NUMERIČNE METODE

UNIVERZITETNI ŠTUDIJ 2005/06

ZAPISKI Z VAJ

asist. Andrej Kotar

1. vaja

Kreiranje nove mape na **d:\vaje\ime_priimek\vaja1**

Navodilo za delo s programom COMPAQ Visual Fortran V 6.6:

1. **Zagon programa:**
- Start
- Programi
- Compaq Visual Fortran 6
- Developer studio

2. **Generiranje novega projekta:**
- **File**
- **New**
- **Projects**
- **Fortran Console Application**
- Vpišite ime projekta v **Project** name (npr. Volumen)
- Pod Location poiščete svojo mapo d:\vaje\ime_priimek\vaja1
- **OK, Finish, OK**

3. **Generiranje novega programa**
- **File**
- **New**
- **Fortran Free Format Source File**
- Vpišite ime datoteke v **File name** (npr. Volumen)
- **OK**

4. **Z urejevalcem teksta napišemo program**

5. **Prevajanje programa**
- **Build**
- **Compile** ime.f90

Odpravljanje sintaktičnih napak:
Prevajalnik napake izpiše v spodnjem oknu
Dvoklik na izpisano napako nam označi vrstico z napako.

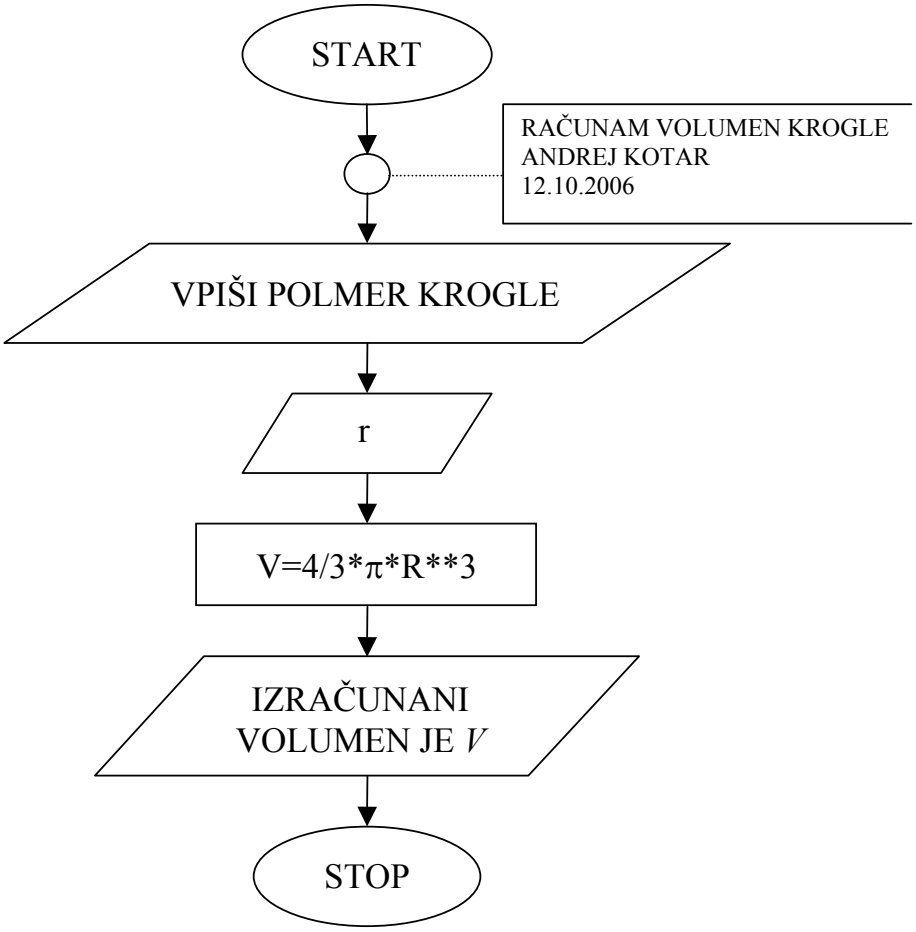
6. **Izgradnja programa in izvajanje**
- **Build**
- **Execute** ime.exe

7. **Odpre se črno DOS okno v katerem se izvaja program.**

Po zaključku dela vse izvorne kode shranite na disketo!

1. naloga:

a) Napišite program za računanje volumna krogle:




```
!RACUNAM VOLUMEN KROGLE
!Andrej Kotar
!12.10.2006
real*4::r,V
real*4,parameter::pi=3.141592
write(*,*)'Podaj radij krogle r:'
read(*,*) r
V=4.*pi*r**3/3.
write(*,*) 'Volumen krogle je ',V
end
```

b) V programu uporabite neskončno DO zanko

2. naloga: Izračunajte razliko in dolžino dveh vektorjev

$$\vec{a} = (1.2,-1.8,3.4)$$
$$\vec{b} = (-3.6,1.9,0.7)$$
$$\vec{c} = \vec{a} - \vec{b}$$
$$dc = |\vec{c}| = \sqrt{c_x^2 + c_y^2 + c_z^2}$$

a) Podatke naj program bere iz ekrana in izpisuje na ekran.
b) Podatki in izpis naj bodo v datoteki.

3. naloga: Izračunajte povprečje števil zapisanih v datoteki.

```
!vaja1.1a
!RACUNAM VOLUMEN KROGLE
real*4::r,V
real*4,parameter::pi=3.141592
write(*,*)'Podaj radij krogle r:'
read(*,*) r
V=4.*pi*r**3/3.
write(*,*) 'Volumen krogle je ',V
end
!test:r=3.5, V=179.5943
```

```
!vaja1.1b
!RACUNAM VOLUMEN KROGLE
real*4::r,V
real*4,parameter::pi=3.141592
do
      write(*,*)'Podaj radij krogle r:'
      read(*,*) r
      V=4.*pi*r**3/3.
      write(*,*) 'Volumen krogle je ',V
enddo
end
!test:r=3.5, V=179.5943
```

```
!vaja1.2a
!RACUNAM RAZLIKO IN DOLZINO VEKTORJEV
real*4,dimension(3)::a,b,c
real*4::dc
write(*,*) 'Podaj komponente vektorjev A in B'
read(*,*) a(1),a(2),a(3)
read(*,*) b(1),b(2),b(3)
c(1)=a(1)-b(1)
c(2)=a(2)-b(2)
c(3)=a(3)-b(3)
dc=sqrt(c(1)*c(1)+c(2)*c(2)+c(3)**2)
write(*,*) 'Razlika vektorjev = (',c(1),',',c(2),',',c(3),')'
write(*,*) 'Dolzina C = ',dc
end
```

```
!vaja1.2b
!RACUNAM RAZLIKO IN DOLZINO VEKTORJEV
real*4,dimension(3)::a,b,c
real*4::dc
open(13,file='podatki.dat')
read(13,*) a(1),a(2),a(3)
read(13,*) b(1),b(2),b(3)
c(1)=a(1)-b(1)
c(2)=a(2)-b(2)
c(3)=a(3)-b(3)
dc=sqrt(c(1)*c(1)+c(2)*c(2)+c(3)**2)
open(26,file='rezultati.dat')
write(26,*) 'Razlika vektorjev = (',c(1),',',c(2),',',c(3),')'
write(26,*) 'Dolzina C = ',dc
end
```

podatki.dat

```
1.2,-1.8 3.4
-3.6
1.9 0.7
```

rezultati.dat

```
Razlika vektorjev = (       4.800000,      -3.700000,       2.700000)
Dolzina C =        6.634757
```

```
      !vaja1.3
      !RACUNAM POVPRECJE STEVIL
      integer::i=0
      real*4::vsota=0,element,povprecje
      open(13,file='podatki.dat')
      do
            read(13,*,end=26) element
            vsota=vsota+element
            i=i+1
      enddo
26    povprecje=vsota/i
      write(*,*)'Povprecje stevil = ',povprecje
      end
```

podatki.dat

```
7
9
2
1
1
```

1. naloga:
Izračunajte vsoto prvih stotih števil.

$$s = \sum_{i=1}^{100} i$$

2. naloga:
Izračunajte naslednjo vsoto za poljuben x.

$$s = \sum_{i=1}^{10} \left(\frac{x}{i} \right)^i$$

3. naloga:
Napišite program za izračun faktorjele.

