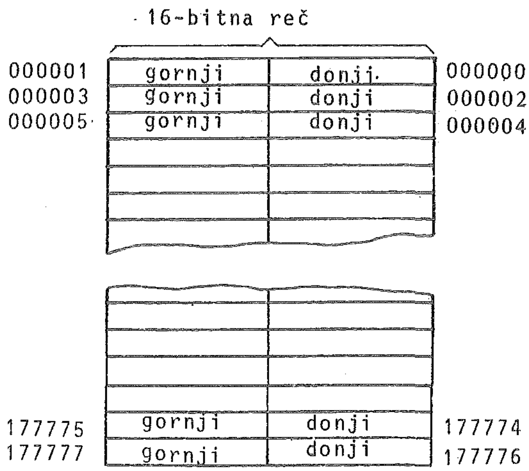
Memorija se može posmatrati kao niz adresiranih bajtova (sl. 2.2), ili niz parova bajtova – niz reči (sl. 2.3).

Ukažimo sada na načine smeštanja podataka u memoriji računara.

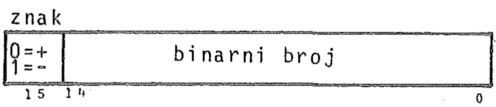
1. **Smeštanje INTEGER brojeva.** Celi brojevi se smeštaju u binarnoj komplementarnoj reprezentaciji (videti sl. 2.4) u jednu reč (dva bajta), što se zahteva pri kompilaciji programa zahtevom „/ON" (ONE WORD = ONE INTEGER). Bez zahteva '„ON"', celobrojne konstante i promenljive zauzeće adresom dve reči, dok će se njihova vrednost smestiti u jednu reč. Celobrojne konstante i promenljive moraju da leže u intervalu od – 32768 do + 32767.



Sl. 2.2

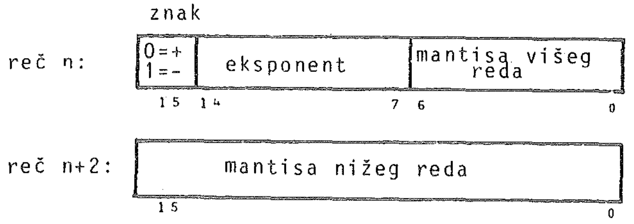


Sl. 2.3



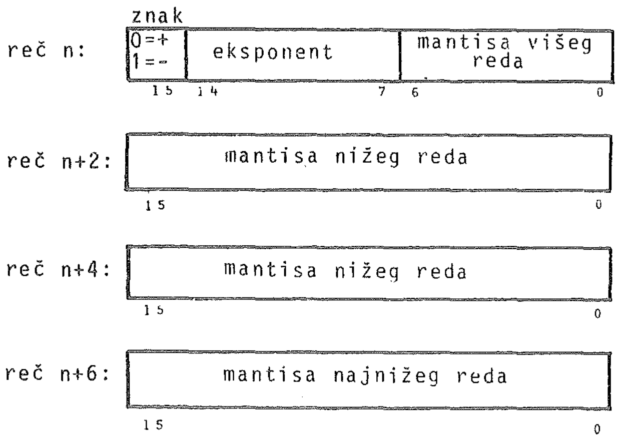
Sl. 2.4

2. **Smeštanje REAL brojeva (obična tačnost)**. REAL brojevi se predstavljaju u pokretnoj tački (Floating Point) i smeštaju u dve reči–četiri bajta (sl. 2.5). Kako je bit najveće težine mantise, posle izvršene normalizacije uvek jednak 1, postiže se efektivna preciznost od 24 bita ili aproksimativno 7 decimalnih cifara. Ovim se postiže registrovanje brojeva koji leže približno u intervalu od $0.14 \cdot 10^{-38}$ do $1.7 \cdot 10^{38}$. Naravno, ovde je reč samo o brojevima sa najviše 7 značajnih decimalnih cifara.



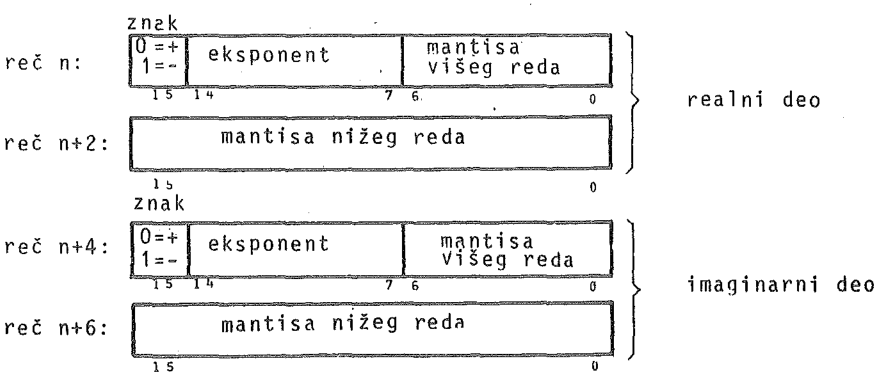
Sl. 2.5

3. Smeštanje REAL brojeva (dvostruka tačnost). Brojevi dvostruke tačnosti se smeštaju u 4 reči–8 bajta (videti sl. 2.6). Efektivna preciznost je 56 bitova ili aproksimativno 16 decimala. Broj po apsolutnoj vrednosti mora da leži u intervalu od $0.14 \cdot 10^{-38}$ do $1.7 \cdot 10^{38}$.



Sl. 2.6

4. Smeštanje kompleksnih brojeva. Kompleksni broj se predstavlja parom realnih brojeva datih u običnoj tačnosti (sl. 2.7).



Sl. 2.7

5. **Smeštanje podataka tipa BYTE (ili LOGICAL*1)**. Brojevi se smeštaju u jedan bajt, prema sl. 2.8, i mora da se nalaze u intervalu od −128 do + 127.

nespecificirano	podatak
15 … 8	7 … 0

Sl. 2.8

6. **Smeštanje Hollerith–podataka**. Tekstovi, alfanumerički znaci (karakteri), smeštaju se u memoriji po jedan karakter u jedan bajt, s time da pretposlednji bajt ostaje prazan u slučaju neparnog broja karaktera. Maksimalan broj karaktera je 255 (videti sl. 2.9).

	15 … 8	7 … 0
reč 0:	2. karakter	1. karakter
reč 2:	4. karakter	3. karakter
⋮		
reč $\frac{n}{2}$:	prazno=40_8	n-ti karakter ($n \leq 255$)

Sl. 2.9

7. **Smeštanje podataka tipa LOGICAL**. Logičke promenljive se memorišu u jednoj reči i mogu se koristiti u logičkim i aritmetičkim izrazima. Logičke vrednosti TRUE i FALSE prikazane su na sl. 2.10.

	15					0
True:	1	7	7	7	7	7
False:	0	0	0	0	0	0

Sl. 2.10

U logičkoj IF naredbi svaka vrednost različita od FALSE tretira se kao TRUE.

U tabeli 3.1 daje se pregled smeštanja predhodno navedenih podataka u memoriji.

Vrsta podatka	Dužina podatka bajt		
	Minimal	Maksim	Stand.
INTEGER	2	4	4
REAL(REAL*8)	4	8	4
COMPLEX	8	8	8
LOGICAL(BYTE)	2	2	2
Hollerith	2	255	2-255

Tabela 3.1

2.3.2. Određivanje dužine podataka

Za sve podatke korišćene u programu pretpostavlja se standardna dužina. Pored standardne dužine, postoji i minimalna i maksimalna dužina podataka. U zavisnosti od programskih zahteva, određuje se optimalna dužina, s ciljem da se zauzme minimalni memorijski prostor.

Tip promenljivih u programu se može definisati opisnom naredbom za implicitnu deklaraciju

IMPLICIT lista,

gde su elementi liste oblika

tip*s (niz početnih slova),

s je dužina podataka u bajtovima koja se odnosi na odgovarajuću vrstu promenljivih, čija imena počinju slovima navedenim u nizu. Ako promenljive koje se implicitno deklarišu imaju standardnu dužinu, lista se piše u obliku:

tip (niz početnih slova).

Implicitno definisanje tipa promenljivih koje vrši sam kompajler odgovara naredbama:

```
IMPLICIT REAL (A–H), (O–Z)
IMPLICIT INTEGER (I–N)
```

Naredba

```
IMPLICIT INTEGER (X, Y, Z), REAL* 8 (R, S, T)
```

znači da će sve promenljive čija imena počinju slovima X, Y, Z biti celobrojne i svaka će zauzeti 4 bajta–2 reči (bez specifikacije „/ON" pri kompiliranju, što znači da se jedna celobrojna promenljiva smešta u jednu reč–dva bajta), a promenljive čija imena počinju slovima R, S, T biće realne promenljive dvostruke tačnosti, dužine 4 reči–8 bajtova.

Pri implicitnom određivanju tipa promenljivih treba obratiti pažnju na to da ne promenimo tip nekih bibliotečkih funkcija (svih osim onih koje daju kompleksan rezultat ili rezultat dvostruke tačnosti). Tako će funkcija FLOAT u programu:

```
IMPLICIT INTEGER (A–Z)
REAL X
I = 1
X = FLOAT (I)
.
.
.
```

da generiše pogrešan rezultat tipa INTEGER i da ga dodeli promenljivoj X. Ovaj program mora da sadrži i naredbu za eksplicitno definisanje tipa promenljive:

```
REAL FLOAT
```

Eksplicitno određivanje tipa promenljivih za svaku određenu promenljivu ili polje (niz, matricu) vrši se naredbama:

```
INTEGER
INTEGER*2 (isto kao INTEGER)
REAL
REAL* 4 (isto kao REAL)
DOUBLE PRECISION
REAL* 8 (isto kao DOUBLE PRECISION)
COMPLEX
LOGICAL
BYTE
LOGICAL* 1
```

Predhodno data razmatranja mogu se koristiti za optimizaciju dužine podataka.

2.3.3. Višestruko korišćenje memorijskog prostora na nivou promenljivih

Višestruko korišćenje memorijskog prostora na nivou promenljivih može se primeniti

– unutar jednog programa;

– od strane više programskih jedinica.

Opisna naredba za višestruko korišćenje memorijskog prostora na nivou promenljivih unutar jednog programa definiše zajednička polja i ima oblik

EQUIVALENCE lista,

gde je lista spisak promenljivih razdvojenih zarezima koji zauzimaju isto mesto u memoriji ako su iste dužine, ili se preklapaju, ako su različitih dužina. Tako, na primer, imamo

EQUIVALENCE (J,K), (TEZ, DUZ), (A, B, C)

što znači da će promenljivim J i K biti dodeljeno isto polje u memoriji dužine 2 bajta, realnim promenljivim TEZ I DUZ isto polje dužine 4 bajta (2 reči), i promenljivim A, B, C polje dužine 4 bajta. Bez korišćenja predhodne naredbe za smeštanje u memoriji bilo bi nam potrebno 24 bajta (J, K, TEZ, DUZ, A, B, C – $2 \times 2 + 5 \times 4 = 24$). Korišćenjem naredbe EQUIVALENCE potreban memorijski prostor se svodi na $2 + 2 \times 4 = 10$ bajtova.

Ako imamo

DIMENSION A (100), B (100)
EQUIVALENCE (A, B)

ušteda u memoriji će biti 400 bajtova.

Za promenljive različitih dužina zajedničko polje određeno je promenljivom najveće dužine. Na primer, ako imamo

COMPLEX C1, C2
REAL*8 A, B, C
REAL I, J, K
INTEGER*2 KOKA, KOLA
LOGICAL L4, L3
EQUIVALENCE (C1, C2, A, B, C, I, J, K, KOKA, KOLA, L4, L3)

polje sa rasporedom promenljivih izgleda kao na sl. 2.11.

Nizovi se takođe mogu smeštati u zajednička polja sa običnim promenljivim ili sa drugim nizovima. Naredbe

DIMENSION Y (20)
EQUIVALENCE (Y, X)

imaju isti efekat kao i naredbe

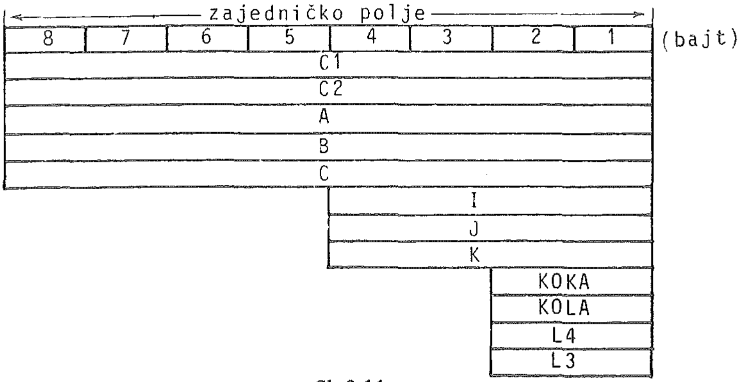
DIMENSION Y (20)
EQUIVALENCE (Y (1), X).

Raspored promenljivih, definisanih naredbama

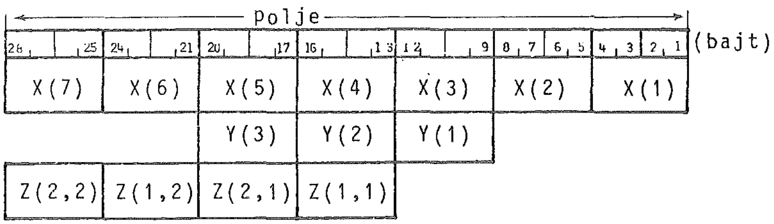
DIMENSION X (7), Y (3), Z (2,2)
EQUIVALENCE (X (5), Y (3), Z (2,1)

u zajedničkom polju izgleda kao na sl. 2.12.

U daljem tekstu razmotrićemo višestruko korišćenje memorijskog prostora na nivou promenljivih od strane više programskih jedinica.



Sl. 2.11



Sl. 2.12

Dok naredba EQUIVALENCE omogućuje višestruko korišćenje istih memorijskih prostora unutar jednog programa ili potprograma, postoji potreba za višestrukim korišćenjem istih delova memorije od strane više programskih jedinica–programa, potprograma ili niza povezanih programa. Naredba za definisanje zajedničke zone je

COMMON lista,

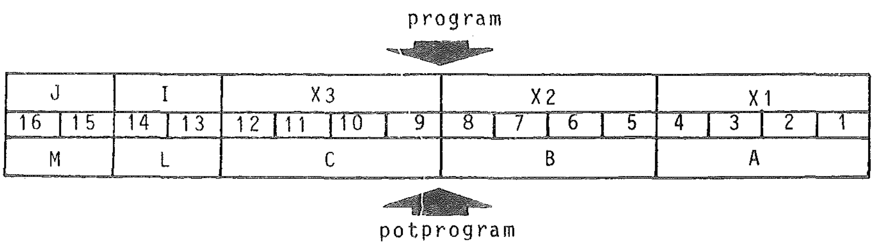
gde je lista spisak promenljivih, među sobom odvojenih zarezima. Ako u programu imamo

COMMON X1, X2, X3, I, J

a u potprogramu naredbu

COMMON A, B, C, L, M

tada će zajednički prostor u memoriji biti kao na sl. 2.13.



Sl. 2.13

Jasno je da naredba COMMON omogućuje ulaz podataka iz programa u potprogram i obrnuto, bez njihovog navođenja kao argumenata potprograma, što doprinosi, zbog direktnog adresiranja, povećanju brzine izvršenja programa.

Pored neimenovanih zona navedenih u COMMON opisu, postoje i imenovane zone, koje sadrže grupu ili blok promenljivih:

COMMON /R/X, Y/ /B, C, D

Ovde imamo blok–zonu R, koja sadrži promenljive X, Y i neimenovanu zonu sa promenljivim B, C, D. Ova naredba se može napisati i ovako:

COMMON B, C, D/R/X, Y

Ako, na primer, program sadrži:

COMMON A, B/R/X, Y, Z

kao prvu naredbu za zajedničko područje, a potprogram sadrži naredbu:

COMMON /R/U, V, W/ /D, E

pristup istoj zoni memorije iz programa i potprograma izgledaće kao na sl. 2.14.

program

B				A				Z				Y				X				
20			17	16			13	12			9	8			5	4			1	← bajt
E				D				W				V				U				

potprogram

Sl. 2.14

Kombinacijom naredbi COMMON i EQUIVALENCE mogu se postići još veće uštede u memorijskom prostoru. Na primer,

COMMON /R/X, Y, Z
DIMENSION A (4)
EQUIVALENCE (A, Y)

prouzrokovaće raspored u memoriji prikazan na sl. 2.15.

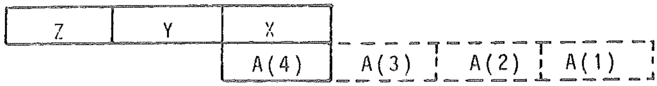
								Z				Y				X				
20			17	16			13	12			9	8			5	4			1	← bajt
A(4)				A(3)				A(2)				A(1)								

Sl. 2.15

Međutim, nepažljivo programiranje može dovesti do pogrešnih zahteva. Neka je, na primer, dato:

COMMON /R/X, Y, Z
DIMENSION A (4)
EQUIVALENCE (X, A (4))

S obzirom da je izvršeno preklapanje prve promenljive bloka R, X i četvrtog člana niza A (4), nisu ostavljena mesta za A (1), A (2) i A (3) (videti sl. 2.16).



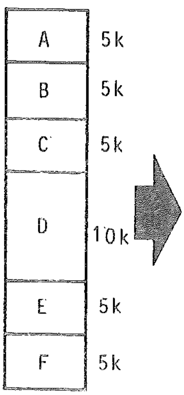
Sl. 2.16

Da ne bi došlo do pogrešnih opisa promenljivih prouzrokovanih nekorektnim redosledom opisanih naredbi, navodimo njihov ispravan redosled.

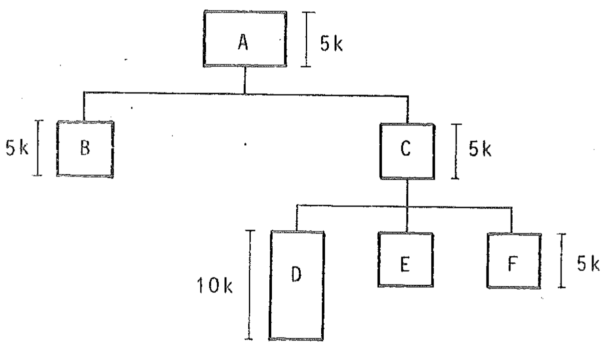
1. Eksplicitna deklaracija vrste promenljivih (INTEGER, REAL, DOUBLE PRECISION, COMPLEX, LOGICAL, BYTE);
2. Implicitna deklaracija vrste promenljivih (IMPLICIT);
3. Navođenje imena potprograma koji se javljaju kao argumenti drugih potprograma (EXTERNAL);
4. Dimenzionisanje polja (DIMENSION),
5. Definisanje zajedničkog memorijskog prostora za više programskih jedinica (COMMON);
6. Definisanje zajedničkog memorijskog prostora u jednoj programskoj jedinici (EQUIVALENCE);
7. Funkcijske naredbe;
8. Programska procedura.

2.3.4 Višestruko korišćenje memorijskog prostora na nivou programskih celina

Vrlo često se dešava da i pored optimalnog korišćenja memorije na nivou promenljivih, veličina programa prevazilazi kapacitet memorije. Taj problem se rešava deobom programa na manje programske jedinice (segmente) i uvođenjem tih segmenata u memoriju prema potrebi–po pozivu. Preklapanje (OWERLAY) programskih jedinica vrši se tek onda kada je prethodna jedinica završila rad i pozvala narednu. Ovakva organizacija se zove LOCAL (load–on–call). Na primer, ako se program sastoji iz više programskih jedinica, sekvencijalna šema programa (sl. 2.17) se može preurediti u razgranatu šemu prema sl. 2.18. Razgranata struktura progra-



Sl. 2.17



Sl. 2.18

ma koja se bazira na preklapanju programskih jedinica B i C, D, E, F i D i E, F, naravno ne u isto vreme, omogućiće smeštanje programske strukture veličine 45 kW (kilo–reči) u memoriji veličine 20 kW. Princip ovakve organizacije (LOCAL–OWERLAY) je da se u jednom trenutku u memoriji nalaze samo aktivni programski delovi, a da se ostali delovi uvode u memoriju samo po potrebi, na poziv operativnog sistema. Uvođenje potrebnih programskih delova u memoriju može se vršiti na kraju izvršene programske jedinice naredbom CALL LOAD („ime nove programske jedinice"), ili automatski, programiranjem šeme redosleda uvođenja programskih segmenata u memoriju, što je podržano operativnim sistemom. Međurezultati programa moraju se čuvati u posebnom delu memorije sa kojim se nikada ne vrši preklapanje i koji se zove osnovni deo programske strukture ili koren (ROOT), označen u šemi sa A. O uticaju ovakve organizacije smeštanja programa u memoriji, na brzinu izvršenja programa, biće reči u daljem tekstu.

2.4. BRZINA IZVRŠENJA

Dve osnovne karakteristike operativne–brze memorije u pogledu brzine su vreme pristupa i memorijski ciklus, što za sistem PDP11/40 iznosi:

vreme pristupa 360 ns;

memorijski ciklus 900 ns.

Za numerički orijentisane probleme od velike su važnosti brzine izvršavanja aritmetičkih operacija (FLOATING POINT), koje su date u tabeli 4.1.

Operacija	Min vreme [μs]	Max vreme [μs]	Srednje vreme [μs]
+	18.78	34.18	26.48
-	19.08	34.18	26.48
*	29.00	37.50	33.25
/	46.72	55.22	50.97

Tabela 4.1

Pri radu sa diskovima (korišćenje virtuelne memorije na disku), važno je znati karakteristike disk jedinice RKO5:

– kapacitet $1.2 \cdot 10^6$ reči;

– vreme pristupa 70 ms;

– vreme prenosa 11 μs.

Sada ćemo ukazati na izvesne načine za povećanje brzine izvršenja programa.

Pri nepažljivom pisanju programa pojedine aritmetičke operacije mogu se nepotrebno ponavljati, što povećava vreme izvršenja. Na primer, umesto

Q1 = A + X * Y
Q2 = B + Y * X
Q3 = X * Y + C

treba pisati

```
      T = X * Y
      Q1 = A + T
      Q2 = B + T
      Q3 = T + C
```

Umesto

```
      DO 10 I = 1,100
10    S = 2.0*Q*A (I) + S
```

treba pisati

```
      P = 2 . 0 * Q
      DO 10 I = 1,100
10    S = P * A (I) + S
```

gde izvlačenjem izraza 2.0* Q ispred petlje eliminišemo 99 operacija množenja.

Zamenom jedne aritmetičke IF naredbe oblika

```
      IF (X – 3.141592) 10, 20, 10
10    CONTINUE
```

naredbom

```
      IF (X.EQ.3.141592) GO TO 20
10    CONTINUE
```

štedimo više operacija testiranja i grananja.

Svi podaci u memoriji računara su adresirani da bi mogli da budu pronađeni i da bi mogli da budu podvrgnuti željenim operacijama. Postoje dva osnovna načina adresiranja podataka u memoriji:

a) direktno adresiranje
b) indirektno adresiranje.

Ove dve vrste adresiranja imaju više različitih podvrsti o kojima neće biti reči. Pristup direktno adresiranim podacima je brži pa iako je programeru u FORTRANu teže da podesi način adresiranja ima primera gde se to može postići:

Primer 2.4.1. Ako imamo potprogram tipa SUBROUTINE ili FUNCTION, adresiranje parametara potprograma je indirektno pa je prenošenje argumenata iz programa u potprogram i obrnuto sporije. Brzina izvršenja se može povećati ako se umesto programa i potprograma oblika

Potprogram:

```
      SUBROUTINE PRIMER (A,B,C,D)

      A = . . .
      B = . . .

      C = . . .
      .
      :
      RETURN
      END
```

Program

```
      .
      :
      CALL PRIMER (X,Y,Z,T)
      .
      :
      END
```

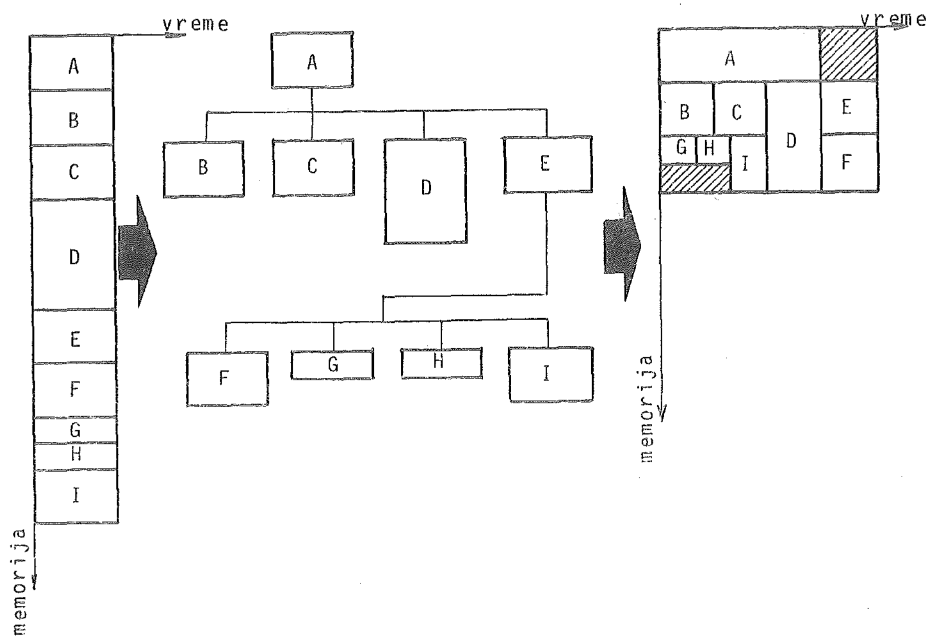
napišu program i potprogram u obliku:

```
SUBROUTINE PRIMER                     COMMON X,Y,Z,T
COMMON A,B,C,D                        .
A = . . .                             :
B = . . .                             CALL PRIMER
C = . . .                             .
.                                     .
:                                     .
RETURN                                END
END
```

Kada radimo sa poljima većih dimenzija od 1, usled toga što računar radi samo sa jednodimenzionalnim nizovima (vektorima), interno preračunavanje adresa dovodi do velikih gubitaka u vremenu. Zato se preporučuje rad sa jednodimenzionalnim nizovima umesto višedimenzionalnih (matrica).

Na kraju, ukažimo kako segmetacija programa utiče na brzinu izvršenja programa.

Kada se, usled veličine programa, vrši njegova podela na segmente koji se po pozivu (load–on–call) uvode u iste blokove interne memorije, dolazi do velikih gubitaka vremena. Jasno je da se preporučuje stvaranje što većih rutina za prekrivanje, jer se tako broj uvođenja u memoriju smanjuje, a time i gubici u vremenu. Tehnikom prekrivanja (OWERLAY), omogućuje se rad većih programa na račun vremena izvršenja, što se vidi sa slike 2.19.



Sl. 2.19

3. PROGRAMIRANJE UVODNIH PROBLEMA

U ovoj glavi daćemo veći broj kompletno rešenih problema, koji imaju za cilj savladavanje osnovnih principa programiranja, kao i postepeno uvođenje čitaoca u složenije numeričke procese.

3.1. Sastaviti blok šemu algoritma i napisati program za izračunavanje sume

$$S = \sum_{i=1}^{n} x_i$$

gde je

$$x_i = \begin{cases} a_i^2 + b_i^2 + c_i^2 & (a_i + b_i > c_i), \\ a_i + 2b_i c_i & (a_i + b_i = c_i), \\ a_i + b_i - c_i & (a_i + b_i < c_i). \end{cases}$$

Trojke ulaznih podataka a_i, b_i, c_i $(i = 1, \ldots, n)$ su zadate na karticama, tako da je na svakoj kartici po jedna, u formatu 3F8.2. Broj n je određen brojem kartica sa podacima. Kraj podataka je označen praznom karticom.

Izlazni rezultat štampati u obliku:

ZBIR XXXX SABIRAKA IZNOSI:

S = ±X. XXXXXXXXE±XX

Rešenje: Blok šema algoritma po kojoj je napisan program data je na sl. 3.1. Kao kriterijum za završetak algoritma iskorišćen je uslov $D = a_i^2 + b_i^2 + c_i^2 = 0$. Određivanje vrednosti za x_i vrši se testiranjem znaka aritmetičkog izraza $\psi = a_i + b_i - c_i$. Primetimo da je algoritam tako organizovan da se sukcesivno obrađuju trojke ulaznih podataka, tako da vođenje računa o indeksu i nije potrebno.

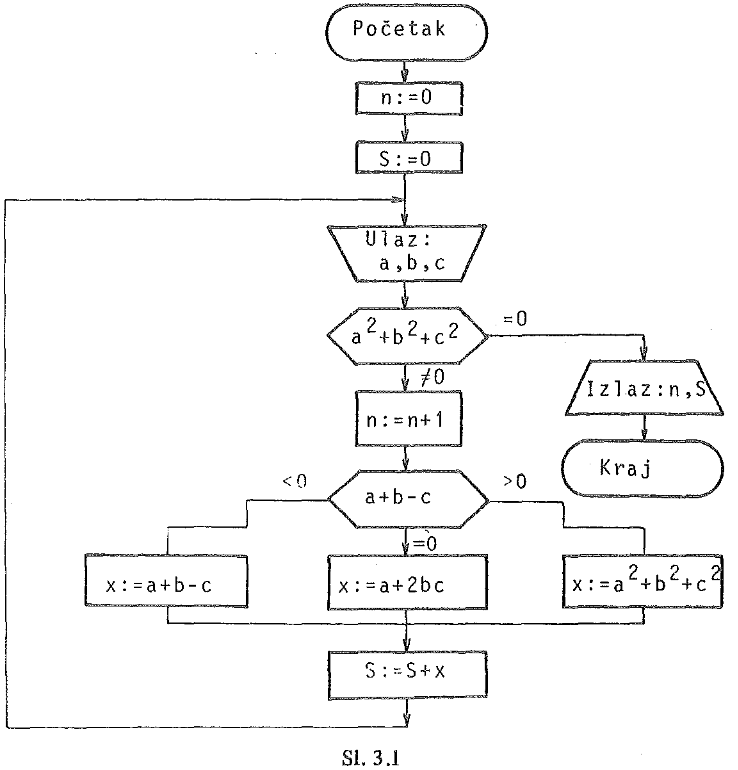
Program ima sledeći oblik:

```
C       IZRACUNAVANJE SUME S
        N=0
        S=0.
      6 READ(8,10)A,B,C
     10 FORMAT(3F8.2)
        D=A*A+B*B+C*C
        IF(D.EQ.0.)GO TO 4
        N=N+1
        IF(A+B-C)1,2,3
      1 X=A+B-C
        GO TO 5
```

```
2  X=A+2.*B*C
   GO TO 5
3  X=D
5  S=S+X
   GO TO 6
4  WRITE(5,20)N,S
20 FORMAT(/9X,'ZBIR  ',I4,'  SABIRAKA  IZNOSI '//
  19X,'S= ',E15.8 // )
   CALL EXIT
   END
```

Za određeni skup ulaznih podataka, pomoću ovog programa, dobijen je izlazni rezultat

```
ZBIR      5  SABIRAKA  IZNOSI

S=  0.46000000E+02
```



Sl. 3.1

3.2. Prema blok šemi algoritma iz primera 1.4 (poglavlje 1.1) napisati program za izračunavanje vrednosti polinoma $F(x)$. U programu predvideti mogućnost da se vrednost polinoma može izračunati za više vrednosti argumenta x.

Program testirati na polinomu

$$F(x) = 1.5 + 2.5x + 3.5x^2 - 3.25x^3 - 1.55x^4 + 2.123x^5 - 5.22x^6,$$

uzimajući za x redom 1., 1.5, – 1., 0.53, –0.55, 1.02.

Rešenje: Program i izlazna lista imaju oblik:

```
      DIMENSION A(10)
      READ(8,5) N
    5 FORMAT(I2)
      N1=N+1
      READ(8,10) (A(I),I=1,N1)
   10 FORMAT(8F10.0)
      WRITE(5,40)
   40 FORMAT(5X,'IZRACUNAVANJE VREDNOSTI POLINOMA'
     1' PO HORNEROVOJ SEMI'//)
   15 READ(8,10,END=30) X
      N=N1-1
      F=A(N+1)
   20 IF(N.EQ.0) GO TO 25
      F=F*X+A(N)
      N=N-1
      GO TO 20
   25 WRITE(5,35) X,F
   35 FORMAT(16X,'X= ',F5.2,5X,'F(X)= 'F10.6)
      GO TO 15
   30 CALL EXIT
      END
```

```
IZRACUNAVANJE VREDNOSTI POLINOMA PO HORNEROVOJ SEMI


                X=  1.00     F(X)=  -0.397000
                X=  1.50     F(X)= -49.028160
                X= -1.00     F(X)=  -3.143000
                X=  0.53     F(X)=   3.175082
                X= -0.55     F(X)=   1.331294
                X=  1.02     F(X)=  -1.030560
```

3.3. Prema algoritmu iz primera 1.6 (poglavlje 1.1) napisati program za izračunavanje i tabeliranje funkcije $f(x) = sh\ x$ za $x = 0.1(0.1)0.8$. Funkcijskim naredbama definisati u_o i $\phi(n, x)$.

Rešenje: Program i izlazna lista imaju oblik:

```
      DIMENSION F(8)
      U0(X)=X
      FI(N,X)=X*X/FLOAT((2*N+1)*2*N)
      DO 10 I=1,8
      X=0.1*FLOAT(I)
      U=U0(X)
      S=0.
      N=1
    5 S=S+U
      U=FI(N,X)*U
      N=N+1
      IF(ABS(U).GT.1.E-6) GO TO 5
   10 F(I)=S
      WRITE(5,15) (I,F(I),I=1,8)
   15 FORMAT(5X,'TABELA VREDNOSTI FUNKCIJE SH(X)'///
     114X,'X',6X,'SH(X)'//(13X,'0.',I1,F12.6))
      CALL EXIT
      END
```

```
TABELA VREDNOSTI FUNKCIJE SH(X)

         X          SH(X)

        0.1        0.100167
        0.2        0.201336
        0.3        0.304520
        0.4        0.410752
        0.5        0.521095
        0.6        0.636654
        0.7        0.758584
        0.8        0.888106
```

3.4. Prema algoritmu iz primera 1.11 (poglavlje 1.1) napisati program za određivanje svih prostih brojeva manjih od 1000.

Rešenje: Program za određivanje prostih brojeva zaključno do 999 i njihovo prebrojavanje ima oblik:

```
C     PROGRAM ZA ODREDJIVANJE PROSTIH BROJEVA
      DIMENSION L(500)
      NN=999
      L(1)=1
      L(2)=2
      L(3)=3
      N=3
      DO 15 I=5,NN,2
      K=3
    5 IF(L(K)-I/L(K-1))1,1,2
    2 N=N+1
      L(N)=I
      GO TO 15
    1 J=I/L(K)
      M=I-J*L(K)
      IF(M)15,15,4
    4 K=K+1
      GO TO 5
   15 CONTINUE
      WRITE(5,28)N,(L(I),I=1,N)
   28 FORMAT(1H1,20X,'TABELA PRVIH' I5,' PROSTIH BROJEVA'// (6I11))
      END
```

TABELA PRVIH 169 PROSTIH BROJEVA

1	2	3	5	7	11
13	17	19	23	29	31
37	41	43	47	53	59
61	67	71	73	79	83
89	97	101	103	107	109
113	127	131	137	139	149
151	157	163	167	173	179
181	191	193	197	199	211
223	227	229	233	239	241
251	257	263	269	271	277
281	283	293	307	311	313

317	331	337	347	349	353
359	367	373	379	383	389
397	401	409	419	421	431
433	439	443	449	457	461
463	467	479	487	491	499
503	509	521	523	541	547
557	563	569	571	577	587
593	599	601	607	613	617
619	631	641	643	647	653
659	661	673	677	683	691
701	709	719	727	733	739
743	751	757	761	769	773
787	797	809	811	821	823
827	829	839	853	857	859
863	877	881	883	887	907
911	919	929	937	941	947
953	967	971	977	983	991
997					

3.5. Nacrtati blok šemu algoritma i napisati program za izračunavanje rastojanja i direkcionog ugla, ako su koordinate tačaka A (x_1, y_1) i B (x_2, y_2) u državnom koordinatnom sistemu date na po jednoj kartici u formatu 4F8.3.

Kada u čitaču nema više kartica završiti program.

Računati po formulama:

$$\Delta x = x_2 - x_1, \quad \Delta y = y_2 - y_1, \quad R = \sqrt{(\Delta x)^2 + (\Delta y)^2},$$

$$\phi = \begin{cases} \operatorname{arctg} \dfrac{\Delta y}{\Delta x} & (\Delta x > 0, \Delta y \geqslant 0), \\ \pi + \operatorname{arctg} \dfrac{\Delta y}{\Delta x} & (\Delta x < 0), \\ 2\pi + \operatorname{arctg} \dfrac{\Delta y}{\Delta x} & (\Delta x > 0, \Delta y < 0), \\ \pi/2 & (\Delta x = 0, \Delta y > 0), \\ 3\pi/2 & (\Delta x = 0, \Delta y < 0). \end{cases}$$

Slučaj $\Delta x = \Delta y = 0$ ne razmatrati. Slučaj $|\Delta x| < 10^{-6}$ tretirati kao $\Delta x = 0$. Ugao dobijen u radijane pretvoriti u stepene, minute i sekunde.

Rezultate štampati u obliku:

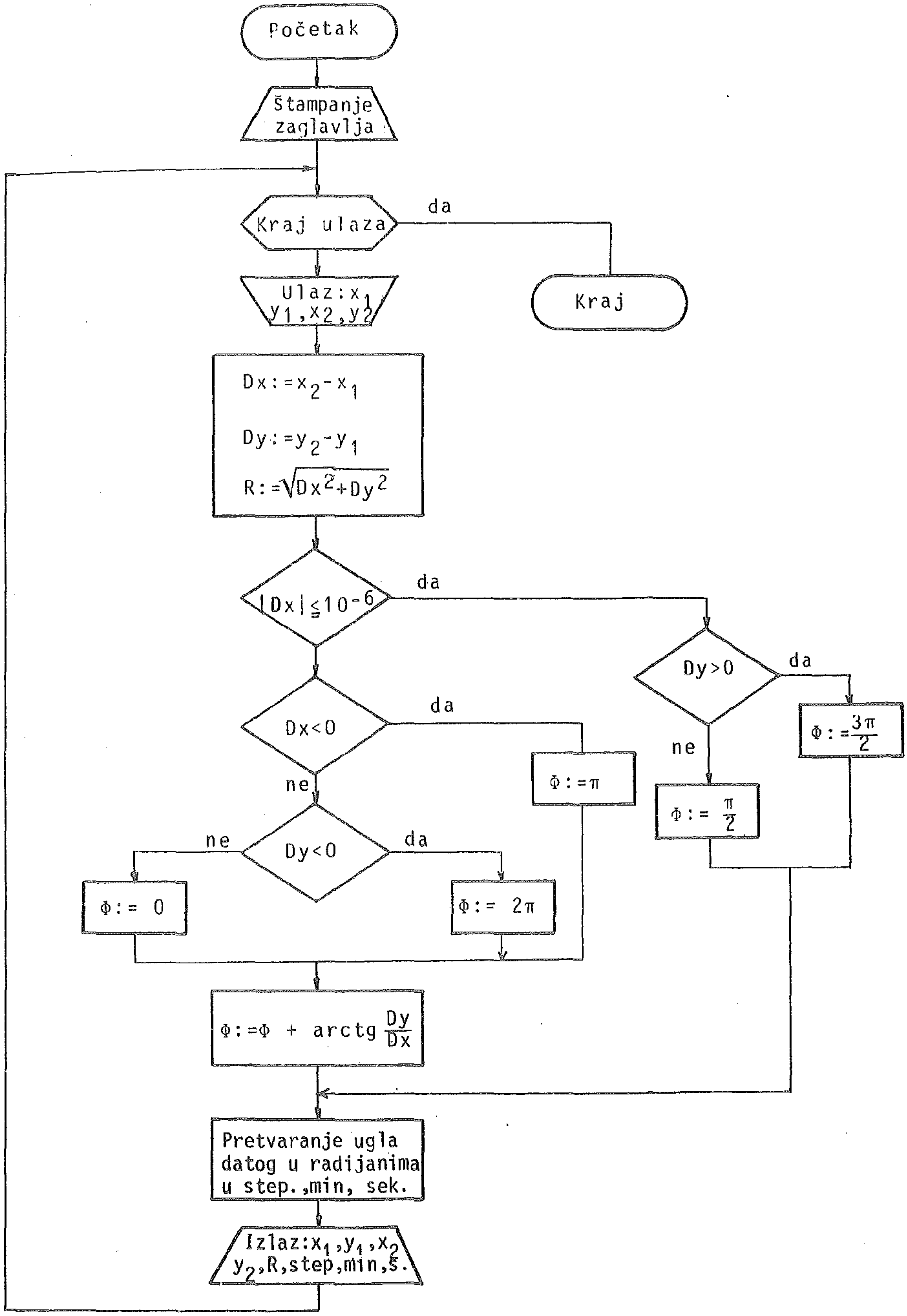
X1/Y1	X2/Y2	R	STEP	MIN	SEC
XXXX.XXX	XXXX.XXX				
XXXX.XXX	XXXX.XXX	XXXXX.XXX	XXX	XX	XX

Rešenje: Prema blok šemi sa sl. 3.2 sastavljen je sledeći program:

```
C====================================================================
C  IZRACUNAVANJE RASTOJANJA I DIREKCIONOG UGLA
C====================================================================
      PI=3.14159265
C      STAMPANJE ZAGLAVLJA
      WRITE(5,20)
   20 FORMAT(1H1//3X,'X1/Y1',5X,'X2/Y2',8X,'R',5X,
     1'STEP MIN SEC'/ )
C    UCITAVANJE KOORDINATA
    7 READ(8,10,END=99)X1,Y1,X2,Y2
   10 FORMAT(4F8.3)
      DX=X2-X1
      DY=Y2-Y1
      R=SQRT(DX*DX+DY*DY)
      IF(ABS(DX).LT.1.E-6)GO TO 1
      IF(DX.LT.0)GO TO 2
      IF(DY.LT.0.)GO TO 3
      FI=0.
      GO TO 4
    2 FI=PI
      GO TO 4
    3 FI=2.*PI
    4 FI=FI+ATAN(DY/DX)
      GO TO 5
    1 IF(DY.LT.0.)GO TO 6
      FI=PI/2.
      GO TO 5
    6 FI=3.*PI/2.
    5 STEP=FI*180./PI
      ISTEP=STEP
      AMIN=(STEP-ISTEP)*60.
      MIN=AMIN
      SEC=(AMIN-MIN)*60.
      ISEC=SEC
      WRITE(5,30) X1,X2,Y1,Y2,R,ISTEP,MIN,ISEC
   30 FORMAT(1X,F8.3,2X,F8.3/1X,F8.3,2X,F8.3,2X,
     1F9.3,2X,I3,2X,I2,2X,I2/)
      GO TO 7
   99 CALL EXIT
      END
```

Primenom ovog programa dobijeni su sledeći rezultati:

X1/Y1	X2/Y2	R	STEP	MIN	SEC
1.500	3.000				
1.000	3.000	2.500	53	7	48
-3.000	2.000				
2.000	5.000	5.831	30	57	49
-2.000	-8.000				
1.500	6.500	7.810	140	11	39
-3.000	-11.000				
-5.000	-24.000	20.616	247	9	58
2.000	7.770				
-3.000	-8.880	8.238	314	27	32


Početak
Štampanje zaglavlja
Kraj ulaza
da
Kraj
Ulaz: x_1, y_1, x_2, y_2
$Dx:=x_2-x_1$
$Dy:=y_2-y_1$
$R:=\sqrt{Dx^2+Dy^2}$
$|Dx|\leq 10^{-6}$
da
$Dy>0$
da
ne
$\Phi:=\frac{3\pi}{2}$
$\Phi:=\frac{\pi}{2}$
$Dx<0$
da
ne
$\Phi:=\pi$
$Dy<0$
ne
da
$\Phi:=0$
$\Phi:=2\pi$
$\Phi:=\Phi+\mathrm{arctg}\frac{Dy}{Dx}$
Pretvaranje ugla datog u radijanima u step., min, sek.
Izlaz: x_1, y_1, x_2, y_2, R, step, min, s.

Sl. 3.2

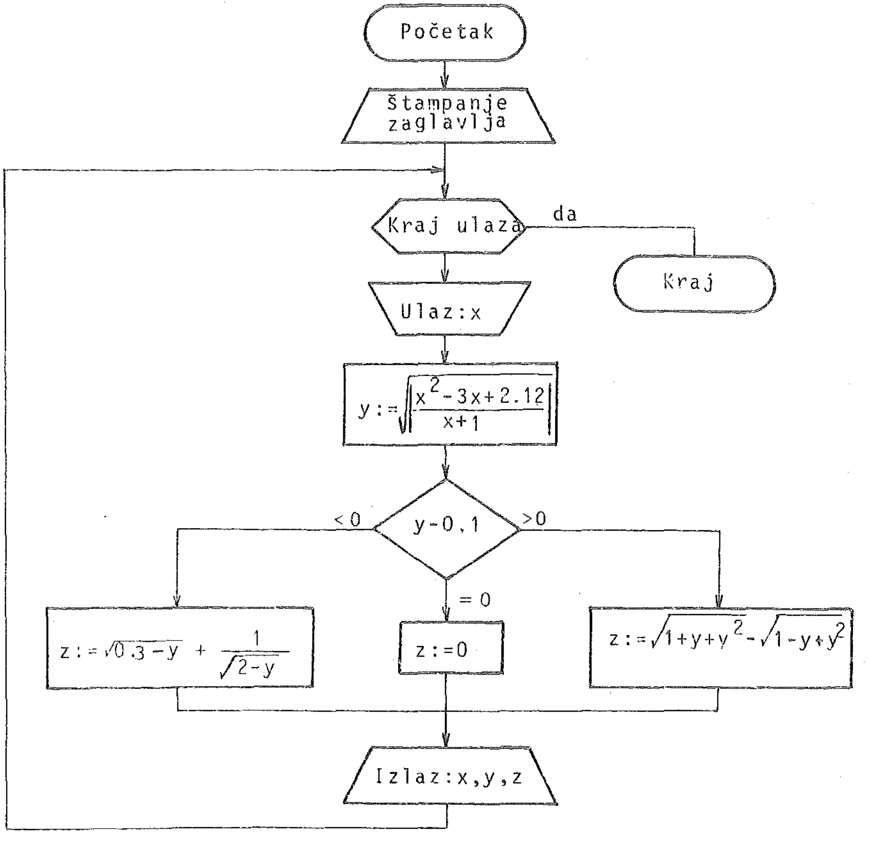
3.6. Nacrtati blok šemu algoritma i napisati program za izračunavanje vrednosti funkcije definisane pomoću

$$z = \begin{cases} \sqrt{0.3 - y} + \dfrac{1}{\sqrt{2 - y}} & (y < 0.1), \\ 0 & (y = 0.1), \\ \sqrt{1 + y + y^2} - \sqrt{1 - y + y^2} & (y > 0.1), \end{cases}$$

gde je $y = \sqrt{\left|\dfrac{x^2 - 3x + 2.12}{x + 1}\right|}$.

Vrednosti za x su date na po jednoj kartici u formatu F8.3.

Rešenje: Blok šema algoritma je data na sl. 3.3.



Sl. 3.3

```
C=======================================================================
C         IZRACUNAVANJE VREDNOSTI FUNKCIJE Z
C=======================================================================
      F(X)=SQRT(ABS((X*X-3.*X+2.12)/(X+1.)))
      WRITE(5,10)
   10 FORMAT(1H1/13X,'X',14X,'Y',14X,'Z'/)
    5 READ(8,20,END=55)X
   20 FORMAT(F8.3)
      Y=F(X)
      IF(Y-0.1)1,2,3
    1 Z=SQRT(0.3-Y)+1./SQRT(2.-Y)
      GO TO 4
    2 Z=0
      GO TO 4
    3 Z=SQRT(1.+Y+Y*Y)-SQRT(1-Y+Y*Y)
    4 WRITE(5,30)X,Y,Z
   30 FORMAT(6X,3(1X,E14.7))
      GO TO 5
   55 CALL EXIT
      END
```

Primenom programa, koji je napisan prema prethodnom algoritmu, na određeni skup ulaznih podataka dobijeni su sledeći rezultati:

X	Y	Z
0.8300000E+01	0.2226671E+01	0.9292163E+00
-0.2300000E+01	0.3317784E+01	0.9669118E+00
0.2500000E+02	0.4608187E+01	0.9826026E+00
0.1300000E+02	0.3071993E+01	0.9615967E+00
0.1420000E+02	0.3256168E+01	0.9656868E+00

(1)E 3.7. Napisati program za tabeliranje vrednosti funkcije

$$y = f(f(f(x))) \qquad (f(x) = 2x^2 + \frac{1}{1-x})$$

za $x = 0.1\ (0.1)\ 0.9$.

Rešenje: Program i izlazna lista imaju sledeći oblik:

```
C=======================================================================
C         IZRACUNAVANJE VREDNOSTI FUNKCIJE Y
C=======================================================================
      F(X)=2.*X*X + 1./(1.-X)
      WRITE(5,1)
    1 FORMAT(1H1/11X,'X',11X,'Y' / )
      DO 10 I=1,9
      X=0.1*I
      Y=F(X)
      Y=F(Y)
      Y=F(Y)
      WRITE(5,2)X,Y
    2 FORMAT(10X,F3.1,4X,E14.7)
   10 CONTINUE
      CALL EXIT
      END
```

```
        X              Y

       0.1         0.5154011E+02
       0.2         0.2545556E+01
       0.3         0.2455243E+02
       0.4         0.9450357E+02
       0.5         0.2799632E+03
       0.6         0.8230203E+03
       0.7         0.2724354E+04
       0.8         0.1238341E+05
       0.9         0.1457512E+06
```

3 + 3.8. Napisati program za tabeliranje vrednosti

$$
y = \begin{cases} a + bx + cx^2 & (a = -1), \\ x^2 \sin^2 x & (a = 0), \\ \sqrt{a + bx} & (a = 1), \\ a \log_e |x| & (a = 2), \\ \frac{1}{4} ax^4 + \frac{1}{2} bx^2 & (a = 3) \end{cases}
$$

ako su vrednosti za a, b, c, x date na karticama u formatu (I2,3F8.2).

Kada u čitaču nema više kartica sa podacima završiti program. Za vrednosti a izvan predviđenog opsega štampati poruku IZVAN OPSEGA.

Rešenje.

```
C=====================================================================
C        IZRACUNAVANJE VREDNOSTI Y
C=====================================================================
         INTEGER A
C         STAMPANJE ZAGLAVLJA
         WRITE(5,40)
      40 FORMAT(1H1/3X,'A',6X,'B',8X,'C',8X,'X',10X,'Y'/ )
C      UCITAVANJE PODATAKA
       1 READ(8,10,END=13)A,B,C,X
      10 FORMAT(I2,3F8.2)
         IF(A.GT.3)GO TO 2
         IF(A.LT.-1)GO TO 2
         I=A+2
         GO TO(3,4,5,6,7),I
       3 Y=A+B*X+C*X*X
         GO TO 8
       4 Y=(X*SIN(X))**2
         GO TO 8
       5 Y=SQRT(A+B*X)
         GO TO 8
       6 Y=A*ALOG(ABS(X))
         GO TO 8
       7 Y=A*X**4/4+B*X*X/2
```

```
 8 WRITE(5,20)A,B,C,X,Y
20 FORMAT(2X,I2,3(1X,F8.2),1X,E13.6)
   GO TO 1
 2 WRITE(5,30)A
30 FORMAT(2X,I2,5X,'IZVAN OPSEGA')
   GO TO 1
13 CALL EXIT
   END
```

Za određeni skup ulaznih podataka dobijeni su sledeći rezultati:

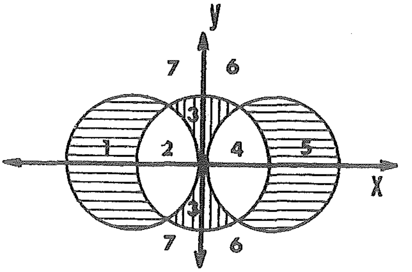
```
 A      B          C          X          Y

-1     1.00       2.00       3.00     0.200000E+02
 0     1.00       2.00       3.00     0.179234E+00
 1     1.00       2.00       3.00     0.200000E+01
 4     IZVAN OPSEGA
 2     1.00       2.00       3.00     0.219722E+01
 3     1.00       2.00       3.00     0.652500E+02
```

3.9. U pravouglom koordinatnom sistemu zadate su tri kružnice, čije su jednačine redom

$$x^2 + y^2 = 1,$$

$$(x-1)^2 + y^2 = 1,$$

$$(x+1)^2 + y^2 = 1.$$



Sl. 3.4

Ove kružnice dele ravan na sedam oblasti prema sl. 3.4.

Napisati potprogram tipa SUBROUTINE koji će određivati oblast u kojoj leže tačke sa koordinatama (x, y).

U glavnom programu učitavati koordinate proizvoljnog broja tačaka zadatih na po jednoj kartici u formatu 2F7.3.

Rešenje: Program i potprogram imaju oblik:

```
C=====================================================================
C        ODREDJIVANJE OBLASTI U KOORDINATNOM SISTEMU
C=====================================================================
      WRITE(5,10)
   10 FORMAT(1H1//10X,'R.BR.',6X,'X',9X,'Y',5X,'OBLAST'/ )
      I=0
   11 READ(8,20,END=77)X,Y
   20 FORMAT(2F7.3)
      I=I+1
      CALL OBLAST(X,Y,J)
      WRITE(5,30)I,X,Y,J
   30 FORMAT(10X,I2,5X,F7.3,3X,F7.3,4X,I1)
      GO TO 11
   77 CALL EXIT
      END
```

U potprogramu OBLAST tački sa koordinatama x i y dodeljuje se kod J, sa vrednostima od 1 do 7, zavisno od položaja tačke u koordinatnoj ravni.

Testiranje programa dalo je sledeće rezultate:

```
      SUBROUTINE OBLAST(X,Y,J)
      A=X*X+Y*Y
      B=(X-1.)**2+Y*Y
      C=(X+1.)**2+Y*Y
      IF(A.GT.1.)GO TO 1567
      IF(B.LE.1.)GO TO 4
      IF(C.LE.1.)GO TO 2
      J=3
      RETURN
    2 J=2
      RETURN
    4 J=4
      RETURN
 1567 IF(B.LE.1.)GO TO 5
      IF(C.LE.1.)GO TO 1
      IF(X.GE.0.)GO TO 6
      J=7
      RETURN
    6 J=6
      RETURN
    5 J=5
      RETURN
    1 J=1
      RETURN
      END
```

R.BR.	X	Y	OBLAST
1	0.000	0.500	3
2	0.750	1.000	6
3	1.500	0.500	5
4	-1.500	0.250	1
5	-1.000	-0.500	1
6	1.000	3.000	6
7	-1.000	-3.000	7
8	-1.000	2.000	7
9	1.000	-1.800	6

1 **3.10.** Sastaviti program za izračunavanje koeficijenta korelacije slučajnih veličina $X=(x_1,\dots,x_n)$ i $Y=(y_1,\dots,y_n)$ po formuli

$$r=\frac{n\left(\sum_{i=1}^{n} x_i y_i\right)-\left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)}{\sqrt{\left(n\sum_{i=1}^{n} x_i^2-\left(\sum_{i=1}^{n} x_i\right)^2\right)\left(n\sum_{i=1}^{n} y_i^2-\left(\sum_{i=1}^{n} y_i\right)^2\right)}}$$

Ulazni podaci su raspoređeni na sledeći način: Na prvoj kartici n u formatu I2, a na ostalim karticama $x_1,\dots,x_n;y_1,\dots,y_n$ u formatu 8F10.3.

Rešenje: Program za izračunavanje koeficijenta korelacije ima oblik:

```
C  PROGRAM ZA IZRACUNAVANJE KOEFICIJENTA KORELACIJE
C=================================================
      DIMENSION X(100),Y(100)
      READ(8,10) N
   10 FORMAT(I2)
      READ(8,11)   (X(I),I=1,N),(Y(I),I=1,N)
   11 FORMAT(8F10.3)
      SX=0.
      SY=0.
      SX2=0.
```

```
      SY2=0.
      SXY=0.
      DO 15 I=1,N
      SX=SX+X(I)
      SY=SY+Y(I)
      SX2=SX2+X(I)*X(I)
      SY2=SY2+Y(I)*Y(I)
   15 SXY=SXY+X(I)*Y(I)
      A=N
      R=(A*SXY-SX*SY)/SQRT((A*SX2-SX*SX)*(A*SY2-
     1 SY*SY))
      WRITE(5,20) R
   20 FORMAT(1H1,'KOEFICIJENT KORELACIJE JE',F9.5)
      CALL EXIT
      END
```

3.11. Neka su date vrednosti x_i $(i = 1, \ldots, 100)$ na deset kartica u formatu 10F8.3. Sastaviti program za izračunavanje vrednosti

$$y_n = \sqrt{\frac{\sum\limits_{i=0}^{k} (x_{n+i} - x_{n-i})^2}{(k+1)(k+2)}} \qquad (n = 1, \ldots, 100),$$

gde je

$$k = \begin{cases} n-1 & \text{ako je } 1 \leqslant n \leqslant 11, \\ 10 & \text{ako je } 11 < n < 90, \\ 100-n & \text{ako je } 90 \leqslant n \leqslant 100. \end{cases}$$

Rezultat štampati prema sledećoj listi:

VREDNOSTI X

XXXX.XXX XXXX.XXX XXXX.XXX XXXX.XXX XXXX.XXX
⋮

VREDNOSTI Y

±X.XXXE±XX ±X.XXXE±XX ±X.XXXE±XX ±X.XXXE±XX ±X.XXXE±XX
⋮

Rešenje:

```
C===============================================================
C      IZRACUNAVANJE VELICINE YN
C===============================================================
      DIMENSION X(100),Y(100)
      READ(8,10)X
   10 FORMAT(10F8.3)
      DO 11 N=1,100
      IF(N.LE.11)GO TO 1
      IF(N.GE.90)GO TO 2
      K=10
      GO TO 3
```

```
  1 K=N-1
    GO TO 3
  2 K=100-N
  3 I=0
    S=0
  4 S=S+(X(N+I)-X(N-I))**2
    IF(I.GE.K)GO TO 5
    I=I+1
    GO TO 4
  5 Y(N)=SQRT(S/(K+1.)/(K+2))
 11 CONTINUE
    WRITE(5,20)X
    WRITE(5,30)Y
 20 FORMAT(1H1/15X,'V R E D N O S T I   X'//(2X,5(F8.3,2X)))
 30 FORMAT(1H0/15X,'V R E D N O S T I   Y'//(2X,5(F8.3,2X)))
    CALL EXIT
    END
```

3.12. Sastaviti program za izračunavanje vrednosti

$$F(x,y)=\begin{cases}\sum\limits_{k=0}^{17}\dfrac{\phi(xy)+k\phi(x/y)^2}{1+k^2} & (x^2+y=0),\\[2ex] \dfrac{1}{2}\log_e(1+(x/y)^2) & (x^2+y\neq 0),\end{cases}$$

gde je $\phi(u)=\sqrt{|1+u^3|}$ ·

Za sve kombinacije ulaznih podataka x_i ($i=1,\ldots,n$) i y_i ($i=1,\ldots,m$) izračunati $F(x_i,y_j)$.

Rešenje: Program ima sledeći oblik:

```
C==================================================================
C                 IZRACUNAVANJE VREDNOSTI F
C==================================================================
C
      DIMENSION  X(100),Y(100)
      FI(U)=SQRT(ABS(1.+U**3))
      READ(8,5)N,M
    5 FORMAT(2I2)
      READ(8,10)(X(I),I=1,N),(Y(I),I=1,M)
   10 FORMAT(10F8.3)
      WRITE(5,20)
   20 FORMAT(1H1/22X,'T A B E L A   V R E D N O S T I'//
     119X,'X',12X,'Y',16X,'F(X,Y)'/)
      DO 11 I=1,N
      DO 11 J=1,M
      IF(X(I)**2+Y(J))1,2,1
    1 F=0.5*ALOG(1.+(X(I)/Y(J))**2)
      GO TO 17
    2 F=0.
```

```
      A=FI(X(I)*Y(J))
      B=FI(X(I)/Y(J))
      B=B*B
      DO 33 L=1,18
      K=L-1
      F=F+(A+K*B)/(1.+K*K)
33    CONTINUE
17    WRITE(5,30)X(I),Y(J),F
30    FORMAT(15X,F8.2,5X,F8.2,9X,E13.6)
11    CONTINUE
      CALL EXIT
      END
```

Za određeni skup ulaznih podataka ($n = 3$, $m = 5$) dobijena je sledeća tabela:

```
                T A B E L A   V R E D N O S T I

        X                Y                F(X,Y)

      1.00             1.00            0.346574E+00
      1.00            -1.00            0.000000E+00
      1.00             2.00            0.111572E+00
      1.00            -2.00            0.111572E+00
      1.00            -4.00            0.303123E-01
      2.00             1.00            0.804719E+00
      2.00            -1.00            0.804719E+00
      2.00             2.00            0.346574E+00
      2.00            -2.00            0.346574E+00
      2.00            -4.00            0.480770E+02
      3.00             1.00            0.115129E+01
      3.00            -1.00            0.115129E+01
      3.00             2.00            0.589328E+00
      3.00            -2.00            0.589328E+00
      3.00            -4.00            0.223144E+00
```

3.13. Napisati program za izračunavanje sume $S = \sum_{i=1}^{30} (a_i - b_i)^2$, gde su a_i i b_i dati pomoću

$$a_i = \begin{cases} i & (i \text{ neparno}) \\ i/2 & (i \text{ parno}) \end{cases} \qquad b_i = \begin{cases} i^2 & (i \text{ neparno}), \\ i^3 & (i \text{ parno}). \end{cases}$$

Rešenje: Program i izlazna lista, u ovom slučaju, imaju oblik:

```
C=========================================
C=========================================
C=========================================
      S=0.
C   STAMPANJE NASLOVA
      WRITE(5,10)
   10 FORMAT(1H1// 11X,'I',7X,'A(I)',
     14X,'B(I)' / )
      DO 11 I=1,30
C   ISPITIVANJE PARNOSTI
      IF(I/2*2.EQ.I)GO TO 2
      A=I
```

```
      B=I*I
      GO TO 3
    2 A=FLOAT(I)/2.
      B=I**3
C   STAMPANJE I,A,B
    3 WRITE(5,20)I,A,B
   20 FORMAT(10X,I2,7X,F4.1,1X,F7.1)
C   SUMIRANJE
      S=S+(A-B)**2
   11 CONTINUE
C   STAMPANJE VREDNOSTI SUME
      WRITE(5,30)S
   30 FORMAT(1H0,10X,'S =   ',E14.7)
      CALL EXIT
      END
```

I	A(I)	B(I)
1	1.0	1.0
2	1.0	8.0
3	3.0	9.0
4	2.0	64.0
5	5.0	25.0
6	3.0	216.0
7	7.0	49.0
8	4.0	512.0
9	9.0	81.0
10	5.0	1000.0
11	11.0	121.0
12	6.0	1728.0
13	13.0	169.0
14	7.0	2744.0
15	15.0	225.0
16	8.0	4096.0
17	17.0	289.0
18	9.0	5832.0
19	19.0	361.0
20	10.0	8000.0
21	21.0	441.0
22	11.0	10648.0
23	23.0	529.0
24	12.0	13824.0
25	25.0	625.0
26	13.0	17576.0
27	27.0	729.0
28	14.0	21952.0
29	29.0	841.0
30	15.0	27000.0

S =   0.1950279E+10

3.14. Napisati program za izračunavanje vrednosti $w = \sqrt[3]{z}$ za $z = 0.5\ (0.5)\ 15.0$, primenom iterativne formule

$$w_{n+1} = w_n + \frac{1}{3}\left(\frac{z}{w_n^2} - w_n\right) \qquad (n = 0, 1, \ldots).$$

Početnu vrednost w_o odrediti po formuli

$$w_o = \begin{cases} z/3 & (z \geqslant 1), \\ z & (z < 1). \end{cases}$$

Kada je zadovoljen uslov $|w_{n+1} - w_n| \leqslant 10^{-5}$ iterativni proces prekinuti i za rezultat uzeti w_{n+1}.

Rešenje: Program i deo izlazne liste imaju sledeći oblik:

```
C================================================================
C                IZRACUNAVANJE   Z**(1./3.)
C================================================================
C
      WRITE(5,10)
   10 FORMAT(1H1//9X,'IZRACUNAVANJE KUBNOG KORENA'//16X,'Z',
     *11X,'W'/)
      DO 11 I=5,150,5
      Z=0.1*I
      IF(Z.GE.1) GO TO 2
      W0=Z
      GO TO 3
    2 W0=Z/3.
    3 W1=W0+1./3.*(Z/W0**2-W0)
      IF(ABS(W1-W0).LE.1.E-5) GO TO 4
      W0=W1
      GO TO 3
    4 WRITE(5,20) Z,W1
   20 FORMAT(14X,F4.1,6X,F9.6)
   11 CONTINUE
      CALL EXIT
      END
```

```
                    IZRACUNAVANJE KUBNOG KORENA

                            Z           W

                           0.5       0.793701
                           1.0       1.000000
                           1.5       1.144714
                           2.0       1.259921
                           2.5       1.357209
                           3.0       1.442250
                           3.5       1.518294
                           4.0       1.587401
                           4.5       1.650964
                           5.0       1.709976
                           5.5       1.765174
                           6.0       1.817121
                           6.5       1.866256
                           7.0       1.912931
                           7.5       1.957434
                           8.0       2.000000
                           8.5       2.040828
                           9.0       2.080084
                           9.5       2.117912
                          10.0       2.154435
```

```
10.5        2.189759
11.0        2.223980
11.5        2.257179
12.0        2.289428
12.5        2.320794
13.0        2.351335
13.5        2.381102
14.0        2.410142
14.5        2.438499
15.0        2.466212
```

(3) 3.15. Sastaviti program za izračunavanje vrednosti sume

$$S_i = \sum_{k=1}^{n} \frac{\sqrt{1+x^2} + \sqrt{|1+xy|}}{k^2 + 3.5},$$

gde su $x = u_1$, $y = u_2$, a u_1 i u_2 realni koreni kvadratne jednačine

$$a_i u^2 + b_i u + c_i = 0 \qquad (a_i \neq 0).$$

Ako je $b_i^2 - 4a_i c_i < 0$ uzeti $u_1 = u_2 = 0$. Brojevi a_i, b_i, c_i $(i = 1, \ldots, n)$ su dati na karticama (po jedna trojka na kartici) u formatu 3F5.2. Broj n je dat na prvoj kartici u formatu I2.

Za rešavanje kvadratne jednačine obrazovati potprogram tipa SUBROUTINE.

Rešenje. U potprogramu QVAD (A, B, C, U1, U2) za dato A, B, C određuju se U1 (= x) i U2 (= y), prema zadatom kriterijumu.

```
      SUBROUTINE QVAD(A,B,C,U1,U2)
      D=B*B-4.*A*C
      IF(D.LT.0.)GO TO 1
      D=SQRT(D)
      U1=(-B+D)/(2.*A)
      U2=(-B-D)/(2.*A)
      RETURN
    1 U1=0.
      U2=0.
      RETURN
      END
```

Glavni program ima sledeći oblik:

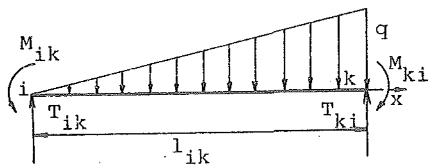
```
C=================================================================
C      IZRACUNAVANJE VREDNOSTI SI
C=================================================================
C    STAMPANJE ZAGLAVLJA
      WRITE(5,10)
   10 FORMAT(1H1/2X,'I    A      B      C      X         Y',10X,'S'
     1/ )
C    UCITAVANJE ULAZNIH PODARAKA A,B,C
      READ(8,20)N
   20 FORMAT(I2)
      I=0
```

```
   41 READ(8,30)A,B,C
   30 FORMAT(3F5.2)
      I=I+1
C     RESAVANJE KVADRATNE JEDNACINE
      CALL QVAD(A,B,C,X,Y)
C     IZRACUNAVANJE SUME S
      S=0.
      DO 11 K=1,20
      DS=(SQRT(1.+X*X)+SQRT(ABS(1.+X*Y)))/(K*K+3.5)
   11 S=S+DS
C     STAMPANJE REZULTATA
      WRITE(5,40)I,A,B,C,X,Y,S
   40 FORMAT(1X,I2,3(1X,F5.2),2(1X,F6.3),1X,E13.6)
      IF(I.LT.N)GO TO 41
      CALL EXIT
      END
```

Za određeni skup ulaznih podataka dobijena je sledeća tabela:

I	A	B	C	X	Y	S
1	1.00	2.00	3.00	0.000	0.000	0.129629E+01
2	3.00	2.00	1.00	0.000	0.000	0.129629E+01
3	1.00	3.00	2.00	-1.000	-2.000	0.203924E+01
4	5.50	8.20	1.00	-0.134	-1.357	0.135855E+01
5	-1.10	-3.30	2.00	-3.517	0.517	0.295614E+01
6	-2.20	3.00	-3.00	0.000	0.000	0.129629E+01

3.16. Za date granične uslove $M_{ik} = 12\ Mpm$, $M_{ki} = 6\ Mpm$ i opterećenje kao na sl. 3.5, napisati program za određivanje maksimalne vrednosti momenta savijanja, kao i presek u kome se on javlja. Za korak uzeti $\Delta x = 0.01\ m$. Za proračun koristiti formule



Sl. 3.5

$$M(x) = -M_{ik} - \frac{q}{l_{ik}} \cdot \frac{x^3}{6} + T_{ik}x, \qquad T_{ik} = \frac{1}{6} q l_{ik} + \frac{M_{ik} - M_{ki}}{l_{ik}} .$$

Ulazne podatke $M_{ik} = 12\ Mpm$, $M_{ki} = 6\ Mpm$, $l_{ik} = 6\ m$, $q = 2\ Mp/m$ učitati u formatu 4F5.2.

Izlaznu listu štampati u obliku:

MAKSIMALNI MOMENT M = ±XX.XXXX U PRESEKU X = ±X.XXXX

Rešenje: Program i izlazna lista imaju oblik:

```
C=================================================================
C        IZRACUNAVANJE MAKSIMALNOG MOMENTA SAVIJANJA
C=================================================================
C
      REAL MIK,MKI,LIK,MMAX,M,LMAX
      READ(8,10)MIK,MKI,LIK,Q
   10 FORMAT(4F5.2)
      MMAX=-1.E5
      TIK=Q*LIK/6+(MIK-MKI)/LIK
      X=0.
    1 X=X+0.01
      M=-MIK-Q*X**3/(6*LIK)+TIK*X
      IF(M.LT.MMAX)GO TO 2
      MMAX=M
      LMAX=X
    2 IF(X.GE.LIK)GO TO 3
      GO TO 1
    3 WRITE(5,20)MMAX,LMAX
   20 FORMAT(4X,'MAKSIMALNI MOMENT   M =   ',F8.4,
     1'   U PRESEKU   X = ',F7.4)
      CALL EXIT
      END

MAKSIMALNI MOMENT   M =   -3.5147   U PRESEKU   X =   4.2400
```

3.17. Sastaviti program za tabeliranje vrednosti koeficijenta α za izračunavanje torzione konstante pravougaonog preseka

$$\alpha = \frac{1}{3}\cdot\frac{b}{a} - \frac{64}{\pi^5}\sum_{n=o}^{+\infty}\frac{th\dfrac{(2n+1)\pi b}{2a}}{(2n+1)^5},$$

gde je a manja stranica pravougaonika, za $b/a = 1\ (0.1)\ 10$.

Proces sumiranja prekinuti kada opšti član reda postane manji od 10^{-3}.
za svako α

Rešenje.

```
C=================================================================
C   IZRACUNAVANJE KOEFICIJENTA ALFA ZA TORZIONU KONS.
C=================================================================
      DIMENSION ALFA(10)
      WRITE(5,10)(I,I=1,9)
   10 FORMAT(1H1,4X,'VREDNOSTI ALFA ZA IZRACUNAVANJE'
     1,' TORZ.KONST.PRAVOUG.PRES.'//6X,'.0',9(4X,'.',I1)/)
      PI=3.14159265
      DO 11 I=1,10
      DO 22 J=1,10
      BA=I+(J-1)*0.1
      S=0
      N=0
    1 A=(2*N+1)*PI*BA/2
      A=(EXP(A)-EXP(-A))/(EXP(A)+EXP(-A))
      A=A/(2*N+1.)**5
```

```
      S=S+A
      IF(ABS(A).LT.1.E-3)GO TO 22
      N=N+1
      GO TO 1
   22 ALFA(J)=8A/3-64/PI**5*S
   11 WRITE(5,20)I,ALFA
   20 FORMAT(1X,I2,10F6.3)
      CALL EXIT
      END
```

$I_t = \alpha a^4$

VREDNOSTI ALFA ZA IZRACUNAVANJE TORZ.KONST.PRAVOUG.PRES.

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9
1	0.141	0.169	0.199	0.230	0.262	0.294	0.326	0.359	0.391	0.424
2	0.457	0.491	0.524	0.557	0.590	0.623	0.657	0.690	0.723	0.757
3	0.790	0.823	0.857	0.890	0.923	0.957	0.990	1.023	1.057	1.090
4	1.123	1.157	1.190	1.223	1.257	1.290	1.323	1.357	1.390	1.423
5	1.457	1.490	1.523	1.557	1.590	1.623	1.657	1.690	1.723	1.757
6	1.790	1.823	1.857	1.890	1.923	1.957	1.990	2.023	2.057	2.090
7	2.123	2.157	2.190	2.223	2.257	2.290	2.323	2.357	2.390	2.423
8	2.457	2.490	2.523	2.557	2.590	2.623	2.657	2.690	2.723	2.757
9	2.790	2.823	2.857	2.890	2.923	2.957	2.990	3.023	3.057	3.090
10	3.123	3.157	3.190	3.223	3.257	3.290	3.323	3.357	3.390	3.423

3.18. Koristeći metod sukcesivnih aproksimacija, sastaviti program za određivanje vrednosti za λ koja zadovoljava uslov

$$\sum_{n=1}^{m} \frac{\alpha_n^2}{\alpha_n^4 - \lambda^4} \leqslant \epsilon \qquad \left(\alpha_n = \frac{n\pi}{L},\ \epsilon = 10^{-3},\ m \leqslant 50\right),$$

uzimajući za $\lambda_{\text{početno}} = \frac{\pi}{L} + \Delta\lambda$ $(\Delta\lambda = 0.01)$ i $L = 5\ (1)\ 10$. Izlaznu listu štampati u obliku:

L	LAM
XX.X	X.XXXXXXXE±XX

Rešenje: Prema blok šemi algoritma sa sl. 3.6 napisan je sledeći program:

```
C=================================================================
C   IZRACUNAVANJE VREDNOSTI LAMDA
C=================================================================
      REAL LAM
      PI=3.1415926
C   STAMPANJE ZAGLAVLJA
      WRITE(5,10)
   10 FORMAT(1H1/5X,'L',8X,'LAM'/ )
C   PETLJA ZA PROMENU L
```

```
      DLAM=0.01
      DO 11 L=5,10
      LAM=PI/L
   16 LAM=LAM+DLAM
      SIGMA=0.
C     PETLJA ZA IZRACUNAVANJE SUMA
      DO 22 N=1,50
      ALFA=N*PI/L
      SIGMA=SIGMA+ALFA*ALFA/(ALFA**4-LAM**4)
      IF(SIGMA.LE.1.E-3)GO TO 15
   22 CONTINUE
      GO TO 16
C    STAMPANJE REZULTATA
   15 WRITE(5,20)L,LAM
   20 FORMAT(4X,I2,3X,E14.7)
   11 CONTINUE
      CALL EXIT
      END
```

```
      L         LAM

      5     0.6383185E+00
      6     0.5335987E+00
      7     0.4587989E+00
      8     0.4026991E+00
      9     0.3590658E+00
     10     0.3241592E+00
```

3.19. Napisati program za rešavanje jednačine

$$f(x) = 1 - x^{-a} + (1 - x)^{-a} = 0 \qquad (a = 0.5\ (0.1)\ 2.8).$$

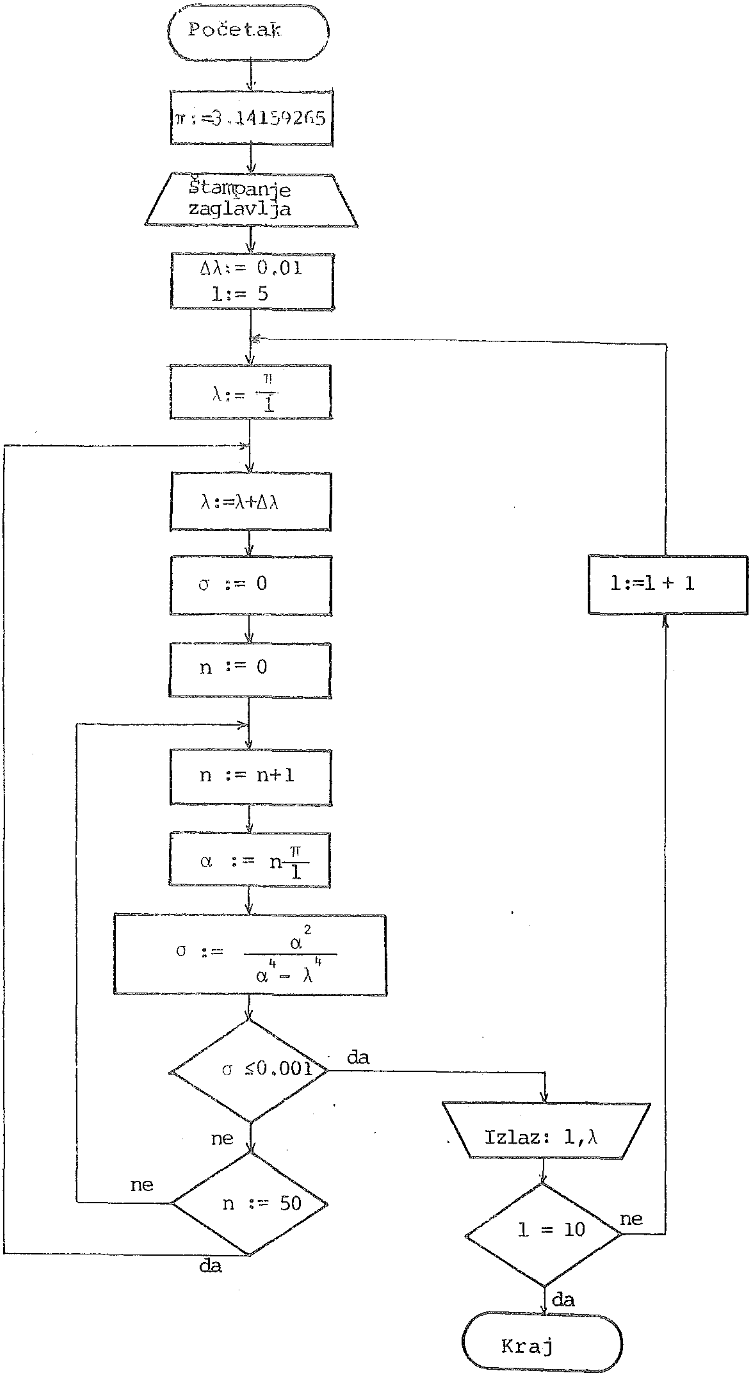
primenom Newtonovog metoda, sa tačnošću $\epsilon = 10^{-5}$. Za početnu aproksimaciju rešenja uzeti $x_o = 0.5$. Na izlazu štampati vrednost parametra a, koren jednačine x i odgovarajuću vrednost $f(x)$ u obliku:

A	X	F (X)
X.X	XX.XXXXXX	XX.XXXXXX
⋮		

Rešenje: Funkciju f i njen izvod f', koji se javljaju u Newtonovom metodu

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})} \qquad (n = 1, 2, \ldots),$$

definisaćemo funkcijskim naredbama. Kriterijum za prekid iterativnog procesa je tačnost ϵ. Naime, smatraćemo da je koren jednačine određen sa tačnošću ϵ, ako $f(x)$ menja znak u intervalu $(x_n - \epsilon, x_n + \epsilon)$. Program ima sledeći oblik:


Početak
π:=3.14159265
Štampanje zaglavlja
Δλ:= 0.01
l:= 5
λ:= π/l
λ:=λ+Δλ
σ := 0
n := 0
n := n+1
α := nπ/l
σ := α²/(α⁴ − λ⁴)
σ ≤0.001
da
ne
n := 50
ne
da
Izlaz: l,λ
l = 10
ne
da
l:=l + 1
Kraj

sl. 3.6

```
C==============================================================
C             RESAVANJE JEDNACINE
C      1 = X**(-A) + (1-X)**(-A) = 0
C            NJUTNOVIM METODOM
C==============================================================
      FUNK(X,A)=1-X**(-A)+(1-X)**(-A)
      PRIZ(X,A)=A*X**(-A-1)+A*(1-X)**(-A-1)
      WRITE(5,10)
   10 FORMAT(1H1//10X,'A',10X,'X',12X,'F(X)'/)
      EPS=1.E-5
      DO 11  I=5,28
      A=I*0.1
      X0=0.5
    6 X=X0-FUNK(X0,A)/PRIZ(X0,A)
      IF(FUNK(X+EPS,A)*FUNK(X-EPS,A).LT.0.) GO TO 7
      X0=X
      GO TO 6
    7 Y=FUNK(X,A)
      WRITE(5,20)A,X,Y
   20 FORMAT(9X,F3.1,5X,F9.6,5X,F9.6)
   11 CONTINUE
      CALL EXIT
      END
```

dok je odgovarajuća izlazna lista:

A	X	F(X)
0.5	0.219949	-0.000015
0.6	0.267609	0.000000
0.7	0.305916	-0.000026
0.8	0.336722	-0.000003
0.9	0.361641	-0.000000
1.0	0.381966	-0.000001
1.1	0.398689	0.000000
1.2	0.412563	0.000001
1.3	0.424159	-0.000084
1.4	0.433933	-0.000044
1.5	0.442217	-0.000023
1.6	0.449281	-0.000012
1.7	0.455337	-0.000007
1.8	0.460554	-0.000003
1.9	0.465068	-0.000004
2.0	0.468990	0.000000
2.1	0.472410	-0.000000
2.2	0.475402	0.000000
2.3	0.478029	0.000001
2.4	0.480340	-0.000001
2.5	0.482380	0.000000
2.6	0.484184	0.000000
2.7	0.485784	-0.000001
2.8	0.487205	-0.000004

3.20. Sastaviti program za približno rešavanje sistema jednačina

$$F(x, y) = 0,$$

$$G(x, y) = 0,$$

gde su F i G neprekidno-diferencijabilne funkcije, primenom Newton–Raphsonovog metoda

$$
(1) \qquad \begin{aligned} x_{n+1} &= x_n - \Delta x_n \\ y_{n+1} &= y_n - \Delta y_n \end{aligned} \qquad (n = 0, 1, 2, \ldots)
$$

polazeći od nekih približnih vrednosti x_o i y_o, pri čemu su

$$
J(x_n, y_n) = \begin{vmatrix} F'_x(x_n, y_n) & F'_y(x_n, y_n) \\ G'_n(x_n, y_n) & G'_y(x_n, y_n) \end{vmatrix} \neq 0,
$$

$$
\Delta x_n = \frac{1}{J(x_n, y_n)} \begin{vmatrix} F(x_n, y_n) & F'_y(x_n, y_n) \\ G(x_n, y_n) & G'_y(x_n, y_n) \end{vmatrix},
$$

$$
\Delta y_n = \frac{1}{J(x_n, y_n)} \begin{vmatrix} F'_x(x_n, y_n) & F(x_n, y_n) \\ G'_x(x_n, y_n) & G(x_n, y_n) \end{vmatrix}.
$$

Parcijalne izvode F'_x, F'_y, G'_x, G'_y izračunavati numerički.