4. naloga:
Napišite program za izračun korenov kvadratne enačbe.

$$ax^2 + bx + c = 0$$

Vrednosti koeficientov a, b in c preberite iz datoteke.

5. naloga:
Izračunajte funkcijo *sin(x)* s pomočjo neskončne vrste na 6 decimalnih mest natančno:

$$\sin(x) = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$$

Vrednost x naj bo v intervalu $0 < x < 2\pi$.

```
!vaja2.1
!RACUNANJE VSOTE PRVIH STOTIH STEVIL
integer::i,s=0
do i=1,100
      s=s+i
enddo
write(*,*) 'Vsota je',s
end
!test: s=5050
```

```
!vaja2.2
!RACUNANJE VSOTE VRSTE
integer::i
real*4::x,s=0
write(*,*) 'Vstavite vrednost x:'
read(*,*) x
do i=1,10
      s=s+(x/i)**i
enddo
write(*,*) 'Vsota je',s
end
!test: x=2.1 s=3.636620
```

```
!vaja2.3
!RACUNANJE FAKTORJELE
integer::N,i
real*4::fakt
write(*,*) 'Vpisite stevilo N:'
read(*,*) N
if(N<0) then
      write(*,*) 'Napaka!'
elseif(N==0.or.N==1) then
      fakt=1
      write(*,'(I6,''! = '',f10.0)') N,fakt
else
      fakt=1
      do i=2,N
            fakt=fakt*i
      enddo
      write(*,'(I6,''! = '',f10.0)') N,fakt
endif
end
!test: N=5 fakt=120.
```

```
  !vaja2.4
  !KVADRATNA ENACBA
  real*4::a,b,c
  real*4::d,x1,x2
  open(14,file='podatki.txt')
  do
      read(14,*,end=7) a,b,c
      if(a==0) then
            if(b==0) then
                  if(c==0) then
                        write(*,*)'Koeficienti so 0, neskoncno resitev!'
                  else
                        write(*,*) 'Samo c je razlicen od 0, protislovje!'
                  endif
            else
                  x1=-c/b
                  write(*,*) 'Linearna enacba!'
                  write(*,*) 'x=',x1
            endif
      else
            d=b**2-4.*a*c
            if(d>=0) then
                  x1=(-b+sqrt(d))/(2.*a)
                  x2=(-b-sqrt(d))/(2.*a)
                  write(*,*) 'x1=',x1
                  write(*,*) 'x2=',x2
            else
                  x1=-b/(2.*a)
                  x2=sqrt(-d)/(2.*a)
                  write(*,'('' x1='',f12.4,'' + '',f12.4,''i'')') x1,x2
                  write(*,'('' x2='',f12.4,'' - '',f12.4,''i'')') x1,x2
            endif
      endif
      write(*,*)
  enddo
7 continue
  end
```

```
podatki.txt
3.4   5.1   0.3
0     1.6   2
4     1     7
0     0     0
0     0     7
```

```
rezultati:

x1= -6.1331216E-02
x2=  -1.438669

Linearna enacba!
x=  -1.250000

x1=     -0.1250 +       1.3170i
x2=     -0.1250 -       1.3170i

Koeficienti so 0, neskoncno resitev!

Samo c je razlicen od 0, protislovje!
```

```
!vaja2.5
!RACUNANJE NESKONCNE VRSTE ZA SINUS
!sin(x)=x/1!-x3/3!+x5/5!-x7/7!+...
integer::i=1
real*4::s=0,clen,x,pi
real*4,parameter::pi=3.141592
write(*,*) 'Vstavi x:'
read(*,*) x
!interval 0<x<2pi
if(x<0) then
      do
            x=x+2*pi
            if(x>0) exit
      enddo
elseif(x>2*pi) then
      do
            x=x-2*pi
            if(x<2*pi) exit
      enddo
endif
!vrsta
clen=x
do
      s=s+clen
      clen=-clen*x**2/((i*2)*(i*2+1))
      if(abs(clen)<5e-7) exit
      i=i+1
enddo
write(*,*) 'Sin(x)=',s
end
!test: sin(1.5)=0.9974950
```

DODATNA NALOGA:
Izračunajte funkcijo *cos(x)* s pomočjo neskončne vrste na 6 decimalnih mest natančno:

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \ldots$$

Vrednost x naj bo v intervalu $0 < x < 2\pi$.

1. naloga:

Preberite vektorja ***a*** = (1.3,2.7,1.8) in ***b*** = (1.8,1.1,-2.3) iz datoteke in nato izračunajte:

a)

$c = \vec{a} \cdot \vec{b}$

$\vec{d} = \vec{a} \times \vec{b}$

$e = (\vec{a}, \vec{b}, \vec{f})$ kjer je $\vec{f} = (1.3, 2.6, 1.8)$

b) Tvorite matriko G po pravilu: $g_{ij} = a_i \, b_j$ in nato izračunajte:

$\vec{x} = G \cdot \vec{a}$

$\vec{y} = G \cdot \vec{b}$

$\vec{z} = G \cdot \vec{f}$

2. naloga:

Preberite matriko A in izračunajte:
a) Vsoto diagonalnih elementov
b) Vsoto izvendiagonalnih elementov
c) Transponirano $B = A^T$ ter izpišite B
d) Prečitajte iz druge datoteke matriko C

$$A = \begin{bmatrix} 1 & 2 & 1 & -1 \\ 2 & 4 & 0 & 2 \\ 1 & 1 & 3 & 1 \\ 2 & -2 & 1 & 2 \end{bmatrix}$$