Iterativni postupak (1) prekinuti kada budu istovremeno ispunjeni uslovi

$$
|x_{n+1} - x_n| \leqslant \epsilon \quad \text{i} \quad |y_{n+1} - y_n| \leqslant \epsilon,
$$

gde je ϵ zadata tačnost.

Za proveru uzeti primer

$$
F(x, y) \equiv 2x^3 - y^2 - 1 = 0,
$$

$$
G(x, y) \equiv xy^3 - y - 4 = 0,
$$

sa $x_o = 1.2$, $y_o = 1.7$ i $\epsilon = 10^{-10}$.

Rešenje: Za izračunavanje parcijalnih izvoda funkcije $f(x, y)$ koristimo sledeće približne izraze

$$
\frac{\partial f}{\partial x} \cong \frac{f(x+h, y) - f(x-h, y)}{2h},
$$

$$
\frac{\partial f}{\partial y} = \frac{f(x, y+h) - f(x, y-h)}{2h},
$$

gde je h dovoljno mali priraštaj (ovde je uzeto $h = 10^{-5}$).

Funkcije F i G zadate su u potprogramu tipa FUNCTION.

S obzirom da se zahteva tačnost 10^{-10}, program ćemo realizovati u dvostrukoj tačnosti.

```
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION R(2)
      READ(8,15) XP,YP,EPS,H
   15 FORMAT(4D10.0)
      WRITE(5,16)
   16 FORMAT(1H1,25X,'NEWTON-RAPHSON-OV METOD ZA RESAVANJE',
     1' SISTEMA JEDNACINA'///41X,'2.*X**3-Y**2-1.=0.'//
     245X,'X*Y**3-Y-4.=0.'///16X,'BR.ITER.',6X,'X',
     315X,'Y',12X,'F(X,Y)',10X,'G(X,Y)'/)
      ITER=0
   20 ITER=ITER+1
      CALL NEWT(XP,YP,XK,YK,FD,H,A,B)
      IF(FD) 5,6,5
    6 WRITE(5,17)
   17 FORMAT(/15X,'JACOBI-EVA MATRICA SINGULARNA')
      GO TO 90
    5 DO 8 I=1,2
    8 R(I)=EEFF(I,XK,YK)
      WRITE(5,18) ITER,XK,YK,(R(J),J=1,2)
   18 FORMAT(15X,I5,4F16.10)
      IF(DABS(A)-EPS) 30,30,40
   30 IF(DABS(B)-EPS) 90,90,40
   40 XP=XK
      YP=YK
      GO TO 20
   90 CALL EXIT
      END

      SUBROUTINE NEWT(XP,YP,XK,YK,FD,H,A,B)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION R(2),DX(2),DY(2)
      X=XP
      Y=YP
      DO 4 I=1,2
      R(I)=EEFF(I,X,Y)
      DX(I)=0.5D0/H*(EEFF(I,X+H,Y)-EEFF(I,X-H,Y))
    4 DY(I)=0.5D0/H*(EEFF(I,X,Y+H)-EEFF(I,X,Y-H))
      FD=DX(1)*DY(2)-DY(1)*DX(2)
      IF(FD) 5,6,5
    5 A=(R(1)*DY(2)-R(2)*DY(1))/FD
      B=(R(2)*DX(1)-R(1)*DX(2))/FD
      XK=X-A
      YK=Y-B
    6 RETURN
      END

      FUNCTION EEFF(J,X,Y)
      IMPLICIT REAL*8 (A-H,O-Z)
      GO TO(50,60),J
   50 EEFF=2.D0*X**3-Y*Y-1.D0
      RETURN
   60 EEFF=X*Y**3-Y-4.D0
      RETURN
      END
```

```
NEWTON-RAPHSON-OV METOD ZA RESAVANJE SISTEMA JEDNACINA

                    2.*X**3-Y**2-1.=0.

                      X*Y**3-Y-4.=0.

BR.ITER.       X               Y             F(X,Y)          G(X,Y)

   1     1.2348762633   1.6609796808   0.0073200054   -0.0022831784
   2     1.2342746753   1.6615262759   0.0000023823   -0.0000008838
   3     1.2342744841   1.6615264668   0.0000000000   -0.0000000000
   4     1.2342744841   1.6615264668  -0.0000000000   -0.0000000000
```

3.21. Neka jednačina $f(x) = 0$ ima koren $x = a$ u intervalu (α, β), pri čemu je $f(\alpha) f(\beta) < 0$. Napisati program za nalaženje korena $x = a$ sa tačnošću ϵ, korišćenjem metoda polovljenja intervala, koji se može iskazati kroz sledeća četiri koraka:

1° $k := 0, \quad x_1 = \alpha, \quad y_1 := \beta;$

2° $k := k + 1, \quad z_k := \frac{1}{2}(x_k + y_k);$

3° Ako je

$$f(z_k) f(x_k) \begin{cases} < 0 & \text{uzeti } x_{k+1} := x_k, \; y_{k+1} := z_k, \\ > 0 & x_{k+1} := z_k, \; y_{k+1} := y_k, \\ = 0 & \text{kraj izračunavanja } a := z_k; \end{cases}$$

4° Ako je

$$|y_{k+1} - x_{k+1}| \begin{cases} \geqslant \epsilon & \text{preći na } 2°, \\ < \epsilon & \text{kraj izračunavanja } a := z_{k+1}. \end{cases}$$

Na izlazu štampati: k, (x_k, y_k), $f(z_k)$, za $k = 0, 5, 10, \ldots$ i za poslednje k, pri kome je postignuta zahtevana tačnost. Na kraju ove tabele štampati vrednost izračunatog korena u obliku:

A = ±X.XXXXXXXXXXXXXD±XX (SA TAČNOŠĆU EPS = 0.XD–XX)

Program realizovati u dvostrukoj tačnosti. Za testiranje programa uzeti:

$$f(x) = e^x - 2(x-1)^2, \quad (\alpha, \beta) = (-0.5, 1.0), \quad \epsilon = 10^{-12}.$$

Rešenje: Program i izlazna lista imaju oblik:

```
C=====================================================================
C                    METOD POLOVLJENJA INTERVALA
C=====================================================================
      DOUBLE PRECISION X,Y,Z,F,FZ,EPS
      F(X)=DEXP(X)-2.*(X-1.)**2
C     STAMPANJE ZAGLAVLJA
      WRITE(5,9)
    9 FORMAT(1H1//2X,'K',2X,'(',8X,'X(K)',8X,',',8X,'Y(K)',
     18X,')',5X,'F(Z(K))'/)
C     UCITAVANJE INTERVALA   (ALFA,BETA)
      READ(8,10)ALFA,BETA
   10 FORMAT(2F5.0)
      EPS=1.D-12
      K=-1
      X=ALFA
      Y=BETA
    5 K=K+1
      Z=0.5*(X+Y)
      FZ=F(Z)
      IF(K/5*5-K.LT.0)GO TO 25
      WRITE(5,20)K,X,Y,FZ
   20 FORMAT(1X,I2,2X,'(',D20.13,',',D20.13,')',2X,D12.5)
   25 IF(FZ*F(X))1,2,3
    1 Y=Z
      GO TO 4
    2 IF(K/5*5-K.EQ.0) GO TO 6
      GO TO 7
    3 X=Z
    4 IF(DABS(Y-X).GE.EPS)GO TO 5
      Z=0.5*(X+Y)
      K=K+1
      FZ=F(Z)
    7 WRITE(5,20)K,X,Y,FZ
    6 WRITE(5,30)Z,EPS
   30 FORMAT(/5X,'A = ',D20.13,' (SA TACNOSCU EPS = ',D7.1,' )')
      CALL EXIT
      END
```

```
 K  (         X(K)          ,          Y(K)          )      F(Z(K))

 0  (-0.5000000000000D+00, 0.1000000000000D+01)   0.15903D+00
 5  ( 0.2031250000000D+00, 0.2500000000000D+00)   0.57870D-01
10  ( 0.2119140625000D+00, 0.2133789062500D+00)  -0.29038D-02
15  ( 0.2132873535156D+00, 0.2133331298828D+00)   0.70475D-05
20  ( 0.2133073806763D+00, 0.2133088111877D+00)  -0.23607D-05
25  ( 0.2133086323738D+00, 0.2133086770773D+00)   0.89352D-07
30  ( 0.2133086337708D+00, 0.2133086351678D+00)   0.53733D-09
35  ( 0.2133086343383D+00, 0.2133086343820D+00)   0.58801D-10
40  ( 0.2133086343465D+00, 0.2133086343479D+00)   0.19760D-11
41  ( 0.2133086343465D+00, 0.2133086343472D+00)   0.48062D-12

    A =  0.2133086343468D+00 (SA TACNOSCU EPS = 0.1D-11 )
```

3.22. Napisati program za rešavanje jednačine $f(x) = 0$ metodom regula–falsi

$$x_i = \frac{x_o f(x_{i-1}) - x_{i-1} f(x_o)}{f(x_{i-1}) - f(x_o)} \qquad (i = 2, 3, \ldots).$$

Funkciju f zadati funkcijskom naredbom. Iterativni proces prekinuti kada je ispunjen uslov $f(x_i - \epsilon) f(x_i + \epsilon) \leqslant 0$ (uzeti, na primer, $\epsilon = 10^{-5}$). Izlaznu listu štampati u obliku:

I	XI	F (XI)
2	±X.XXXXXXX±YY	±X.XXXXXXXE±YY
⋮		

Za testiranje programa uzeti jednačinu iz prethodnog primera.

Rešenje: U ovom slučaju uzećemo $x_o = -0.5, x_1 = 1$. Program i izlazna lista imaju oblik:

```
C=================================================================
C                 RESAVANJE JEDNACINE
C              EXP(X) - 2*(X-1)**2 = 0
C                 METODOM REGULA-FALSI
C=================================================================
C
      F(X)=EXP(X)-2*(X-1)**2
      WRITE(5,10)
   10 FORMAT(1H1//9X,'I',10X,'XI',14X,'F(XI)'/)
      X0=-0.5
      X1=1
      I=2
    3 X=(X0*F(X1)-X1*F(X0))/(F(X1)-F(X0))
      Y=F(X)
      WRITE(5,20)I,X,Y
   20 FORMAT(8X,I2,5X,E14.7,2X,E14.7)
      IF(F(X-1.E-5)*F(X+1.E-5))1,1,2
    2 X1=X
      I=I+1
      GO TO 3
    1 CALL EXIT
      END
```

```
I          XI                F(XI)

2     0.3833066E+00     0.7065065E+00
3     0.2476403E+00     0.1489089E+00
4     0.2200995E+00     0.2971089E-01
5     0.2146460E+00     0.5861640E-02
6     0.2135718E+00     0.1153708E-02
7     0.2133604E+00     0.2268553E-03
8     0.2133188E+00     0.4470348E-04
9     0.2133106E+00     0.8702278E-05
```

3.23. Neka je dat polinom $P(x) = a_1 z^3 + a_2 z^2 + a_3 z + a_4$ $(a_1 \neq 0)$. Napisati program za određivanje nula ovog polinoma po sledećem algoritmu:

1° Jednu realnu nulu odrediti primenom Newtonovog metoda (videti primer 3.19) sa tačnošću 10^{-7} $(|x_{n+1} - x_n| < 10^{-7})$;

2° Sa tako nađenom nulom z_1 odrediti koeficijente polinoma $Q(z) = P(z) / (z - z_1)$;

3° Rešiti kvadratnu jednačinu $Q(z) = 0$ pomoću standardne formule (videti primer 1.10 iz poglavlja 1.1).

Za izračunavanje vrednosti polinoma napisati potprogram tipa FUNCTION. Algoritamske korake 1° i 2° obaviti u jednom potprogramu tipa SUBROUTINE. Takođe, za algoritamski korak 3° obrazovati potprogram tipa SUBROUTINE.

U programu predvideti mogućnost rešavanja proizvoljnog broja jednačina. Na izlazu štampati koeficijente polinoma P i Q i korene jednačine $P(z) = 0$.

Program testirati sa $P(z) = 3z^3 - 7z^2 + 8z - 2$ i $P(z) = z^3 - 5z^2 - z + 5$.

Rešenje: Za izračunavanje vrednosti polinoma obrazovan je funkcijski potprogram PL, korišćenjem Hornerove šeme. Argumenti u potprogramskoj listi imaju sledeće značenje:

Z – vrednost argumenta;
A – koeficijenti polinoma;
N – stepen polinoma.

```
      FUNCTION PL(Z,A,N)
      DIMENSION A(1)
      PL=A(1)
      DO 10 I=1,N
   10 PL=PL*Z+A(I+1)
      RETURN
      END

      SUBROUTINE KJ(A,B,C,X1,Y1,X2,Y2)
      D=B*B-4.*A*C
      IF(D)25,10,20
   10 X1=-B/(2.*A)
      X2=X1
   15 Y1=0.
      Y2=0.
      RETURN
   20 X1=(-B+SQRT(D))/(2.*A)
      X2=(-B-SQRT(D))/(2.*A)
      GO TO 15
   25 X1=-B/(2.*A)
      X2=X1
      Y1=SQRT(-D)/(2.*A)
      Y2=-Y1
      RETURN
      END
```

Potprogram PL koristimo za izračunavanje vrednosti polinoma $P(z)$ i polinoma $P'(z)$.

Za rešavanje kvadratne jednačine $Q(z) = az^2 + bz + c = 0$ obrazovan je potprogram KJ. Argumenti u potprogramskoj listi imaju sledeće značenje:

A, B, C – koeficijenti jednačine;
X1, Y1 – realni i imaginarni deo prvog korena jednačine;
X2, Y2 – realni i imaginarni deo drugog korena jednačine.

Za realizaciju algoritamskih koraka 1° i 2° obrazovan je potprogram NEWT, sa argumentima:

A – koeficijenti polinoma P;
B – koeficijenti polinoma P', a zatim polinoma Q;
N – stepen polinoma P (N = 3);
Z1 – realni koren jednačine $P(z) = 0$ određen Newtonovim metodom.

```
      SUBROUTINE NEWT(A,B,N,Z1)
      DIMENSION A(1),B(1)
C     ODREDJIVANJE KOEFICIJENATA P'(Z)
      DO 5 I=1,3
    5 B(I)=A(I)*FLOAT(4-I)
C     ODREDJIVANJE REALNOG KORENA Z(1)
      Z0=0.
   10 Z1=Z0-PL(Z0,A,N)/PL(Z0,B,N-1)
      IF(ABS(Z1-Z0)-1.E-7)20,15,15
   15 Z0=Z1
      GO TO 10
C     ODREDJIVANJE KOEFICIJENATA Q(Z)
   20 B(1)=A(1)
      DO 25 I=2,3
   25 B(I)=A(I)+B(I-1)*Z1
      RETURN
      END
```

Glavni program i izlazna lista imaju oblik:

```
C==================================================================
C       RESAVANJE JEDNACINE TRECEG STEPENA
C==================================================================
      DIMENSION A(4),B(3),ZR(3),ZI(3)
    5 READ(8,10,END=99)(A(I),I=1,4)
   10 FORMAT(4F10.0)
      IF(A(1)) 15,99,15
   15 CALL NEWT(A,B,3,Z1)
      ZR(1)=Z1
      ZI(1)=0.
      WRITE(5,20) (I,A(I),I=1,4)
   20 FORMAT(1H1,22X,'KOEFICIJENTI POLINOMA P(Z)'//5X,
     *4('A(',I1,')=',F8.5,3X)//)
      WRITE(5,25) (I,B(I),I=1,3)
   25 FORMAT(/23X,'KOEFICIJENTI POLINOMA Q(Z)'//5X,
     *3('B(',I1,')=',F8.5,3X)//)
      WRITE(5,30)
   30 FORMAT(/23X,'  NULE POLINOMA P(Z)'//27X,'REAL',8X,'IMAG'/)
      CALL KJ(B(1),B(2),B(3),ZR(2),ZI(2),ZR(3),ZI(3))
      WRITE(5,35) (I,ZR(I),ZI(I),I=1,3)
   35 FORMAT(/18X,'Z(',I1,')=',2F12.7)
      GO TO 5
   99 CALL EXIT
      END
```

```
                  KOEFICIJENTI POLINOMA P(Z)
A(1)= 3.00000   A(2)=-7.00000   A(3)= 8.00000   A(4)=-2.00000

                  KOEFICIJENTI POLINOMA Q(Z)
B(1)= 3.00000   B(2)=-6.00000   B(3)= 6.00000

                  NULE POLINOMA P(Z)
                   REAL        IMAG

         Z(1)=   0.3333333   0.0000000
         Z(2)=   1.0000000   1.0000001
         Z(3)=   1.0000000  -1.0000001

                  KOEFICIJENTI POLINOMA P(Z)
A(1)= 1.00000   A(2)=-5.00000   A(3)=-1.00000   A(4)= 5.00000

                  KOEFICIJENTI POLINOMA Q(Z)
B(1)= 1.00000   B(2)= 0.00000   B(3)=-1.00000

                  NULE POLINOMA P(Z)
                   REAL        IMAG

         Z(1)=   5.0000000   0.0000000
         Z(2)=   1.0000000   0.0000000
         Z(3)=  -1.0000000   0.0000000
```

3.24. Sastaviti program za određivanje koeficijenata polinoma

(1) $P(z) = C_{n+1}z^n + C_n z^{n-1} + \ldots + C_2 z + C_1 \qquad (C_{n+1} = 1)$

ako su poznate sve njegove nule $z_k = x_k + iy_k \quad (k = 1, \ldots, n)$.

Rešenje: Neka je

$$P_k(z) \overset{\text{def}}{=} \prod_{i=1}^{k}(z - z_i) = C_{k+1}^{(k)} z^k + C_k^{(k)} z^{k-1} + \ldots + C_2^{(k)} z + C_1^{(k)}$$

Tada za polinom (1) važi $P(z) = P_n(z)$, tj. $C_i = C_i^{(n)}$ $(i = 1, \ldots, n+1)$.
Kako je

$$P_k(z) = (z - z_k) P_{k-1}(z)$$

važe sledeće rekurentne relacije

$$
\begin{aligned}
& C_1^{(k)} = -z_k C_1^{(k-1)} \\
(2) \qquad & C_i^{(k)} = C_{i-1}^{(k-1)} - z_k C_i^{(k-1)} \quad (i = 2, \ldots, k), \\
& C_{k+1}^{(k)} = C_k^{(k-1)}
\end{aligned}
$$

pri čemu se polazi od $C_1^{(1)} = -z_1$, $C_2^{(1)} = 1$.

Na osnovu (1) obrazovan je potprogram VIETE, čiji argumenti u listi imaju sledeće značenje:

Z – vektor zadatih nula dužine *N*;
N – stepen polinoma;
C – koeficijenti polinoma;
KB – kontrolni broj (KB = 0 korektno zadat stepen polinoma, KB = 1 stepen polinoma manji od jedinice).

```
      SUBROUTINE VIETE(Z,N,C,KB)
       COMPLEX Z(1),C(1),A,B
       IF(N.GE.1) GO TO 5
       KB=1
       RETURN
    5 KB=0
       C(1)=-Z(1)
       C(2)=1.
       IF(N.GE.2) GO TO 20
       RETURN
   20 DO 15 K=2,N
       A=C(1)
       C(1)=-Z(K)*C(1)
       DO 10 I=2,K
       B=C(I)
       C(I)=A-Z(K)*B
   10 A=B
   15 C(K+1)=A
       RETURN
       END
```

Potprogram, kao i glavni program, realizovani su u kompleksnoj aritmetici. Glavni program i izlazna lista (za jedan konkretan slučaj) imaju oblik:

```
C=================================================================
C      FORMIRANJE KOEFICIJENATA POLINOMA NA OSNOVU
C                   ZADATIH NULA
C=================================================================
       COMPLEX Z(10),C(11)
C      UCITAVANJE NULA POLINOMA
     2 READ(8,10,END=99)N,(Z(I),I=1,N)
    10 FORMAT(I2/(10F8.2))
C      POZIV PODPROGRAMA VIETE
       CALL VIETE(Z,N,C,KB)
       IF(KB.EQ.0) GO TO 1
       WRITE(5,20)
    20 FORMAT(1H1/5X,'GRESKA U PODACIMA: STEPEN POLINOMA '
      1'MANJI OD 1'//)
       GO TO 2
C      STAMPANJE NULA I KOEFICIJENATA POLINOMA
     1 WRITE(5,25)(I,Z(I),I=1,N)
    25 FORMAT(16X,'NULE POLINOMA'//19X,'REAL',5X,'IMAG'//
      1(10X,'Z(',I<I/10+1>,')=',4X,F10.7,1X,F10.7))
       N=N+1
       WRITE(5,30)(I,C(I),I=1,N)
    30 FORMAT(/16X,'KOEFICIJENTI POLINOMA'//23X,'REAL',5X,
      1'IMAG'//(10X,'C(',I<I/10+1>,')=',4X,F10.7,1X,F10.7))
       GO TO 2
    99 CALL EXIT
       END
```

```
                NULE POLINOMA

                   REAL      IMAG

          Z(1)=    1.0000000  1.0000000
          Z(2)=    1.0000000  1.0000000
          Z(3)=    1.0000000 -1.0000000

                KOEFICIJENTI POLINOMA

                       REAL      IMAG

          C(1)=   -2.0000000 -2.0000000
          C(2)=    4.0000000  2.0000000
          C(3)=   -3.0000000 -1.0000000
          C(4)=    1.0000000  0.0000000
```

3.25. Sastaviti program za određivanje kompleksnog korena transcendentne jednačine $f(z) = 0$ primenom Newtonovog metoda

$$(1) \quad z_{n+1} = z_n - \frac{f(z_n)}{f'(z_n)} \qquad (f'(z_n) \neq 0),$$

gde je $z_n = x_n + iy_n$ $(n = 0, 1, \ldots)$. Iterativni proces prekinuti kada istovremeno budu ispunjeni uslovi

$$(2) \quad \begin{aligned} |x_{n+1} - x_n| &\leqslant \epsilon, \\ |y_{n+1} - y_n| &\leqslant \epsilon, \end{aligned}$$

gde je ϵ unapred zadata tačnost.

Program organizovati na sledeći način:

1° U potprogramu tipa FUNCTION zadaju se funkcije: Re $(f(z))$, Im $(f(z))$, Re $(f'(z))$, Im $(f'(z))$.

2° U potprogramu tipa SUBROUTINE izračunava se jedan iterativni korak po formuli (1).

3° Glavni program treba da sadrži:

a) Učitavanje ulaznih podataka x_o, y_o, ϵ;

b) Pozivanje potprograma tipa SUBROUTINE i ispitivanje uslova (2);

c) Štampanje naslova i tabele u obliku:

NEWTONOV METOD ZA REŠAVANJE TRANSCENDENTNE JEDNAČINE

F (Z) = EXP (Z) – 0.2 ☆ Z + 1 = 0

BR.ITER	REAL (Z)	IMAG (Z)	REAL (F(Z))	IMAG (F(Z))
0	X.XXXXXXX	X.XXXXXXX	X.XXXXXX	X.XXXXXX
⋮				

ZADATA TAČNOST IZRAČUNAVANJA EPSILON = 0.XE±XX

Ukoliko je $f'(z_n) = 0$ štampati:

PRVI IZVOD FUNKCIJE JEDNAK NULI

i prekinuti iterativni proces.

Za rešavanje uzeti primer: $f(z) = e^z - 0.2\,z + 1$, $z_o = 1 + \pi i$, $\epsilon = 10^{-6}$

Problem rešiti u realnoj i kompleksnoj aritmetici.

Rešenje: S obzirom da je $z = x + iy$, iz formule

$$z_{n+1} = z_n - \frac{f(z_n)}{f'(z_n)} \qquad (n = 0, 1, \ldots),$$

razdvajanjem realnog i imaginarnog dela, sleduje

$$x_{n+1} = x_n - \frac{1}{\Delta}(\mathrm{Re}\,(f(z_n))\,\mathrm{Re}\,(f'(z_n)) + \mathrm{Im}\,(f(z_n))\,\mathrm{Im}\,(f'(z_n)))$$

i

$$y_{n+1} = y_n - \frac{1}{\Delta}(\mathrm{Im}\,(f(z_n))\,\mathrm{Re}\,(f'(z_n)) - \mathrm{Re}\,(f(z_n))\,\mathrm{Im}\,(f'(z_n))),$$

gde je $\Delta = |f'(z_n)|^2 = |\mathrm{Re}\,(f'(z_n))|^2 + |\mathrm{Im}\,(f'(z_n))|^2$.

Kako je u našem slučaju

$$f(z) = e^z - 0.2z + 1 \quad \text{i} \quad f'(z) = e^z - 0.2,$$

imamo redom

$$\begin{aligned}
\text{Re}\ (f(z)) &= e^x \cos y - 0.2\,x + 1,\\
\text{Im}\ (f(z)) &= e^x \sin y - 0.2y\\
\text{Re}\ (f'(z)) &= e^x \cos y - 0.2\\
\text{Im}\ (f'(z)) &= e^x \sin y.
\end{aligned}$$

Potprogrami i glavni program u realnoj aritmetici, kao i odgovarajući rezultati imaju oblik:

```
      FUNCTION EF(X,Y,I)
      GO TO(10,20,30,40),I
   10 EF=EXP(X)*COS(Y)-.2*X+1.
      RETURN
   20 EF=EXP(X)*SIN(Y)-.2*Y
      RETURN
   30 EF=EXP(X)*COS(Y)-.2
      RETURN
   40 EF=EXP(X)*SIN(Y)
      RETURN
      END
```

```
      SUBROUTINE TRANS(X0,Y0,A,B,R)
      C=EF(X0,Y0,3)
      D=EF(X0,Y0,4)
      R=C*C+D*D
      IF(R) 5,10,5
    5 X0=X0-(A*C-B*D)/R
      Y0=Y0-(B*C-A*D)/R
   10 RETURN
      END
```

```
C=====================================================================
C   PROGRAM ZA NALAZANJE KOMPLEKSNOG KORENA TRANSCENDENTNE
C   JEDNACINE  F(Z) = 0  PRIMENOM NEWTONOVOG METODA
C               REALNA ARITMETIKA
C=====================================================================
      READ(8,5) X0,Y0,EPS
    5 FORMAT(2F10.0,E5.0)
      WRITE(5,10)
   10 FORMAT(//10X,'NEWTONOV METOD ZA RESAVANJE TRANSCENDE'
     1,'NTNE JEDNACINE'//22X,'F(Z)=EXP(Z)-0.2*Z+1=0'//
     25X,'BR.ITER.',4X,'REAL(Z)',5X,'IMAG(Z)',4X,'REAL(F(Z))'
     3,2X,'IMAG(F(Z))'/)
      ITER=0
      KBR=1
   15 A=EF(X0,Y0,1)
      B=EF(X0,Y0,2)
      WRITE(5,20) ITER,X0,Y0,A,B
   20 FORMAT(5X,I4,2X,2F15.7,2F12.6)
      GO TO (22,50),KBR
   22 ITER=ITER+1
      XS=X0
      YS=Y0
      CALL TRANS(X0,Y0,A,B,R)
      IF(R) 25,25,35
   25 WRITE(5,30)
```

```
30 FORMAT(//5X,'PRVI IZVOD FUNKCIJE JEDNAK NULI')
   GO TO 50
35 IF(ABS(XO-XS)-EPS) 40,40,15
40 IF(ABS(YS-YO)-EPS) 45,45,15
45 KBR=2
   GO TO 15
50 WRITE(5,55) EPS
55 FORMAT(/5X,'ZADATA TACNOST IZRACUNAVANJA EPSILON ='
  1,F7.1)
   CALL EXIT
   END
```

NEWTONOV METOD ZA RESAVANJE TRANSCENDENTNE JEDNACINE

F(Z)=EXP(Z)-0.2*Z+1=0

BR.ITER.	REAL(Z)	IMAG(Z)	REAL(F(Z))	IMAG(F(Z))
0	1.0000000	3.1415920	-1.918282	-0.628316
1	0.3426672	2.9262881	-0.444709	-0.284296
2	0.0372189	2.7002838	0.054076	-0.096737
3	0.0497325	2.6425617	0.067235	-0.025535
4	0.0911206	2.6459618	0.018186	-0.008234
5	0.1015006	2.6458960	0.006090	-0.002721
6	0.1049548	2.6459036	0.002026	-0.000908
7	0.1060993	2.6459043	0.000678	-0.000304
8	0.1064821	2.6459045	0.000227	-0.000102
9	0.1066100	2.6459043	0.000077	-0.000034
10	0.1066533	2.6459045	0.000025	-0.000012
11	0.1066675	2.6459043	0.000009	-0.000004
12	0.1066727	2.6459045	0.000003	-0.000001
13	0.1066741	2.6459043	0.000001	-0.000000
14	0.1066747	2.6459045	0.000000	-0.000000

ZADATA TACNOST IZRACUNAVANJA EPSILON =0.1E-05

Navedeni potprogrami i glavni program mogu se znatno jednostavnije realizovati u kompleksnoj aritmetici. Interesantno je primetiti da je u ovom slučaju za određivanje kompleksnog korena posmatrane jednačine sa istom tačnošću (10^{-6}) potrebno samo 5 iteracija. Do ove razlike u broju iteracija dolazi zbog tačnijeg izvršavanja potprograma CEXP, u odnosu na potprograme EXP, SIN i COS.

```
   COMPLEX FUNCTION F(Z,I)
   COMPLEX Z
   GO TO(1,2),I
 1 F=CEXP(Z)-0.2*Z+1.
   RETURN
 2 F=CEXP(Z)-0.2
   RETURN
   END
```

```
   SUBROUTINE NEW(Z,Z0)
   COMPLEX Z,Z0,F
   Z=Z0- F(Z0,1)/F(Z0,2)
   RETURN
   END
```

```
C=====================================================================
C  PROGRAM ZA NALAZANJE KOMPLEKSNOG KORENA TRANSCENDENTNE
C  JEDNACINE  F(Z) = 0  PRIMENOM NEWTONOVOG METODA
C             KOMPLEKSNA ARITMETIKA
C=====================================================================
      COMPLEX Z,Z0,F,Y,A
      READ(8,10)Z0
   10 FORMAT(2E14.7)
      EPS=1.E-6
      WRITE(5,20)
   20 FORMAT(//10X,'NEWTONOV METOD ZA RESAVANJE TRANSCENDE'
     1,'NTNE JEDNACINE'//22X,'F(Z)=EXP(Z)-0.2*Z+1=0'//
     25X,'BR.ITER.',4X,'REAL(Z)',5X,'IMAG(Z)',4X,'REAL(F(Z))'
     3,2X,'IMAG(F(Z))'/)
      ITER=0
       Y=F(Z0,1)
   13 WRITE(5,30)ITER,Z0,Y
   30 FORMAT(5X,I4,2X,2F13.7,2F12.6)
      Y= F(Z0,2)
      B=CABS(Y)
      IF(B.EQ.0.)GO TO 99
      CALL NEW(Z,Z0)
      ITER=ITER+1
      Y=F(Z,1)
      A=Z-Z0
      IF(ABS(REAL(A)).GT.EPS)GO TO 95
      IF(ABS(AIMAG(A)).LE.EPS)GO TO 98
   95 Z0=Z
      GO TO 13
   99 WRITE(5,40)
   40 FORMAT(10X,'PRVI IZVOD FUNKCIJE JEDNAK NULI'//)
      GO TO 97
   98 WRITE(5,30)ITER,Z ,Y
   97 WRITE(5,55) EPS
   55 FORMAT(/5X,'ZADATA TACNOST IZRACUNAVANJA EPSILON ='
     1,E7.1)
      CALL EXIT
      END
```

```
          NEWTONOV METOD ZA RESAVANJE TRANSCENDENTNE JEDNACINE

                      F(Z)=EXP(Z)-0.2*Z+1=0

BR.ITER.     REAL(Z)       IMAG(Z)     REAL(F(Z))  IMAG(F(Z))

    0       1.0000000     3.1415920    -1.918282   -0.628316
    1       0.3426675     2.9262881    -0.444709   -0.284296
    2       0.1036774     2.7002838    -0.023705   -0.066273
    3       0.1054018     2.6458714     0.001517   -0.000634
    4       0.1066756     2.6459045    -0.000001    0.000000
    5       0.1066748     2.6459043     0.000000    0.000000

ZADATA TACNOST IZRACUNAVANJA EPSILON =0.1E-05
```

3.26. Dat je niz od n brojeva x_i ($i = 1, \ldots, n$), na svakoj kartici po 10 brojeva, u formatu 10F8.2. Broj n je dat na prvoj kartici u formatu I3.

Napisati program za preuređenje niza tako da na početku niza budu pozitivni brojevi u neopadajućem redosledu, a zatim negativni brojevi u nerastućem redosledu.

Rešenje: Program i izlazna lista za određeni skup ulaznih podataka imaju oblik:

```
C=====================================================================
C      PROGRAM ZA PREUREDJENJE NIZA
C=====================================================================
C
      DIMENSION X(999)
      READ(8,10)N
   10 FORMAT(I3)
C  UCITAVANJE NIZA
      READ(8,20)(X(I),I=1,N)
   20 FORMAT(10F8.2)
C  UREDJENJE NIZA PO NERASTUCEM REDOSLEDU
      K=N-1
      DO 11 I=1,K
      L=I+1
      DO 11 J=L,N
      IF(X(I).GT.X(J))GO TO 11
      R=X(I)
      X(I)=X(J)
      X(J)=R
   11 CONTINUE
       NP=0
C PREBROJAVANJE POZITIVNIH BROJEVA
      DO 22 I=1,N
      IF(X(I).LT.0.)GO TO 33
   22 NP=NP+1
C  UREDJENJE POZITIVNIH BROJEVA U NEOPADAJUCEM REDOSLEDU
   33 L=NP/2
      DO 44 I=1,L
      R=X(I)
      X(I)=X(NP+1-I)
      X(NP+1-I)=R
   44 CONTINUE
C  STAMPANJE UREDJENOG NIZA
      WRITE(5,30)(X(I),I=1,N)
   30 FORMAT(1H1/ 14X,'UREDJENI NIZ'//(1X,4(1X,F8.2)))
      CALL EXIT
      END
```

```
              UREDJENI NIZ

    1.00      2.00      3.00      4.00
    4.00      5.00      5.00      7.00
    7.00      8.00      8.00      9.00
   10.00      1.00     12.00     13.00
   13.00     13.00     13.13     14.00
   18.00     18.00     19.00     20.00
   21.00     22.00     23.00     25.00
   30.00     33.00     60.00     -8.00
   -9.00     -9.00    -10.00    -11.00
  -12.00    -12.00    -15.00    -16.00
  -22.00    -25.00    -31.00    -32.00
  -33.00    -35.00    -40.00    -45.00
  -50.00   -169.00
```

3.27. Neka je funkcija f definisana pomoću $f(x, y) = e^{3x}\sin xy + x^2 y - y \log x$. Parcijalni izvodi funkcije f u tački (x_0, y_0) mogu se približno izračunati pomoću sledećih formula:

$$\frac{\partial f(x_0, y_0)}{\partial x} \cong \frac{f(x_0 + h, y_0) - f(x_0, y_0)}{h},$$

$$\frac{\partial f(x_0, y_0)}{\partial y} \cong \frac{f(x_0, y_0 + h) - f(x_0, y_0)}{h},$$

gde je h dovoljno mali priraštaj argumenta.

Napisati program za tabeliranje vrednosti funkcije i njenih parcijalnih izvoda za $x = 1\ (0.2)\ 1.6$ i $y = 0\ (0.2)\ 0.4$, uzimajući $h = 0.01$. Paralelno računati i tačne vrednosti parcijalnih izvoda (radi provere datih formula). Izlaznu listu dati u obliku:

```
X   Y     F(X,Y)       DFXP         DFXT         DFYP         DFYT

X.X X.X  X.XXXXE±XX  X.XXXXE±XX  X.XXXXE±XX  X.XXXXE±XX  X.XXXXE±XX
 ⋮
```

gde su DFXP i DFYP, DFXT i DFYT redom približne vrednosti parcijalnih izvoda po x i y, odnosno njihove tačne vrednosti.

Funkciju f i njene parcijalne izvode zadati u potprogramu tipa FUNCTION.

Rešenje: Potprogram, program i odgovarajuća izlazna lista imaju oblik:

```
      FUNCTION FP(X,Y,I)
C     ZA I=1 RACUNAJU SE VREDNOSTI FUNKCIJE F
C     ZA I=2 RACUNAJU SE VREDNOSTI PARCIJALNIH IZVODA DF/DX
C     ZA I=3 RACUNAJU SE VREDNOSTI PARCIJALNIH IZVODA DF/DY
      U=EXP(3.*X)
      GO TO (1,2,3),I
    1 FP=U*SIN(X*Y)+X*X*Y-Y*ALOG(X)
      RETURN
    2 FP=U*(3.*SIN(X*Y)+Y*COS(X*Y))+2.*X*Y-Y/X
      RETURN
    3 FP=X*U*COS(X*Y)+X*X-ALOG(X)
      RETURN
      END

C==============================================================
C  PROGRAM ZA TABELIRANJE VREDNOSTI FUNKCIJE I NJENIH
C   PARCIJALNIH IZVODA
C==============================================================
      READ(8,5) XP,XK,DX,YP,YK,DY,H
    5 FORMAT(7F5.0)
      N=IFIX((XK-XP)/DX)+1
      M=IFIX((YK-YP)/DY)+1
      WRITE(5,10)
   10 FORMAT(1H1,5X, 'X',3X,'Y',5X,'F(X,Y)',7X,'DFXP',
     18X,'DFXT',8X,'DFYP',8X,'DFYT'/)
      DO 15 I=1,N
      X=XP+DX*FLOAT(I-1)
```

```
      DO 15 J=1,M
      Y=YP+DY*FLOAT(J-1)
      F=FP(X,Y,1)
      DFXP=(FP(X+H,Y,1)-F)/H
      DFYP=(FP(X,Y+H,1)-F)/H
      DFXT=FP(X,Y,2)
      DFYT=FP(X,Y,3)
   15 WRITE(5,20) X,Y,F,DFXP,DFXT,DFYP,DFYT
   20 FORMAT(4X,2F4.1,5E12.4)
      CALL EXIT
      END
```

X	Y	F(X,Y)	DFXP	DFXT	DFYP	DFYT
1.0	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.2109E+02	0.2109E+02
1.0	0.2	0.4190E+01	0.1641E+02	0.1611E+02	0.2066E+02	0.2069E+02
1.0	0.4	0.8222E+01	0.3185E+02	0.3127E+02	0.1946E+02	0.1950E+02
1.2	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.4517E+02	0.4518E+02
1.2	0.2	0.8951E+01	0.3413E+02	0.3352E+02	0.4385E+02	0.4392E+02
1.2	0.4	0.1740E+02	0.6547E+02	0.6431E+02	0.4009E+02	0.4021E+02
1.4	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.9498E+02	0.9498E+02
1.4	0.2	0.1875E+02	0.6975E+02	0.6852E+02	0.9116E+02	0.9135E+02
1.4	0.4	0.3607E+02	0.1320E+03	0.1297E+03	0.8037E+02	0.8072E+02
1.6	0.0	0.0000E+00	0.0000E+00	0.0000E+00	0.1965E+03	0.1965E+03
1.6	0.2	0.3864E+02	0.1407E+03	0.1383E+03	0.1861E+03	0.1866E+03
1.6	0.4	0.7340E+02	0.2621E+03	0.2577E+03	0.1571E+03	0.1580E+03

3.28. Sastaviti program za tabeliranje funkcije greške

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

za $x = 0\ (0.5)\ 10$ sa tačnošću 10^{-5}. Izračunavanje izvoditi u potprogramu tipa SUBROUTINE na sledeći način:

a) za $x < 3$ podintegralnu funkciju razviti u potencijalni red u okolini nule i integraliti dobijeni red član po član;

b) za $x \geqslant 3$ koristiti približnu asimptotsku formulu

$$(1) \qquad erf(x) = 1 - \frac{e^{-x^2}}{x\sqrt{\pi}} \left(1 - \frac{1}{2x^2} + \frac{1 \cdot 3}{2^2 x^4} - \frac{1 \cdot 3 \cdot 5}{2^3 x^6} + \ldots\right)$$

Glavni program treba da obezbedi potrebnu promenu argumenta x i štampanje tabele u obliku:

X	ERF (X)	X	ERF (X)
X.X	X.XXXX	X.X	X.XXXXX
⋮			

Rešenje: S obzirom na razvoj

$$e^{-t^2} = \sum_{k=0}^{+\infty} \frac{(-1)^k t^{2k}}{k!}$$

imamo

$$(2) \qquad erf(x) = \frac{2}{\sqrt{\pi}} \sum_{k=0}^{+\infty} \frac{(-1)^k x^{2k+1}}{k!\,(2k+1)} \cdot$$

Prema uslovu zadatka, za izračunavanje vrednosti funkcije greške, koristimo razvoj (2) za $x < 3$ i razvoj (1) za $x \geqslant 3$. Za sumiranje ovih redova (videti primer 1.6, poglavlje 1.1), funkcija ϕ biće

$$\phi(k,x) = -\frac{2k-1}{k(2k+1)}\, x^2 \qquad (x < 3),$$

$$\phi(k,x) = -\frac{2k-1}{2x^2} \qquad (x \geqslant 3).$$

Odgovarajući programi i izlazna lista imaju oblik:

```
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION ER(20),X(20)
      WRITE(5,15)
   15 FORMAT(1H1,2(6X,'X',4X,'ERF(X)')/)
      DO 10 I=5,100,5
      K=I/5
      X(K)=0.1D0*DBLE(FLOAT(I))
   10 CALL FG(X(K),ER(K))
      DO 20 I=1,10
      X(I)=X(I)+0.001D0
      X(I+10)=X(I+10)+0.001D0
   20 WRITE(5,25)X(I),ER(I),X(I+10),ER(I+10)
   25 FORMAT(5X,F4.1,F9.5,4X,F4.1,F9.5)
      CALL EXIT
      END

      SUBROUTINE FG(X,ERF)
      IMPLICIT REAL*8 (A-H,O-Z)
      PI=3.1415926535D0
      PI=DSQRT(PI)
      Y=X*X
      EPS=1.D-5
      S=0.D0
      N=1
      IF(X-3.D0)5,10,10
    5 U=X
   16 S=S+U
      A=DBLE(FLOAT(N))
      U=-(2.D0*A-1.D0)/(A*(2.D0*A+1.D0))*Y*U
      N=N+1
      IF(DABS(U)-EPS)15,16,16
   15 ERF=2.D0/PI*S
      RETURN
   10 F=DEXP(-Y)/(X*PI)
      U=1.D0
   20 S=S+U
      A=DBLE(FLOAT(N))
      U=-(2.D0*A-1.D0)/(2.D0*Y)*U
      N=N+1
      IF(DABS(U*F)-EPS)21,21,20
   21 ERF=1.D0-F*S
      RETURN
      END
```

X	ERF(X)	X	ERF(X)
0.5	0.52049	5.5	1.00000
1.0	0.84270	6.0	1.00000
1.5	0.96610	6.5	1.00000
2.0	0.99532	7.0	1.00000
2.5	0.99960	7.5	1.00000
3.0	0.99998	8.0	1.00000
3.5	1.00000	8.5	1.00000
4.0	1.00000	9.0	1.00000
4.5	1.00000	9.5	1.00000
5.0	1.00000	10.0	1.00000

3.29. Lagrangeov interpolacioni polinom za funkciju $x \to f(x)$, za interpolacione čvorove x_i $(i = 1, \ldots, n)$, ima oblik

$$P(x) = \sum_{i=1}^{n} \frac{(x - x_1)\ldots(x - x_{i-1})(x - x_{i+1})\ldots(x - x_n)}{(x_i - x_1)\ldots(x_i - x_{i-1})(x_i - x_{i+1})\ldots(x_i - x_n)} f(x_i).$$

Uzimajući za interpolacione čvorove $x_i = x_o + (i - 1)h$ $(i = 1, \ldots, n)$, $x_o = 0$, $h = 0.5$, $n = 10$, sastaviti program za izračunavanje vrednosti polinoma P (za funkciju $f(x) = e^x \sin x$) u m zadatih tačaka.

Rešenje: Program i izlazni rezultati imaju oblik:

```
C     ==================================================
C     LAGRANGE-OVA   INTERPOLACIJA
C     ==================================================
      REAL L(20)
      DIMENSION X(20),Y(20),XP(20)
      FF(X)=EXP(X)*SIN(X)
      READ(8,5) N,X0,DX
      READ(8,5) M,(XP(I),I=1,M)
    5 FORMAT(I2/(8F10.0))
      DO 7 I=1,N
      X(I)=X0+DX*FLOAT(I-1)
    7 Y(I)=FF(X(I))
      WRITE(5,40)
   40 FORMAT(1H1,8X,'K',3X,'X',11X,'F',14X,'FT'/)
      DO 25 K=1,M
      DO 10 J=1,N
      L(J)=1.
      DO 10 I=1,N
      IF(I-J) 15,10,15
   15 L(J)=L(J)*(XP(K)-X(I))/(X(J)-X(I))
   10 CONTINUE
      F=0.
      DO 20 I=1,N
   20 F=F+L(I)*Y(I)
      FT=FF(XP(K))
   25 WRITE(5,30) K,XP(K),F,FT
   30 FORMAT(I10,F6.2,2F15.7)
      CALL EXIT
      END
```

K	X	F	FT
1	0.22	0.2703108	0.2719308
2	0.76	1.4733748	1.4731042
3	1.28	3.4455431	3.4456384
4	1.94	6.4898572	6.4898362
5	2.73	6.1342893	6.1342211
6	3.14	0.0367036	0.0367950
7	3.65	-18.7286777	-18.7289619
8	4.35	-72.4501190	-72.4464188

3.30. Napisati program za sumiranje sporokonvergentnog reda $\sum_{k=o}^{+\infty} u_k(x)$, gde je $u_k(x) = (-1)^k \dfrac{4}{2k+1+x}$, za $x = 0\ (0.5)\ 5$, primenom nelinearne transformacije

$$T_n = T_n(S_n) = S_{n+1} - \frac{S_{n+1}S_{n-1} - S_n^2}{S_{n+1} + S_{n-1} - 2S_n} \quad (n = 1, 2, \ldots).$$

gde je $S_n = \sum_{k=o}^{n} u_k(x)$. Sumiranje prekinuti kada je ispunjen uslov $|T_{n+1} - T_n| < \epsilon$, gde je $\epsilon = 10^{-5}$. Na izlazu štampati

X	N	S (N)	T (N)
X.X	XX	X.XXXXXXXE±XX	X.XXXXXXXE±XX
⋮			

Rešenje: Program i izlazna lista imaju oblik:

```
C==================================================================
C    IZRACUNAVANJE SUME SPOROKONVERGENTNOG REDA
C==================================================================
      FI(I,X)=-(FLOAT(2*I-1)+X)/(FLOAT(2*I+1)+X)
      WRITE(5,100)
  100 FORMAT(4X,'SUMIRANJE SPOROKONVERGENTNOG REDA'//
     13X,'X',4X,'N',7X,'S(N)',11X,'T(N)'/)
      READ(8,200)XP,XK,DX,EPS
  200 FORMAT(3F5.2,E8.1)
      X=XP
   22 U0=4./(1.+X)
      U=-4./(3.+X)
      I=0
      S1=U0
      A1=1.E10
      I=1
      S=U0+U
      S2=S
   10 I=I+1
      U=U*FI(I,X)
      S=S+U
      S3=S
      A2=S3-(S3-S2)**2/(S1+S3-2.*S2)
      IF(ABS(A2-A1)-EPS) 20,20,15
```

```
15 S1=S2
   S2=S3
   A1=A2
   GO TO 10
20 WRITE(5,30)X,I,S3,A2
30 FORMAT(2X,F3.1,2X,I3,2(1X,E14.7))
   IF(X.GE.XK)GO TO 11
   X=X+DX
   GO TO 22
11 CALL EXIT
   END
```

SUMIRANJE SPOROKONVERGENTNOG REDA

X	N	S(N)	T(N)
0.0	38	0.3167229E+01	0.3141597E+01
0.5	38	0.1975456E+01	0.1949987E+01
1.0	37	0.1360324E+01	0.1386289E+01
1.5	37	0.1038414E+01	0.1064211E+01
2.0	37	0.8327705E+00	0.8584028E+00
2.5	37	0.6912112E+00	0.7166803E+00
3.0	36	0.6396754E+00	0.6137105E+00
3.5	36	0.5615863E+00	0.5357888E+00
4.0	36	0.5005631E+00	0.4749309E+00
4.5	36	0.4516462E+00	0.4261771E+00
5.0	35	0.3603248E+00	0.3862897E+00

4 3.31. Date su matrice $A = [a_{ij}]_{n\times m}$ i $B = [b_{ij}]_{n\times m}$ $(n \leqslant 20,\ m \leqslant 8)$. Odrediti matricu C čiji su elementi određeni formulama

$$c_{ij} = \begin{cases} a_{ij} + b_{ij} & (a_{ij} < b_{ij}), \\ a_{ij} - b_{ij} & (a_{ij} > b_{ij}), \\ a_{ij} b_{ij} & (a_{ij} = b_{ij}). \end{cases}$$

Na listi štampati elemente matrica A, B, C.

```
C====================================================================
C
C====================================================================
      DIMENSION A(20,8),B(20,8),C(20,8)
C     UCITAVANJE MATRICA A I B
      READ(8,1)N,M
    1 FORMAT(I2,I1)
      READ(8,2)((A(I,J),J=1,M),I=1,N),((B(I,J),J=1,M),I=1,N)
    2 FORMAT(<M>F8.2)
C   IZRACUNAVANJE C
      DO 11 I=1,N
      DO 11 J=1,M
      IF(A(I,J)-B(I,J))10,20,30
   10 C(I,J)=A(I,J)+B(I,J)
      GO TO 11
```

```
20 C(I,J)=A(I,J)*B(I,J)
   GO TO 11
30 C(I,J)=A(I,J)-B(I,J)
11 CONTINUE
C   STAMPANJE IZLAZNIH REZULTATA
   I1=1HA
   I2=1HB
   I3=1HC
   WRITE(5,100)I1,((A(I,J),J=1,M),I=1,N)
   WRITE(5,100)I2,((B(I,J),J=1,M),I=1,N)
   WRITE(5,100)I3,((C(I,J),J=1,M),I=1,N)
100 FORMAT(//<M*5-4>X,'MATRICA ',A1 //(<M>(2X,F8.2) ))
   CALL EXIT
   END
```

```
                 MATRICA A

      1.00      2.00      3.00      4.00
      5.00      6.00      7.00      8.00
      9.00     10.00     11.00     12.00

                 MATRICA B

      9.00     10.00     11.00     12.00
      5.00      6.00      7.00      8.00
      1.00      2.00      3.00      4.00

                 MATRICA C

     10.00     12.00     14.00     16.00
     25.00     36.00     49.00     64.00
      8.00      8.00      8.00      8.00
```

3.32. Napisati program za transponovanje matrica $A = [a_{ij}]_{m \times n}$ ne koristeći pomoćna polja u centralnoj memoriji, već pomoću upisa na disk i učitavanja sa diska direktno u transponovanom obliku. Na izlazu štampati matricu A i njenu transponovanu matricu.

Rešenje: Program i izlazni rezultat za konkretno datu matricu A tipa 3x4 ima oblik:

```
C===================================================================
C   TRANSPONOVANJE MATRICE UPISOM NA DISK I ODGOVARAJUCIM
C                 UCITAVANJEM SA DISKA
C===================================================================
      DIMENSION A(50,50)
      DEFINE FILE 1(50,100,U,II)
C     UCITAVANJE MATRICE A
      READ(8,10)M,N
   10 FORMAT(2I2)
      READ(8,20)((A(I,J),J=1,N),I=1,M)
   20 FORMAT(<N>F8.2)
C   STAMPANJE MATRICE A
      WRITE(5,30)((A(I,J),J=1,N),I=1,M)
   30 FORMAT(//<N*5-4>X,'MATRICA A'//(<N>(2X,F8.2)  ))
```

```
C    UPIS NA DISK
      II=1
      DO 11 I=1,M
   11 WRITE(1'II)(A(I,J),J=1,N)
C    UCITAVANJE SA DISKA
      II=1
      DO 22 I=1,M
   22 READ(1'II)(A(J,I),J=1,N)
C    STAMPANJE TRANSPONOVANE MATRICE A
      WRITE(5,40)((A(I,J),J=1,M),I=1,N)
   40 FORMAT(//<M*5-4>X,'MATRICA A - TRANSP.'//
     1(<M>(2X,F8.2)   ))
      CALL EXIT
      END
```

```
                MATRICA A

     1.00      2.00      3.00      4.00
     5.00      6.00      7.00      8.00
     9.00     10.00     11.00     12.00

           MATRICA A - TRANSP.

     1.00      5.00      9.00
     2.00      6.00     10.00
     3.00      7.00     11.00
     4.00      8.00     12.00
```

4. NUMERIČKI METODI U LINEARNOJ ALGEBRI

4.1. ELEMENTI MATRIČNOG RAČUNA

4.1.1. LR faktorizacija kvadratne matrice

Često se kod rešavanja sistema linearnih jednačina javlja problem predstavljanja kvadratne matrice kao proizvod dve trougaone matrice. Ovaj odeljak posvećen je ovom problemu.

Teorema 1.1.1. Ako su sve determinante

$$\Delta_k = \begin{vmatrix} a_{11} & \dots & a_{1k} \\ \cdot & & \\ \cdot & & \\ \cdot & & \\ a_{k1} & & a_{kk} \end{vmatrix} \qquad (k = 1, \dots, n-1)$$

različite od nule, matrica $A = [a_{ij}]_{n \times n}$ može se predstaviti u obliku

(1.1.1) $\quad A = LR,$

gde je L donja i R gornja trougaona matrica.

Trougaone matrice L i R reda n, imaju oblike:

(1.1.2) $\quad L = [l_{ij}]_{n \times n} \quad (l_{ij} = 0 \Leftarrow i < j),$

(1.1.3) $\quad R = [r_{ij}]_{n \times n} \quad (r_{ij} = 0 \Leftarrow i > j).$

Razlaganje (1.1.1), poznato kao LR faktorizacija (dekompozicija), nije jedinstveno, s obzirom na jednakost

$$LR = (cL)\left(\frac{1}{c}R\right) \qquad (\forall\, c \neq 0).$$

Međutim, ako se dijagonalnim elementima matrice R (ili L) fiksiraju vrednosti od kojih nijedna nije jednaka nuli, razlaganje je jedinstveno.

S obzirom na (1.1.2) i (1.1.3) i imajući u vidu da je

$$a_{ij} = \sum_{k=1}^{\max(i,j)} l_{ik} r_{kj} \qquad (i, j = 1, \dots, n)$$

elementi matrica L i R mogu se lako odrediti rekurzivnim postupkom, ukoliko se unapred zadaju elementi r_{ii} $(\neq 0)$ ili l_{ii} $(\neq 0)$ $(i = 1, \ldots, n)$.

Tako, na primer, neka su dati brojevi r_{ii} $(\neq 0)$ $(i = 1, \ldots, n)$. Tada važi

$$l_{11} = \frac{a_{11}}{r_{11}}$$

$$\left.\begin{aligned} r_{1i} &= \frac{a_{1i}}{l_{11}} \\ l_{i1} &= \frac{a_{i1}}{r_{11}} \end{aligned}\right\} \quad (i = 2, \ldots, n);$$

$$\left.\begin{aligned} &\text{(i)} \quad l_{ii} = \frac{1}{r_{ii}}\Big(a_{ii} - \sum_{k=1}^{i-1} l_{ik} r_{ki}\Big), \\ &\left.\begin{aligned} r_{ij} &= \frac{1}{l_{ii}}\Big(a_{ij} - \sum_{k=1}^{i-1} l_{ik} r_{kj}\Big) \\ l_{ji} &= \frac{1}{r_{ii}}\Big(a_{ji} - \sum_{k=1}^{i-1} l_{jk} r_{ki}\Big) \end{aligned}\right\} \quad (j = i+1, \ldots, n); \end{aligned}\right\} \quad (i = 2, \ldots, n).$$

Slično bismo mogli iskazati i rekurzivni postupak za određivanje elemenata matrica L i R ako su unapred dati brojevi l_{ii} $(\neq 0)$ $(i = 1, \ldots, n)$.

U primenama, najčešće se uzima $r_{ii} = 1$ $(i = 1, \ldots, n)$ ili $l_{ii} = 1$ $(i = 1, \ldots, n)$.

U primenama vrlo često se javljaju višedijagonalne matrice, tj. matrice čiji su elementi različiti od nule samo na glavnoj dijagonali i oko glavne dijagonale. Na primer, ako je $a_{ij} \neq 0$ za $|i-j| \leqslant 1$ i $a_{ij} = 0$ za $|i-j| > 1$, matrica je trodijagonalna. Obično elemente ovakve matrice predstavljamo vektorima $(a_2, \ldots, a_n)$, $(b_1, \ldots b_n)$, $(C_1, \ldots, C_{n-1})$, tj.

$$\text{(1.1.4)} \qquad A = \begin{bmatrix} b_1 & c_1 & 0 \ldots 0 & 0 \\ a_2 & b_2 & c_2 \quad 0 & 0 \\ 0 & a_3 & b_3 \quad 0 & 0 \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ 0 & 0 & 0 \quad a_n & b_n \end{bmatrix}.$$

Ako je $a_{ij} \neq 0$ $(|i-j| \leqslant 2)$ i $a_{ij} = 0$ $(|i-j| > 2)$, imamo slučaj pentodijagonalne matrice.

Pretpostavimo sada da trodijagonalna matrica (1.1.4) ispunjava uslove teoreme 1.1.1. Za dekompoziciju ovakve matrice dovoljno je pretpostaviti da su

$$L = \begin{bmatrix} \beta_1 & 0 & 0 \ldots 0 & 0 \\ \alpha_2 & \beta_2 & 0 & 0 & 0 \\ 0 & \alpha_3 & \beta_3 & 0 & 0 \\ . & & & & \\ . & & & & \\ . & & & & \\ 0 & 0 & 0 & \alpha_n & \beta_n \end{bmatrix} \qquad (\beta_1\beta_2 \ldots \beta_n \neq 0)$$

i

$$R = \begin{bmatrix} 1 & \gamma_1 & 0 \ldots 0 & 0 \\ 0 & 1 & \gamma_2 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ . & & & & \\ . & & & & \\ . & & & & \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Upoređivanjem odgovarajućih elemenata matrica A i matrice

$$LR = \begin{bmatrix} \beta_1 & \beta_1\gamma_1 & 0 & \ldots & 0 & 0 \\ \alpha_2 & \alpha_2\gamma_1+\beta_2 & \beta_2\gamma_2 & & 0 & 0 \\ 0 & \alpha_3 & \alpha_3\gamma_2+\beta_3 & & 0 & 0 \\ . & & & & & \\ . & & & & & \\ . & & & & & \\ 0 & 0 & 0 & & \alpha_n & \alpha_n\gamma_{n-1}+\beta_n \end{bmatrix}$$

dobijamo sledeće rekurzivne formule za određivanje elemenata $\alpha_i, \beta_i, \gamma_i$:

$$\beta_1 = b_1, \qquad \gamma_1 = \frac{c_1}{\beta_1},$$

$$(1.1.5) \quad \alpha_i = a_i, \qquad \beta_i = b_i - \alpha_i\gamma_{i-1}, \qquad \gamma_i = \frac{c_i}{\beta_i} \quad (i = 2, \ldots, n-1),$$

$$\alpha_n = a_n, \qquad \beta_n = b_n - \alpha_n\gamma_{n-1}.$$

4.1.2. Sopstveni vektori i sopstvene vrednosti matrica

Definicija 1.2.1. Neka je A kompleksna kvadratna matrica reda n. Svaki vektor $\vec{x} \in C^n$, koji je različit od nula–vektora, naziva se sopstveni vektor matrice A ako postoji skalar $\lambda \epsilon C$ takav da je

$$(1.2.1) \qquad A\vec{x} = \lambda \vec{x} .$$

Skalar λ naziva se odgovarajuća sopstvena vrednost.
S obzirom da se (1.2.1) može predstaviti u obliku

$$(A - \lambda I) \vec{x} = \vec{0} ,$$

zaključujemo da jednačina (1.2.1) ima netrivijalna rešenja (po $\vec{x}$) ako i samo ako je det $(A - \lambda I) = 0$.

4.2. DIREKTNI METODI U LINEARNOJ ALGEBRI

4.2.1. Uvodne napomene

Numerički problemi u linearnoj algebri mogu se klasifikovati u nekoliko grupa:

1^{o} Rešavanje sistema linearnih algebarskih jednačina

$$A \vec{x} = \vec{b}$$

sa regularnom matricom A, izračunavanje determinante od A i inverzija matrice A;

2^{o} Rešavanje proizvoljnog sistema linearnih jednačina metodom najmanjih kvadrata;

3^{o} Određivanje sopstvenih vrednosti i sopstvenih vektora date kvadratne matrice;

4^{o} Rešavanje zadatka linearnog programiranja.

Za rešavanje ovih problema razvijen je čitav niz metoda, koji se mogu podeliti u dve klase.

Prvu klasu ovih metoda čine tzv. direktni metodi ili kako se ponekad nazivaju tačni metodi. Osnovna karakteristika ovih metoda je ta da se posle konačnog broja transformacija (koraka) dolazi do rezultata. Ukoliko bi se sve računske operacije izvodile tačno, dobijeni rezultat bi bio apsolutno tačan. Naravno, kako se proces računanja izvodi sa zaokrugljivanjem međurezultata, konačan rezultat je ograničene tačnosti.

Drugu klasu metoda čine iterativni metodi, kod kojih se rezultat dobija posle beskonačnog broja koraka. Kao početne vrednosti rešenja, kod primene iterativnih metoda, najčešće se koriste rezultati dobijeni nekim od direktnih metoda. O opštoj teoriji iterativnih procesa bilo je reči u trećoj glavi. U poglavlju 4.3 izložićemo glavne osobine iterativnih metoda koji se koriste u linearnoj algebri. Napomenimo da se kod rešavanja sistema sa velikim brojem jednačina, kakvi se javljaju pri rešavanju parcijalnih diferencijalnih jednačina, koriste uglavnom iterativni metodi.

4.2.2. Gaussov metod eliminacije sa izborom glavnog elementa

Posmatrajmo sistem linearnih algebarskih jednačina

$$
\begin{aligned}
&a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1,\\
&a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2,\\
&\qquad\vdots\\
&a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n,
\end{aligned}
\tag{2.2.1}
$$

ili u matričnom obliku

$$A\vec{x} = \vec{b}, \tag{2.2.2}$$

gde su

$$
A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & & a_{2n} \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ a_{n1} & a_{n2} & & a_{nn} \end{bmatrix}, \quad \vec{b} = \begin{bmatrix} b_1 \\ b_2 \\ \cdot \\ \cdot \\ \cdot \\ b_n \end{bmatrix}, \quad \vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix}.
$$

Za sistem jednačina (2.2.2) pretpostavljamo da ima jedinstveno rešenje.

Poznato je da se rešenja sistema (2.2.1), tj. (2.2.2), mogu izraziti pomoću Cramerovih formula

$$x_i = \frac{\det A_i}{\det A} \qquad (i = 1, \ldots, n),$$

gde je A_i matrica dobijena iz matrice A zamenom i–te kolone vektorom $\vec{b}$. Međutim, ove formule su nepogodne za praktična izračunavanja, s obzirom da je za izračunavanje $n + 1$ determinanata potreban veliki broj računskih operacija. Naime, ako bismo vrednost determinante n–tog reda izračunavali razvijanjem determinante po vrstama ili kolonama, potrebno je izvršiti $S_n = n!-1$ sabiranja i $M_n \cong n!\,(e-1)$ množenja ($n > 4$), što znači da je ukupan broj računskih operacija $P_n = M_n + S_n \cong \cong n!e$. Pod pretpostavkom da je za obavljanje jedne računske operacije potrebno 100 μs, što je slučaj kod brzih računara, to bi za izračunavanje vrednosti determinante tridesetog reda ($n = 30$) bilo potrebno oko $2.3 \cdot 10^{20}$ godina. Uopšteno govoreći ovakav postupak je praktično neprimenljiv, već za determinante reda $n > 5$.

Jedan od najpogodnijih direktnih metoda za rešavanje sistema linearnih jednačina je Gaussov metod eliminacije. Ovaj metod se zasniva na redukciji sistema (2.2.2), primenom ekvivalentnih transformacija, na trougaoni sistem

$$R\vec{x}=\vec{c} \quad , \tag{2.2.3}$$

gde su

$$R=\begin{bmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ & r_{22} & & r_{2n} \\ & & \ddots & \vdots \\ & & & r_{nn} \end{bmatrix} \quad \text{i} \quad \vec{c}=\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}.$$

Sistem (2.2.3) se rešava sukcesivno polazeći od poslednje jednačine. Naime,

$$x_n=\frac{c_n}{r_{nn}},$$

$$x_i=\frac{1}{r_{ii}}\Big(c_i-\sum_{k=i+1}^{n} r_{ik}x_k\Big) \qquad (i=n-1,\dots 1).$$

Napomenimo da su koeficijenti $r_{ii}\neq 0$, jer po pretpostavci sistem (2.2.2), tj. (2.2.3) ima jedinstveno rešenje.

Pokazaćemo sada kako se sistem (2.2.1) može redukovati na ekvivalentan sistem sa trougaonom matricom.

Pod pretpostavkom da je $a_{11}\neq 0$, izračunajmo najpre faktore

$$m_{i1}=\frac{a_{i1}}{a_{11}} \qquad (i=2,\dots,n),$$

a zatim množenjem prve jednačine u sistemu (2.2.1) sa m_{i1} i oduzimanjem od i–te jednačine, dobijamo sistem od $n-1$ jednačina

$$\begin{gathered} a_{22}^{(2)}x_2+\dots+a_{2n}^{(2)}x_n=b_2^{(2)}\ , \\ \vdots \\ a_{n2}^{(2)}x_2+\dots+a_{nn}^{(2)}x_n=b_n^{(2)}\ , \end{gathered} \tag{2.2.4}$$

gde su

$$a_{ij}^{(2)}=a_{ij}-m_{i1}a_{1j}\ , \quad b_i^{(2)}=b_i-m_{i1}b_1 \quad (i,j=2,\dots,n).$$

Pod pretpostavkom da je $a_{22}^{(2)} \neq 0$, primenjujući isti postupak na (2.2.4) sa $m_{i2} = a_{i2}^{(2)}/a_{22}^{(2)}$ $(i = 3, \ldots, n)$ dobijamo sistem od $n-2$ jednačine

$$a_{33}^{(3)} x_3 + \ldots + a_{3n}^{(3)} x_n = b_3^{(3)} ,$$

$$a_{n3}^{(2)} x_3 + \ldots + a_{nn}^{(n)} x_n = b_n^{(3)} ,$$

gde su

$$a_{ij}^{(3)} = a_{ij}^{(2)} - m_{i2} a_{2j}^{(2)} , \quad b_i^{(3)} = b_i^{(2)} - m_{i2} b_2^{(2)} \quad (i, j = 3, \ldots, n).$$

Nastavljajući ovaj postupak, posle $n-1$ koraka dolazimo do jednačine

$$a_{nn}^{(n)} x_n = b_n^{(n)} .$$

Iz dobijenih sistema, uzimanjem prvih jednačina, dolazimo do sistema jednačina

$$\begin{aligned} a_{11}^{(1)} x_1 + a_{12}^{(1)} x_2 + a_{13}^{(1)} x_3 + \ldots + a_{1n}^{(1)} x_n &= b_1^{(1)} , \\ a_{22}^{(2)} x_2 + a_{23}^{(2)} x_3 + \ldots + a_{2n}^{(2)} x_n &= b_2^{(2)} , \\ a_{33}^{(3)} x_3 + \ldots + a_{3n}^{(3)} x_n &= b_3^{(3)} , \\ &\vdots \\ a_{nn}^{(n)} x_n &= b_n^{(n)} , \end{aligned}$$

pri čemu smo stavili $a_{ij}^{(1)} = a_{ij}$, $b_i^{(1)} = b_i$.

Navedena trougaona redukcija ili kako se često kaže Gaussova eliminacija, se svodi na izračunavanje koeficijenata

$$m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}, \quad a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)} , \quad b_i^{(k+1)} = b_i^{(k)} - m_{ik} b_k^{(k)} \quad (i, j = k+1, \ldots, n)$$

za $k = 1, 2, \ldots, n-1$. Primetimo da su elementi matrice R i vektora $\vec{c}$ dati sa

$$r_{ij} = a_{ij}^{(i)} , \quad c_i = b_i^{(i)} \qquad (i = 1, \ldots, n ; \; j = i, \ldots, n) .$$

Da bi navedena redukcija egzistirala, potrebno je obezbediti uslov $a_{kk}^{(k)} \neq 0$. Elementi $a_{kk}^{(k)}$ su poznati kao glavni elementi ili stožerski elementi*. Pod pretpostavkom da je matrica A sistema (2.2.2) regularna, uslove $a_{kk}^{(k)} \neq 0$ moguće je obezbediti permutacijom jednačina u sistemu.

Štaviše, sa stanovišta tačnosti rezultata potrebno je koristiti tzv. strategiju izbora glavnog elementa. Modifikacija Gaussovog eliminacionog metoda u ovom smislu, naziva se Gaussov metod sa izborom glavnog elementa. Prema ovom metodu za glavni element u k–tom eliminacionom koraku uzimamo element $a_{rk}^{(k)}$, za koji je $|a_{rk}^{(k)}| = \max\limits_{k \leqslant i \leqslant n} |a_{ik}^{(k)}|$, uz permutaciju k–te i r–te vrste.

Ako dozvolimo i permutaciju nepoznatih najbolje je za glavni element u k–tom eliminacionom koraku uzeti element $a_{rs}^{(k)}$, za koji je $|a_{rs}^{(k)}| = \max\limits_{k \leqslant i,j \leqslant n} |a_{ij}^{(k)}|$ uz permutaciju k–te i r–te vrste i k–te i s–te kolone. Ovakav postupak se naziva metod sa totalnim izborom glavnog elementa.

Može se pokazati (videti [2]) da ukupan broj računskih operacija u Gaussovom metodu iznosi

$$N(n) = \frac{1}{6}(4n^3 + 9n^2 - 7n).$$

Za dovoljno veliko n imamo $N(n) \cong 2n^3/3$. Dugo vremena se mislilo da je Gaussov metod najoptimalniji u pogledu broja računskih operacija. U novije vreme V. Strassen je, uvodeći iterativni algoritam za množenje i inverziju matrica, dao jedan metod za rešavanje sistema linearnih jednačina, kod koga je broj računskih operacija reda $n^{\log_2 7}$. Strassenov metod je, dakle, optimalniji od Gaussovog metoda ($\log_2 7 < 3$).

Trougaona redukcija obezbeđuje lako izračunavanje determinante sistema. Naime, važi

$$\det A = a_{11}^{(1)} a_{22}^{(2)} \ldots a_{nn}^{(n)}.$$

Ukoliko je korišćen Gaussov metod sa izborom glavnog elementa treba samo voditi računa o broju permutacija vrsta (i kolona kod metoda sa totalnim izborom glavnog elementa), koje utiču na znak determinante. Ovakav način za izračunavanje determinante je veoma efikasan. Na primer, za izračunavanje determinante reda $n = 30$, potrebno je 0.18 s, ako se jedna računska operacija obavlja za 10 μs.

*) Na engleskom jeziku pivotal element, ili prosto pivot.

4.2.3. Inverzija matrica pomoću Gaussovog metoda

Neka je $A = [a_{ij}]_{n \times n}$ regularna matrica i neka je

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & & x_{2n} \\ . & & & \\ . & & & \\ . & & & \\ x_{n1} & x_{n2} & & x_{nn} \end{bmatrix} = [\vec{x}_1\ \vec{x}_2 \dots \vec{x}_n]$$

njena inverzna matrica. Vektori $\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n$ su redom prva, druga, ..., n–ta kolona matrice X. Definišimo vektore $\vec{e}_1, \vec{e}_2, \dots, \vec{e}_n$ pomoću

$$\vec{e}_1 = [1\ 0 \dots 0]^T,\ \vec{e}_2 = [0\ 1 \dots 0]^T, \dots, \vec{e}_n = [0\ 0 \dots 1]^T.$$

S obzirom na jednakost $AX = [A\vec{x}_1\ A\vec{x}_2 \dots A\vec{x}_n] = I = [\vec{e}_1\ \vec{e}_2 \dots \vec{e}_n]$, problem određivanja inverzne matrice može se svesti na rešavanje n sistema linearnih jednačina

$$(2.3.1) \qquad A\vec{x}_i = \vec{e}_i \qquad (i = 1, \dots, n).$$

Za rešavanje sistema (2.3.1) pogodno je koristiti Gaussov metod, s obzirom da se matrica A pojavljuje kao matrica svih sistema, pa njenu trougaonu redukciju treba izvršiti samo jednom. Pri ovome sve elementarne transformacije koje su potrebne za trougaonu redukciju matrice A treba primeniti i na jediničnu matricu $I = [\vec{e}_1\ \vec{e}_2 \dots \vec{e}_n]$. Na taj način se matrica A transformiše u trougaonu matricu R, a matrica I u matricu $C = [\vec{c}_1\ \vec{c}_2 \dots \vec{c}_n]$. Najzad, ostaje da se reše trougaoni sistemi

$$R\vec{x}_i = \vec{c}_i \qquad (i = 1, \dots, n).$$

4.2.4. Faktorizacioni metodi

Faktorizacioni metodi za rešavanje sistema linearnih jednačina zasnivaju se na razlaganju matrice sistema na proizvod dve matrice čiji je oblik takav da omogućava svođenje sistema na dva sistema jednačina koji se jednostavno sukcesivno rešavaju. U ovom odeljku ukazaćemo na metode zanovane na LR faktorizaciji matrice (videti odeljak 4.1.1).

Neka je dat sistem jednačina

$$(2.4.1) \qquad A\vec{x} = \vec{b},$$

sa kvadratnom matricom A, čiji su svi glavni dijagonalni minori različiti od nule. Tada, na osnovu teoreme 1.1.1, postoji faktorizacija matrice $A = LR$, gde je L donja i R gornja trougaona matrica. Faktorizacija je jednoznačno određena, ako se, na primer, usvoji da matrica L ima jediničnu dijagonalu. U tom slučaju, sistem (2.4.1), tj. sistem $LR\vec{x} = \vec{b}$, se može predstaviti u ekvivalentnom obliku

(2.4.2) $\qquad L\vec{y} = \vec{b}, \qquad R\vec{x} = \vec{y}.$

Na osnovu prethodnog, za rešavanje sistema jednačina (2.4.1), može se formulisati sledeći metod:

1° Stavimo $l_{ii} = 1$ $(i = 1, \ldots, n)$;

2° Odredimo ostale elemente matrice $L = [\,l_{ij}]_{n \times n}$ i matrice $R = [\,r_{ij}]_{n \times n}$ (videti odeljak 4.1.1);

3° Rešimo prvi sistem jednačina u (2.4.2);

4° Rešimo drugi sistem jednačina u (2.4.2).

Koraci 3° i 4° se jednostavno izvode. Naime, neka su

$$\vec{b} = [b_1\ b_2 \ldots b_n]^{\mathrm{T}}, \vec{y} = [y_1\ y_2 \ldots y_n]^{\mathrm{T}}, \vec{x} = [x_1\ x_2 \ldots x_n]^{\mathrm{T}}.$$

Tada je

$$y_1 = b_1,\ y_i = b_i - \sum_{k=1}^{i-1} l_{ik} y_k \qquad (i = 2, \ldots, n)$$

i

$$x_n = \frac{y_n}{r_{nn}},\ x_i = \frac{1}{r_{ii}}\Big(y_i - \sum_{k=i+1}^{n} r_{ik} x_k\Big) \qquad (i = n-1, \ldots, 1).$$

Izloženi metod se u literaturi sreće kao metod Haleckog. U slučaju kada je matrica A normalna, tj. kada je simetrična i pozitivno definitna, metod Haleckog se može uprostiti. Naime, tada se može uzeti da je $L = R^{\mathrm{T}}$. Dakle, treba odrediti faktorizaciju matrice A u obliku $A = R^{\mathrm{T}} R$. Na osnovu formula iz odeljka 4.1.1 za elemente matrice R važe formule

$$r_{11} = \sqrt{a_{11}},\ r_{1j} = \frac{a_{1j}}{r_{11}} \qquad (j = 2, \ldots, n),$$

$$r_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} r_{ki}^2}$$

$$(i = 2, \ldots, n).$$

$$r_{ij} = \frac{1}{r_{ii}}\Big(a_{ij} - \sum_{k=1}^{i-1} r_{ki} r_{kj}\Big) \qquad (j = i+1, \ldots, n)$$

U ovom slučaju sistemi (2.4.2) postaju

$$R^{\mathrm{T}}\vec{y} = \vec{b}, \qquad R\vec{x} = \vec{y}.$$

Primedba 2.4.1. Determinanta normalne matrice se može izračunati po metodi kvadratnog korena kao

$$\det A = (r_{11} r_{22} \dots r_{nn})^2 .$$

Faktorizacioni metodi su naročito pogodni za rešavanje sistema linearnih jednačina, kod kojih se matrica sistema ne menja, već samo vektor slobodnih članova $\vec{b}$. Ovakvi sistemi se često javljaju u tehnici.