$$C = \begin{bmatrix} 2 & 1 \\ -3 & -6 \\ -4 & 1 \\ -1 & 2 \end{bmatrix}$$

in izračunajte produkt $BC = D$.
e) Negativne elemente v D zamenjajte z nič in nato izpišite spremenjeno matriko D.

```
!vaja3.1
!VEKTORJI
integer,parameter::n=3
real*4,dimension(n,n)::G
real*4,dimension(n)::a,b,d,f,x,y,z
real*4::c,e
integer::i,j
!branje vektorjev a, b in f iz datoteke
open(12,file='vektorji.txt')
read(12,*) (a(i),i=1,n)
read(12,*) (b(i),i=1,n)
read(12,*) (f(i),i=1,n)
!a)
c=a(1)*b(1)+a(2)*b(2)+a(3)*b(3)
d(1)=a(2)*b(3)-a(3)*b(2)
d(2)=a(3)*b(1)-a(1)*b(3)
d(3)=a(1)*b(2)-a(2)*b(1)
e=d(1)*f(1)+d(2)*f(2)+d(3)*f(3)
!b)
do i=1,n
      do j=1,n
            G(i,j)=a(i)*b(j)
      enddo
enddo
do i=1,n
      x(i)=G(i,1)*a(1)+G(i,2)*a(2)+G(i,3)*a(3)
      y(i)=G(i,1)*b(1)+G(i,2)*b(2)+G(i,3)*b(3)
      z(i)=G(i,1)*f(1)+G(i,2)*f(2)+G(i,3)*f(3)
enddo
!izpis
write(*,*) 'Skalarni produkt c = a.b = ',c
write(*,*) 'Vektorski produkt d = axb = (',d(1),',',d(2),',',d(3),')'
write(*,*) 'Mesani produkt e = (a,b,c) = ',e
write(*,*) 'Matrika Gij:'
do i=1,n
      write(*,*) (G(i,j),j=1,n)
enddo
write(*,*) 'Vektor x = G.a = (',x(1),',',x(2),',',x(3),')'
write(*,*) 'Vektor y = G.b = (',y(1),',',y(2),',',y(3),')'
write(*,*) 'Vektor z = G.f = (',z(1),',',z(2),',',z(3),')'
end
```

```
vektorji.txt
1.3  2.7  1.8
1.8  1.1 -2.3
1.3  2.6  1.8
```

```
rezultati:
Skalarni produkt c = a.b =        1.170000
Vektorski produkt d = axb = (      -8.190000,       6.230000,    -3.430000)
Mesani produkt e = (a,b,c) =   -6.230008E-01
Matrika Gij:
       2.340000        1.430000       -2.990000
       4.860000        2.970000       -6.210000
       3.240000        1.980000       -4.140000
Vektor x = G.a = (       1.521000,       3.159000,       2.106000)
Vektor y = G.b = (      12.662000,      26.298000,      17.532000)
Vektor z = G.f = (       1.378000,       2.862000,       1.908000)
```

```
!vaja3.2
!MATRIKA
integer,parameter::ndat=5
real*4,dimension(ndat,ndat)::A,B,C,D
real*4::diag=0,izven=0
integer::i,j,na,nc,mc
!branje matrike A
open(12,file='a.txt')
read(12,*) na
do i=1,na
      read(12,*) (a(i,j),j=1,na)
enddo
do i=1,na
      do j=1,na
            if(i==j) then
                  !a)
                  diag=diag+A(i,j)
            else
                  !b)
                  izven=izven+A(i,j)
            endif
      enddo
enddo
write(*,*) 'Vsota diagonalnih elementov:',diag
write(*,*) 'Vsota izvendiagonalnih elementov:',izven
!c)
do i=1,na
      do j=1,na
            B(j,i)=A(i,j)
      enddo
enddo
write(*,*) 'Transponirana matrika B = AT:'
do i=1,na
       write(*,*) (B(i,j),j=1,na)
enddo
!d)
open(14,file='c.txt')
read(14,*) nc,mc
do i=1,nc
      read(14,*) (C(i,j),j=1,mc)
enddo
call prodab(B,na,na,ndat,C,nc,mc,ndat,D,ndat)
write(*,*) 'Produkt matrik D=B*C:'
do i=1,na
      write(*,*) (D(i,j),j=1,mc)
enddo
!e)
do i=1,na
      do j=1,mc
            if(D(i,j)<1e-20) D(i,j)=0.
      enddo
enddo
write(*,*) 'Spremenjena matrika D:'
do i=1,na
      write(*,*) (D(i,j),j=1,mc)
enddo
end
```

```
!Podprogram mnozi matriki
subroutine prodab(A,n,p,av,B,r,m,bv,C,cv)
integer::p,av,r,bv,cv
real*4::A(av,*),B(bv,22),c(cv,1)
if(r/=p) then
      stop 'Dimenzije niso primerne'
endif
do i=1,n
      do j=1,m
            C(i,j)=0.
            do k=1,r
                  C(i,j)=C(i,j)+A(i,k)*B(k,j)
            enddo
      enddo
enddo
return
end
```

```
a.txt
4
1  2  1 -1
2  4  0  2
1  1  3  1
2 -2  1  2
```

```
c.txt
 4  2
 2  1
-3 -6
-4  1
-1  2
```

```
Rezultati:
Vsota diagonalnih elementov:      10.000000
Vsota izvendiagonalnih elementov:      10.000000
Transponirana matrika B = AT:
       1.000000        2.000000        1.000000        2.000000
       2.000000        4.000000        1.000000       -2.000000
       1.000000    0.000000E+00        3.000000        1.000000
      -1.000000        2.000000        1.000000        2.000000
Produkt matrik D=B*C:
     -10.000000       -6.000000
     -10.000000      -25.000000
     -11.000000        6.000000
     -14.000000       -8.000000
Spremenjena matrika D:
   0.000000E+00    0.000000E+00
   0.000000E+00    0.000000E+00
   0.000000E+00        6.000000
   0.000000E+00    0.000000E+00
```

DODATNA NALOGA:

Napišite program za izračun determinante matrike, ki je sestavljena iz treh stolpcev in treh vrstic. Matriko preberite iz datoteke.

1. naloga:

Iz datoteke preberite števila, nato pa jih v obratnem vrstnem redu izpišite na zaslon.

števila v datoteki:
2.1
6.8
3.5
9.1
4.2
7.1
1.8

2. naloga:

Uredite števila iz prve naloge, ki so zapisana v datoteki, po naraščajočem vrstnem redu nato pa jih izpišite na zaslon. Izpišite tudi največje in najmanjše število.

3. naloga:

Matriko *A* preberite in naredite:
a) V matriki poiščite največji element.
b) Matriko normirajte.
c) Seštejte diagonalne elemente.
d) Izračunajte skalarni produkt druge vrstice in tretjega stolpca.

$$A = \begin{bmatrix} 1 & 2 & 1 & -1 \\ 2 & 4 & 0 & 2 \\ 1 & 1 & 3 & 1 \\ 2 & -2 & 1 & 2 \end{bmatrix}$$