Sada ćemo pokazati da se Gaussov metod eliminacije može interpretirati kao LR faktorizacija matrice A. Uzmimo matricu A takvu, da prilikom eliminacije ne treba vrširi permutaciju vrsta i kolona. Polazni sistem označimo sa $A^{(1)} \vec{x} = \vec{b}^{(1)}$. Gaussov eliminacioni postupak daje $n-1$ ekvivalentnih sistema $A^{(2)} \vec{x} = \vec{b}^{(2)}, \dots, A^{(n)} \vec{x} = \vec{b}^{(n)}$, pri čemu matrica $A^{(k)}$ ima oblik

$$A^{(k)} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} \dots & a_{1k}^{(1)} \dots & a_{1n}^{(1)} \\ & a_{22}^{(2)} & a_{2k}^{(2)} & a_{2n}^{(2)} \\ & \ddots & \vdots & \\ & & a_{kk}^{(k)} & a_{kn}^{(k)} \\ & & \vdots & \\ & & a_{nk}^{(k)} & a_{nn}^{(k)} \end{bmatrix}$$

Analizirajmo modifikaciju elementa a_{ij} $(= a_{ij}^{(1)})$ u procesu trougaone redukcije. Kako je, za $k = 1,2, \dots, n-1$,

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)} \qquad (i, j = k+1, \dots, n),$$

i

$$a_{i1}^{(k+1)} = a_{i2}^{(k+1)} = \dots = a_{ik}^{(k+1)} = 0 \qquad (i = k+1, \dots, n),$$

sumiranjem dobijamo

$$a_{ij} = a_{ij}^{(1)} = a_{ij}^{(i)} + \sum_{k=1}^{i-1} m_{ik} a_{kj}^{(k)} \qquad (i \leqslant j)$$

i

$$a_{ij} = a_{ij}^{(1)} = 0 + \sum_{k=1}^{j} m_{ik} a_{kj}^{(k)} \qquad (i > j).$$

Definišući $m_{ii} = 1$ $(i = 1, \ldots, n)$, poslednje dve jednakosti se mogu predstaviti u obliku

$$(2.4.3) \qquad a_{ij} = \sum_{k=1}^{p} m_{ik} a_{kj}^{(k)} \qquad (i, j = 1, \ldots, n),$$

gde je $p = \min\,(i, j)$. Jednakost (2.4.3) ukazuje da Gaussova eliminacija daje LR faktorizaciju matrice A, gde su

$$L = \begin{bmatrix} 1 & & & \\ m_{21} & 1 & & \\ & \cdot & & \\ & & \cdot & \\ m_{n1} & m_{n2} & \ldots & 1 \end{bmatrix}, \quad R = \begin{bmatrix} r_{11} & r_{12} & \ldots & r_{1n} \\ & r_{22} & & r_{2n} \\ & & \cdot & \\ & & & \cdot \\ & & & r_{nn} \end{bmatrix}$$

i $r_{kj} = a_{kj}^{(k)}$. Pri programskoj realizaciji Gaussovog metoda u cilju dobijanja LR faktorizacije matrice A, nije potrebno koristiti nove memorijske elemente za pamćenje matrice L, već je pogodno faktore m_{ik} smeštati na mesto koeficijenata matrice A koji se anuliraju u procesu trougaone redukcije. Na taj način, posle završene trougaone redukcije, na mesto matrice A biće memorisane matrice L i R, prema sledećoj šemi

Uočimo da se dijagonalni elementi matrice L, koji su svi jednaki jedinici, ne moraju memorisati.

Metod Haleckog, zasnovan na LR faktorizaciji, primenjuje se u slučajevima kada matrica A ispunava uslove teoreme 1.1.1. Međutim, primenljivost ovog metoda može se proširiti i na druge sisteme sa regularnom matricom, uzimajući u obzir permutaciju jednačina u sistemu. Za faktorizaciju iskoristimo Gaussov eliminacioni metod sa izborom glavnog elementa. Pri ovome biće $LR = A'$, gde se matrica A' dobija iz matrice A konačnim brojem razmena vrsta. Ovo znači da u procesu eliminacije treba memorisati niz indeksa glavnih elemenata $I = (p_1, \ldots, p_{n-1})$, pri čemu je p_k broj vrste iz koje se uzima glavni element u k–tom eliminacionom koraku. Kod rešavanja sistema $A \cdot \vec{x} = \vec{b}$, neposredno posle faktorizacije treba, u skladu sa nizom indeksa I, permutovati koordinate vektora $\vec{b}$. Na taj način se dobija transformisani vektor $\vec{b}'$, pa se rešavanje datog sistema svodi na sukcesivno rešavanje trougaonih sistema

$$L\vec{y} = \vec{b}' \quad \text{i} \quad R\vec{x} = \vec{y}.$$

4.3. ITERATIVNI METODI U LINEARNOJ ALGEBRI

4.3.1. Uvod

Posmatrajmo sistem linearnih jednačina

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 , \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 , \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n , \end{aligned} \tag{3.1.1}$$

koji se može predstaviti i u matričnom obliku

$$A\vec{x} = \vec{b}, \tag{3.1.2}$$

gde su

$$A = \begin{bmatrix} a_{11} & a_{12} \ldots & a_{1n} \\ a_{21} & a_{22} & a_{2n} \\ . & & \\ . & & \\ . & & \\ a_{n1} & a_{n2} & a_{nn} \end{bmatrix}, \quad \vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{bmatrix}, \quad \vec{b} = \begin{bmatrix} b_1 \\ b_2 \\ . \\ . \\ . \\ b_n \end{bmatrix}.$$

Uvek u ovom poglavlju, pretpostavljamo da sistem (3.1.1), tj. (3.1.2) ima jedinstveno rešenje.

Iterativni metodi za rešavanje sistema (3.1.2) imaju za cilj određivanje rešenja $\vec{x}$ sa unapred zadatom tačnošću. Naime, polazeći od proizvoljnog vektora $\vec{x}^{(0)}$ ($= [\, x_1^{(0)} \ldots x_n^{(0)}]^T$), iterativnim metodama se određuje niz $\vec{x}^{(k)}$ ($\vec{x}^{(k)} = [\, x_1^{(k)} \ldots x_n^{(k)}\,]^T$) takav da je

$$\lim_{k \to +\infty} \vec{x}^{(k)} = \vec{x} .$$

4.3.2. Metod proste iteracije

Jedan od najprostijih metoda za rešavanje sistema linearnih jednačina je metod proste iteracije. Za primenu ovog metoda, potrebno je prethodno sistem (3.1.2) predstaviti u ekvivalentnom obliku

$$\vec{x} = B\vec{x} + \vec{\beta} . \tag{3.2.1}$$

Tada je metod proste iteracije dat sa

$$\vec{x}^{(k)} = B\vec{x}^{(k-1)} + \vec{\beta} \qquad (k = 1,2, \ldots). \tag{3.2.2}$$

Ako se pođe od proizvoljnog vektora $\vec{x}^{(0)}$, pomoću (3.2.2) generiše se niz $\{\vec{x}^{(k)}\}$, koji pod izvesnim uslovima konvergira rešenju datog sistema.

Ako je

$$B=\begin{bmatrix} b_{11} & b_{12} \ldots & b_{1n} \\ b_{21} & b_{22} & b_{2n} \\ . & & \\ . & & \\ . & & \\ b_{n1} & b_{n2} & b_{nn} \end{bmatrix} \quad \text{i} \quad \beta=\begin{bmatrix} \beta_1 \\ \beta_2 \\ . \\ . \\ . \\ \beta_n \end{bmatrix}$$

iterativni metod (3.2.2) može se predstaviti skalarno

$$x_1^{(k)} = b_{11}x_1^{(k-1)} + \ldots + b_{1n}x_n^{(k-1)} + \beta_1 ,$$

$$x_2^{(k)} = b_{21}\, x_1^{(k-1)} + \ldots + b_{2n}x_n^{(k-1)} + \beta_2 ,$$

$$\vdots$$

$$x_n^{(k)} = b_{n1}x_1^{(k-1)} + \ldots + b_{nn}x_n^{(k-1)} + \beta_n ,$$

gde je $k = 1,2, \ldots$.

Može se pokazati (videti [2]) da iterativni proces (3.2.2) konvergira ako su sve sopstvene vrednosti matrice B po modulu manje od jedinice. S obzirom da je određivanje sopstvenih vrednosti matrice dosta komplikovano to se kod praktične primene metoda proste iteracije ispituju samo tzv. dovoljni uslovi za konvergenciju. Naime, za matricu B se mogu definisati različite norme, kao na primer,

$$\| B \|_1 = (\sum_{i,j} b_{ij}^2)^{1/2} ,$$

(3.2.3) $$\| B \|_2 = \max_i \sum_{j=1}^{n} |b_{ij}| ,$$

$$\| B \|_3 = \max_j \sum_{i=1}^{n} |b_{ij}|$$

Nije teško pokazati da iterativni proces (3.2.2) konvergira ako je $\| B \| < 1$, pri proizvoljnom početnom vektoru $\vec{x}^{(0)}$.

4.3.3. Gauss–Seidelov metod

Gauss–Seidelov metod se dobija modifikacijom metoda proste iteracije. Kao što smo ranije videli, kod metoda proste iteracije, vrednost i–te komponente $x_i^{(k)}$

vektora $\vec{x}^{(k)}$ izračunava se na osnovu vrednosti $x_1^{(k-1)}, \ldots, x_n^{(k-1)}$, tj.

$$x_i^{(k)} = \sum_{j=1}^{n} b_{ij} x_j^{(k-1)} + \beta_i \qquad (i = 1, \ldots, n;\ k = 1,2, \ldots).$$

Ovaj metod može se modifikovati na taj način što bi se za izračunavanje vrednosti $x_1^{(k)}$ koristile vrednosti $x_1^{(k)}, \ldots, x_{i-1}^{(k)}, x_i^{(k-1)}, \ldots, x_n^{(k-1)}$, tj.

$$(3.3.1) \quad x_i^{(k)} = \sum_{j=1}^{i-1} b_{ij} x_j^{(k)} + \sum_{j=i}^{n} b_{ij} x_j^{(k-1)} + \vec{\beta}_i \quad (i = 1, \ldots, n;\ k = 1,2, \ldots).$$

Navedena modifikacija metoda proste iteracije poznata je kao Gauss–Seidelov metod.

Iterativni proces (3.3.1) može se predstaviti i u matričnoj formi. Naime, neka je

$$B = B_1 + B_2,$$

gde su

$$B_1 = \begin{bmatrix} 0 & 0 \ldots & 0 & 0 \\ b_{21} & 0 & 0 & 0 \\ . & & & \\ . & & & \\ . & & & \\ b_{n1} & b_{n2} & b_{n,n-1} & 0 \end{bmatrix} \quad \text{i } B_2 = \begin{bmatrix} b_{11} & b_{12} \ldots & b_{1n} \\ 0 & b_{22} & b_{2n} \\ . & & \\ . & & \\ . & & \\ 0 & 0 & b_{nn} \end{bmatrix}.$$

Tada (3.3.1) postaje

$$(3.3.2) \qquad \vec{x}^{(k)} = B_1 \vec{x}^{(k)} + B_2 \vec{x}^{(k-1)} + \vec{\beta} \qquad (k = 1,2, \ldots).$$

Teorema 3.3.1. Pri proizvoljnom vektoru $\vec{x}^{(0)}$, iterativni proces (3.3.2) konvergira ako i samo ako su svi koreni jednačine

$$\det [B_2 - (I - B_1)\lambda] = \begin{bmatrix} b_{11} - \lambda & b_{12} \ldots & b_{1n} \\ b_{21}\lambda & b_{22} - \lambda & b_{2n} \\ . & & \\ . & & \\ . & & \\ b_{n1}\lambda & b_{n2}\lambda & b_{nn} - \lambda \end{bmatrix} = 0$$

po modulu manji od jedinice.

4.4. PROGRAMSKA REALIZACIJA

Ovo poglavlje je posvećeno programskoj realizaciji metoda izloženih u ovoj glavi. Za uspešno praćenje materije u okviru ovog poglavlja neophodno je poznavanje celokupne materije izložene u prethodnim poglavljima ove glave.

U svim potprogramima koje dajemo, matrice se tretiraju kao vektori.

4.4.1. Potprogram za transponovanje matrice MTRN ima oblik

```
      SUBROUTINE MTRN(A,B,N,M)
C
C     TRANSPONOVANJE MATRICE A
C
      DIMENSION A(1),B(1)
      IC=0
      DO 5 I=1,N
      IJ=I-N
      DO 5 J=1,M
      IJ=IJ+N
      IC=IC+1
    5 B(IC)=A(IJ)
      RETURN
      END
```

Parametri u potprogramskoj listi imaju sledeće značenje:

A ulazna matrica tipa NxM, koja se tretira kao vektor dužine NM (uzet kolona po kolona);

B izlazna matrica tipa MxN ($B = A^T$). Tretman matrice B je isti kao i tretman matrice A.

4.4.2. Potprogram za množenje matrica A (tipa NxM) i B (tipa MxL) ima oblik

```
      SUBROUTINE MMAT(A,B,C,N,M,L)
C
C     MATRICA  A TIPA N*M
C     MATRICA  B TIPA M*L
C     MATRICA  C TIPA N*L
C     MNOZENJE MATRICA  C=A*B
C
      DIMENSION A(1),B(1),C(1)
      IC=0
      I2=-M
      DO 5 J=1,L
      I2=I2+M
      DO 5 I=1,N
      IC=IC+1
      IA=I-N
      IB=I2
      C(IC)=0.
      DO 5 K=1,M
      IA=IA+N
      IB=IB+1
    5 C(IC)=C(IC)+A(IA)*B(IB)
      RETURN
      END
```

pri čemu je C izlazna matrica ($C = AB$) tipa NxL.

4.4.3. Sastavimo program za određivanje matrice $B^T A$, korišćenjem prethodnih potprograma, ako su matrice A i B date. Neka je matrica A tipa NxM, a matrica B tipa

$N \times K$ (broj elemenata jedne i druge matrice nije veći od 100).

Program ima oblik

```
      DIMENSION A(100),B(100),C(100)
      READ(8,10) N,M,K
   10 FORMAT(3I2)
      NM=N*M
      NK=N*K
      KM=K*M
      READ(8,20) (A(I),I=1,NM),(B(I),I=1,NK)
   20 FORMAT(16F5.0)
      CALL MTRN(B,C,N,K)
      CALL MMAT(C,A,B,K,N,M)
      WRITE(5,30) ((B(J),J=I,KM,K),I=1,K)
   30 FORMAT(5X,'MATRICA C=B(TR)*A'//(2X,<M>F6.1))
      CALL EXIT
      END
```

Testirajući ovaj program sa matricama

$$A = \begin{bmatrix} -1 & 3 & 0 & 2 \\ 1 & 4 & 1 & 5 \\ 0 & 1 & -2 & 0 \\ -2 & 3 & 1 & 3 \end{bmatrix} \quad \text{i } B = \begin{bmatrix} 1 & -3 & 0 \\ 0 & 4 & -6 \\ 2 & -1 & 2 \\ -1 & 5 & 1 \end{bmatrix},$$

dobijamo sledeći rezultat:

```
  MATRICA C=B(TR)*A

  1.0    2.0    -5.0   -1.0
 -3.0   21.0    11.0   29.0
 -8.0  -19.0    -9.0  -27.0
```

4.4.4. Metod Haleckog za rešavanje sistema linearnih jednačina (videti odeljak 4.2.4) može se programski realizovati na sledeći način:

```
C=====================================================================
C                          METOD HALECKOG
C=====================================================================
      DIMENSION A(10,10),B(10)
   33 READ(8,100) N
  100 FORMAT(I2)
      IF(N)11,22,11
   11 READ(8,101) (B(I),I=1,N)
  101 FORMAT(8F10.4)
      READ(8,101) ((A(I,J),J=1,N),I=1,N)
      WRITE(5,102)
  102 FORMAT(///5X,'MATRICA A',<(N-1)*12+3>X,'VEKTOR B' /)
      WRITE(5,103) ((A(I,J),J=1,N),B(I),I=1,N)
  103 FORMAT(1X,<N>F12.7,F13.7)
```

```
C FAKTORIZACIJA MATRICE A U OBLIKU A=L*R
      DO 10 I=2,N
   10 A(1,I)=A(1,I)/A(1,1)
      DO 25 I=2,N
      I1=I-1
      S=A(I,I)
      DO 20 K=1,I1
   20 S=S-A(I,K)*A(K,I)
      A(I,I)=S
      IF(I.EQ.N) GO TO 40
      IJ=I+1
      DO 25 J=IJ,N
      S=A(I,J)
      T=A(J,I)
      DO 30 K=1,I1
      S=S-A(I,K)*A(K,J)
   30 T=T-A(J,K)*A(K,I)
      A(I,J)=S/A(I,I)
   25 A(J,I)=T
   40 WRITE(5,107)
  107 FORMAT(//5X,'MATRICA L'/)
      DO 111 I=1,N
  111 WRITE(5,103)(A(I,J),J=1,I)
      WRITE(5,108)
  108 FORMAT(//5X,'MATRICA R'/)
      N1=N-1
      DO 222 I=1,N1
      II=I+1
      M=N-I
  222 WRITE(5,99)(A(I,J),J=II,N)
      WRITE(5,99)
   99 FORMAT(<12*I-8>X,'1.0000000',<M>F12.7)
C NALAZENJE VEKTORA RESENJA
      B(1)=B(1)/A(1,1)
      DO 55 I=2,N
      I1=I-1
      DO 45 K=1,I1
   45 B(I)=B(I)-A(I,K)*B(K)
   55 B(I)=B(I)/A(I,I)
      DO 50 J=1,N1
      I=N-J

      I1=I+1
      DO 50 K=I1,N
   50 B(I)=B(I)-A(I,K)*B(K)
      WRITE(5,109)
  109 FORMAT(//13X,'VEKTOR RESENJA'/)
      WRITE(5,104) (B(I),I=1,N)
  104 FORMAT(12X,F12.7  )
      GO TO 33
   22 CALL EXIT
      END
```

Kod faktorizacije matrice A ($= LR$) uzeli smo u gornje trougaonoj matrici R jediničnu dijagonalu, tj. $r_{ii} = 1$ ($i = 1, \ldots, n$). Program je organizovan tako da se matrica A transformiše u matricu A_1, čiji se donji trougao (uključujući i glavnu dijagonalu) poklapa sa matricom L, a strogo gornji trougao sa matricom R. Primetimo da se dijagonalni elementi u matrici R ne pamte, već se samo kod štampanja, pomoću naredbe FORMAT formalno štampaju. Takođe, primetimo da je u odeljku 4.2.4. uzeta jedinična dijagonala u matrici L.

Primenom ovog programa na jedan konkretan sistem jednačina dobijaju se sledeći rezultati:

```
MATRICA A                                                          VEKTOR B

1.0000000    4.0000000    1.0000000    3.0000000     9.0000000
0.0000000   -1.0000000    2.0000000   -1.0000000     0.0000000
3.0000000   14.0000000    4.0000000    1.0000000    22.0000000
1.0000000    2.0000000    2.0000000    9.0000000    14.0000000

MATRICA L

1.0000000
0.0000000   -1.0000000
3.0000000    2.0000000    5.0000000
1.0000000   -2.0000000   -3.0000000    2.0000000

MATRICA R

1.0000000    4.0000000    1.0000000    3.0000000
             1.0000000   -2.0000000    1.0000000
                          1.0000000   -2.0000000
                                       1.0000000

          VEKTOR RESENJA

            1.0000000
            1.0000000
            1.0000000
            1.0000000
```

4.4.5. Slično se realizuje i metod kvadratnog korena za rešavanje sistema linearnih jednačina sa simetričnom pozitivno definitnom matricom. U ovom slučaju dovoljno je učitati samo elemente matrice A sa glavne dijagonale i, na primer, elemente iz gornjeg trougla.

Program i izlazna lista za konkretan sistem jednačina su dati u daljem tekstu. Napomenimo da je sa stanovišta uštede memorijskog prostora pogodnije matricu A tretirati kao vektor. Međutim, zbog lakšeg razumevanja čitaoca, na ovom mestu, nismo poštovali ovu pogodnost.

Organizacija programa je takva da se pored rešavanja sistema jednačina izračunava i determinanta matrice sistema. U izlaznoj listi donji trougao simetrične matrice A je izostavljen.

```
C================================================================
C  RESAVANJE SISTEMA LINEARNIH JEDNACINA METODOM KVADRATNOG
C     KORENA
C================================================================
         DIMENSION A(10,10),B(10)
      3 READ(8,100) N
    100 FORMAT(I2)
        IF(N) 1,2,1
C UCITAVANJE VEKTORA B
      1 READ(8,101) (B(I),I=1,N)
    101 FORMAT(8F10.4)
```

```
C  UCITAVANJE GORNJEG TROUGLA MATRICE A
      READ(8,101) ((A(I,J),J=I,N),I=1,N)
      WRITE(5,102)
  102 FORMAT(///5X,'MATRICA SISTEMA    '/)
      WRITE(5,99) ((A(I,J),J=I,N),I=1,N)
   99 FORMAT(<12*I-11>X,<N-I+1>F12.7)
      WRITE(5,105)
  105 FORMAT(//5X,'VEKTOR SLOBODNIH CLANOVA    '/)
      WRITE(5,133) (B(I),I=1,N)
  133 FORMAT(1X,10F12.7)
C NALAZENJE ELEMENATA GORNJE TROUGAONE MATRICE
      A(1,1)=SQRT(A(1,1))
      DO 11 J=2,N
   11 A(1,J)=A(1,J)/A(1,1)
      DO 12 I=2,N
      S=0.
       IM1=I-1
      DO 13 K=1,IM1
   13 S=S+A(K,I)*A(K,I)
      A(I,I)=SQRT(A(I,I)-S)
      IF(I-N) 29,12,29
   29 IP1=I+1
      DO 14 J=IP1,N
      S=0.
      DO 15 K=1,IM1
   15 S=S+A(K,I)*A(K,J)
   14 A(I,J)=(A(I,J)-S)/A(I,I)
   12 CONTINUE
C  IZRACUNAVANJE DETERMINANTE SISTEMA
      DET=1.
      DO 60 I=1,N
   60 DET=DET*A(I,I)
      DET=DET*DET
C RESAVANJE SISTEMA L*Y=B
      B(1)=B(1)/A(1,1)
      DO 7 I=2,N
      IM1=I-1
      S=0.
      DO 8 K=1,IM1
    8 S=S+A(K,I)*B(K)
      P=1./A(I,I)
    7 B(I)=P*(B(I)-S)
C RESAVANJE SISTEMA R*X=Y
C REZULTAT SE SMESTA U VEKTOR B


      B(N)=B(N)/A(N,N)
      NM1=N-1
      DO 30 II=1,NM1
      JJ=N-II
      S=0.
      JJP1=JJ+1
      DO 50 K=JJP1,N
   50 S=S+A(JJ,K)*B(K)
   30 B(JJ)=(B(JJ)-S)/A(JJ,JJ)
C
C STAMPANJE REZULTATA
      WRITE(5,201)
  201 FORMAT(//5X,'MATRICA R'/)
      DO 222 I=1,N
  222 WRITE(5,99)(A(I,J),J=I,N)
      WRITE(5,208) DET
  208 FORMAT(//5X,'DETERMINANTA SISTEMA D=',F11.7/)
      WRITE(5,109)
```

```
109 FORMAT(//5X,'RESENJE SISTEMA    '/)
    WRITE(5,133) (B(I),I=1,N)
    GO TO 3
  2 CALL EXIT
    END
```

```
     MATRICA SISTEMA

    3.0000000    0.0000000    1.0000000
                 2.0000000    1.0000000
                              1.0000000


     VEKTOR SLOBODNIH CLANOVA

    4.0000000    3.0000000    3.0000000


     MATRICA R

    1.7320509    0.0000000    0.5773503
                 1.4142135    0.7071068
                              0.4082483


     DETERMINANTA SISTEMA D=  1.0000002



     RESENJE SISTEMA

    0.9999999    0.9999998    1.0000002



     MATRICA SISTEMA

    1.0000000    0.0000000    1.0000000
                 4.0000000    2.0000000
                              9.0000000


     VEKTOR SLOBODNIH CLANOVA

    2.0000000    6.0000000   12.0000000


     MATRICA R

    1.0000000    0.0000000    1.0000000
                 2.0000000    1.0000000
                              2.6457515


     DETERMINANTA SISTEMA D= 28.0000038



     RESENJE SISTEMA

    1.0000001
```

4.4.6. Faktorizacioni metod za rešavanje sistema linearnih jednačina baziran na Gaussovoj eliminaciji sa izborom glavnog elementa (videti odeljke 4.2.2 i 4.2.4) može se programski realizovati pomoću sledećih potprograma:

```
      SUBROUTINE LRFAK(A,N,IP,DET,KB)
      DIMENSION A(1),IP(1)
      KB=0
      N1=N-1
      INV=0
      DO 45 K=1,N1
      IGE=(K-1)*N+K
C
C     NALAZENJE GLAVNOG ELEMENTA U K-TOM
C     ELIMINACIONOM KORAKU
C
      GE=A(IGE)
      I1=IGE+1
      I2=K*N
      IMAX=IGE
      DO 20 I=I1,I2
      IF(ABS(A(I))-ABS(GE)) 20,20,10
   10 GE=A(I)
      IMAX=I
   20 CONTINUE
      IF(GE)25,15,25
   15 KB=1
C
C     MATRICA SISTEMA JE SINGULARNA
C
      RETURN
   25 IP(K)=IMAX-N*(K-1)
      IF(IP(K)-K) 30,40,30
   30 I=K
      IK=IP(K)
C
C     PERMUTACIJA VRSTA U MATRICI
C
      DO 35 J=1,N
      S=A(I)
      A(I)=A(IK)
      A(IK)=S
      I=I+N
   35 IK=IK+N
      INV=INV+1
C
C     K-TI ELIMINACIONI KORAK
C
   40 DO 45 I=I1,I2
      A(I)=A(I)/GE
      IA=I
      IC=IGE
      K1=K+1
      DO 45 J=K1,N
      IA=IA+N
      IC=IC+N
   45 A(IA)=A(IA)-A(I)*A(IC)
C
C     IZRACUNAVANJE DETERMINANTE
C
      DET=1.
      DO 50 I=1,N
      IND=I+(I-1)*N
```

```
50 DET=DET*A(IND)
   IF(INV-INV/2*2) 55,55,60
60 DET=-DET
55 RETURN
   END


      SUBROUTINE RSTS(A,N,IP,B)
      DIMENSION A(1),IP(1),B(1)
C
C     SUKCESIVNO RESAVANJE TROUGAONIH SISTEMA
C
      N1=N-1
C     PERMUTACIJA VEKTORA B
      DO 10 I=1,N1
      I1=IP(I)
      IF(I1-I)5,10,5
    5 S=B(I)
      B(I)=B(I1)
      B(I1)=S
   10 CONTINUE
C     RESAVANJE DONJE TROUGAONOG SISTEMA
      DO 15 K=2,N
      IA=-N+K
      K1=K-1
      DO 15 I=1,K1
      IA=IA+N
   15 B(K)=B(K)-A(IA)*B(I)
C     RESAVANJE GORNJE TROUGAONOG SISTEMA
      NN=N*N
      B(N)=B(N)/A(NN)
      DO 25 KK=1,N1
      K=N-KK
      IA=NN-KK
      I=N+1
      DO 20 J=1,KK
      I=I-1
      B(K)=B(K)-A(IA)*B(I)
   20 IA=IA-N
   25 B(K)=B(K)/A(IA)
      RETURN
      END
```

Parametri u potprogramskoj listi kod potprograma LRFAK imaju sledeće značenje:

A – Ulazna matrica reda N memorisana kao niz kolona po kolona. Posle N–1 eliminacionih koraka matrica A se transformiše u matricu koja sadrži trougaone matrice L i R (videti odeljak 4.2.4);

N – red matrice A;

IP – vektor dužine N–1, koji se formira u procesu eliminacije i predstavlja niz indeksa glavnih elemenata (videti odeljak 4.2.4).

DET – izlazna veličina koja daje vrednost determinante matrice sistema A, kao proizvod elemenata na glavnoj dijagonali u matrici R, sa tačnošću do na znak. Ova vrednost se koriguje znakom, na kraju potprograma, imajući u vidu broj permutacija vrsta u matrici u toku eliminacionog procesa.

KB – kontrolni broj sa vrednostima KB = 0 ako je faktorizacija korektno izvedena i KB = 1 ako je matrica sistema singularna. U poslednjem slučaju LR faktorizacija ne egzistira.

Potprogram *RSTS* sukcesivno rešava sisteme jednačina (2.4.4). Parametri u potprogramskoj listi imaju sledeće značenje:

A – matrica dobijena u potprogramu LRFAK;

N – red matrice *A*;

IP – vektor dobijen u potprogramu LRFAK;

B – vektor slobodnih članova u sistemu jednačina koji se rešava. Ovaj vektor se transformiše u vektor rešenja datog sistema.

Glavni program je organizovan tako da se, najpre, data matrica *A* faktorizuje, pomoću potprograma LRFAK, a zatim je mogućno rešiti sistem jednačina $A\vec{x}=\vec{b}$ za proizvoljan broj različitih vektora $\vec{b}$, pozivanjem potprograma RSTS. Glavni program i izlazna lista imaju oblik:

```
      DIMENSION A(100),B(10),IP(9)
      READ(8,5) N
    5 FORMAT(I2)
      NN=N*N
      READ(8,10) (A(I),I=1,NN)
   10 FORMAT(16F5.0)
      WRITE(5,34)
   34 FORMAT(1H1,5X,'MATRICA A'/)
      DO 12 I=1,N
   12 WRITE(5,15) (A(J),J=I,NN,N)
   15 FORMAT(10F10.5)
      CALL LRFAK(A,N,IP,DET,KB)
      IF(KB) 20,25,20
   20 WRITE(5,30)
   30 FORMAT(1H0,'MATRICA A JE SINGULARNA'//)
      GO TO 70
   25 WRITE(5,35)
   35 FORMAT(1H0,5X,'FAKTORIZOVANA MATRICA'/)
      DO 55 I=1,N
   55 WRITE(5,15) (A(J),J=I,NN,N)
      WRITE(5,75) DET
   75 FORMAT(/5X,'DETERMINANTA MATRICE A ='F10.6/)
   50 READ(8,10,END=70) (B(I),I=1,N)
      WRITE(5,40) (B(I),I=1,N)
   40 FORMAT(/5X,'VEKTOR B'//(10F10.5))
      CALL RSTS(A,N,IP,B)
      WRITE(5,45) (B(I),I=1,N)
   45 FORMAT(/5X,'RESENJE'//(10F10.5))
      GO TO 50
   70 CALL EXIT
      END
```

```
     MATRICA A

  3.00000   1.00000   6.00000
  2.00000   1.00000   3.00000
  1.00000   1.00000   1.00000

     FAKTORIZOVANA MATRICA

  3.00000   1.00000   6.00000
  0.33333   0.66667  -1.00000
  0.66667   0.50000  -0.50000

    DETERMINANTA MATRICE A =  1.000000
```

```
VEKTOR B

2.00000   7.00000   4.00000

RESENJE

19.00000  -7.00000  -8.00000

VEKTOR B

1.00000   1.00000   1.00000

RESENJE

0.00000   1.00000   0.00000
```

4.4.7. Korišćenjem potprograma LRFAK i RSTS i imajući u vidu odeljak 4.2.3, lako se može obrazovati program za inverziju matrica. Odgovarajući program i izlazni rezultat (za matricu iz prethodnog primera) imaju oblik:

```
C=================================================================
C                         INVERZIJA MATRICE
C=================================================================
      DIMENSION A(100),B(10),IP(9),AINV(100)
      READ(8,5) N
    5 FORMAT(I2)
      NN=N*N
      READ(8,10) (A(I),I=1,NN)
   10 FORMAT(16F5.0)
      WRITE(5,34)
   34 FORMAT(1H1,5X,'MATRICA A'/)
      DO 12 I=1,N
   12 WRITE(5,15) (A(J),J=I,NN,N)
   15 FORMAT(10F10.5)
      CALL LRFAK(A,N,IP,DET,KB)
      IF(KB) 20,25,20
   20 WRITE(5,30)
   30 FORMAT(1H0,'MATRICA A JE SINGULARNA'//)
      GO TO 70
   25 DO 45 I=1,N
      DO 40 J=1,N
   40 B(J)=0.
      B(I)=1.
      CALL RSTS(A,N,IP,B)
      IN=(I-1)*N
      DO 45 J=1,N
      IND=IN+J
   45 AINV(IND)=B(J)
      WRITE(5,50)
   50 FORMAT(1H0,5X,'INVERZNA MATRICA'/)
      DO 55 I=1,N
   55 WRITE(5,15)(AINV(J),J=I,NN,N)
   70 CALL EXIT
      END
```

```
MATRICA A

3.00000   1.00000   6.00000
2.00000   1.00000   3.00000
1.00000   1.00000   1.00000
```

```
INVERZNA MATRICA

-2.00000   5.00000  -3.00000
 1.00000  -3.00000   3.00000
 1.00000  -2.00000   1.00000
```

4.4.8. Sastavimo sada program za rešavanje sistema linearnih jednačina oblika $\vec{x} = B\vec{x} + \vec{\beta}$, metodom proste iteracije (videti odeljak 4.3.2). S obzirom da metod proste iteracije konvergira kada je norma matrice B manja od jedinice, to ćemo za ispitivanje ovog uslova obrazovati potprogram NORMA, po kome se u zavisnosti od k (u potprogramu K) izračunavaju norme $\| B \|_k$ $(k = 1,2,3)$ saglasno formuli (3.2.3) iz odeljka 4.3.2. Parametri u listi imaju sledeće značenje:

A – matrica memorisana kao vektor, čija se norma traži;
N – red matrice;
K – broj koji definiše normu ($K = 1, 2, 3$);
$ANOR$ – odgovarajuća norma matrice A.

```
      SUBROUTINE NORMA(A,N,K,ANOR)
      DIMENSION A(1)
      NU=N*N
      ANOR=0
      GO TO(10,20,40),K
   10 DO 15 I=1,NU
   15 ANOR=ANOR+A(I)**2
      ANOR=SQRT(ANOR)
      RETURN
   20 DO 25 I=1,N
      L=-N
      S=0.
      DO 30 J=1,N
      L=L+N
      IA=L+I
   30 S=S+ABS(A(IA))
      IF(ANOR-S) 35,25,25
   35 ANOR=S
   25 CONTINUE
      RETURN
   40 L=-N
      DO 50 J=1,N
      S=0.
      L=L+N
      DO 45 I=1,N
      LI=L+I
   45 S=S+ABS(A(LI))
      IF(ANOR-S) 55,50,50
   55 ANOR=S
   50 CONTINUE
      RETURN
      END
```

Glavni program je organizovan tako da se pre početka iterativnog procesa ustanovljava konvergencija. Naime, ukoliko je bar jedna od normi $\| B \|_k < 1$ $(k = 1, 2, 3)$, prelazi se na iterativni proces, dok se u protivnom slučaju štampa poruka da uslovi za konvergenciju nisu zadovoljeni i u tom slučaju se program završava.

Za množenje matrice B vektorom $\vec{x}^{(k-1)}$ koristimo potprogram MMAT, koji je dat u 4.4.2. Za početni vektor $\vec{x}^{(0)}$ uzimamo vektor $\vec{\beta}$.

Kao kriterijum za završetak iterativnog procesa usvojili smo

$$|x_i^{(k)} - x_i^{(k-1)}| \leqslant \epsilon \ (i = 1, \ldots, n).$$

Na izlazu štampamo poslednju iteraciju koja zadovoljava gornji kriterijum.

```
      DIMENSION B(100),BETA(10),X(10),X1(10)
      READ(8,5) N,EPS
    5 FORMAT(I2,E5.0)
      NN=N*N
      READ(8,10) (B(I),I=1,NN),(BETA(I),I=1,N)
   10 FORMAT(16F5.1)
      WRITE(5,13)
   13 FORMAT(1H1,5X,'MATRICA B',24X,'VEKTOR BETA')
      DO 15 I=1,N
   15 WRITE(5,20) (B(J),J=I,NN,N),BETA(I)
   20 FORMAT(/2X,4F8.1,5X,F8.1)
      DO 30 K=1,3
      CALL NORMA(B,N,K,ANOR)
      IF(ANOR-1.) 25,30,30
   30 CONTINUE
      WRITE(5,35)
   35 FORMAT(5X,'USLOVI ZA KONVERGENCIJU NISU ZADOVOLJENI')
      GO TO 75
   25 ITER=0
      DO 40 I=1,N
   40 X(I)=BETA(I)
   62 ITER=ITER+1
      CALL MMAT(B,X,X1,N,N,1)
      DO 45 I=1,N
   45 X1(I)=X1(I)+BETA(I)
      DO 55 I=1,N
      IF(ABS(X1(I)-X(I))-EPS)55,55,60
   55 CONTINUE
      WRITE(5,42) ITER
   42 FORMAT( /3X,I3,'.ITERACIJA'/)
      WRITE(5,50)(I,X1(I),I=1,N)
   50 FORMAT(3X,4(1X,'X(',I2,')=',F9.5))
      GO TO 75
   60 DO 65 I=1,N
   65 X(I)=X1(I)
      GO TO 62
   75 CALL EXIT
      END
```

Uzimajući tačnost $\epsilon = 10^{-5}$, za jedan konkretan sistem jednačina četvrtog reda (videti izlaznu listu) dobijamo rešenje u četrnaestoj iteraciji.

```
MATRICA B                                   VEKTOR BETA

-0.1      0.4      0.1      0.1             0.7

 0.4     -0.1      0.1      0.1             0.7

 0.1      0.1     -0.2      0.2             1.2

 0.1      0.1      0.2     -0.2            -1.6

14.ITERACIJA

X( 1)=  1.00000 X( 2)=  1.00000 X( 3)=  1.00000 X( 4)= -1.00000
```

4.4.9. Kod rešavanja sistema sa velikim brojem jednačina pojavljuje se problem smeštanja matrice sistema u centralnoj memoriji računara. U tim slučajevima koristimo virtuelnu memoriju na disku. Sledeći program je realizovan za takve sisteme, a zasnivan je na Gaussovoj eliminaciji (bez izbora glavnog elementa). U programu je obezbeđeno mesto za matricu stotog reda. Međutim, po potrebi se, bez teškoća, mogu rešavati i sistemi jednačina proizvoljnog reda, uz proširenje datoteke na disku. Program je realizovan u dvostrukoj tačnosti.

```
C     PROGRAM ZA RESAVANJE VELIKIH SISTEMA JEDNACINA
C     DATOTEKA VIRTUELNE MEMORIJE
        REAL*8 A(100),B(100),X(100),DT,E1,E2,Z1
        DEFINE FILE 7(102,400,U,KL)
C       N = BROJ JEDNACINA LINEARNOG SISTEMA (N>1)
        READ(8,10)N
   10   FORMAT(I3)
C     UPISIVANJE PARAMETARA LINEARNOG SISTEMA
C     U DATOTEKU 7 NA DISKU
        DO 11 L=1,N
        READ(8,20)(A(I),I=1,N)
   20   FORMAT(5D16.10)
        KL=L
   11   WRITE(7'KL)(A(I),I=1,N)
        READ(8,20)(B(I),I=1,N)
        KL=N+1
        WRITE(7'KL)(B(I),I=1,N)
C     RESAVANJE LINEARNOG SISTEMA
C     PRIMENOM VIRTUELNE MEMORIJE
        KL=N+1
        READ(7'KL)B
        N1=N-1
        DO 1 K=1,N1
        KL=K
        READ(7'KL)A
        N2=K+1
        IF(A(K).NE.0.D0)GO TO 111
        DO 2 K1=N2,N
        KL=K1
        READ(7'KL)X
        IF(X(K).NE.0.D0)GO TO 21
        DO 3 L=K,N
        Z1=A(L)
        A(L)=X(L)
        X(L)=Z1
```

```
    3 CONTINUE
      KL=K
      WRITE(7'KL)A
      KL=K1
      WRITE(7'KL)X
      Z1=B(K)
      B(K)=B(K1)
      B(K1)=Z1
      GO TO 111
   21 CONTINUE
    2 CONTINUE
      GO TO 222
  111 E1=B(K)/A(K)
      DO 4 J=N2,N
      KL=J
      READ(7'KL)X
      B(J)=B(J)-X(K)*E1
      E2=X(K)/A(K)
      DO 5 I=K,N
      X(I)=X(I)-A(I)*E2
    5 CONTINUE
      KL=J
      WRITE(7'KL)X
    4 CONTINUE
    1 CONTINUE
      DO 6 KIN=2,N
      K=N+2-KIN
      KL=K
      READ(7'KL)A
      E1=B(K)/A(K)
      N3=K-1
      DO 7 J=1,N3
      KL=J
      READ(7'KL)X
      B(J)=B(J)-X(K)*E1
      X(K)=0.D0
      KL=J
      WRITE(7'KL)X
    7 CONTINUE
    6 CONTINUE
      DT=1.D0
      DO 8 L=1,N
      KL=L
      READ(7'KL)A
      X(L)=B(L)/A(L)
      DT=DT*A(L)
    8 CONTINUE
      KL=N+1
      WRITE(7'KL)B
      KL=N+2
      WRITE(7'KL)X
      GO TO 333
  222 DT=0.D0
      DO 9 L=1,N
      X(L)=0.D0
    9 CONTINUE
  333 CONTINUE
C     STAMPANJE RESENJA SISTEMA JEDNACINA
      KL=N+2
      READ(7'KL)X
      WRITE(5,30)(X(I),I=1,N)
   30 FORMAT(2D24.16)
      CALL EXIT
      END
```

4.4.10. Obrazujmo program za nalaženje matrice $S = e^A$, gde je A data kvadratna matrica reda n, korišćenjem formule

$$(1) \qquad e^A = \sum_{k=0}^{+\infty} \frac{1}{k!} A^k .$$

Neka je S_k k–ta parcijalna suma reda (1), a U_k njen opšti član. Tada važe jednakosti

$$(2) \qquad U_k = \frac{1}{k} U_{k-1} A, \; S_k = S_{k-1} + U_k \qquad (k = 1, 2, \ldots).$$

pri čemu je $U_0 = S_0 = I$ (jedinična matrica reda n).

Korišćenjem jednakosti (2) može se obrazovati program za sumiranje reda (1), pri čemu se, kao kriterijum za prekidanje procesa sumiranja, obično uzima slučaj kada je norma matrice U_k manja od unapred zadatog malog pozitivnog broja ϵ. U našem slučaju uzećemo normu $\|\cdot\|_2$ (videti formulu (3.2.3)) i $\epsilon = 10^{-5}$.

Korišćenjem potprograma MMAT za množenje matrica (videti 4.4.2) i potprograma NORMA za izračunavanje norme matrica (videti 4.4.8), sastavili smo sledeći program za nalaženje matrice e^A.

```
C     ================================================================
C                       ODREDJIVANJE MATRICE EXP(A)
C     ================================================================
      DIMENSION A(100),S(100),U(100),P(100)
      READ(8,10) N,EPS
   10 FORMAT(I2,E5.0)
      NN=N*N
      READ(8,15) (A(I),I=1,NN)
   15 FORMAT(16F5.0)
C     FORMIRANJE JEDINICNE MATRICE
      DO 20 I=1,NN
      S(I)=0.
   20 U(I)=0.
      N1=N+1
      DO 25 I=1,NN,N1
      S(I)=1.
   25 U(I)=1.
C     SUMIRANJE MATRICNOG REDA
      K=0
   30 K=K+1
      CALL MMAT(U,A,P,N,N,N)
      B=1./K
      DO 35 I=1,NN
      U(I)=B*P(I)
   35 S(I)=S(I)+U(I)
C     ISPITIVANJE USLOVA ZA PREKID SUMIRANJA
      CALL NORMA(U,N,2,ANOR)
      IF(ANOR.GT.EPS) GO TO 30
      WRITE(5,40)((A(I),I=J,NN,N),J=1,N)
   40 FORMAT(1H0,<5*N-9>X,'M A T R I C A   A'// (<N>F10.5))
      WRITE(5,45)((S(I),I=J,NN,N),J=1,N)
   45 FORMAT( //<5*N-9>X,'M A T R I C A   EXP(A)'//(<N>F10.5))
      CALL EXIT
      END
```

Dobijeni program testirali smo na primeru

$$A = \begin{bmatrix} 4 & 3 & -3 \\ 2 & 3 & -2 \\ 4 & 4 & -3 \end{bmatrix}$$

za koji se analitički može dobiti

(3)
$$e^{A} = e \begin{bmatrix} 3e-2 & 3e-3 & -3e+3 \\ 2e-2 & 2e-1 & -2e+2 \\ 4e-4 & 4e-4 & -4e+5 \end{bmatrix}$$

Izlazna lista ima oblik

```
            M A T R I C A   A

4.00000     3.00000   -3.00000
2.00000     3.00000   -2.00000
4.00000     4.00000   -3.00000

          M A T R I C A   EXP(A)

16.73060   14.01232  -14.01233
 9.34155   12.05983   -9.34155
18.68310   18.68310  -15.96482
```

Korišćenjem (3) nije teško proveriti da su sve decimale u dobijenim elementima matrice e^{A} tačne.

5. NUMERIČKI METODI ZA INTEGRACIJU

5.1. KVADRATURNE FORMULE

5.1.1. Uvodne napomene

Numerička integracija funkcija sastoji se u približnom izračunavanju određenih integrala na osnovu niza vrednosti podintegralne funkcije po određenoj formuli.

Formule za numeričko izračunavanje jednostrukih integrala nazivaju se kvadraturne formule. Slično, formule za dvostruke integrale nazivaju se kubaturne formule. U našem izlaganju zadržaćemo se uglavnom na kvadraturnim formulama.

Potreba za numeričkom integracijom javlja se u velikom broju slučajeva. Naime, Newton–Leibnitzova formula

$$(1.1.1) \qquad \int_a^b f(x)\, dx = F(b) - F(a),$$

gde je F primitivna funkcija za funkciju f, ne može se uvek uspešno primeniti. Navešćemo neke od tih slučajeva:

1° Funkcija F se ne može predstaviti pomoću konačnog broja elementarnih funkcija (na primer, kada je $f(x) = e^{-x^2}$).

2° Primena formule (1.1.1) često dovodi do vrlo složenog izraza, čak i kod izračunavanja integrala jednostavnijih funkcija; na primer

$$\int_0^a \frac{dx}{1+x^3} = \log \sqrt[3]{|a+1|} - \frac{1}{6} \log (a^2 - a + 1) + \frac{1}{\sqrt{3}} \operatorname{arctg} \frac{a\sqrt{3}}{2-a}.$$

3° Kod integracije funkcija, čije su vrednosti poznate na diskretnom skupu tačaka (dobijene, na primer, eksperimentalno), nije moguće primeniti formulu (1.1.1).

Veliki broj kvadraturnih formula ima oblik

$$(1.1.2) \qquad \int_a^b f(x)\, dx \cong \sum_{k=0}^{n} A_k f_k,$$

gde je $f_k = f(x_k)$ $(a \leqslant x_o < \ldots < x_n \leqslant b)$. Ako je $x_o = a$ i $x_n = b$, za formulu (1.1.2) kažemo da je zatvorenog tipa, dok u ostalim slučajevima kažemo da je otvorenog tipa. Napomenimo da se za integraciju diferencijabilnih funkcija koriste i formule u kojima se pored vrednosti funkcije pojavljuju i vrednosti izvoda.

Od interesa su i formule za izračunavanje integrala

$$\int_a^b p(x) f(x) \, dx,$$

gde je $x \to p(x)$ data težinska funkcija.

Jedan prost način za konstrukciju kvadraturnih formula zasniva se na primeni interpolacije. Formule dobijene na ovaj način nazivaju se kvadraturne formule interpolacionog tipa.

Neka su vrednosti funkcije f u datim tačkama $x_o, x_1, \ldots, x_n$ $(\in [a, b])$ redom $f_o, f_1, \ldots, f_n$, tj.

$$f_k \equiv f(x_k) \qquad (k = 0, 1, \ldots, n).$$

Na osnovu ovih podataka, možemo konstruisati Lagrangeov interpolacioni polinom

$$P_n(x) = \sum_{k=o}^{n} f_k \frac{\omega(x)}{(x - x_k)\ \omega'(x_k)},$$

gde je $\omega(x) = (x - x_o)(x - x_1) \ldots (x - x_n)$.

Tada je

$$\int_a^b p(x) \ f(x) dx = \int_a^b p(x) \ P_n(x) \ dx + R_{n+1}(f),$$

tj.

$$\text{(1.1.3)} \qquad \int_a^b p(x) \ f(x) \, dx = \sum_{k=o}^{n} A_k f_k + R_{n+1}(f),$$

gde smo stavili

$$A_k = \int_a^b \frac{p(x)\,\omega(x)}{(x - x_k)\,\omega'(x_k)} dx \qquad (k = 0, 1, \ldots, n).$$

U formuli (1.1.3), veličina $R_{n+1}(f)$ naziva se ostatak kvadraturne formule i predstavlja grešku koja se čini zamenom integrala konačnom sumom. Indeks $n + 1$ u ostatku označava da se integral približno izračunava na osnovu vrednosti podintegralne funkcije u $n + 1$ tačaka.

Sa π_n označimo skup svih polinoma ne višeg stepena od n.

Kako je za $f(x) = x^k$ $(k = 0, 1, \ldots, n)$, $f(x) \equiv P_n(x)$, imamo

$$R_{n+1}(x^k) \equiv 0 \qquad (k = 0, 1, \ldots, n),$$

odakle zaključujemo da je formula (1.1.3) tačna za svako $f \in \pi_n$, bez obzira na izbor interpolacionih čvorova x_k $(k = 0, 1, \ldots, n)$ i u ovom slučaju kažemo da formula (1.1.3) ima algebarski stepen tačnosti n.

5.1.2. Newton–Cotesove formule

U ovom odeljku izvešćemo kvadraturne formule zatvorenog tipa u kojima su interpolacioni čvorovi $x_k = x_o + kh$ $(k = 0, 1, \ldots, n)$ uzeti ekvidistantno sa korakom $h = \dfrac{b-a}{n}$.

Ako uvedemo smenu $x - x_o = ph$, imamo

$$(1.2.1) \qquad \omega(x) = (x - x_o)(x - x_1)\ldots(x - x_n) = h^{n+1} p(p-1)\ldots(p-n)$$

i

$$(1.2.2) \qquad \begin{aligned} \omega'(x_k) &= (x - x_o)\ldots(x_k - x_{k-1})(x_k - x_{k+1})\ldots(x_k - x_n) \\ &= h^n(-1)^{n-k} k!\,(n-k)! \end{aligned}$$

Uvođenjem oznake za uopšteni stepen $x^{(s)} = x(x-1)\ldots(x-s+1)$, na osnovu (1.2.1), (1.2.2) i rezultata iz prethodnog odeljka dobijamo

$$A_k = \int_o^n \frac{(-1)^{n-k} p^{(n+1)} h}{(p-k)k!\,(n-k)!}\,dp \qquad (k = 0, 1, \ldots, n),$$

tj.

$$A_k = (b-a)H_k \qquad (k = 0, 1, \ldots, n),$$

gde smo stavili

$$(1.2.3) \qquad H_k \equiv H_k(n) = \frac{(-1)^{n-k}}{n!\,n}\binom{n}{k}\int_o^n \frac{p^{(n+1)}}{p-k}\,dp \quad (k = 0, 1, \ldots, n).$$

Koeficijenti H_k u literaturi (videti na primer [2]) poznati su kao Newton–Cotesovi koeficijenti, a odgovarajuće formule

$$(1.2.4) \qquad \int_{x_o=a}^{x_n=b} f(x)\,dx = (b-a)\sum_{k=o}^{n} H_k f\left(a + k\frac{b-a}{n}\right) \quad (k \in N)$$

kao Newton–Cotesove formule.

U daljem izlaganju daćemo pregled Newton–Cotesovih formula za $n \leqslant 4$. Pri ovome koristimo oznake $h = \dfrac{b-a}{n}$, $f_k = f(x_k)$ $(k = 0, 1, \ldots, n)$.

1) $n = 1$ (trapezno pravilo)

$$\int_{x_o}^{x_1} f(x)\,dx = \frac{h}{2}(f_o + f_1) - \frac{h^3}{12} f''(\xi_1);$$

2) $n = 2$ (Simpsonovo pravilo)

$$\int_{x_0}^{x_2} f(x)\ dx = \frac{h}{3}(f_0 + 4f_1 + f_2) - \frac{h^5}{90} f^{IV}(\xi_2)$$

3) $n = 3$ (Simpsonovo pravilo $\frac{3}{8}$)

$$\int_{x_0}^{x_3} f(x)\ dx = \frac{3h}{8}(f_0 + 3f_1 + 3f_2 + f_3) - \frac{3h^5}{80} f^{IV}(\xi_3)$$

4) $n = 4$ (Booleovo pravilo)

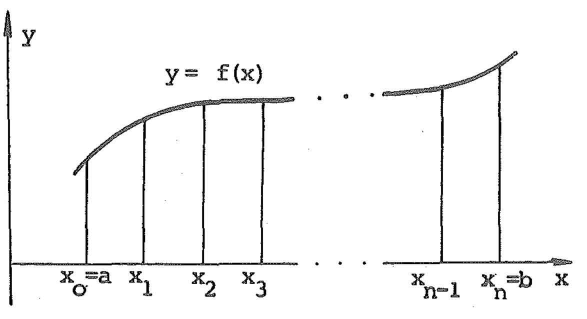
$$\int_{x_0}^{x_4} f(x)\ dx = \frac{2h}{45}(7f_0 + 32f_1 + 12f_2 + 32f_3 + 7f_4) - \frac{8h^7}{945} f^{(6)}(\xi_4)$$

gde $\xi_k \in (x_0, x_k)$ $(k = 1, \ldots, 4)$.

5.1.3. Uopštene kvadraturne formule

Da bismo tačnije izračunali vrednost integrala potrebno je podeliti segment $[a,b]$ na niz podsegmenata, a zatim na svakom od njih primeniti neku od kvadraturnih formula. Na taj način dobijamo uopštene ili kompozitne formule. U ovom odeljku razmotrićemo uopštene formule dobijene na bazi trapezne i Simpsonove formule.

Podelimo segment $[a, b]$ na niz podsegmenata $[x_{i-1}, x_i]$ $(i = 1, \ldots, n)$ tako da je $x_i = a + ih$ $(i = 0, 1, \ldots, n)$ i $h = (b - a)/n$.



sl.1.3.1

Primenom trapezne formule na svaki od podsegmenata dobijamo

$$\int_a^b f(x)\ dx = \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} f(x)\ dx = \sum_{i=1}^{n} \left(\frac{h}{2}(f_{i-1} + f_i) - \frac{h^3}{12} f''(\xi_i)\right),$$

tj.

$$\int_a^b f(x)\ dx = T_n - \frac{h^3}{12} \sum_{i=1}^{n} f''(\xi_i),$$

gde su

$$T_n \equiv T_n\,(f;\,h) = h\,(\tfrac{1}{2}f_o + f_1 + \ldots + f_{n-1} + \tfrac{1}{2}f_n)$$

i $\quad x_{i-1} < \xi_i < x_i\;(i = 1, \ldots, n).$

Teorema 1.3.1. Ako $f \in C^2\,[a, b]$ važi jednakost

$$\int_a^b f\,(x)\;dx - T_n = -\frac{(b-a)^2}{12n^2}\,f\,''(\xi) \qquad (a < \xi < b).$$

Kvadraturna formula

$$\int_a^b f\,(x)\;dx \cong T_n\,(f;h) \qquad (h = \frac{b-a}{n})$$

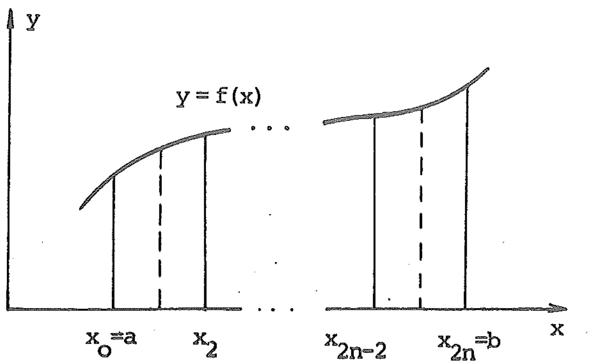
naziva se uopštena trapezna formula.

Pretpostavimo sada da je $h = \dfrac{b-a}{2n}$, tj. $x_i = a + ih\;(i = 0, 1, \ldots, 2n)$ (videti sl. 1.3.2), a zatim na podsegmente $[x_o, x_2], \ldots, [x_{2n-2}, x_{2n}]$ primenimo Simpsonovu formulu. Na ovaj način dobijamo uopštenu Simpsonovu formulu

$$\int_a^b f\,(x)\;dx \cong S_n\,(f;h) \qquad (h = \frac{b-a}{2n}),$$

gde je

$$S_n \equiv S_n\,(f;h) = \frac{h}{3}\left\{ f_o + 4\,(f_1 + \ldots + f_{2n-1}) + 2\,(f_2 + \ldots + f_{2n-2}) + f_{2n}\right\}.$$



sl.1.3.2

Teorema 1.3.2. Ako $f \in C^4\,[a, b]$ važi jednakost

$$\int_a^b f\,(x)\,dx - S_n = -\frac{(b-a)^2}{2880n^4}\,f^{IV}(\xi) \qquad (a < \xi < b).$$

5.1.4. Rombergova integracija

Za izračunavanje određenih integrala, u praksi se najčešće koristi uopštena trapezna formula u jednom specijalnom obliku, koji je poznat kao Rombergova integracija (videti [2]).

Sa $T_k^{(o)}$ označimo trapeznu aproksimaciju $T_n\ (f;\ h)\ (n = 2^k)$, tj. $h = (b - a)/2^k$). Rombergova integracija se sastoji u konstrukciji dvodimenzionalnog niza $T_k^{(m)}$ $(m = 0, 1, \ldots, k;\ k = 0, 1, \ldots)$ pomoću

$$(1.4.1) \qquad T_k^{(m)} = \frac{4^m T_{k+1}^{(m-1)} - T_k^{(m-1)}}{4^m - 1}.$$

Na osnovu (1.4.1) može se konstruisati tzv. T tabela

$$\begin{array}{ccccccc}
T_o^{(o)} & — & T_o^{(1)} & — & T_o^{(2)} & — & T_o^{(3)} \\
 & / & & / & & / & \vdots \\
T_1^{(o)} & — & T_1^{(1)} & — & T_1^{(2)} & — & \vdots \\
 & / & & / & \vdots & & \\
T_2^{(o)} & — & T_2^{(1)} & — & \vdots & & \\
 & / & \vdots & & & & \\
T_3^{(o)} & — & \vdots & & & &
\end{array}$$

uzimajući $k = 0, 1, \ldots$ i $m = 1, 2, \ldots$. U prvoj koloni ove tabele nalaze se redom približne vrednosti integrala I dobijene primenom trapezne formule sa $h_k = (b - a)/2^k$ $(k = 0, 1, \ldots)$. Druga kolona ove tabele dobija se na osnovu prve, korišćenjem formule (1.4.1), treća na osnovu druge, itd.

Iterativni proces, definisan sa (1.4.1) predstavlja standardni Rombergov metod za numeričku integraciju. Može se dokazati da nizovi $\{T_k^{(m)}\}_{k \in N_o}$ i $\{T_k^{(m)}\}_{m \in N_o}$ (po kolonama i vrstama u T–tabeli) konvergiraju ka I. Kod praktične primene Rombergove integracije, iterativni proces (1.4.1) se najčešće prekida kada je $|T_o^{(m)} - T_o^{(m-1)}| \leqslant \epsilon$, gde je ϵ unapred data dozvoljena greška, i tada se uzima $I \cong T_o^{(m)}$.

5.1.5. Programska realizacija

U ovom odeljku dajemo programsku realizaciju Simpsonove i Rombergove integracije.

1.5.1. Za integraciju po uopštenoj Simpsonovoj formuli realizovan je potprogram INTEG. Parametri u listi imaju značenje koje je objašnjeno C–karticama u potprogramu. Podintegralna funkcija se zadaje u potprogramu FUN, i može zavisiti od jednog parametra Z. Celobrojnim parametrom J je obezbeđeno istovremeno zadavanje više podintegralnih funkcija.

Potprogram INTEG je organizovan tako da se početni broj podsegmenata može povećavati (redukcijom koraka h na $h/2$) do MAX = 1000. U slučaju kada je relativna razlika u vrednosti integrala, dobijenog sa korakom h i korakom $h/2$, manja od 10^{-5}, računski proces se prekida i vrednost integrala izračunata sa dotad najmanjim korakom se uzima kao definitivna vrednost integrala. Ukoliko se ovaj kriterijum ne može ispuniti sa manje od MAX podsegmenata daje se poruka KBR = 1 (u protivnom je KBR = 0).

Za testiranje ovog potprograma uzeli smo izračunavanje sledećih integrala:

$$\int_0^1 \frac{e^{zx}}{x^2+z^2}\,dx \qquad (z = 1.0\,(0.1)\,1.5),$$

$$\int_0^{1/2} \pi\sin(\pi z x)\,dx \qquad (z = 1.0\,(0.2)\,1.4),$$

$$\int_1^2 \frac{\log(x+z)}{z^2+e^x}\cdot\frac{\sin x}{x}\,dx \qquad (z = 0.\,(0.1)\,0.5).$$

Potprogrami, glavni program i izlazna lista imaju oblik:

```
C     =================================================
C     IZRACUNAVANJE ODREDENOG INTEGRALA FUNKCIJE
C     F(X,Z,J)  SIMPSONOVOM FORMULOM
C     =================================================
      SUBROUTINE INTEG(A,B,S,F,J,KBR,Z)
C     A = DONJA GRANICA INTEGRALA
C     B = GORNJA GRANICA INTEGRALA
C     S = VREDNOST INTEGRALA SA TACNOSCU EPS=1.E-5
C     KBR = KONTROLNI BROJ
C     KBR=0 INTEGRAL KOREKTNO IZRACUNAT
C     KBR=1 INTEGRAL NIJE IZRACUNAT SA ZAHTEVANOM TACNOSCU
C     Z = PARAMETAR U PODINTEGRALNOJ FUNKCIJI
C     POCETNI BROJ PODEOKA JE 2*MP,A MAKSIMALNI MAX=1000
      MP=15
      MAX=1000
      KBR=0
      N=2.*MP
      S0=0.
      SAB=F(A,Z,J)+F(B,Z,J)
      H=(B-A)/FLOAT(N)
      X=A
      S1=0.
      N2=N-2
      DO 5 I=2,N2,2
      X=X+2.*H
    5 S1=S1+F(X,Z,J)
   10 S2=0.
      X=A-H
      N1=N-1
      DO 15 I=1,N1,2
      X=X+2.*H
   15 S2=S2+F(X,Z,J)
      S=H/3.*(SAB+2.*S1+4.*S2)
      REL=(S-S0)/S
      IF(ABS(REL)-1.E-5) 35,35,20
   20 IF(N-MAX) 25,25,30
   25 N=2*N
      H=0.5*H
```

```
C       BROJ PODEOKA SE UVECAVA DVA PUTA I
C       IZRACUNAVA SE NOVA VREDNOST ZA S1
        S1=S1+S2
        S0=S
        GO TO 10
   30   KBR=1
   35   RETURN
        END

      FUNCTION FUN(X,Z,J)
      GO TO(10,20,30),J
   10 FUN=EXP(Z*X)/(X*X+Z*Z)
      RETURN
   20 PI=3.1415926535
      FUN=PI*SIN(PI*X*Z)
      RETURN
   30 FUN=ALOG(X+Z)/(Z*Z+EXP(X))*SIN(X)/X
      RETURN
      END

      EXTERNAL FUN
      WRITE(5,5)
    5 FORMAT(1H1,2X,'IZRACUNAVANJE VREDNOSTI INTEGRALA PRIMENOM ',
     1'SIMPSONOVE FORMULE'//14X,'TACNOST IZRACUNAVANJA EPS = 1.E-5'
     2///11X,'J',4X,'DONJA',5X,'GORNJA',3X,'PARAMETAR',3X,'VREDNOST'/
     316X,'GRANICA',3X,'GRANICA',5X,'Z',7X,'INTEGRALA'//)
      DO 40 J=1,3
      READ(8,15) DG,GG,ZP,DZ,ZK
   15 FORMAT(5F5.1)
      Z=ZP-DZ
   18 Z=Z+DZ
      IF(Z.GT.ZK) GO TO 40
      CALL INTEG(DG,GG,S,FUN,J,KBR,Z)
      IF(KBR) 20,25,20
   20 WRITE(5,30)
   30 FORMAT(/11X,'INTEGRAL NIJE KOREKTNO IZRACUNAT'/)
      GO TO 18
   25 WRITE(5,35) J,DG,GG,Z,S
   35 FORMAT(11X,I1,F8.1,2F10.1,F15.6/)
      GO TO 18
   40 CONTINUE
      CALL EXIT
      END
```

IZRACUNAVANJE VREDNOSTI INTEGRALA PRIMENOM SIMPSONOVE FORMULE

TACNOST IZRACUNAVANJA EPS = 1.E-5

J	DONJA GRANICA	GORNJA GRANICA	PARAMETAR Z	VREDNOST INTEGRALA
1	0.0	1.0	1.0	1.270724
1	0.0	1.0	1.1	1.153890
1	0.0	1.0	1.2	1.059770
1	0.0	1.0	1.3	0.983069
1	0.0	1.0	1.4	0.920013

1	0.0	1.0	1.5	0.867848
2	0.0	0.5	1.0	1.000000
2	0.0	0.5	1.2	1.090848
2	0.0	0.5	1.4	1.134133
3	1.0	2.0	0.0	0.048047
3	1.0	2.0	0.1	0.059595
3	1.0	2.0	0.2	0.069940
3	1.0	2.0	0.3	0.079052
3	1.0	2.0	0.4	0.086920
3	1.0	2.0	0.5	0.093558

1.5.2. Sada dajemo programsku realizaciju Rombergove integracije (videti odeljak 5.1.4) u dvostrukoj tačnosti. Lista u potprogramu ROMBI ima sledeće značenje:

DG – donja granica integrala;
GG – gornja granica integrala;
FUN – ime funkcijskog potprograma kojim se definiše podintegralna funkcija;
EPS – zahtevana tačnost izračunavanja;
VINT – vrednost integrala sa tačnošću EPS, ukoliko je KB = 0;
KB – kontrolni broj (KB = 0 integral korektno izračunat; KB = 1 tačnost izračunavanja integrala nije postignuta sa 15 predviđenih koraka, tj. sa brojem podsegmenata 2^{15}).

Za testiranje ovog potprograma uzeto je tabeliranje funkcije

$$F(x) = \int_0^x e^{-t^2}\,dt \qquad (x = 0.1\ (0.1)\ 1.0),$$

sa tačnošću 10^{-8}. Programi i izlazna lista imaju oblik:

```
C=================================================================
C             ROMBERGOVA INTEGRACIJA
C=================================================================
      DOUBLE PRECISION GG,FUN,VINT
      EXTERNAL FUN
      EPS=1.E-8
      WRITE(5,11)
   11 FORMAT(1H0,5X,'X',7X,'INTEGRAL(0.,X)'/)
      DO 10 I=1,10
      GG=0.1*I
      CALL ROMBI(0.D0,GG,FUN,EPS,VINT,KB)
      IF(KB)5,15,5
    5 WRITE(5,20)GG
   20 FORMAT(5X,F3.1,4X,'TACNOST NE ZADOVOLJAVA'//)
      GO TO 10
   15 WRITE(5,25)GG,VINT
   25 FORMAT(5X,F3.1,4X,F14.9)
   10 CONTINUE
      CALL EXIT
      END
```

```
      SUBROUTINE ROMBI(DG,GG,FUN,EPS,VINT,KB)
      DOUBLE PRECISION FUN,VINT,T(15),DG,GG,H,A,POM,B,X
      KB=0
      H=GG-DG
      A=(FUN(DG)+FUN(GG))/2.
      POM=H*A
      DO 50 K=1,15
      X=DG+H/2.
   10 A=A+FUN(X)
      X=X+H
      IF(X.LT.GG)GO TO 10
      T(K)=H/2.*A
      B=1.
      IF(K.EQ.1)GO TO 20
      K1=K-1
      DO 15 M=1,K1
      I=K-M
      B=4.*B
   15 T(I)=(B*T(I+1)-T(I))/(B-1.)
   20 B=4.*B
      VINT=(B*T(1)-POM)/(B-1.)
      IF(DABS(VINT-POM).LE.EPS) RETURN
      POM=VINT
   50 H=H/2.
      KB=1
      RETURN
      END


      FUNCTION FUN(X)
      DOUBLE PRECISION FUN,X
      FUN=DEXP(-X*X)
      RETURN
      END
```

X	INTEGRAL(0.,X)
0.1	0.099667666
0.2	0.197365034
0.3	0.291237894
0.4	0.379652845
0.5	0.461281006
0.6	0.535153543
0.7	0.600685661
0.8	0.657669863
0.9	0.706241531
1.0	0.746824133

5.1.6. O numeričkom izračunavanju jedne klase dvostrukih integrala

U ovom odeljku ukazaćemo na jedan način za približno izračunavanje dvostrukih integrala oblika

$$(1.6.1) \qquad \iint_G f(x, y)\, dxdy,$$

gde je oblast integracije jedinični krug, tj. $G = \{(x, y) \mid x^2 + y^2 \leqslant 1\}$. Naime, za numeričko izračunavanje integrala (1.6.1) u literaturi ([3]) je poznata formula

$$(1.6.2) \qquad \iint\limits_{G} f(x,y)\,dxdy \cong \frac{\pi}{8}\,(2\,f(0) + \sum_{i=1}^{n} f(M_i)),$$

gde 0 predstavlja koordinatni početak, tj. 0 = (0,0), dok tačke M_i imaju polarne koordinate

$$r_i = \sqrt{\frac{2}{3}}\,,\ \phi_i = \frac{\pi}{3}\ (i-1) \quad (i = 1, 2, \ldots, 6).$$

Prema formuli (1.6.2) realizovaćemo program za izračunavanje dvostrukih integrala, gde je oblast integracije jedinični krug. Organizacija programa biće takva da se funkcijskim potprogramom EF mogu definisati više različitih podintegralnih funkcija f. Parametri u listi imaće sledeće značenje:

X – vrednost argumenta x;
Y – vrednost argumenta y;
K – celobrojni parametar kojim se difinišu različite podintegralne funkcije.

Formula (1.6.2) realizovana je kroz potprogram DVINT, čiji parametri u listi imaju sledeće značenje:

EF – ime funkcijskog potprograma;
K – celobrojni parametar, kao u potprogramu EF;
VRINT – izračunata vrednost integrala, prema kubaturnoj formuli (1.6.2).

```
      SUBROUTINE DVINT(EF,K,VRINT)
      PI=3.1415926535
      RO=SQRT(2./3.)
      PI3=PI/3.
      FI=-PI3
      VRINT=2.*EF(0.,0.,K)
      DO 10 I=1,6
      FI=FI+PI3
      X=RO*COS(FI)
      Y=RO*SIN(FI)
   10 VRINT=VRINT+EF(X,Y,K)
      VRINT=PI/8.*VRINT
      RETURN
      END
```

Glavni program ima oblik:

```
C==================================================================
C     IZRACUNAVANJE DVOSTRUKOG INTEGRALA
C==================================================================
      EXTERNAL EF
      WRITE(5,5)
    5 FORMAT(1H1//10X,'IZRACUNAVANJE DVOSTRUKOG INTEGRALA'//)
      DO 10 K=1,3
      CALL DVINT(EF,K,VRINT)
   10 WRITE(5,15)K,VRINT
   15 FORMAT(15X,I1,'. PRIMER'//10X,'VREDNOST INTEGRALA ='
     1,F12.6//)
      CALL EXIT
      END
```

Primenom ovog programa približno smo izračunali vrednosti sledećih integrala:

1° $\iint\limits_G \frac{16x^2 y^2}{1+x^2+y^2}\, dxdy;$

2° $\iint\limits_G \sqrt{1+(1+x)^2+y^2}\, dxdy;$

3° $\iint\limits_G \frac{24x^2}{\sqrt{2-x^2-y^2}}\, dxdy.$

Funkcijski potprogram EF i izlazna lista imaju oblik:

```
      FUNCTION EF(X,Y,K)
      GO TO(10,20,30),K
   10 EF=(16.*X*X*Y*Y)/(1.+X*X+Y*Y)
      RETURN
   20 EF=SQRT(1.+Y*Y+(1.+X)**2)
      RETURN
   30 EF=(24.*X*X)/SQRT(2.-X*X-Y*Y)
      RETURN
      END
```

```
IZRACUNAVANJE DVOSTRUKOG INTEGRALA

      1. PRIMER

VREDNOST INTEGRALA =     1.256638

      2. PRIMER

VREDNOST INTEGRALA =     4.858377

      3. PRIMER

VREDNOST INTEGRALA =    16.324198
```

5.2. INTEGRALNE JEDNAČINE

5.2.1. Uvod

Jednačina

$$(2.1.1) \quad y(x)=f(x)+\lambda \int_a^b K(x,t)\,y(t)\,dt,$$

gde su f i K poznate funkcije, y nepoznata funkcija i λ numerički parametar, naziva se Fredholmova integralna jednačina druge vrste.

Funkcija K se naziva jezgro integralne jednačine (2.1.1). U našim razmatranjima pretpostavljaćemo uvek da je jezgro definisano i neprekidno na $D = \{(x, t) | a \leqslant x \leqslant b, \ a \leqslant t \leqslant b\}$.

Ako je $f(x) \not\equiv 0$, jednačina (2.1.1) se naziva nehomogenom, a u slučaju kada je $f(x) \equiv 0$, jednačina je homogena.

Integralna jednačina oblika

$$f(x) + \lambda \int_a^b K(x, t)\, y(t)\, dt = 0$$

naziva se Fredholmova integralna jednačina prve vrste.

Volterraove integralne jednačine prve i druge vrste imaju oblike

$$f(x) + \lambda \int_a^x K(x, t)\, y(t)\, dt = 0$$

i

$$y(x) = f(x) + \lambda \int_a^x K(x, t)\, y(t)\, dt$$

respektivno.

Za rešavanje Fredholmove jednačine (2.1.1) često se koristi metod sukcesivnih aproksimacija koji se zasniva na jednakosti

$$(2.1.2) \qquad y_n(x) = f(x) + \lambda \int_a^b K(x, t)\, y_{n-1}(t)\, dt \qquad (n = 1, 2, \ldots,),$$

pri čemu se uzima $y_0 = f(x)$. Naime, ako definišemo niz funkcija $\{\bar{y}_k\}$ pomoću

$$\bar{y}_0(x) = y_0(x) = f(x), \quad \bar{y}_k(x) = \int_a^b K(x, t)\, \bar{y}_{k-1}(t)\, dt \quad (k = 1, 2, \ldots).$$

tada se (2.1.2) može predstaviti u obliku

$$(2.1.3) \qquad y_n(x) = \sum_{k=0}^{n} \lambda^k \bar{y}_k(x) \qquad (n = 1, 2, \ldots).$$

Može se pokazati da niz y_n konvergira ka tačnom rešenju jednačine (2.1.1) ako je ispunjen uslov $|\lambda| < \dfrac{1}{M(b-a)}$ gde je $M = \max\limits_{x,t \in [a,b]} |K(x, t)|$.

5.2.2. Primena kvadraturnih formula na rešavanje Fredholmove integralne jednačine druge vrste

U cilju rešavanja jednačine (2.1.1) uzmimo kvadraturnu formulu

$$(2.2.1) \qquad \int_a^b F(x)\, dx = \sum_{j=1}^{n} A_j F(x_j) + R_n(F),$$

gde su apscise $x_1, \ldots, x_n$ iz $[a, b]$, A_j težinski koeficijenti koji ne zavise od F i $R_n(F)$ odgovarajući ostatak.

Ako u (2.1.1) stavimo redom $x = x_i$ $(i = 1, \ldots, n)$, imamo

$$y(x_i) = f(x_i) + \lambda \int_a^b K(x, t)\, y(t)\, dt \qquad (i = 1, \ldots, n),$$

odakle primenom kvadraturne formule (2.2.1) sleduje

$$(2.2.2) \quad y(x_i) = f(x_i) + \lambda \sum_{k=1}^{n} A_j K(x_i, x_j)\, y(x_j) + R_n(F) \quad (i = 1, \ldots, n),$$

gde je $F_i(t) = K(x_i, t)\, y(t)$ $(i = 1, \ldots, n)$. Odbacivanjem članova $R_n(F_i)$ $(i = 1, \ldots, n)$, na osnovu (2.2.2) dobijamo sistem linearnih jednačina

$$(2.2.3) \qquad y_i - \lambda \sum_{i=1}^{n} A_j K_{ij} y_j = f_i \qquad (i = 1, \ldots, n),$$

gde smo stavili $y_i = y(x_i)$, $f_i = f(x_i)$, $K_{ij} = K(x_i, x_j)$.

Sistem (2.2.3) se može predstaviti i u matričnom obliku

$$\begin{bmatrix} 1 - \lambda A_1 K_{11} & -\lambda A_2 K_{12} & \ldots & -\lambda A_n K_{1n} \\ -\lambda A_1 K_{21} & 1 - \lambda A_2 K_{22} & & -\lambda A_n K_{2n} \\ \vdots & & & \\ -\lambda A_1 K_{n1} & -\lambda A_2 K_{n2} & & 1 - \lambda A_n K_{nn} \end{bmatrix} \cdot \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix}.$$

Rešavanjem dobijenog sistema linearnih jednačina po $y_1, \ldots, y_n$, približno rešenje jednačine (2.1.1) se može predstaviti u obliku

$$(2.2.4) \qquad \tilde{y}(x) = f(x) + \lambda \sum_{j=1}^{n} A_j K(x, x_j)\, y_j.$$

5.2.3. Programska realizacija

Metod izložen u prethodnom odeljku realizovaćemo korišćenjem uopštene Simpsonove kvadraturne formule, kod koje je

$$h = \frac{b-a}{2m}, \quad n = 2m + 1, \quad x_i = a + (i-1)h \quad (i = 1, \ldots, n),$$

$$A_1 = A_{2m+1} = \frac{h}{3}, \quad A_2 = A_4 = \ldots = A_{2m} = \frac{4h}{3}, \quad A_3 = A_5 = \ldots = A_{2m-1} = \frac{2h}{3}$$

Za rešavanje sistema linearnih jednačina (2.2.3) koristićemo potprograme LRFAK i RSTS iz 4.4.6 (poglavlje 4.4).

U potprogramu FRED formira se sistem jednačina (2.2.3). Parametri u listi ovog potprograma imaju sledeće značenje:

X – vektor apscisa kvadraturne formule;
A – vektor težinskih koeficijenata kvadraturne formule;
FK – ime funkcijskog potprograma kojim se zadaje funkcija f i jezgro K;
PL – vrednost parametra λ;
C – matrica sistema (2.2.3), memorisana kao vektor (kolona po kolona);
F – vektor slobodnih članova u sistemu jednačina (2.2.3).

```
      SUBROUTINE FRED(X,A,N,FK,PL,C,F)
      DIMENSION X(1),A(1),C(1),F(1)
      IND=-N
      DO 15 J=1,N
      IND=IND+N
      DO 10 I=1,N
      IJ=IND+I
      C(IJ)=-PL*A(J)*FK(X(I),X(J),2)
      IF(I-J)10,5,10
    5 C(IJ)=1+C(IJ)
   10 CONTINUE
   15 F(J)=FK(X(J),X(I),1)
      RETURN
      END
```

Funkcijski potprogram FK ima sledeće parametre u listi:

X i T – vrednosti argumenata x i t respektivno;
M – celobrojni parametar, koji definiše izračunavanje vrednosti funkcije $f(M = 1)$ i izračunavanje vrednosti jezgra K ($M = 2$) pri zadatim vrednostima argumenata.

```
      FUNCTION FK(X,T,M)
      GO TO(10,20),M
   10 FK=EXP(X)
      RETURN
   20 FK=X*EXP(X*T)
      RETURN
      END
```

Uzimajući kao primer jednačinu

$$y(x) = e^x - \int_0^1 xe^{xt} y(t)\, dt$$

i $M = 1, 2$ ($N = 3, 5$) dobijeni su rezultati koje navodimo iza odgovarajućeg programa. Primetimo da je tačno rešenje date jednačine $y(x) = 1$.

```
      EXTERNAL FK
      DIMENSION X(10),A(10),C(100),B(10),IP(9)
      READ(8,5) PL,DG,GG
    5 FORMAT(3F5.0)
   10 READ(8,15,END=60) M
   15 FORMAT(I2)
      N=2*M+1
      H=(GG-DG)/(2.*FLOAT(M))
      X(1)=DG
      DO 20 I=2,N
   20 X(I)=X(I-1)+H
      Q=H/3.
      A(1)=Q
      A(N)=Q
      DO 25 I=1,M
   25 A(2*I)=4.*Q
      DO 30 I=2,M
   30 A(2*I-1)=2.*Q
      CALL FRED(X,A,N,FK,PL,C,B)
      CALL LRFAK(C,N,IP,DET,KB)
      IF(KB) 35,40,35
   35 WRITE(5,45)
   45 FORMAT(1H0,'MATRICA SISTEMA SINGULARNA'//)
      GO TO 60
   40 CALL RSTS(C,N,IP,B)
      WRITE(5,50) (B(I),I=1,N)
   50 FORMAT(/5X,'RESENJE'//(10F10.5))
      GO TO 10
   60 CALL EXIT
      END
```

```
     RESENJE

 1.00000    0.94328    0.79472

     RESENJE

 1.00000    1.00000    1.00000    1.00000    0.99997
```

5.3. OBIČNE DIFERENCIJALNE JEDNAČINE

5.3.1. Uvod

Ovo poglavlje je posvećeno, uglavnom, rešavanju Cauchyevog problema kod običnih diferencijalnih jednačina, tj. problema sa početnim uslovima. Metodi su razvrstani u dve opšte klase i to:

1° Klasa linearnih višekoračnih metoda,
2° Klasa metoda Runge–Kutta.