Vse rezultate izpišite na zaslon.

```
      !vaja4.1
      !OBRATNI VRSTNI RED
      real*4,dimension(100)::a
      integer::n,i=1
      open(10,file='podatki.txt')
      do
            read(10,*,end=20) a(i)
            i=i+1
      enddo
20    n=i-1
      !izpis v obratnem redu
      open(30,file='rezultati.txt')
      do i=n,1,-1
            write(30,*) a(i)
            write(*,*) a(i)
      enddo
      end
```

```
podatki.txt
2.1
6.8
3.5
9.1
4.2
7.1
1.8
```

```
Rezultati:
   1.800000
   7.100000
   4.200000
   9.100000
   3.500000
   6.800000
   2.100000
```

```
      !vaja4.2
      !UREJANJE STEVIL
      real*4,dimension(1:100)::a
      real::min
      integer::i=1,j,n,m
      open(14,file='podatki.txt')
      do
            read(14,*,end=10) a(i)
            i=i+1
      enddo
10    n=i-1
      !sortiranje stevil
      do i=1,n-1
            min=a(i)    !najmanjse stevilo
            m=i         !indeks najmanjsega stevila
            do j=i+1,n
                  if(a(j)<min) then
                        min=a(j)
                        m=j
                  endif
            enddo
            !zamenjava stevila
            a(m)=a(i)
            a(i)=min
      enddo
      write(*,*) 'Najvecje stevilo je ',a(n)
      write(*,*) 'Najmanjse stevilo je ',a(1)
      open(13,file='rezultati.txt')
      write(*,*) 'Urejena stevila:'
      do i=1,n
            write(13,'(f10.5)') a(i)
            write(*,'(f10.5)') a(i)
      enddo
      close(13)
      end
```

```
podatki.txt
2.1
6.8
3.5
9.1
4.2
7.1
1.8
```

```
 Najvecje stevilo je    9.100000
 Najmanjse stevilo je    1.800000
 Urejena stevila:
   1.80000
   2.10000
   3.50000
   4.20000
   6.80000
   7.10000
   9.10000
```

```
!vaja4.3
integer,parameter::ndat=10
real*4,dimension(ndat,ndat)::A
open(12,file='matrika.txt')
write(*,*) 'Prebrana matrika:'
read(12,*) n,m
do i=1,n
      read(12,*) (A(i,j),j=1,m)
      write(*,'(6f10.4)') (A(i,j),j=1,m)
enddo
Anaj=abs(a(1,1))
do i=1,n
      do j=1,m
            if(abs(A(i,j))>Anaj) then
                  Anaj=abs(A(i,j))
            endif
      enddo
enddo
write(*,*) 'Absolutno najvecji clen je ', Anaj
!normiramo matriko
do i=1,n
      do j=1,m
            A(i,j)=A(i,j)/Anaj
      enddo
enddo
!vsota diagonalnih elementov
if(n/=m) stop 'm je razlicno od n'
s=0
do i=1,n
      s=s+A(i,i)
enddo
!skalarni produkt
sp23=0
do k=1,n
      sp23=sp23+A(2,k)*A(k,3)
enddo
!izpis rezultatov
write(*,*) 'Normirana matrika:'
do i=1,n
      write(*,'(6f10.4)') (A(i,j),j=1,m)
enddo
write(*,*) 'Vsota diagonalnih elementov je ',s
write(*,*) 'Skalarni produkt druge vrstice in tretjega stolpca je ',sp23
end

 Prebrana matrika:
    1.0000    2.0000    1.0000   -1.0000
    2.0000    4.0000    0.0000    2.0000
    1.0000    1.0000    3.0000    1.0000
    2.0000   -2.0000    1.0000    2.0000
 Absolutno najvecji clen je    4.000000
 Normirana matrika:
    0.2500    0.5000    0.2500   -0.2500
    0.5000    1.0000    0.0000    0.5000
    0.2500    0.2500    0.7500    0.2500
    0.5000   -0.5000    0.2500    0.5000
 Vsota diagonalnih elementov je    2.500000
 Skalarni produkt druge vrstice in tretjega stolpca je   0.2500000
```

DODATNA NALOGA:
Elemente matrike A v tretji nalogi uredite po velikosti, nato pa matriko izpišite v datoteko.

5. vaja

1. naloga:

Napišite program za reševanje linearne enačbe v kompleksnem.

$$(2+0i)x + (1+0i) = 0$$

$$(7.3+8.1i)x + (9.8+7.7i) = 0$$

Vrednosti koeficientov *a* in *b* preberite iz datoteke.

2. naloga:

S pomočjo neskončne vrste izračunajte e^x na *5* cifer natančno?

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \ldots$$

3. naloga:

V pravokotniku poznamo dolžino diagonale *d* in ploščino *p*. Dolžina diagonale je *d=2.0* enoti, ploščina *p* pa je velikosti *1*, *0.1*, *0.01*, …,10^{-8} ploščinskih enot. Za dane vrednosti diagonale *d* in ploščin *p* izračunajte stranici *a* in *b* po formulah

$$a = \frac{1}{2}\left(\sqrt{d^2 + 2p} + \sqrt{d^2 - 2p}\right),$$

$$b = \frac{1}{2}\left(\sqrt{d^2 + 2p} - \sqrt{d^2 - 2p}\right).$$

Nato izračunajte še manjšo stranico *b* po formuli *b=p/a*.

4. naloga:

Z uporabo rekurzijske formule izračunajte integrale

$$I_n = \int_0^1 x^n e^{x-1} dx,$$

kjer je *n=0,1,2,...* Integriranje po delih nam pri $n > 0$ da

$$I_n = x^n e^{x-1}\Big|_0^1 - n\int_0^1 x^{n-1} e^{x-1} dx = 1 - nI_{n-1}$$

$$I_0 = \int_0^1 e^{x-1} dx = e^{x-1}\Big|_0^1 = 1 - \frac{1}{e}$$

```
      !vaja5.1
      !linearna enacba v kompleksnem
      complex::a,b,x
      open(14,file='koeficienti.txt')
      do
            read(14,*,end=3) a,b
            if(abs(a)==0) stop 'Ni kvadratna enacba...'
            x=-b/a
            write(*,*) 'x=',x
      enddo
3     continue
      end
```

```
koeficienti.txt
(2.,0.),(1.,0.)
(7.3,8.1),(9.8,7.7)
```

```
rezultati:
x = (-0.5000000,0.0000000E+00)
x = (-1.126240,0.1948697)
```

```
!vaja5.2
!racunanje EXP(x)
real*4::x,s=1,clen=1
integer::i=1
write(*,*) 'Podaj x'
read(*,*) x
do
      clen=clen*x/i
      s=s+clen
      if(abs(clen)<5e-6) exit
      i=i+1
enddo
write(*,*) 'Exp(',x,')=',s
write(*,*) 'Tocna vrednost=',exp(x)
end
!test: exp(3)=20.085540
```

```
!vaja5.3
!Racunanje stranic pravokotnika
real*4::d,p,a,b1,b2
integer::i
d=2.
p=1.
write(*,*) '      d           p            a            b1              b2'
do i=1,9
      a=0.5*(sqrt(d**2+2.*p)+sqrt(d**2-2.*p))
      b1=0.5*(sqrt(d**2+2.*p)-sqrt(d**2-2.*p))
      b2=p/a
      write(*,*) d,p,a,b1,b2
      p=0.1*p
enddo
end
```

```
        d             p             a             b1             b2
   2.000000       1.000000      1.931852      0.5176381      0.5176381
   2.000000      0.1000000      1.999375      5.0015606E-02  5.0015643E-02
   2.000000      1.0000001E-02  1.999994      4.9999715E-03  5.0000162E-03
   2.000000      1.0000000E-03  2.000000      5.0001318E-04  5.0000002E-04
   2.000000      1.0000000E-04  2.000000      5.0034265E-05  5.0000002E-05
   2.000000      1.0000001E-05  2.000000      5.0033982E-06  5.0000003E-06
   2.000000      1.0000001E-06  2.000000      4.8841866E-07  5.0000006E-07
   2.000000      1.0000002E-07  2.000000      2.5000004E-08  5.0000008E-08
   2.000000      1.0000002E-08  2.000000      2.5000004E-09  5.0000009E-09
```

```
!vaja5.4a
!Rekurzijska formula - nestabilen algoritem
integer::n=0
real*4::I
I=1.-1./exp(1.)
write(*,*) 'n     In'
write(*,'(I3,f14.7)') n,I
do n=1,15
      I=1.-n*I
      write(*,'(I3,f14.7)') n,I
enddo
end
```

```
n      In
  0     0.6321205
  1     0.3678795
  2     0.2642411
  3     0.2072767
  4     0.1708932
  5     0.1455340
  6     0.1267958
  7     0.1124296
  8     0.1005630
  9     0.0949326
 10     0.0506744
 11     0.4425812
 12    -4.3109741
 13    57.0426636
 14  -797.5972900
 15 11964.9589844
```

```
!vaja5.4b
!Rekurzijska formula - stabilen algoritem
integer::n=16
real*4::I
I=0.
write(*,*) 'n     In'
write(*,'(I3,f14.7)') n-1,I
do n=15,1,-1
      I=(1.-I)/n
      write(*,'(I3,f14.7)') n-1,I
enddo
end
```

```
n      In
 15     0.0000000
 14     0.0666667
 13     0.0666667
 12     0.0717949
 11     0.0773504
 10     0.0838772
  9     0.0916123
  8     0.1009320
  7     0.1123835
  6     0.1268024
  5     0.1455329
  4     0.1708934
  3     0.2072766
  2     0.2642411
  1     0.3678795
  0     0.6321205
```