Takođe, ukazaćemo i na rešavanje konturnih problema kod običnih diferencijalnih jednačina.

5.3.2. Eulerov metod

Eulerov metod je najprostiji numerički metod za rešavanje Cauchyevog problema

(3.2.1) $\quad y' = f(x, y), \quad y(x_o) = y_o$

i bazira se na približnoj jednakosti

$$y(x) = y(x_o) + (x - x_o)\, y'(x_o),$$

tj.

(3.2.2) $\quad y(x) = y(x_o) + (x - x_o)\, f(x_o, y_o),$

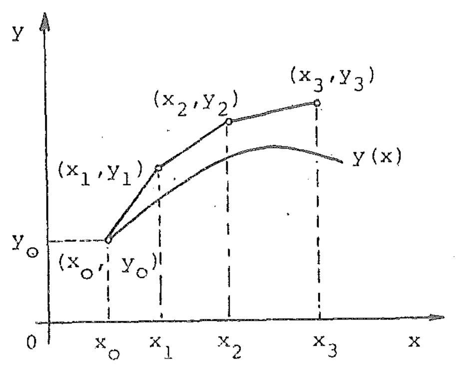
s obzirom na (3.2.1). Ako sa y_1 označimo približnu vrednost za $y(x_1)$, na osnovu (3.2.2) imamo

$$y_1 = y_o + (x_1 - x_o)\, f(x_o, y_o).$$

U opštem slučaju, za proizvoljan skup tačaka $x_o < x_1 < x_2 < \ldots$, približne vrednosti za $y(x_n)$, u oznaci y_n, možemo odrediti pomoću

(3.2.3) $\quad y_{n+1} = y_n + (x_{n+1} - x_n)\, f(x_n, y_n) \qquad (n = 0, 1, \ldots).$

Poslednja formula definiše Eulerov metod, čija je geometrijska interpretacija data na sl. 3.2.1.



sl.3.2.1

Poligonalna linija $(x_o, y_o) - (x_1, y_1) - (x_2, y_2) - \ldots$ poznata je kao Eulerov poligon.

Najčešće se tačke x_n biraju ekvidistantno, tj. $x_{n+1} - x_n = h = \text{const}\ (> 0)$ $(n = 0, 1, \ldots)$ i u tom slučaju (3.2.3) se svodi na

$$y_{n+1} = y_n + hf(x_n, y_n) \qquad (n = 0, 1, \ldots).$$

5.3.3. Opšti linearni višekoračni metod

U ovom i narednim odeljcima ovog poglavlja razmatraćemo jednu opštu klasu metoda za rešavanje Cauchyevog problema

$$(3.3.1) \qquad y' = f(x, y), \quad y(x_o) = y_o \qquad (x_o \leqslant x \leqslant b).$$

Ako segment $[x_o, b]$ podelimo na N podsegmenta dužine $h = \dfrac{b - x_o}{N}$, dobijamo niz tačaka x_n određen sa

$$x_n = x_o + nh \qquad (n = 0, 1, \ldots, N).$$

Neka y_n označava niz približnih vrednosti rešenja problema (3.3.1) u tačkama x_n i neka je $f_n \equiv f(x_n, y_n)$. Pred nama se postavlja problem određivanja niza y_n. Za rešavanje ovog problema razrađen je veliki broj metoda. Jedan od ovih metoda je i Eulerov metod koji je razmatran u prethodnom odeljku. Kod Eulerovog metoda niz y_n se izračunava rekurzivno pomoću

$$(3.3.2) \qquad y_{n+1} - y_n = hf_n \qquad (n = 0, 1, \ldots).$$

pri čemu postoji linearna veza između y_n, y_{n+1} i f_n. U opštem slučaju za izračunavanje niza y_n mogu se koristiti složenije rekurentne relacije, nego što je (3.3.2). Među metodima, koji proističu iz ovih relacija, važnu ulogu igraju metodi kod kojih postoji linearna veza između y_{n+i}, f_{n+i} $(i = 0, 1, \ldots, k)$ i oni čine klasu tzv. linearnih višekoračnih metoda*.

Opšti linearni višekoračni metod može se predstaviti u obliku

$$(3.3.3) \qquad \sum_{i=o}^{k} \alpha_i y_{n+1} = h \sum_{i=o}^{k} \beta_i f_{n+i} \qquad (n = 0, 1, \ldots),$$

gde su α_i i β_i konstantni koeficijenti određeni sa tačnošću do na multiplikovanu konstantu. Da bismo obezbedili njihovu jednoznačnost uzećemo $\alpha_k = 1$.

Ako je $\beta_k = 0$, kažemo da je metod (3.3.3) otvorenog tipa ili da je eksplicitan; u protivnom metod je zatvorenog tipa ili implicitan.

U opštem slučaju (3.3.3) predstavlja nelinearnu diferencnu jednačinu, s obzirom da je $f_{n+i} \equiv f(x_{n+i}, y_{n+i})$.

Za određivanje niza y_n, primenom metoda (3.3.3) potrebno je poznavanje startnih vrednosti y_i $(i = 0, 1, \ldots, k - 1)$. Kako nam je unapred poznata jedino vrednost y_o, poseban problem u primeni višekoračnih metoda (3.3.3) pred-

*) Na engleskom: multistep method.

stavlja određivanje ostalih startnih vrednosti. Ovom problemu biće posvećen poseban odeljak.

Pod pretpostavkom da su poznate startne vrednosti y_i $(i = 0, 1, \ldots, k-1)$, kod eksplicitnih metoda direktno se izračunavaju $y_k, y_{k+1}, \ldots, y_N$ pomoću

$$y_{n+k} = h \sum_{i=0}^{k-1} \beta_i f_{n+i} - \sum_{i=0}^{k-1} \alpha_i y_{n+i} \quad (n = 0, 1, \ldots, N-k).$$

Međutim, kod implicitnih metoda za određivanje vrednosti y_{n+k} treba rešiti jednačinu

$$(3.3.4) \qquad y_{n+k} = h\beta_k f(x_{n+k}, y_{n+k}) + \phi$$

gde je

$$\phi = h \sum_{i=0}^{k-1} \beta_i f_{n+i} - \sum_{i=0}^{k-1} \alpha_i y_{n+i}.$$

Kada je $(x, y) \to f(x, y)$ nelinearna funkcija koja zadovoljava Lipschitzov uslov po y sa konstantom L, jednačina (3.3.4) se može rešiti iterativnim procesom

$$(3.3.5) \qquad y_{n+k}^{[s+1]} = h\beta_k f(x_{n+k}, y_{n+k}^{[s]}) + \phi$$

polazeći od proizvoljne vrednosti $y_{n+k}^{[o]}$ ako je

$$h\,|\beta_k|\,L < 1$$

Uslov dat ovom nejednakošću obezbeđuje konvergenciju iterativnog procesa (3.3.5).

Za metod (3.3.3) definišimo diferencni operator $L_h : C^1[x_o, b] \to C[x_o, b]$ pomoću

$$(3.3.6) \qquad L_h[y] = \sum_{i=0}^{k} [\alpha_i y(x+ih) - h\beta_i y'(x+ih)].$$

Neka funkcija $g \in C^\infty[x_o, b]$. Tada se $L_h[g]$ može predstaviti u obliku

$$(3.3.7) \qquad L_h[g] = C_o g(x) + C_1 h g'(x) + C_2 h^2 g''(x) + \ldots,$$

gde su C_j $(j = 0, 1, \ldots)$ konstante, koje ne zavise od h i g.

Definicija 3.3.1. Linearni višekoračni metod (3.3.3) ima red p ako je u razvoju (3.3.7)

$$C_o = C_1 = \ldots = C_p = 0 \quad \text{i} \quad C_{p+1} \neq 0.$$

Neka je $x \to y(x)$ tačno rešenje problema (3.3.1) i y_n niz približnih vrednosti ovog rešenja u tačkama $x_n = x_o + nh$ $(n = 0, 1, \ldots, N)$ dobijen primenom metoda (3.3.3), sa startnim vrednostima $y_i = s_i(h)$ $(i = 0, 1, \ldots, k-1)$.

Definicija 3.3.1. Za linearni višekoračni metod (3.3.3) se kaže da je konvergentan ako je za svako $x \in [x_o, b]$

$$\lim_{\substack{x \to 0 \\ x - x_o = nh}} y_n = y(x)$$

i ako za startne vrednosti važi $\lim_{h \to 0} s_i(h) = y_o$ $(i = 0, 1, \ldots, k-1)$.

Linearni višekoračni metod (3.3.3) se može okarakterisati prvim i drugim karakterističnim polinomom koji su dati respektivno pomoću

$$\rho(\xi) = \sum_{i=0}^{k} \alpha_i \xi^i \quad \text{i} \quad \sigma(\xi) = \sum_{i=o}^{k} \beta_i \xi^i .$$

Dve važne klase konvergentnih višekoračnih metoda koje se sreću u primenama su:

1° Metodi kod kojih je $\rho(\xi) = \xi^k - \xi^{k-1}$;

2° Metodi kod kojih je $\rho(\xi) = \xi^k - \xi^{k-2}$.

Eksplicitni metodi prve klase nazivaju se Adams–Bashfortovi, a implicitni Adams–Moultonovi. Slično, eksplicitni metodi druge klase nose naziv Nystromovi metodi, dok se odgovarajući implicitni metodi nazivaju generalisani Milne–Simpsonovi.

Naravno, postoje metodi koji ne pripadaju ovim klasama.

5.3.4. Izbor startnih vrednosti

Kao što je ranije napomenuto, kod primene linearnih višekoračnih metoda na rešavanje problema (3.3.1), potrebno je poznavanje startnih vrednosti $y_i = s_i(h)$, takvih da je $\lim_{h \to o} s_i(h) = y_o$ $(i = 1, \ldots, k-1)$. Naravno, ovaj problem se postavlja kada je $k > 1$.

Ako je metod (3.3.3) reda p, tada očigledno startne vrednosti $s_i(h)$ treba birati tako da je

$$s_i(h) - y(x_i) = 0(h^{p+1}) \qquad (i = 1, \ldots, k-1),$$

gde je $x \to y(x)$ tačno rešenje problema (3.3.1).

U ovom odeljku navešćemo jednu klasu metoda za određivanje potrebnih startnih vrednosti.

Pretpostavimo da je funkcija f u diferencijalnoj jednačini (3.3.1) dovoljan broj puta diferencijabilna. Tada na osnovu Taylorovog metoda imamo

$$y(x_o+h) = y(x_o) + hy'(x_o) + \frac{h^2}{2!}y''(x_o) + \ldots + \frac{h^p}{p!}y^{(p)}(x_o) + 0(h^{p+1}).$$

Poslednja jednakost ukazuje na to da se može uzeti

$$s_1(h) = y(x_o) + hy'(x_o) + \frac{h^2}{2!}y''(x_o) + \ldots + \frac{h^p}{p!}y^{(p)}(x_o),$$

s obzirom da je tada $s_i(h) - y(x_1) = 0(h^{p+1})$ $(x_1 = x_o + h)$. Isti postupak se može primeniti i na određivanje ostalih startnih vrednosti. Naime, u opštem slučaju, imamo

$$s_i(h) = y(x_{i-1}) + hy'(x_{i-1}) + \frac{h^2}{2!}y''(x_{i-1}) + \ldots + \frac{h^p}{p!}y^{(p)}(x_{i-1}) \quad (i = 1, \ldots, k-1),$$

pri čemu za $y(x_{i-1})$ uzimamo $s_{i-1}(h)$.

5.3.5. Prediktor–korektor metodi

Kao što je navedeno u odeljku 5.3.3 primena implicitnih metoda je skopčana sa rešavanjem jednačine (3.3.4) na svakom koraku integracije, pri čemu se za ovo rešavanje koristi iterativni proces (3.3.5). Bez obzira na ovu teškoću implicitni metodi se dosta koriste za rešavanje Cauchyevog problema, s obzirom da imaju niz prednosti nad eksplicitnim metodima (viši red, bolja numerička stabilnost). Početna vrednost $y_{n+k}^{[o]}$, u primenama se određuje korišćenjem nekog eksplicitnog metoda, koji tada nazivamo prediktor. Implicitni metod (3.3.4) nazivamo korektor. Metod dobijen ovakvom kombinacijom nazivamo prediktor–korektor metod.

Za određivanje y_{n+k}, iterativni proces (3.3.5) treba primenjivati sve dok ne bude ispunjen uslov

$$\left|y_{n+k}^{[s+1]} - y_{n+k}^{[s]}\right| < \epsilon,$$

gde je ϵ dozvoljena greška, obično reda lokalne greške zaokrugljivanja. Tada se za y_{n+k} može uzeti $y_{n+k}^{[s+1]}$.

Međutim, ovakav način se najčešće ne primenjuje u praksi, s obzirom da zahteva veliki broj izračunavanja vrednosti funkcije f po jednom koraku i uz to je ovaj broj promenljiv od koraka do koraka. Da bi se smanjio ovaj broj izračunavanja, broj iteracija u (3.3.5) se fiksira. Dakle, uzima se samo $s = 0, 1, \ldots, m-1$.

5.3.6. Programska realizacija višekoračnih metoda

U ovom odeljku daćemo programsku realizaciju kako eksplicitnih tako i implicitnih metoda. Dobijene programe testiraćemo na primeru (sa $h = 0.1$).

$$y' = x^2 + y, \qquad y(1) = 1 \qquad (1 \leqslant x \leqslant 2).$$

Tačno rešenje ovog problema je $y(x) = 6e^{x-1} - x^2 - 2x - 2$.

3.6.1. Eulerov metod

$$y_{n+1} - y_n = hf_n \qquad (n = 0, 1, \ldots).$$

čiji je red $p = 1$, i Adams–Bashfortov metod trećeg reda

$$y_{n+3} - y_{n+2} = \frac{h}{12}(23f_{n+2} - 16f_{n+1} + 5f_n) \qquad (n = 0, 1, \ldots),$$

realizovani su pomoću potprograma EULER i ADAMS respektivno.

```
      SUBROUTINE EULER(XP,XK,H,Y,FUN)
      DIMENSION Y(1)
      N=(XK-XP)/H
      X=XP
      DO 11 I=1,N
      Y(I+1)=Y(I)+H*FUN(X,Y(I))
   11 X=X+H
      RETURN
      END


      FUNCTION FUN(X,Y)
      FUN=X*X+Y
      RETURN
      END


      SUBROUTINE ADAMS(XP,XK,H,Y,FUN)
      DIMENSION Y(1)
      N=(XK-XP)/H
      X=XP
      F0=FUN(X,Y(1))
      F1=FUN(X+H,Y(2))
      N2=N-2
      DO 11 I=1,N2
      F2=FUN(X+2.*H,Y(I+2))
      Y(I+3)=Y(I+2)+H*(23.*F2-16.*F1+5.*F0)/12.
      F0=F1
      F1=F2
   11 X=X+H
      RETURN
      END
```

Parametri u potprogramskoj listi imaju sledeće značenje:

XP i *XK* – početna i krajnja tačka intervala integracije;
H – korak integracije;
Y – vektor približnih vrednosti rešenja dobijen višekoračnim metodom, pri čemu kod Eulerovog metoda $Y(1)$ predstavlja datu početnu vrednost, a kod Adamsovog metoda startne vrednosti su date kroz $Y(1)$, $Y(2)$ i $Y(3)$;
FUN – ime funkcijskog potprograma kojim se definiše desna strana diferencijalne jednačine $f(x, y)$.

Startne vrednosti za Adamsov metod određujemo primenom Taylorovog metoda za $p = 3$ (videti odeljak 5.3.4). Naime, kako je

$$y(1) = 1, \quad y'(1) = 2, \quad y''(1) = 4, \quad y'''(1) = 6, \quad h = 0.1,$$

dobijamo $Y(1) = 1.$, $Y(2) = 1.221$, $Y(3) = 1.48836$.

Glavni program i izlazna lista imaju oblik:

```
C================================================================
C  RESAVANJE DIFERENCIJALNIH JEDNACINA EKSPLICITNIM METODIMA
C================================================================
      EXTERNAL FUN
      DIMENSION Y(100),Z(100)
      F(X)=6.*EXP(X-1.)-X*X-2.*X-2.
      WRITE(5,10)
   10 FORMAT(8X,'RESAVANJE DIFERENCIJAL.JED.EKSPLICITNIM',
     1' METODIMA'//8X,'XN',8X,'YN(I)',5X,'GRESKA(%)',3X,
     2'YN(II)',4X,'GRESKA(%)'/)
      XP=1.
      XK=2.
      H=0.1
      Y(1)=1.
      CALL EULER(XP,XK,H,Y,FUN)
      Z(1)=Y(1)
      Z(2)=1.221
      Z(3)=1.48836
      CALL ADAMS(XP,XK,H,Z,FUN)
      N=(XK-XP)/H
      NN=N+1
      X=XP
      DO 22 I=1,NN
      G1=ABS((Y(I)-F(X))/F(X))*100.
      G2=ABS((Z(I)-F(X))/F(X))*100.
      WRITE(5,20)X,Y(I),G1,Z(I),G2
   22 X=X+H
   20 FORMAT(8X,F3.1,2(4X,F9.5,4X,F5.2))
      CALL EXIT
      END
```

```
        RESAVANJE DIFERENCIJAL.JED.EKSPLICITNIM METODIMA

        XN        YN(I)     GRESKA(%)    YN(II)     GRESKA(%)

        1.0      1.00000     0.00       1.00000      0.00
        1.1      1.20000     1.72       1.22100      0.00
        1.2      1.44100     3.19       1.48836      0.00
```

10

1.3	1.72910	4.42	1.80883	0.02
1.4	2.07101	5.47	2.19028	0.03
1.5	2.47411	6.37	2.64126	0.04
1.6	2.94652	7.13	3.17116	0.05
1.7	3.49717	7.79	3.79040	0.06
1.8	4.13589	8.36	4.51045	0.06
1.9	4.87348	8.87	5.34403	0.07
2.0	5.72183	9.32	6.30518	0.07

3.6.2. Uzimajući Eulerov metod kao prediktor i trapezno pravilo ($p = 2$)

$$y_{n+1} - y_n = \frac{h}{2}(f_n + f_{n+1}) \qquad (n = 0, 1, \ldots)$$

kao korektor (sa brojem iteracija $m = 2$) obrazovan je potprogram PREKOR. Glavni program, potprogram i izlazni rezultati imaju oblik:

```
C=======================================================================
C  RESAVANJE DIF.JED. METODOM PREDIKTOR-KOREKTOR
C=======================================================================
      EXTERNAL FUN
      DIMENSION Y(100)
      F(X)=6.*EXP(X-1.)-X*X-2.*X-2.
      WRITE(5,10)
   10 FORMAT(8X,'RESAVANJE DIF.JED. METODOM PREDIKTOR-KOREK'
     1,'TOR'     //15X,'XN',13X,'YN',10X,'GRESKA(%)'/)
      READ(8,5)XP,XK,YP,H
    5 FORMAT(4F6.1)
      CALL PREKOR(XP,XK,YP,H,Y,FUN)
      N=(XK-XP)/H
      NN=N+1
      X=XP
      DO 11 I=1,NN
      G=ABS((Y(I)-F(X))/F(X))*100.
      WRITE(5,15)X,Y(I),G
   15 FORMAT(15X,F3.1,8X,F9.5,8X,F5.2)
   11 X=X+H
      CALL EXIT
      END


      SUBROUTINE PREKOR(XP,XK,YP,H,Y,FUN)
      DIMENSION Y(100)
      N=(XK-XP)/H
      X=XP
      Y(1)=YP
      DO 10 I=1,N
C     PROGNOZIRANJE VREDNOSTI
      FXY=FUN(X,Y(I))
      YP=Y(I)+H*FXY
C     KOREKCIJA VREDNOSTI
      DO 20 M=1,2
   20 YP=Y(I)+H/2.*(FXY+FUN(X+H,YP))
      Y(I+1)=YP
   10 X=X+H
      RETURN
      END
```

RESAVANJE DIF.JED. METODOM PREDIKTOR-KOREKTOR

XN	YN	GRESKA(%)
1.0	1.00000	0.00
1.1	1.22152	0.04
1.2	1.48952	0.07
1.3	1.81097	0.10
1.4	2.19363	0.12
1.5	2.64602	0.14
1.6	3.17760	0.15
1.7	3.79881	0.17
1.8	4.52118	0.18
1.9	5.35747	0.18
2.0	6.32177	0.19

5.3.7. Metodi Runge–Kutta

U prethodnim odeljcima razmatrani su linearni višekoračni metodi za rešavanje Cauchyevog problema (3.3.1). Red ovih metoda se može povećati povećanjem broja koraka. Međutim, ukoliko se žrtvuje linearnost koju poseduju ovi metodi, moguće je konstruisati jednokoračne metode sa proizvoljnim redom.

Za rešavanje Cauchyevog problema oblika (3.3.1) sa dovoljno puta diferencijabilnom funkcijom f, moguće je, takođe, konstruisati jednokoračne metode višeg reda (na primer, Taylorov metod).

Posmatrajmo opšti eksplicitni jednokoračni metod

$$y_{n+1} - y_n = h\,\phi\,(x_n, y_n, h) \tag{3.7.1}$$

Definicija 3.7.1. Metod (3.7.1) je reda p ako je p najveći ceo broj za koji važi

$$y\,(x+h) - y\,(x) - h\,\phi\,(x, y\,(x), h) = 0\,(h^{p+1}),$$

gde je $x \to y\,(x)$ tačno rešenje problema (3.3.1).

Definicija 3.7.2. Metod (3.7.1) je konzistentan ako je $\phi\,(x, y, 0) \equiv f\,(x, y)$.

Primetimo da je Taylorov metod specijalan slučaj metoda (3.7.1). Naime, kod Taylorovog metoda reda p imamo

$$\phi\,(x, y, h) = \phi_T\,(x, y, h) = \sum_{i=0}^{p-1} \frac{h^i}{(i+1)\,!}\left(\frac{\partial}{\partial x} + f\,\frac{\partial}{\partial y}\right)^i f\,(x, y). \tag{3.7.2}$$

U specijalnom slučaju, kod Eulerovog metoda je $\phi\,(x, y, h) = f\,(x, y)$.

U ovom odeljku razmatraćemo jednu specijalnu klasu metoda, oblika (3.7.1), koju je 1895. godine predložio C. Runge ([4]). Kasnije, ovu klasu metoda razvili su W. Kutta ([5]) i K. Heun ([6]).

Kao što ćemo kasnije videti, svi ovi metodi sadrže slobodne parametre. S obzirom na vreme u kome su se pojavili ovi metodi, slobodni parametri su birani tako da se dobiju što jednostavnije formule za praktično računanje. Međutim, ovakve vrednosti parametra ne obezbeđuju optimalne karakteristike posmatranih metoda. U daljem tekstu ove metode zvaćemo klasičnim.

Opšti eksplicitni metod Runge–Kutta ima oblik

$$(3.7.3) \qquad y_{n+1} - y_n = h\,\phi\,(x_n, y_n, h),$$

gde su

$$\phi\,(x, y, h) = \sum_{i=1}^{m} c_i\, k_i,$$

$$k_1 = f\,(x, y),$$

$$k_i = f\,(x + a_i h,\ y + b_i\ h) \qquad (i = 2, \ldots, m).$$

$$(3.7.4) \qquad a_i = \sum_{j=1}^{i-1} \alpha_{ij}, \quad b_i = \sum_{j=1}^{i-1} \alpha_{ij}\, k_j$$

Primetimo da iz uslova konzistencije metoda (3.7.3) sleduje $\sum_{i=1}^{m} c_i = 1$.

Nepoznate koeficijente koji figurišu u ovom metodu, određujemo iz uslova da metod ima maksimalni red. Pri ovome, koristimo sledeću činjenicu: Ako se $\phi\,(x, y, h)$, razvijeno po stepenima od h, može predstaviti u obliku

$$\phi\,(x, y, h) = \phi_T\,(x, y, h) + 0\,(h^p),$$

gde je ϕ_T definisano pomoću (3.7.2), tada je metod (3.7.3) reda p.

Prethodno nađimo razvoj $\phi_T\,(x, y, h)$ po stepenim od h. Korišćenjem Monge-ovih oznaka za parcijalne izvode imamo

$$\left(\frac{\partial}{\partial x} + \frac{\partial}{\partial y}\right) f = f_x + f f_y = F$$

i

$$\left(\frac{\partial}{\partial x} + f\,\frac{\partial}{\partial y}\right)^2 f = \left(\frac{\partial}{\partial x} + f\,\frac{\partial}{\partial y}\right) F = G + f_y F,$$

gde smo stavili $G = f_{xx} + 2 f f_{xy} + f^2 f_{yy}$. Tada iz (3.7.2) sleduje

$$(3.7.5) \qquad \phi_T\,(x, y, h) = f + \frac{1}{2}\,hF + \frac{1}{6}\,h^2\,(G + f_y F) + 0\,(h^3).$$

Razmotrićemo sada samo metode Runge–Kutta, čiji je red $p \leqslant 3$. Pokazuje se da je za dobijanje metoda trećeg reda dovoljno uzeti $m = 3$. U tom slučaju, formule (3.7.3) se svode na

$$\phi(x, y, h) = c_1 k_1 + c_2 k_2 + c_3 c_4$$

$$k_1 = f(x, y),$$

$$k_2 = f(x + a_2 h, \quad y + b_2 h),$$

$$k_3 = f(x + a_3 h, \quad y + b_3 h)$$

i

$$a_2 = \alpha_{21}, \quad b_2 = \alpha_{21} k_1,$$

$$a_3 = \alpha_{31} + \alpha_{32}, \quad b_3 = \alpha_{31} k_1 + \alpha_{32} k_2.$$

Razvijanjem funkcije k_2 u Taylorov red, u okolini tačke (x, y), dobijamo

$$k_2 = f + a_2 Fh + \frac{1}{2} a_2^2 Gh^2 + 0(h^3).$$

Kako je

$$b_3 = \alpha_{31} k_1 + \alpha_{32} k_2 = \alpha_{31} f + \alpha_{32} (f + a_2 Fh + \frac{1}{2} a_2^2 Gh^2) + 0(h^3)$$

imamo

$$b_3 = a_3 f + a_2 \alpha_{32} Fh + 0(h^2) \quad \text{i} \quad b_3^2 = a_3^2 f^2 + 0(h).$$

Razvijanjem funkcije k_3 u okolini tačke (x, y) i korišćenjem poslednjih jednakosti imamo

$$k_3 = f + a_3 Fh + \frac{1}{2} (2 a_3 \alpha_{32} Ff_y + a_3^2 G) h^2 + 0(h^3).$$

Najzad, zamenom dobijenih izraza za k_1, k_2, k_3 u izrazu za $\phi(x, y, h)$ dobijamo

$$\phi(x, y, h) = (c_1 + c_2 + c_3) f + (c_2 a_2 + c_3 c_3) Fh$$
$$+ (c_2 a_2^2 G + 2 c_3 a_2 \alpha_{32} Ff_y + c_3 a_3^2 G) \frac{h^2}{2} + 0(h^3).$$

Poslednja jednakost dozvoljava konstrukciju metoda za $m = 1, 2, 3$.

Slučaj m = 1. Kako je $c_2 = c_3 = 0$ imamo

$$\phi(x, y, h) = c_1 f + 0(h^3).$$

Upoređivanjem sa (3.7.5) dobijamo

$$\phi_T(x, y, h) - \phi(x, y, h) = (1 - c_1)f + \frac{1}{2}h^2(G + f_y F) + 0(h^3),$$

odakle zaključujemo da se za $c_1 = 1$, dobija metod

$$y_{n+1} - y_n = hf_n,$$

čiji je red $p = 1$. S obzirom da je ovo Eulerov metod mi vidimo da on pripada i klasi metoda Runge–Kutta.

Slučaj m = 2. Ovde je $c_3 = 0$ i

$$\phi(x, y, h) = (c_1 + c_2)f + c_2 a_2 Fh + \frac{1}{2} c_2 a_2^2 Gh^2 + 0(h^3)$$

Kako je

$$\phi(x, y, h) - \phi_T(x, y, h) = (c_1 + c_2 - 1)f + (c_2 a_2 - \frac{1}{2})Fh +$$
$$+ \frac{1}{6}[(3c_2 a_2^2 - 1)G - f_y F]h^2 + 0(h^3),$$

zaključujemo da se pod uslovima

(3.7.6) $\qquad c_1 + c_2 = 1 \quad$ i $\quad c_2 a_2 = \frac{1}{2}$,

dobija metod drugog reda sa jednim slobodnim parametrom. Naime, iz sistema jednakosti (3.7.6) sleduje

$$c_2 = \frac{1}{2a_2} \quad \text{i} \quad c_1 = \frac{2a_2 - 1}{2a_2},$$

gde je a_2 ($\neq 0$) slobodan parametar. Dakle, sa $m = 2$ imamo jednoparametarsku familiju metoda

$$y_{n+1} - y_n = \frac{h}{2a_2}((2a_2 - 1)k_1 + k_2),$$

$$k_1 = f(x_n, y_n),$$

$$k_2 = f(x_n + a_2 h, y_n + a_2 k_1 h).$$

U specijalnom slučaju, za $a_2 = \frac{1}{2}$, dobijamo Euler–Cauchyev metod

$$y_{n+1} - y_n = hf(x_n + \frac{1}{2}h,\ y_n + \frac{1}{2}hf(x_n, y_n)).$$

Slično, za $a_2 = 1$, dobijamo tzv. poboljšan Euler–Cauchyev metod

$$y_{n+1} - y_n = \frac{h}{2}[f(x_n, y_n) + f(x_n + h,\ y_n + hf(x_n, y_n))].$$

O geometrijskoj interpretaciji dobijenih metoda videti, na primer, [7].

Slučaj m = 3. Kako je

$$\phi(x, y, h) - \phi_T(x, y, h) = (c_1 + c_2 + c_3 - 1) f + (c_2 a_2 + c_3 a_3 - \frac{1}{2}) Fh +$$

$$+ [(c_2 a_2^2 + c_3 a_3^2 - \frac{1}{3}) G + (2 c_3 a_2 \alpha_{32} - \frac{1}{3}) Ff_y] \frac{h^2}{2} + 0(h^3)$$

zaključujemo da su za dobijanje metoda trećeg reda dovoljni uslovi

$$\begin{aligned} &c_1 + c_2 + c_3 = 1, \\ &c_2 a_2 + c_3 a_3 = \frac{1}{2}, \\ &c_2 a_2^2 + c_3 a_3^2 = \frac{1}{3}, \\ &c_3 a_2 \alpha_{32} = \frac{1}{6}. \end{aligned} \tag{3.7.7}$$

S obzirom da imamo četiri jednačine sa šest nepoznatih, izlazi da, u slučaju $m = 3$, imamo dvoparametarsku familiju metoda Runge–Kutta. Može se pokazati da među metodima ove familije ne postoji ni jedan metod čiji je red veći od tri.

U specijalnom slučaju kada je $a_2 = \frac{1}{3}$ i $a_3 = \frac{2}{3}$, iz (3.7.7) sleduje $c_1 = \frac{1}{4}$, $c_2 = 0$, $c_3 = \frac{3}{4}$, $\alpha_{32} = \frac{2}{3}$. Dakle dobili smo metod

$$\begin{aligned} &y_{n+1} - y_n = \frac{h}{4}(k_1 + 3k_3), \\ &k_1 = f(x_n, y_n), \\ &k_2 = f(x_n + \frac{h}{3}, y_n + \frac{h}{3} k_1), \\ &k_3 = f(x_n + \frac{2h}{3}, y_n + \frac{2h}{3} k_2), \end{aligned}$$

koji se u literaturi sreće kao Heunov metod.

Za $a_2 = \frac{1}{2}$, $a_3 = 1$ $(\Rightarrow c_1 = c_3 = \frac{1}{6}$, $c_2 = \frac{2}{3}$, $\alpha_{32} = 2)$

dobijamo metod

$$y_{n+1} - y_n = \frac{h}{6}(k_1 + 4k_2 + k_3),$$

$$k_1 = f(x_n, y_n),$$
$$k_2 = f\left(x_n + \frac{h}{2},\ y_n + \frac{h}{2}k_1\right),$$
$$k_3 = f(x_n + h,\ y_n - hk_1 + 2hk_2),$$

koji je najpopularniji među metodima trećeg reda sa stanovišta ručnog izračunavanja.

U slučaju kada je $m = 4$, dobijamo dvoparametarsku familiju metode četvrtog reda. Naime, ovde se, analogno sistemu (3.7.7), javlja sistem od 11 jednačina sa 13 nepoznatih.

Sada navodimo, bez dokaza, metod Runge–Kutta četvrtog reda

$$y_{n+1} - y_n = \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4), \tag{3.7.8}$$
$$k_1 = f(x_n, y_n),$$
$$k_2 = f\left(x_n + \frac{h}{2},\ y_n + \frac{h}{2}k_1\right),$$
$$k_3 = f\left(x_n + \frac{h}{2},\ y_n + \frac{h}{2}k_2\right),$$
$$k_4 = f(x_n + h,\ y_n + hk_3),$$

koji se u primenama tradicionalno najviše koristi.

Od metoda četvrtog reda često se koristi i tzv. Gillova varijanta ([8]), koja se može iskazati sledećim rekurzivnim postupkom:

$n := 0, Q_o := 0$

(*) $Y_o := y_n,\ k_1 := hf(x_n, Y_o),\quad Y_1 := Y_o + \frac{1}{2}(k_1 - 2Q_o),$

$Q_1 := Q_o + \frac{3}{2}(k_1 - 2Q_o) - \frac{1}{2}k_1,\quad k_2 := hf\left(x_n + \frac{h}{2}, Y_1\right),$

$Y_2 := Y_1 + (1 - \sqrt{1/2})(k_2 - Q_1),\ Q_2 := Q_1 + 3(1 - \sqrt{1/2})(k_2 - Q_1)(1 - \sqrt{1/2})k_2$

$k_3 := hf\left(x_n + \frac{h}{2}, Y_2\right),\ Y_3 := Y_2 + (1 + \sqrt{1/2})(k_3 - Q_2),$

$Q_3 := Q_2 + 3(1 + \sqrt{1/2})(k_3 - Q_2) - (1 + \sqrt{1/2})k_3,\ k_4 := hf(x_n + h, Y_3),$

$Y_4 := Y_3 + \frac{1}{6}(k_4 - 2Q_3),\ Q_o := Q_3 + \frac{1}{2}(k_4 - 2Q_3) - \frac{1}{2}k_4,\ y_{n+1} := Y_4$

$n := n + 1$

Preći na (*).

Za razliku od linearnih višekoračnih metoda, metodi Runge–Kutta ne zahtevaju poznavanje startnih vrednosti (sem $y(x_0) = y_0$, koja, inače, definiše Cauchyev problem), ali su za praktičnu primenu znatno komplikovaniji, s obzirom da zahtevaju m izračunavanja vrednosti funkcije f u svakom koraku.

5.3.8. Programska realizacija metoda Runge–Kutta

U ovom odeljku dajemo programsku realizaciju Euler–Cauchyevog, poboljšanog Euler–Cauchyevog metoda, kao i metoda četvrtog reda (3.7.8) i Gillove varijante metoda Runge–Kutta. Dobijene programe testiraćemo na primeru iz odeljka 5.3.6.

3.8.1. Potprogramom EULCAU realizovani su Euler–Cauchyev i poboljšan Euler–Cauchyev metod. Parametri u listi imaju sledeće značenje

- XP – početna tačka intervala integracije;
- H – korak integracije;
- N – ceo broj, takav da je $N + 1$ dužina vektora Y;
- M – ceo broj koji definiše način konstrukcije vektora Y. Naime, u vektoru Y se redom memoriše svaka M–ta vrednost rešenja dobijena u procesu integracije;
- Y – vektor rešenja dužine $N + 1$, pri čemu $Y(1)$ predstavlja zadati početni uslov y_0, $Y(2)$ je vrednost rešenja dobijena integracijom u tački $XP + M \cdot H$, itd.
- FUN – ime funkcijskog potprograma, kojim se definiše desna strana diferencijalne jednačine $f(x, y)$;
- K – ceo broj, sa vrednostima $K = 1$ i $K = 2$, kojim se zadaje integracija po Euler–Cauchyevom i poboljšanom Euler–Cauchyevom metodu respektivno.

Potprogram EULCAU ima oblik:

```
      SUBROUTINE EULCAU(XP,H,N,M,Y,FUN,K)
      DIMENSION Y(1)
      X=XP
      Y1=Y(1)
      NN=N+1
      DO 10 I=2,NN
      DO 20 J=1,M
      Y0=Y1
      Y1=FUN(X,Y0)
      GO TO(1,2),K
    1 Y1=Y0+H*FUN(X+0.5*H,Y0+0.5*H*Y1)
      GO TO 20
    2 Y1=Y0+H*(Y1+FUN(X+H,Y0+H*Y1))/2.
   20 X=X+H
   10 Y(I)=Y1
      RETURN
      END
```

Glavni program i izlazna lista su dati u daljem tekstu. Kao ulazne parametre za integraciju smo uzeli $H = 0.1$, $N = 10$, $M = 1$, a u drugom slučaju $H = 0.05$, $N = 10$, $M = 2$. Kolone Y1N i Y2N, u izlaznoj listi daju vrednosti za rešenje datog Cauchyevog problema, po običnom i poboljšanom Euler–Cauchyevom metodu respektivno. Pored ovih kolona, u izlaznoj listi su date i kolone sa odgovarajućim greškama (izražene u %) u odnosu na tačno rešenje.

```
C==================================================================
C  RESAVANJE DIF.JED.EULER-CAUCHYEVIM I POBOLJSANIM METODOM
C==================================================================
      EXTERNAL FUN
      DIMENSION Y(100),Z(100)
      F(X)=6.*EXP(X-1.)-X*X-2.*X-2.
      WRITE(5,10)
   10 FORMAT(10X,'RESAVANJE DIF.JED. EULER-CAUCHYEVIM I PO',
     1'BOLJSANIM METODOM'  )
   20 READ(8,25,END=99)XP,Y(1),H,N,M
   25 FORMAT(3F6.1,2I3)
      CALL EULCAU(XP,H,N,M,Y,FUN,1)
      Z(1)=Y(1)
      CALL EULCAU(XP,H,N,M,Z,FUN,2)
      WRITE(5,30)H
   30 FORMAT(1H0,30X,'(H=',F6.4,')'//15X,'XN', 8X,'Y1N', 4X,
     1'GRESKA(%)', 5X,'Y2N', 4X,'GRESKA(%)'/)
      NN=N+1
      X=XP
      DO 11 I=1,NN
      G1=ABS((Y(I)-F(X))/F(X))*100.
      G2=ABS((Z(I)-F(X))/F(X))*100.
      WRITE(5,15)X,Y(I),G1,Z(I),G2
   15 FORMAT(15X,F3.1,3X,F9.6,2X,F7.5,3X,F9.6,2X,F7.5)
   11 X=X+H*M
      GO TO 20
   99 CALL EXIT
      END
```

RESAVANJE DIF.JED. EULER-CAUCHYEVIM I POBOLJSANIM METODOM

(H=0.1000)

XN	Y1N	GRESKA(%)	Y2N	GRESKA(%)
1.0	1.000000	0.00000	1.000000	0.00000
1.1	1.220250	0.06349	1.220500	0.04302
1.2	1.486676	0.11696	1.487203	0.08161
1.3	1.806227	0.16179	1.807059	0.11583
1.4	2.186581	0.19934	2.187750	0.14598
1.5	2.636222	0.23109	2.637764	0.17274
1.6	3.164526	0.25814	3.166479	0.19656
1.7	3.781851	0.28130	3.784260	0.21778
1.8	4.499645	0.30134	4.502557	0.23682
1.9	5.330558	0.31909	5.334026	0.25424
2.0	6.288567	0.33481	6.292649	0.27012

(H=0.0500)

XN	Y1N	GRESKA(%)	Y2N	GRESKA(%)
1.0	1.000000	0.00000	1.000000	0.00000
1.1	1.220824	0.01652	1.220888	0.01128
1.2	1.487963	0.03049	1.488098	0.02143
1.3	1.808391	0.04219	1.808604	0.03041
1.4	2.189811	0.05192	2.190111	0.03824
1.5	2.640738	0.06019	2.641133	0.04522
1.6	3.170581	0.06728	3.171082	0.05148
1.7	3.789740	0.07329	3.790357	0.05700
1.8	4.509705	0.07845	4.510452	0.06190
1.9	5.343177	0.08311	5.344067	0.06643
2.0	6.304192	0.08717	6.305239	0.07058

3.8.2. Prema formularna (3.7.8) za standardni metod Runge–Kutta četvrtog reda obrazovan je potprogram RK4:

```
      SUBROUTINE RK4(X0,Y0,H,M,N,YVEK,F)
C     ==================================
C     METOD RUNGE-KUTTA CETVRTOG REDA
C     ==================================
      DIMENSION YVEK(1)
      T=H/2.
      X=X0
      Y=Y0
      DO 20 I=1,N
      DO 10 J=1,M
      A=F(X,Y)
      B=F(X+T,Y+T*A)
      C=F(X+T,Y+T*B)
      D=F(X+H,Y+H*C)
      X=X+H
   10 Y=Y+H/6.*(A+2.*B+2.*C+D)
   20 YVEK(I)=Y
      RETURN
      END
```

Parametri u listi imaju sledeće značenje:

*X*0, *X*0 – definišu zadati početni uslov ($Y0 = y\,(X0)$;

H – korak integracije;

M, N – celi brojevi sa značenjem sličnim kao u potprogramu EULCAU;

YVEK – vektor dužine *N* koji se dobija kao rezultat numeričke integracije, pri čemu je Y (1) vrednost dobijena u tački $X0 + M \cdot H$, Y (2) vrednost u tački $X0 + 2M \cdot H$, itd.