1. naloga:

Narišite funkcijo *f(x)* na intervalu od *a* do *b* z uporabo podprograma »SCRSHO.FOR«.

2. naloga:

Narišite funkcijo *f(x)* na intervalu od *a* od *b* v grafičnem načinu. Pri nalogi si pomagajte s programom »num5.for«.

3. naloga:

Izračunajte korene nelinearne enačbe

$$x + 0.3 \cos x = e^{-x}$$

na štiri mesta natančno z linearno iterativno metodo.

4. naloga:

Izračunajte korene nelinearne enačbe

$$x + \sin x = e^{-x}$$

na štiri mesta natančno s popravljeno iterativno metodo.

5. naloga:

Izračunajte korene nelinearne enačbe

$$x + 0.3 \cos x = e^{-x}$$

na šest mest natančno z Newtonovo (tangentno) metodo.

```
!vaja 6.1
! Program narise grobi graf na ekran,
! poklice podprogram scrsho(fun), Numerical Recipes,
! ki sprejme kot argument funkcijo.
! V projekt vkljucimo podprogram 'scrsho.for'.
! Lepso sliko dobimo s Quick Win projektom.
! Vsa okna moramo maksimalno povecati.
! Kurzor ni viden.
!     glavni program
!     USE MSFLIB  !dodamo, ce zelimo kurzor
      EXTERNAL fun
      REAL*4::fun
!     I=DISPLAYCURSOR($GCURSORON)   !dodamo, ce zelimo kurzor
      call scrsho(fun)
      END


! Vpisemo funkcijo, ki jo zelimo narisati
! Podprogramska funkcija
      FUNCTION fun(x)
            fun=exp(0.8*x)*sin(x)
      END
```

```
!Risanje funkcije na zaslon v tekstovnem nacinu
      SUBROUTINE scrsho(fx)
      INTEGER ISCR,JSCR
      REAL fx
      EXTERNAL fx
      PARAMETER (ISCR=60,JSCR=21)
      INTEGER i,j,jz
      REAL dx,dyj,x,x1,x2,ybig,ysml,y(ISCR)
      CHARACTER*1 scr(ISCR,JSCR),blank,zero,yy,xx,ff
      SAVE blank,zero,yy,xx,ff
      DATA blank,zero,yy,xx,ff/' ','-','l','-','x'/
1     continue
      write (*,*) ' Enter x1,x2 (= to stop)'
      read (*,*) x1,x2
      if(x1.eq.x2) return
      do 11 j=1,JSCR
      scr(1,j)=yy
      scr(ISCR,j)=yy
11    continue
      do 13 i=2,ISCR-1
            scr(i,1)=xx
            scr(i,JSCR)=xx
            do 12 j=2,JSCR-1
                  scr(i,j)=blank
12          continue
13    continue
      dx=(x2-x1)/(ISCR-1)
      x=x1
      ybig=0.
      ysml=ybig
      do 14 i=1,ISCR
            y(i)=fx(x)
            if(y(i).lt.ysml) ysml=y(i)
            if(y(i).gt.ybig) ybig=y(i)
            x=x+dx
14    continue
      if(ybig.eq.ysml) ybig=ysml+1.
      dyj=(JSCR-1)/(ybig-ysml)
      jz=1-ysml*dyj
      do 15 i=1,ISCR
            scr(i,jz)=zero
            j=1+(y(i)-ysml)*dyj
            scr(i,j)=ff
15    continue
      write (*,'(1x,1pe10.3,1x,80a1)') ybig,(scr(i,JSCR),i=1,ISCR)
      do 16 j=JSCR-1,2,-1
            write (*,'(12x,80a1)') (scr(i,j),i=1,ISCR)
16    continue
      write (*,'(1x,1pe10.3,1x,80a1)') ysml,(scr(i,1),i=1,ISCR)
      write (*,'(12x,1pe10.3,40x,e10.3)') x1,x2
      goto 1
      END
! (C) Copr. 1986-92 Numerical Recipes Software *1:%0Nkp#.
```

```
!vaja6.2
!Risanje funkcije v graficnem nacinu
USE MSFLIB
EXTERNAL fun
LOGICAL*2::lk=.TRUE.
REAL*8::xzl,yzl,xsd,ysd,x,y,dx,dy,fun
INTEGER::test
TYPE(wxycoord)::xys
CHARACTER*6::ndat
test=DISPLAYCURSOR($GCURSORON)
test=SETBKCOLOR(15) !bela
CALL CLEARSCREEN($GCLEARSCREEN)
test = SETTEXTCOLOR(0) ! crna
WRITE(*,*)' Podaj xzl,yzl,xsd,ysd...'
READ(*,*) xzl,yzl,xsd,ysd
test=DISPLAYCURSOR($GCURSOROFF)
test=INITIALIZEFONTS()
test=SETFONT('t''Times New Roman CE''h18w10pvib')
IF(SETWINDOW(lk,xzl,yzl,xsd,ysd)==0) THEN
      STOP ' okna ne morem postaviti'
ENDIF
test=SETCOLOR(12) !Rdeca
IF(RECTANGLE_W($GBORDER,xzl,yzl,xsd,ysd)==0) THEN
      STOP ' okvirja ne bo'
ENDIF
test=SETCOLOR(9) !Modra
CALL MOVETO_W(xzl,DBLE(0.),xys)
test=LINETO_W(xsd,DBLE(0.))
CALL MOVETO_W(DBLE(0.),ysd,xys)
test=LINETO_W(DBLE(0.),yzl)
dy=(yzl-ysd)/20
WRITE(NDAT,'(F6.1)') xzl
CALL MOVETO_W(xzl,dy,xys)
CALL OUTGTEXT(NDAT)
WRITE(NDAT,'(F6.1)') xsd
CALL MOVETO_W(xsd-(xsd-xzl)/10,dy,xys)
CALL OUTGTEXT(NDAT)
test=SETCOLOR(10) !zelena
dx=(xsd-xzl)/1000
DO x=xzl,xsd,dx
      y=fun(x)
      test=SETPIXEL_W(x,y)
ENDDO
OPEN(12,FILE='USER',TITLE='Tabela funkcije')
DX=(XSD-XZL)/10
DO X=XZL,XSD,DX
      WRITE(12,'(f12.3,E15.7)') x,fun(x)
ENDDO
CLOSE(12,STATUS='KEEP')
READ(*,*)
END



!Funkcija, ki jo zelimo narisati
REAL*8 FUNCTION fun(x)
REAL*8::x
fun=SIN(x-0.356)
END
```

```
      !vaja6.3
      real*4 function g(x)
      g=exp(-x)-0.3*cos(x)
      endfunction


      program linearna_iterativna_metoda
      external g
      write(*,*) 'Podaj x0, n, eps'
      read(*,*) x0,n,eps
      do i=1,n
            x1=g(x0)
            if(abs(x1)<1.e-30) stop 'Rel. napake ne morem izraziti'
            if(abs((x1-x0)/x1)<eps) goto 1      !L<=1/2
            x0=x1
      enddo
      write(*,*) 'Koren',x1,'je narobe.'
      stop
1     x2=g(x1)
      al=(x2-x1)/(x1-x0)
      al1=al/(1-al)
      write(*,*) 'Tocna ocena napake:', al1*abs((x2-x1)/x2)
      write(*,*) 'Koren=',x2
      stop
      end
      !test: x0=0.4, n=100, eps=0.5e-4
      !Tocna ocena napake:  -8.6918299E-06
      !Koren=  0.3961468



      !vaja6.4
      real*4 function g(x)
      g=x-(x+sin(x)-exp(-x))/2.634
      endfunction


      program popravljena_iterativna_metoda
      external g
      write(*,*) 'Podaj x0, n, eps'
      read(*,*) x0,n,eps
      do i=1,n
            x1=g(x0)
            if(abs(x1)<1.e-30) stop 'Rel. napake ne morem izraziti'
            if(abs((x1-x0)/x1)<eps) goto 1      !L<=1/2
            x0=x1
      enddo
      write(*,*) 'Koren',x1,'je narobe.'
      stop
1     x2=g(x1)
      al=(x2-x1)/(x1-x0)
      al1=al/(1-al)
      write(*,*) 'Tocna ocena napake:', al1*abs((x2-x1)/x2)
      write(*,*) 'Koren=',x2
      stop
      end
      !test: x0=0.36, n=10, eps=0.5e-4
      !Tocna ocena napake: -4.7501336E-10
      !Koren=  0.3544631
```

```
      !vaja6.5
      real*4 function g(x)
      f=x+0.3*cos(x)-exp(-x)
      df=1-0.3*sin(x)+exp(-x)
      g=x-f/df
      endfunction

      program newtonova_metoda
      external g
      write(*,*) 'Podaj x0, n, eps'
      read(*,*) x0,n,eps
      do i=1,n
            x1=g(x0)
            if(abs(x1)<1.e-30) stop 'Rel. napake ne morem izraziti'
            if(abs((x1-x0)/x1)<eps) goto 1      !L<=1/2
            x0=x1
      enddo
      write(*,*) 'Koren',x1,'je narobe.'
      stop
1     x2=g(x1)
      al=(x2-x1)/(x1-x0)
      al1=al/(1-al)
      write(*,*) 'Tocna ocena napake:', al1*abs((x2-x1)/x2)
      write(*,*) 'Koren=',x2
      stop
      end
      !test: x0=0.4, n=10, eps=0.5e-6
      !Tocna ocena napake: -3.7615575E-08
      !Koren=  0.3961434
```

1. naloga:
Izračunajte korene nelinearne enačbe

$$x + 0.3 \cos x = e^{-x}$$

na ε natančno s *Sekantno metodo*.

2. naloga:
Izračunajte korene nelinearne enačbe

$$x + 0.3 \cos x = e^{-x}$$

na ε natančno z *Bisekcijsko metodo*. Nato prilagodite program metodi *Regula Falsi*.

3. naloga:
Izračunajte korene nelinearne enačbe

$$x + 0.3 \cos x = e^{-x}$$

na ε natančno s kombinacijo *Newtonove* in *Bisekcijske metode*. Pri izračunu uporabite podprogram »rtsafe.for«.

4. naloga:
Poiščite ničle polinoma

$$2x^4 + 7x^3 - 4x^2 + 29x + 14 = 0$$

na 6 točnih cifer z uporabo *Newtonove metode* in *Hornerjevega algoritma*.

```
      !vaja7.1
      !Sekantna metoda

      !funkcija
      function g(x0,xm1)
      f(x)=exp(-x)-0.3*cos(x)-x
      g=x0-(x0-xm1)/(f(x0)-f(xm1))*f(x0)
      end

      !glavni program
      external g
      write(*,*) 'Podaj xm1, x0, eps in n'
      read(*,*) xm1,x0,eps,n
      do i=1,n
            x1=g(x0,xm1)
            if(abs((x1-x0)/x1)<eps) then
                  goto 1
            else
                  xm1=x0
                  x0=x1
            endif
      enddo
      write(*,*) 'Narobe koren = ',x1
      stop
1     continue
      al=(x1-x0)/(x0-xm1)
      al1=al/(1-al)
      write(*,*) 'Tocnejsa ocena napake = ',al1*abs((x1-x0)/x1)
      write(*,*) 'Koren = ',x1
      end
      ! xm1=0.3, x0=0.35, eps=0.00001, n=20
      !  Tocnejsa ocena napake =  -3.1751476E-10
      !  Koren =   0.3961434
```

```
      !vaja7.2
      !Bisekcijska metoda

      !glavni program
      external f
      write(*,*) 'Podaj x1, x2 in eps:'
      read(*,*) x1,x2,eps
      write(*,*) 'Koren = ', x(f,x1,x2,eps)
      end
      ! x1=0.3, x2=0.5, rel_nap=0.00001
      ! Koren = 0.3961426


      !funkcija
      real function f(x)
      f=x+0.3*cos(x)-exp(-x)
      end


      !bisekcijska metoda
      real function x(f,x1,x2,eps)
      f1=f(x1)
      f2=f(x2)
      if(f1*f2>=0) then
            stop 'Nepravilna izbira x1 in x2!'
      else
1           continue
            x3=0.5*(x1+x2)                ! Bisekcijska metoda
!           x3=x2-f2*(x2-x1)/(f2-f1)      ! metoda Regula Falsi
            f3=f(x3)
            if(abs(f3)==0) then
                  x=x3
                  return
            endif
            if(f2*f3<0) then
                  x1=x3
                  f1=f3
            else
                  x2=x3
                  f2=f3
            endif
            if(abs((x2-x1)/(x2+x1))<eps) then
                  napaka=0
                  if(abs(f2)<abs(f1)) then
                        x=x2
                  else
                        x=x1
                  endif
            else
                  goto 1
            endif
      endif
      end
```

```
!vaja7.3
!kombinacija Newtonove in bisekcijske metode


!glavni program
external rtsafe,funcd
write(*,*) 'Podaj x1, x2 in eps'
read(*,*) x1,x2,eps
x=rtsafe(funcd,x1,x2,eps)
write(*,*) 'Koren = ',x
end
! x1=0.3, x2=0.5, eps=0.00001
! Koren =  0.3961434


!podprogram
subroutine funcd(x,f,df)
f=x+0.3*cos(x)-exp(-x)
df=1-0.3*sin(x)+exp(-x)
return
end
```

```
      FUNCTION rtsafe(funcd,x1,x2,xacc)
      INTEGER MAXIT
      REAL rtsafe,x1,x2,xacc
      EXTERNAL funcd
      PARAMETER (MAXIT=100)
      INTEGER j
      REAL df,dx,dxold,f,fh,fl,temp,xh,xl
      call funcd(x1,fl,df)
      call funcd(x2,fh,df)
      if((fl.gt.0..and.fh.gt.0.).or.(fl.lt.0..and.fh.lt.0.))pause
     *'root must be bracketed in rtsafe'
      if(fl.eq.0.)then
        rtsafe=x1
        return
      else if(fh.eq.0.)then
        rtsafe=x2
        return
      else if(fl.lt.0.)then
        xl=x1
        xh=x2
      else
        xh=x1
        xl=x2
      endif
      rtsafe=.5*(x1+x2)
      dxold=abs(x2-x1)