F – ime funkcijskog potprograma kojim se definiše desna strana diferencijalne jednačine $f\,(x, y)$.

Glavni program ima oblik:

```
C=================================================================
C  RESAVANJE DIF.JED. METODOM RUNGE-KUTTA
C=================================================================
      EXTERNAL FUN
      DIMENSION Y(100)
      F(X)=6.*EXP(X-1.)-X*X-2.*X-2.
      WRITE(5,10)
   10 FORMAT(14X,'RESAVANJE DIF.JED. METODOM RUNGE-KUTTA' )
   20 READ(8,5,END=99)X0,Y0,H,N,M
    5 FORMAT(3F6.1,2I3)
      CALL RK4(X0,Y0,H,M,N,Y,FUN)
      G=0.
      WRITE(5,25)H,X0,Y0,G
   25 FORMAT(1H0,28X,'(H=',F6.4,')'//15X,'XN',13X,'YN',10X,
     1'GRESKA(%)'//15X,F3.1,8X,F9.6,7X,F7.5)
      X=X0
      DO 11 I=1,N
      X=X+H*M
      G=ABS((Y(I)-F(X))/F(X))*100.
   11 WRITE(5,15)X,Y(I),G
   15 FORMAT(15X,F3.1,8X,F9.6,7X,F7.5)
      GO TO 20
   99 CALL EXIT
      END
```

Uzimajući $H = 0.1, N = 10, M = 1$ dobijeni su sledeći rezultati:

```
RESAVANJE DIF.JED. METODOM RUNGE-KUTTA

                (H=0.1000)

XN              YN              GRESKA(%)

1.0             1.000000        0.00000
1.1             1.221025        0.00000
1.2             1.488416        0.00008
1.3             1.809152        0.00014
1.4             2.190947        0.00009
1.5             2.642325        0.00011
1.6             3.172710        0.00019
1.7             3.792512        0.00018
1.8             4.513240        0.00012
1.9             5.347612        0.00019
2.0             6.309683        0.00015
```

3.8.3. Gillovu varijantu metoda Runge–Kutta realizovaćemo u dvostrukoj tačnosti. Parametri u listi potprograma GILL, X0, H, N, M. Y, FUN imaju isto značenje respektivno kao parametri HP, H, N, M, Y, FUN u potprogramu EULCAU. Primetimo da je ovaj potprogram realizovan tako da je izvršena optimizacija u pogledu broja promenljivih.

Ulazne parametre za integraciju smo uzeli kao u 3.8.1.

```
C=================================================================
C  RESAVANJE DIF.JED.METODOM RUNGE-KUTTA(GILLOVA VARIJANTA)
C=================================================================
      EXTERNAL FUN
      REAL*8 Y(100),F,FUN,X0,X,H,G
      F(X)=6.*DEXP(X-1.)-X*X-2.*X-2.
      WRITE(5,10)
   10 FORMAT(8X,'RESAVANJE DIF.JED. METODOM RUNGE-KUTTA (GILL'
     1,'OVA VARIJANTA)' )
   20 READ(8,25,END=99)X ,Y(1),H,N,M
   25 FORMAT(3F6.1,2I3)
      X0=X
      CALL GILL(X0,H,N,M,Y,FUN)
      WRITE(5,30)H
   30 FORMAT(1H0,28X,'(H=',F6.4,')'//15X,'XN',13X,'YN',10X,
     1'GRESKA(%)'/ )
      NN=N+1
      DO 11 I=1,NN
      G=DABS((Y(I)-F(X))/F(X))*100.
      WRITE(5,15)X,Y(I),G
   15 FORMAT(15X,F3.1,8X,F9.6,6X,D10.3)
   11 X=X+H*M
      GO TO 20
   99 CALL EXIT
      END

      SUBROUTINE GILL(X0,H,N,M,Y,FUN)
      REAL*8 Y(1),H,FUN,X0,Y0,Q,K,A,B
      B=DSQRT(0.5D0)
      Q=0.D0
      Y0=Y(1)
      NN=N+1
```

```
      DO 10 I=2,NN
      DO 20 J=1,M
      K=H*FUN(X0,Y0)
      A=0.5*(K-2.*Q)
      Y0=Y0+A
      Q=Q+3.*A-0.5*K
      K=H*FUN(X0+H/2.,Y0)
      A=(1.-B)*(K-Q)
      Y0=Y0+A
      Q=Q+3.*A-(1.-B)*K
      K=H*FUN(X0+H/2.,Y0)
      A=(1.+B)*(K-Q)
      Y0=Y0+A
      Q=Q+3.*A-(1.+B)*K
      K=H*FUN(X0+H,Y0)
      A=(K-2.*Q)/6.
      Y0=Y0+A
      Q=Q+3.*A-K/2.
   20 X0=X0+H
   10 Y(I)=Y0
      RETURN
      END

      FUNCTION FUN(X,Y)
      REAL*8 FUN,X,Y
      FUN=X*X+Y
      RETURN
      END
```

RESAVANJE DIF.JED. METODOM RUNGE-KUTTA (GILLOVA VARIJANTA)

(H=0.1000)

XN	YN	GRESKA(%)
1.0	1.000000	0.000D+00
1.1	1.221025	0.246D-04
1.2	1.488416	0.460D-04
1.3	1.809152	0.647D-04
1.4	2.190946	0.808D-04
1.5	2.642325	0.949D-04
1.6	3.172709	0.107D-03
1.7	3.792512	0.118D-03
1.8	4.513240	0.128D-03
1.9	5.347611	0.136D-03
2.0	6.309682	0.144D-03

(H=0.0500)

XN	YN	GRESKA(%)
1.0	1.000000	0.000D+00
1.1	1.221025	0.162D-05
1.2	1.488417	0.303D-05
1.3	1.809153	0.425D-05
1.4	2.190948	0.531D-05
1.5	2.642327	0.623D-05
1.6	3.172713	0.704D-05
1.7	3.792516	0.775D-05
1.8	4.513245	0.838D-05
1.9	5.347618	0.894D-05
2.0	6.309690	0.946D-05

5.3.9. Rešavanje sistema jednačina i jednačina višeg reda

Metodi koji su bili razmatrani u prethodnim odeljcima mogu se uopštiti u tom smislu da budu primenljivi za rešavanje Cauchyevog problema za sistem od p jednačina prvog reda

$$y_i' = f_i\,(x; y_1, \dots y_p),\ y_i\,(x_o) = y_{io}\ (i = 1, \dots, p). \tag{3.9.1}$$

U ovom slučaju, sistem jednačina (3.9.1) treba predstaviti u vektorskom obliku

$$\vec{y}\,' = \vec{f}(x, \vec{y}),\ \vec{y}\,(x_o) = \vec{y}_o \tag{3.9.2}$$

gde su

$$\vec{y} = \begin{bmatrix} y_1 \\ \cdot \\ \cdot \\ \cdot \\ y_p \end{bmatrix},\quad \vec{y}_o = \begin{bmatrix} y_{1o} \\ \cdot \\ \cdot \\ y_{po} \end{bmatrix},\ \vec{f}(x, \vec{y}) = \begin{bmatrix} f_1\,(x; y_1, \dots, y_p) \\ \cdot \\ \cdot \\ \cdot \\ f_p\,(x; y_1, \dots, y_p) \end{bmatrix},$$

Od interesa je i rešavanje Cauchyevog problema za diferencijalne jednačine višeg reda. Primetimo, međutim, da se ovaj problem može svesti na prethodni. Naime, neka je data diferencijalna jednačina reda p

$$y^{(p)} = f\,(x, y, y', \dots, y^{(p-1)}) \tag{3.9.3}$$

sa početnim uslovima

$$y^{(i)}\,(x_o) = y_{io} \qquad (i = 0, 1, \dots, p-1). \tag{3.9.4}$$

Tada se, supstitucijama

$$z_1 = y,\ \ z_2 = y', \dots, z_p = y^{(p-1)},$$

jednačina (3.9.3) sa uslovima (3.9.4), svodi na sistem

$$\begin{aligned} z_1' &= z_2, \\ z_2' &= z_3, \\ &\ \ \vdots \\ z_{p-1}' &= z_m, \\ z_p' &= f\,(x; z_1, z_2, \dots, z_p), \end{aligned}$$

sa uslovima $z_i\,(x_o) = z_{io} = y_{io}$ $(i = 1, \dots, p)$.

Linearni višekoračni metodi, koje smo do sada razmatrali, mogu se formalno generalisati na vektorski oblik

$$\sum_{i=o}^{k} \alpha_i \vec{y}_{n+i} = h \sum_{i=o}^{k} \beta_i \vec{f}_{n+i}$$

gde je $\vec{f}_{n+i} = \vec{f}(x_{n+i}, \vec{y}_{n+i})$, a zatim se kao takvi mogu primeniti na rešavanje Cauchyevog problema (3.9.2).

Takođe, metodi Runge–Kutta za rešavanje Cauchyevog problema (3.9.2) imaju oblik

$$\vec{y}_{n+1} - \vec{y}_n = h\vec{\phi}(x_n, \vec{y}_n, h),$$

gde su

$$\vec{\Phi}(x, \vec{y}, h) = \sum_{i=1}^{m} c_i \vec{k}_i ,$$

$$\vec{k}_1 = \vec{f}(x, \vec{y}) ,$$

$$\vec{k}_i = \vec{f}(x + a_i h, \vec{y} + \vec{b}_i h)$$

$$a_i = \sum_{j=1}^{i-1} \alpha_{ij} , \quad \vec{b}_i = \sum_{j=1}^{i-1} \alpha_{ij} \vec{k}_j \qquad (i = 2, \ldots, m) .$$

Sva analiza, koja je data u prethodnim poglavljima formalno se može preneti na navedene vektorske metode.

Kao primer realizujmo standardni metod Runge-Kutta četvrtog reda (3.7.8) za rešavanje sistema od dve diferencijalne jednačine

$$y' = f_1(x, y, z), \quad z' = f_2(x; y, z) ,$$

pri uslovima $y(x_o) = y_o$ i $z(x_o) = z_o$.

Odgovarajući potprogram ima oblik:

```
SUBROUTINE RKS(XP,XKRAJ,YP,ZP,H,N,YY,ZZ)
REAL KY1,KY2,KY3,KY4,KZ1,KZ2,KZ3,KZ4
DIMENSION YY(1),ZZ(1)
K=(XKRAJ-XP)/(H*FLOAT(N))
N1=N+1
X=XP
Y=YP
Z=ZP
T=H/2.
YY(1)=Y
ZZ(1)=Z
```

```
      DO 6 I=2,N1
      DO 7 J=1,K
      KY1=FUN(1,X,Y,Z)
      KZ1=FUN(2,X,Y,Z)
      KY2=FUN(1,X+T,Y+T*KY1,Z+T*KZ1)
      KZ2=FUN(2,X+T,Y+T*KY1,Z+T*KZ1)
      KY3=FUN(1,X+T,Y+T*KY2,Z+T*KZ2)
      KZ3=FUN(2,X+T,Y+T*KY2,Z+T*KZ2)
      KY4=FUN(1,X+H,Y+H*KY3,Z+H*KZ3)
      KZ4=FUN(2,X+H,Y+H*KY3,Z+H*KZ3)
      Y=Y+H*(KY1+2.*(KY2+KY3)+KY4)/6.
      Z=Z+H*(KZ1+2.*(KZ2+KZ3)+KZ4)/6.
    7 X=X+H
      YY(I)=Y
    6 ZZ(I)=Z
      RETURN
      END
```

Koristeći ovaj potprogram rešili smo sistem jednačina

$$y' = xyz, \quad z' = xy/z \, ,$$

pri uslovima $y(1) = 1/3$ i $z(1) = 1$, na segmentu $[1, 2.5]$ uzimajući korak integracije $h = 0.01$, dok na izlazu štampamo x sa korakom 0.1 i odgovarajuće vrednosti za y, y_T, z, z_T, gde su y_T i z_T tačna rešenja ovog sistema i data su sa

$$y_T = \frac{72}{(7-x^2)^3} \qquad \text{i} \qquad z_T = \frac{6}{7-x^2}.$$

```
C=================================================================
C  RESAVANJE SISTEMA DIF.JED.METODOM RUNGE-KUTTA
C=================================================================
      DIMENSION YT(16),ZT(16),YY(16),ZZ(16),X(16)
      YEG(X)=72./(7.-X*X)**3
      ZEG(X)=6./(7.-X*X)
      READ(8,15)N,XP,YP,ZP,XKRAJ
   15 FORMAT(I2,4F3.1)
      YP=YP/3.
      H=0.1
      N1=N+1
      DO 5 I=1,N1
      X(I)=XP+H*FLOAT(I-1)
      YT(I)=YEG(X(I))
    5 ZT(I)=ZEG(X(I))
      WRITE(5,22)
      H=0.01
      CALL RKS(XP,XKRAJ,YP,ZP,H,N,YY,ZZ)
      WRITE(5,18) H,(X(I),YY(I),YT(I),ZZ(I),ZT(I),I=1,N1)
   18 FORMAT(//7X,'KORAK INTEGRACIJE H=',F6.3///7X,'X',
     111X,'Y',10X,'YTACNO',11X,'Z',10X,'ZTACNO'//(F10.2,4F14.7))
   22 FORMAT(1H1, 9X,'RESAVANJE SISTEMA SIMULTANIH',
     1' DIFERENCIJALNIH JEDNACINA'//33X,'Y''=XYZ'//33X,
     2'Z''=XY/Z')
      CALL EXIT
      END
```

```
      FUNCTION FUN(J,X,Y,Z)
      GO TO(50,60),J
   50 FUN=X*Y*Z
      RETURN
   60 FUN=X*Y/Z
      RETURN
      END
```

Odgovarajući program i izlazna lista imaju sledeći oblik:

```
RESAVANJE SISTEMA SIMULTANIH DIFERENCIJALNIH JEDNACINA

                        Y'=XYZ

                        Z'=XY/Z

KORAK INTEGRACIJE H= 0.010
```

X	Y	YTACNO	Z	ZTACNO
1.00	0.3333333	0.3333333	1.0000000	1.0000000
1.10	0.3709342	0.3709341	1.0362694	1.0362694
1.20	0.4188979	0.4188979	1.0791367	1.0791367
1.30	0.4808936	0.4808936	1.1299436	1.1299435
1.40	0.5623944	0.5623943	1.1904763	1.1904762
1.50	0.6718182	0.6718181	1.2631581	1.2631578
1.60	0.8225904	0.8225905	1.3513514	1.3513515
1.70	1.0370675	1.0370678	1.4598541	1.4598541
1.80	1.3544686	1.3544689	1.5957446	1.5957447
1.90	1.8481333	1.8481344	1.7699113	1.7699116
2.00	2.6666656	2.6666667	1.9999998	2.0000000
2.10	4.1441259	4.1441321	2.3166018	2.3166029
2.20	7.1444836	7.1444917	2.7777767	2.7777779
2.30	14.3993673	14.3994160	3.5087693	3.5087738
2.40	37.7629280	37.7631035	4.8387012	4.8387108
2.50	170.6634674	170.6666718	7.9999280	8.0000000

5.3.10. Konturni problemi

U ovom odeljku ukazaćemo na diferencni metod za rešavanje konturnog problema

$$y'' + p(x)y' + q(x)y = f(x); \quad y(a) = A, \quad y(b) = B, \tag{3.10.1}$$

gde su funkcije p, q, f neprekidne na $[a,b]$.

Segment $[a,b]$ podelimo na $N+1$ podsegmenata dužine $h = \dfrac{b-a}{N+1}$, tako da je $x_n = a + nh$ $(n = 0,1,\ldots,N+1)$. U tačkama x_n $(n = 1,\ldots,N)$ diferencijalnu jednačinu iz (3.10.1) aproksimirajmo sa

$$\frac{y_{n+1} - 2y_n + y_{n-1}}{h^2} + p_n \frac{y_{n+1} - y_{n-1}}{2h} + q_n y_n = f_n \quad (n=1,\ldots,N). \tag{3.10.2}$$

gde su $p_n \equiv p(x_n)$, $q_n \equiv q(x_n)$, $f_n \equiv f(x_n)$.

Ako uvedemo smene

$$a_n = 1 - \frac{h}{2} p_n, \quad b_n = h^2 q_n - 2, \quad c_n = 1 + \frac{h}{2} p_n ,$$

(3.10.2) se može predstaviti u obliku

$$a_n y_{n-1} + b_n y_n + c_n y_{n+1} = h^2 f_n \qquad (n=1, \ldots, N). \tag{3.10.3}$$

S obzirom da se konturni uslovi $y_o = A$ i $Y_{N+1} = B$, pred nama se postavlja problem rešavanja sistema linearnih jednačina $T\vec{y} = \vec{d}$, gde su

$$\vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_2 \end{bmatrix}, \quad \vec{d} = \begin{bmatrix} h^2 f_1 - Aa_1 \\ h^2 f_2 \\ \cdot \\ \vdots \\ h^2 f_N - Bc_N \end{bmatrix}, \quad T = \begin{bmatrix} b_1 & c_1 & 0 \ldots & 0 \\ a_2 & b_2 & c_2 & 0 \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ 0 & 0 & 0 & b_N \end{bmatrix}.$$

Matrica sistema je trodijagonalna. Za rešavanje ovog sistema pogodno je izvršiti dekompoziciju matrice T u obliku $T = LR$ (videti odeljak 4.1.1), čime se problem svodi na sukcesivno rešavanje dva trougaona sistema linearnih jednačina. Ovakav postupak za rešavanje konturnog problema (3.10.1), u literaturi je poznat kao matrična faktorizacija.

Sledeći program je realizovan na osnovu izloženog postupka.

```
      DIMENSION A(100),B(100),C(100),D(100)
C     ===================================
C     MATRICNA FAKTORIZACIJA ZA RESAVANJE
C     KONTURNIH PROBLEMA KOD LINEARNIH
C     DIFERENCIJALNIH JEDNACINA II REDA
C     Y'' + P(X)Y' + Q(X)Y = F(X)
C     Y(DG) = YA, Y(GG) = YB
C     ===================================
      READ(8,5) DG,YA,GG,YB
    5 FORMAT(4F10.5)
C     UCITAVANJE BROJA MEDJUTACAKA PREKO
C     TERMINALA LA 30
   10 READ(6,15) N
   15 FORMAT(I2)
      N1=N+1
      IF(N.EQ.0) GO TO 60
      H=(GG-DG)/FLOAT(N1)
      HH=H*H
      X=DG
      DO 20 I=1,N
      X=X+H
```

```
      Y=H/2.*PQF(X,1)
      A(I)=1.-Y
      C(I)=1.+Y
      B(I)=HH*PQF(X,2)-2.
   20 D(I)=HH*PQF(X,3)
      D(1)=D(1)-YA*A(1)
      D(N)=D(N)-YB*C(N)
      C(1)=C(1)/B(1)
      DO 25 I=2,N
      B(I)=B(I)-A(I)*C(I-1)
   25 C(I)=C(I)/B(I)
      D(1)=D(1)/B(1)
      DO 30 I=2,N
   30 D(I)=(D(I)-A(I)*D(I-1))/B(I)
      NM=N-1
      DO 35 I=1,NM
      J=NM-I+1
   35 D(J)=D(J)-C(J)*D(J+1)
      WRITE(5,40) N,(I,I=1,N1)
   40 FORMAT(///5X,'BROJ MEDJUTACAKA N =',I3///5X,'I',6X,'0',9I10)
      DO 45 I=1,N
      C(I)=DG+H*FLOAT(I)
   45 B(I)=PQF(C(I),4)
      WRITE(5,50) DG,(C(I),I=1,N),GG
      WRITE(5,55) YA,(D(I),I=1,N),YB
      WRITE(5,65) YA,(B(I),I=1,N),YB
   50 FORMAT(/5X,'X(I)',10(F6.2,4X))
   55 FORMAT(/5X,'Y(I)',10F10.6)
   65 FORMAT(/5X,'YEGZ',10F10.6)
      GO TO 10
   60 CALL EXIT
      END
```

Primetimo da je program tako realizovan da se broj međutačaka N učitava preko terminala *LA* 30. U slučaju kada je $N = 0$ program se završava. Takođe, u programu je predviđeno i tabeliranje tačnog rešenja u posmatranim tačkama, radi kontrole. Jasno je, međutim, da ovo poslednje ima smisla samo u školskim primerima gde nam je rešenje poznato. Tako je, na primer, za konturni problem

$$y'' - 2xy' - 2y = -4x \; ; \; y(0) = 1, \; y(1) = 1 + e \cong 3.7182818,$$

tačno rešenje $\quad y = x + \exp(x^2)$.

Za ovaj konturni problem funkcijski potprogram za definisanje funkcija p, q, f, kao i tačnog rešenja, nazvali smo PQF. Za slučaj $N = 4$, dobili smo sledeće rezultate

```
      FUNCTION PQF(X,M)
      GO TO(10,20,30,40),M
   10 PQF=-2.*X
      RETURN
   20 PQF=-2.
      RETURN
   30 PQF=-4.*X
      RETURN
   40 PQF=X+EXP(X*X)
      RETURN
      END
```

BROJ MEDJUTACAKA N = 4

I	0	1	2	3	4	5
X(I)	0.00	0.20	0.40	0.60	0.80	1.00
Y(I)	1.000000	1.243670	1.577952	2.038018	2.699739	3.718282
YEGZ	1.000000	1.240811	1.573511	2.033330	2.696481	3.718282

5.4. PARCIJALNE DIFERENCIJALNE JEDNAČINE

5.4.1. Metod mreža

U ovom poglavlju ukazaćemo samo na jedan način za numeričko rešavanje parcijalnih jednačina. Naime, razmotrićemo samo kako se metodom mreža rešava Laplaceova jednačina (eliptičkog tipa) i talasna jednačina (hiperboličkog tipa). Slično se rešavaju i ostale jednačine; na primer jednačina provođenja toplote (paraboličkog tipa).

Metod mreža ili diferencni metod, kako se često naziva, predstavlja osnovni metod za rešavanje jednačina matematičke fizike (parcijalne jednačine koje se javljaju u fizici i tehnici).

Neka je data linearna parcijalna diferencijalna jednačina

(4.1.1) $Lu = f$

i neka se u oblasti D, koja je ograničena krivom Γ ($D = int\ \Gamma$), traži ono njeno rešenje koje na krivoj Γ zadovoljava dati konturni uslov

(4.1.2) $Ku = \psi \qquad ((x,y) \in \Gamma)$.

U primeni metoda mreža, najpre, treba izabrati diskretni skup tačaka D_h, koji pripada oblasti $\bar{D}$ ($= D \cup \Gamma$) i koji se naziva mrežom. Najčešće se u primenama za mrežu uzima familija paralelnih pravih $x_i = x_0 + ih$, $y_j = y_0 + jl$ ($i,j = 0, \pm 1, \pm 2, \ldots$). Tačke preseka ovih pravih se nazivaju čvorovima mreže, a veličine h i l koracima mreže. Dva čvora mreže se nazivaju susednim ako su udaljena po x i y osi samo za jedan korak. Ako sva četiri susedna čvora nekog čvora pripadaju oblasti $\bar{D}$, onda se taj čvor naziva unutrašnjim; u protivnom čvor mreže D_h se naziva graničnim čvorom. U primenama pored pravougaone mreže koriste se i drugi oblici mreža.

Metod mreža se sastoji u aproksimaciji jednačina (4.1.1) i (4.1.2) pomoću odgovarajućih diferencnih jednačina. Naime, operator L možemo aproksimirati diferencnim operatorom, veoma jednostavno, zamenom izvoda odgovarajućim diferencama, u unutrašnjim čvorovima mreže. Pri ovome se koriste sledeće formule

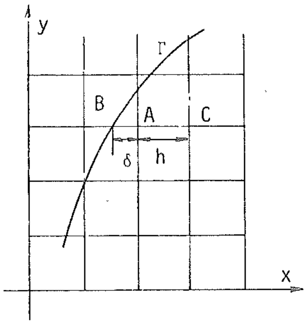
$$\frac{\partial u(x_i, y_j)}{\partial x} \cong \frac{u_{i+1,j} - u_{i,j}}{h}, \quad \frac{\partial u(x_i, y_j)}{\partial y} \cong \frac{u_{i+1,j} - u_{i-1,j}}{2h},$$

$$\frac{\partial^2 u(x_i, y_j)}{\partial x^2} \cong \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} \quad , \text{ itd.}$$

Potpuno simetrične su formule za parcijalne izvode po promenljivoj y.

Aproksimacija konturnih uslova, u nekim slučajevima, može biti vrlo složen problem, što zavisi od oblika operatora K i konture Γ. Kod tzv. konturnih uslova prve vrste, kod kojih je $Ku = u$, jedan praktičan način za aproksimaciju predložio je L. Collatz i on se sastoji u sledećem:

Neka je graničnom čvoru A najbliža tačka sa konture Γ, tačka B i neka je njihovo rastojanje δ (videti sl. 4.1.1).



Sl. 4.1.1

Na osnovu vrednosti funkcije u tačkama B i C, linearnom interpolacijom dobijamo

$$u(A) \cong \frac{h\psi(B) + \delta u(C)}{h + \delta} \, .$$

Aproksimacija konturnog uslova (4.1.2), u ovom slučaju se sastoji u definisanju jednačina gornjeg oblika za svaki granični čvor.

Jednačine dobijene aproksimacijom jednačine (4.1.1) i konturnog uslova (4.1.2) predstavljaju sistem linearnih jednačina, čijim se rešavanjem dobijaju tražena numerička rešenja postavljenog problema.

U daljem izlaganju ukazaćemo na dva osnovna primera.

5.4.2. Laplaceova jednačina

Neka je potrebno naći rešenje Laplaceove jednačine

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \qquad ((x, y) \in D) \, ,$$

koje na konturi kvadrata $D = \{ (x, y) \mid 0 < x < 1, \; 0 < y < 1 \}$ ispunjava određeni uslov $u(x, y) = \psi(x, y)$ ($(x, y) \in \Gamma$). Izaberimo mrežu D_h kod koje je $l = h = \frac{1}{N-1}$, tako da su čvorovi mreže tačke $(x_i, y_j) = ((i-1)h, (j-1)l)$ $(i, j = 1, \ldots, N)$. Standardna diferencna aproksimacija (šema) za rešavanje Laplaceove jednačine ima oblik

$$\frac{1}{h^2}(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}) = 0,$$

ili u obliku

$$u_{i,j} = \frac{1}{4}(u_{i,j+1} + u_{i,j-1} + u_{i-1,j} + u_{i+1,j}).$$

Uzimajući $i,j = 2, \ldots, N-1$ u poslednjoj jednakosti dobijamo sistem od $(N-2)^2$ linearnih jednačina. Za rešavanje ovog sistema obično se koristi metod proste iteracije ili, što je još prostije, Gauss–Seidelov metod (videti odeljak 4.3.3).

Odgovarajući program za rešavanje posmatranog problema ima oblik

```
      ==================================
      RESAVANJE LAPLACE-OVE JEDNACINE
      ==================================
      DIMENSION U(25,25)
      READ(8,4) N
    4 FORMAT(I2)
      M=N-1
      READ(8,1) (U(1,J),J=1,N),(U(N,J),J=1,N),
     1(U(I,1),I=2,M),(U(I,N),I=2,M)
    1 FORMAT(8F10.0)
      DO 10 I=2,M
      DO 10 J=2,M
   10 U(I,J)=0.
      IMAX=0
   20 READ(6,4) MAX
      IF(MAX.EQ.0)GO TO 100
      DO 30 ITER=1,MAX
      DO 30 I=2,M
      DO 30 J=2,M
   30 U(I,J)=(U(I,J+1)+U(I,J-1)+U(I-1,J)+U(I+1,J))/4.
      IMAX=IMAX+MAX
      WRITE(5,65) IMAX,(J,J=1,N)
   65 FORMAT(//26X,'BROJ ITERACIJA JE',I3//17X,
     14(5X,'J =',I2))
      DO 60 I=1,N
   60 WRITE(5,66) I,(U(I,J),J=1,N)
   66 FORMAT(13X,'I =',I2,6F10.4)
      GO TO 20
  100 CALL EXIT
      END
```

Za rešavanje sistema linearnih jednačina koristili smo Gauss–Seidelov metod sa početnim uslovima $u_{i,j} = 0$ $(i, j = 2, \ldots, N-1)$, pri čemu se na broj iteracija može uticati preko terminala *LA* 30. Za $N = 4$ i konturne uslove

$u_{11} = 0,\quad u_{12} = 30,\quad u_{13} = 60,\quad u_{14} = 90,$

$u_{41} = 180,\quad u_{42} = 120,\quad u_{43} = 60,\quad u_{44} = 0,$

$u_{21} = 60,\quad u_{31} = 120,\quad u_{24} = 60,\quad u_{34} = 30\,,$

dobijeni su sledeći rezultati:

BROJ ITERACIJA JE 2

	J = 1	J = 2	J = 3	J = 4
I = 1	0.0000	30.0000	60.0000	90.0000
I = 2	60.0000	47.8125	53.9062	60.0000
I = 3	120.0000	83.9063	56.9531	30.0000
I = 4	180.0000	120.0000	60.0000	0.0000

BROJ ITERACIJA JE 7

	J = 1	J = 2	J = 3	J = 4
I = 1	0.0000	30.0000	60.0000	90.0000
I = 2	60.0000	59.9881	59.9940	60.0000
I = 3	120.0000	89.9940	59.9970	30.0000
I = 4	180.0000	120.0000	60.0000	0.0000

BROJ ITERACIJA JE 9

	J = 1	J = 2	J = 3	J = 4
I = 1	0.0000	30.0000	60.0000	90.0000
I = 2	60.0000	59.9993	59.9996	60.0000
I = 3	120.0000	89.9996	59.9998	30.0000
I = 4	180.0000	120.0000	60.0000	0.0000

BROJ ITERACIJA JE 10

	J = 1	J = 2	J = 3	J = 4
I = 1	0.0000	30.0000	60.0000	90.0000
I = 2	60.0000	59.9998	59.9999	60.0000
I = 3	120.0000	89.9999	60.0000	30.0000
I = 4	180.0000	120.0000	60.0000	0.0000

BROJ ITERACIJA JE 11

	J = 1	J = 2	J = 3	J = 4
I = 1	0.0000	30.0000	60.0000	90.0000
I = 2	60.0000	60.0000	60.0000	60.0000
I = 3	120.0000	90.0000	60.0000	30.0000
I = 4	180.0000	120.0000	60.0000	0.0000

5.4.3. Talasna jednačina

Posmatrajmo talasnu jednačinu

$$\frac{\partial^2 u}{\partial x^2} = \frac{1}{a^2} \cdot \frac{\partial^2 u}{\partial t^2} \tag{4.3.1}$$

sa početnim uslovima

$$(4.3.2) \qquad u(x,0)=f(x), \quad u_t(x,0)=g(x) \qquad (0<x<b)$$

i konturnim uslovima

$$(4.3.3) \qquad u(0,t)=\phi(t), \quad u(b,t)=\psi(t) \qquad (t\geqslant 0).$$

Korišćenjem konačnih razlika, jednačina (4.3.1) se može aproksimirati pomoću

$$(4.3.4) \qquad u_{i+1,j}-2u_{i,j}+u_{i-1,j}=\frac{1}{r^2}\,(u_{i,j+1}-2u_{i,j}+u_{i,j-1}),$$

gde je $r=a\dfrac{l}{h}$ (h i l su koraci po x i t osi respektivno) i $u_{i,j}\cong u(x_i,t_j)$. Na osnovu prve jednakosti u (4.3.2) imamo

$$(4.3.5) \qquad u_{i,o}=f(x_i)=f_i\,.$$

Uvođenjem fiktivnog sloja $j=-1$, drugi početni uslov u (4.3.2) može se jednostavno aproksimirati pomoću

$$(4.3.6) \qquad u_t(x_i,0)=g(x_i)=g_i\cong\frac{u_{i,1}-u_{i,-1}}{2l}\,.$$

Ako u (4.3.4) stavimo $j=0$ dobijamo

$$f_{i+1}-2f_i+f_{i-1}-\frac{1}{r^2}\,(u_{i,1}-2f_i+u_{i,-1})=0,$$

odakle, s obzirom na (4.3.6) sleduje

$$u_{i,1}=lg_i+f_i+\frac{1}{2}r^2\,(f_{i+1}-2f_i+f_{i-1}),$$

tj.

$$(4.3.7) \qquad u_{i,1}=lg_i+(1-r^2)f_i+\frac{1}{2}r^2\,(f_{i+1}+f_{i-1}).$$

S druge strane iz (4.3.4) sleduje

$$(4.3.8) \qquad u_{i,j+1}=\frac{1}{r^2}\,(u_{i+1,j}+u_{i-1,j})-u_{i,j-1}+2\left(\frac{1}{r^2}-1\right)u_{i,j}\,.$$

Ako stavimo $h=b/N$ i $x_i=(i-1)h$ $(i=1,2,\ldots,N+1)$, na osnovu konturnih uslova (4.3.3) imamo

$$(4.3.9) \qquad u_{1,j}=\phi(t_j)=\phi_j, \quad u_{N+1,j}=\psi(t_j)=\psi_j\,,$$

gde je $j = 0,1, \ldots$ Za određivanje rešenja u pravougaoniku $P = \{(x, t) \mid 0 < x < b,\ 0 < t < T_{max}\}$, maksimalna vrednost indeksa j je celobrojni deo od T_{max}/l, tj. $j_{max} = M = [\,T_{max}/l\,]$.

Na osnovu jednakosti (4.3.5), (4.3.7), (4.3.8), (4.3.9) jednostavno se nalaze približna rešenja datog problema u čvorovima mreže pravougaonika P, što je realizovano sledećim programom.

```
C=================================================================
C  RESAVANJE PARCIJALNE DIF.JED.HIPERBOLICNOG TIPA
C=================================================================
      DIMENSION U(3,9)
      READ(8,5)N,A,B,R,TMAX
    5 FORMAT(I2,4F5.2)
      N1=N+1
      WRITE(5,10)(I,I=1,N1)
   10 FORMAT(1H1,10X,1HJ,<N+1>(4X,'U(',I1,',J)')/)
      H=B/FLOAT(N)
      EL=R*H/A
      M=TMAX/EL
      T=0.
      DO 15 K=1,2
      U(K,1)=FF(T,B,3)
      U(K,N1)=FF(T,B,4)
   15 T=T+EL
      X=0.
      R2=R*R
      DO 20 I=2,N
      X=X+H
      U(1,I)=FF(X,B,1)
   20 U(2,I)=EL*FF(X,B,2)+(1.-R2)*U(1,I)
      DO 25 I=2,N
   25 U(2,I)=U(2,I)+R2/2.*(U(1,I+1)+U(1,I-1))
      J=0
   30 WRITE(5,35)J,(U(1,I),I=1,N1)
   35 FORMAT(7X,I5,<N1>F10.4)
      IF(J.EQ.M)GO TO 50
      J=J+1
      U(3,1)=FF(T,B,3)
      U(3,N1)=FF(T,B,4)
      DO 40 I=2,N
   40 U(3,I)=(U(2,I+1)+U(2,I-1))/R2-U(1,I)-2.*(1./R2-1.)*U(2,I)
      T=T+EL
      DO 45 I=1,N1
      U(1,I)=U(2,I)
   45 U(2,I)=U(3,I)
      GO TO 30
   50 CALL EXIT
      END
```

Primetimo da se vrednosti rešenja u tri uzastopna sloja $j-1$, j, $j+1$, pamte u prvoj, drugoj i trećoj vrsti matrice U, respektivno.

Funkcije f, g, ϕ, ψ definišu se funkcijskim potprogramom FF za $I = 1, 2, 3, 4$, respektivno.

U konkretnom slučaju za $a = 2$, $b = 4$, $T_{max} = 6$, $f(x) = x(4-x)$, $g(x) = 0$, $\phi(t) = 0$, $\psi(t) = 0$, $N = 4$ i $r = 1$, potprogram FF i odgovarajući rezultati imaju oblik:

```
      FUNCTION FF(X,B,I)
      GO TO(10,20,30,40),I
10    FF=X*(B-X)
      RETURN
20    FF=0.
      RETURN
30    FF=0.
      RETURN
40    FF=0.
      RETURN
      END
```

J	U(1,J)	U(2,J)	U(3,J)	U(4,J)	U(5,J)
0	0.0000	3.0000	4.0000	3.0000	0.0000
1	0.0000	2.0000	3.0000	2.0000	0.0000
2	0.0000	0.0000	0.0000	0.0000	0.0000
3	0.0000	-2.0000	-3.0000	-2.0000	0.0000
4	0.0000	-3.0000	-4.0000	-3.0000	0.0000
5	0.0000	-2.0000	-3.0000	-2.0000	0.0000
6	0.0000	0.0000	0.0000	0.0000	0.0000
7	0.0000	2.0000	3.0000	2.0000	0.0000
8	0.0000	3.0000	4.0000	3.0000	0.0000
9	0.0000	2.0000	3.0000	2.0000	0.0000
10	0.0000	0.0000	0.0000	0.0000	0.0000
11	0.0000	-2.0000	-3.0000	-2.0000	0.0000
12	0.0000	-3.0000	-4.0000	-3.0000	0.0000

LITERATURA

1. N. Parezanović: Računske mašine i programiranje – Programski jezik Fortran IV. Beograd, 1973.
2. G. V. Milovanović: Numerička analiza – I deo. Niš, 1979.
3. Л. А. ЛЮСТЕРНИК: Некоторые кубатурные формулы для двукратных интегралов. ДАН СССР 62, 6 /1948/, 449-459.
4. C. Runge: Über die numerische Auflösung von Differentialgleichungen. Math. Ann. 46 (1985), 167–178.
5. W. Kutta: Beitrag zur näherungsweisen Integration totaler Differentialgleichungen. Z. Math. Phys. 46 (1901), 435–453.
6. K. Heun: Neue Methode zur approximativen Integration der Differentialgleichungen einer unabhängigen Veränderlichen. ibid. 45 (1900), 23–38.
7. M. Bertolino: Numerička analiza. Beograd, 1977.
8. S. Gill: A Process for the Step–by–Step Integration of Differential Equations in an Automatic Computing Machine. Proc. Cambridge Phil. Soc. 47 (1951), 96–108.
9. G. V. Milovanović: Numerička analiza – Diferencijalne i integralne jednačine Niš, 1981.
10. D. D. McCracken and W. S. Dorn: Numerical Methods and Fortran Programming. New York, 1964.
11. B. Carnahan, H. A. Luther, J. O. Wilkes: Applied Numerical Methods. New York, 1969.