      dx=dxold
      call funcd(rtsafe,f,df)
      do 11 j=1,MAXIT
        if(((rtsafe-xh)*df-f)*((rtsafe-xl)*df-f).ge.0..or. abs(2.*
     *f).gt.abs(dxold*df) ) then
          dxold=dx
          dx=0.5*(xh-xl)
          rtsafe=xl+dx
          if(xl.eq.rtsafe)return
        else
          dxold=dx
          dx=f/df
          temp=rtsafe
          rtsafe=rtsafe-dx
          if(temp.eq.rtsafe)return
        endif
        if(abs(dx).lt.xacc) return
        call funcd(rtsafe,f,df)
        if(f.lt.0.) then
          xl=rtsafe
        else
          xh=rtsafe
        endif
11    continue
      pause 'rtsafe exceeding maximum iterations'
      return
      END
!  (C) Copr. 1986-92 Numerical Recipes Software *1:%0Nkp#.
```

```
      !vaja7.4
      !Iskanje nicel polinoma z Newtonovo metodo in Hornerjevim algoritmom

      !podprogram Horner
      subroutine horner(a,x,n)
      real,dimension(10)::a
      do i=1,n
            a(i+1)=a(i)*x+a(i+1)
      enddo
      end

      !glavni program
      real,dimension(10)::a,b,c
      a(1)=2.;a(2)=7.;a(3)=-4.;a(4)=29.;a(5)=14.
      n=4
      x=2.
      b=a
2     a=b
      call horner(a,x,n)
      write(*,*)
      write(*,*) 'P(x)=',a(n+1)
      write(*,*) (a(i),i=1,n+1)
      c=a
      call horner(a,x,n-1)
      write(*,*) 'dP(x)=',a(n)
      x1=x-a(n+1)/a(n) !Newtonova metoda
      write(*,*) 'priblizek: x1=',x1
      if(abs((x1-x)/x1)<1e-6) then
            write(*,*) 'Koren=',x1
            b=c
            n=n-1
            write(*,*) 'po korenu...',(b(i),i=1,n+1)
            if(n<=2) goto 4
      else
            x=x1
            read(*,*)
      end if
      goto 2
      !resevanje kvadratne enacbe
4     a=b
      d=a(2)**2-4*a(1)*a(3)
      if(d>0) then
            d=sqrt(d)
            x1=(-a(2)+d)/(2*a(1))
            x2=(-a(2)-d)/(2*a(1))
            write(*,*) 'Korena sta:',x1,x2
      elseif(d<0) then
            d=sqrt(-d)
            x1=-a(2)/2/a(1)
            x2=d/2/a(1)
            write(*,*) 'Korena sta:'
            write(*,*) x1,'+',x2,'i'
            write(*,*) x1,'-',x2,'i'
      else
            write(*,*) 'Samo en koren=',a(2)/2/a(1)
      endif
      end
      !x1=-0.4384472
      !x2=-4.561553
      !x3=0.7500002 + 1.713914i
      !x4=0.7500002 - 1.713914i
```

1. naloga:

Izračunajte rešitev sistema linearnih enačb po postopku *Gaussove eliminacije*. Uporabite podprogram za Gaussovo metodo brez pivotiranja *GAUS.FOR*.

$$x_1 + x_2 + 3x_4 = 4$$
$$2x_1 + x_2 - x_3 + x_4 = 1$$
$$3x_1 - x_2 - x_3 + 2x_4 = -3$$
$$-x_1 + 2x_2 + 3x_3 - x_4 = 4$$

2. naloga:

Izračunajte determinanto matrike *A*. Uporabite podprogram za razcep matrike na zgornjo in spodnjo trikotno matriko *LUDCMP.FOR*.

$$A = \begin{bmatrix} 1 & -1 & 2 & -1 \\ 2 & -2 & 3 & -3 \\ 1 & 1 & 1 & 0 \\ 1 & -1 & 4 & 3 \end{bmatrix}$$

Izpišite spodnjo *L* in zgornjo *U* trikotno matriko.

3. naloga:

Izračunajte rešitev sistema linearnih enačb iz prve naloge po *Gauss-Jordanovi metodi*.

4. naloga:

Z Gauss-Jordanovo metodo izračunajte inverzno matriko matrike *A* iz druge naloge. Uporabite podprogram za Gauss-Jordanovo metodo *GAUSSJ.FOR*.

```
!vaja8.1
!resevanje sistema linearnih enacb z Gaussovo metodo
integer, parameter::ndat=10
real, dimension(ndat,ndat)::A
real, dimension(ndat)::x
integer::i,j,n
open(10,file='matrika.txt')
open(20,file='resitev.txt')
read(10,*) n
do i=1,n
      read(10,*) (A(i,j),j=1,n+1)
enddo
call gaus(A,x,n,ndat)
do i=1,n
      write(*,'(a,i1,a,f10.4)') 'x(',i,')=',x(i)
      write(20,'(a,I1,a,f10.4)') 'x(',i,')=',x(i)
enddo
end
```

```
matrika.txt
4
 1  1  0  3  4
 2  1 -1  1  1
 3 -1 -1  2 -3
-1  2  3 -1  4

resitev.txt
x(1)=   -1.0000
x(2)=    2.0000
x(3)=    0.0000
x(4)=    1.0000
```

Gaussova metoda:

```
      subroutine gaus(a,x,n,nd)
      dimension a(nd,1),x(nd)
      do 20 i=1,n-1
      do 20 j=i+1,n
      f=a(j,i)/a(i,i)
      do 20 k=i,n+1
20    a(j,k)=a(j,k)-f*a(i,k)
      x(n)=a(n,n+1)/a(n,n)
      do 30 i=n-1,1,-1
      s=a(i,n+1)
      do 40 j=i+1,n
40    s=s-a(i,j)*x(j)
30    x(i)=s/a(i,i)
      return
      end
```

```
!vaja8.2
!Izracun determinante s pomocjo LU razcepa
integer,parameter::np=10
integer,dimension(np)::indx
real*4,dimension(np,np)::A,U,L
open(10,file='matrika.txt')
read(10,*) n
do i=1,n
      read(10,*) (A(i,j),j=1,n)
enddo
call ludcmp(A,n,np,indx,d)
do i=1,n
      d=d*A(i,i)
enddo
write(*,*) 'Determinanta = ',d
!zapis matrike L in U
do i=1,n
      do j=1,i-1
            U(i,j)=0.
            L(i,j)=A(i,j)
      enddo
      do j=i,n
            U(i,j)=A(i,j)
            L(i,j)=0.
      enddo
      L(i,i)=1.
enddo
write(*,*) 'Matrika L:'
do i=1,n
      write(*,'(4f8.3)') (L(i,j),j=1,n)
enddo
write(*,*) 'Matrika U:'
do i=1,n
      write(*,'(4f8.3)') (U(i,j),j=1,n)
enddo
end
```

```
matrika.txt
4
1 -1  2 -1
2 -2  3 -3
1  1  1  0
1 -1  4  3
```

```
Determinanta =    4.000000
 Matrika L:
   1.000   0.000   0.000   0.000
   2.000   1.000   0.000   0.000
   1.000   0.500   1.000   0.000
   1.000   0.500   0.200   1.000
 Matrika U:
   1.000   1.000   1.000   0.000
   0.000  -4.000   1.000  -3.000
   0.000   0.000   2.500   4.500
   0.000   0.000   0.000  -0.400
```

Podprogram LUDCMP.FOR

```
      SUBROUTINE ludcmp(a,n,np,indx,d)
      INTEGER n,np,indx(n),NMAX
      REAL d,a(np,np),TINY
      PARAMETER (NMAX=500,TINY=1.0e-20)
      INTEGER i,imax,j,k
      REAL aamax,dum,sum,vv(NMAX)
      d=1.
      do 12 i=1,n
        aamax=0.
        do 11 j=1,n
          if (abs(a(i,j)).gt.aamax) aamax=abs(a(i,j))
11      continue
        if (aamax.eq.0.) pause 'singular matrix in ludcmp'
        vv(i)=1./aamax
12    continue
      do 19 j=1,n
        do 14 i=1,j-1
          sum=a(i,j)
          do 13 k=1,i-1
            sum=sum-a(i,k)*a(k,j)
13        continue
          a(i,j)=sum
14      continue
        aamax=0.
        do 16 i=j,n
          sum=a(i,j)
          do 15 k=1,j-1
            sum=sum-a(i,k)*a(k,j)
15        continue
          a(i,j)=sum
          dum=vv(i)*abs(sum)
          if (dum.ge.aamax) then
            imax=i
            aamax=dum
          endif
16      continue
        if (j.ne.imax)then
          do 17 k=1,n
            dum=a(imax,k)
            a(imax,k)=a(j,k)
            a(j,k)=dum
17        continue
          d=-d
          vv(imax)=vv(j)
        endif
        indx(j)=imax
        if(a(j,j).eq.0.)a(j,j)=TINY
        if(j.ne.n)then
          dum=1./a(j,j)
          do 18 i=j+1,n
            a(i,j)=a(i,j)*dum
18        continue
        endif
19    continue
      return
      END
C  (C) Copr. 1986-92 Numerical Recipes Software *1:%0Nkp#.
```

```
!vaja8.3
!Gauss-Jordanova metoda
real*4,dimension(4,5)::A
open(10,file='matrika.txt')
read(10,*) n
do i=1,n
      read(10,*) (A(i,j),j=1,n+1)
enddo
do ii=1,n
      write(*,'(5f8.2)') (A(ii,jj),jj=1,n+1)
enddo
write(*,*)
do i=1,n
      !normiranje
      del=A(i,i)
      do j=1,n+1
            A(i,j)=A(i,j)/del
      enddo
      do ii=1,n
            write(*,'(5f8.2)') (A(ii,jj), jj=1,n+1)
      enddo
      pause 'Delili z vodilnim koeficientom.'
      !eliminacija
      do k=1,n
            if(k==i) cycle
            fakt=A(k,i)
            do j=i,n+1
                  A(k,j)=A(k,j)-fakt*A(i,j)
            enddo
      enddo
      do ii=1,n
            write(*,'(5f8.2)') (A(ii,jj),jj=1,n+1)
      enddo
      pause 'Po koraku'
enddo
write(*,*) 'Resitev...'
do i=1,n
      write(*,*) A(i,n+1)
enddo
end
!rezultat:
!x1 = -1.000000
!x2 = 2.000000
!x3 = 0.000000E+00
!x4 = 1.000000
```

```
matrika.txt
 4
 1  1  0  3  4
 2  1 -1  1  1
 3 -1 -1  2 -3
-1  2  3 -1  4
```

```
!vaja 8.4
!Izracun inverzne matrike
integer, parameter::ndat=4
real, dimension(ndat,ndat)::A,E
integer::i,j,n
open(10,file='matrika.txt')
open(20,file='inverzna.txt')
read(10,*) n
do i=1,n
      read(10,*) (A(i,j),j=1,n)
enddo
!tvorba enotske matrike
do i=1,n
      do j=1,n
            E(i,j)=0.
      enddo
E(i,i)=1.
enddo
!inverzna matrika
call gaussj(A,n,ndat,E,n,ndat)
do i=1,n
      write(*,'(4f8.4)') (E(i,j),j=1,n)
      write(10,'(4f8.4)') (E(i,j),j=1,n)
enddo
end
```

```
matrika.txt
4
1 -1  2 -1
2 -2  3 -3
1  1  1  0
1 -1  4  3
```

```
inverzna.txt
  -7.500   3.500   0.500   1.000
   3.000  -1.500   0.500  -0.500
   4.500  -2.000   0.000  -0.500
  -2.500   1.000   0.000   0.500
```

Gauss-Jordanova metoda:

```
      SUBROUTINE gaussj(a,n,np,b,m,mp)
            INTEGER m,mp,n,np,NMAX
            REAL a(np,np),b(np,mp)
            PARAMETER (NMAX=50)
            INTEGER i,icol,irow,j,k,l,ll,indxc(NMAX),indxr(NMAX),ipiv(NMAX)
            REAL big,dum,pivinv
            do 11 j=1,n
              ipiv(j)=0
      11    continue
            do 22 i=1,n
              big=0.
              do 13 j=1,n
                if(ipiv(j).ne.1)then
                  do 12 k=1,n
                    if (ipiv(k).eq.0) then
```

```
                if (abs(a(j,k)).ge.big)then
                  big=abs(a(j,k))
                  irow=j
                  icol=k
                endif
              else if (ipiv(k).gt.1) then
                pause 'singular matrix in gaussj'
              endif
12          continue
          endif
13      continue
        ipiv(icol)=ipiv(icol)+1
        if (irow.ne.icol) then
          do 14 l=1,n
            dum=a(irow,l)
            a(irow,l)=a(icol,l)
            a(icol,l)=dum
14        continue
          do 15 l=1,m
            dum=b(irow,l)
            b(irow,l)=b(icol,l)
            b(icol,l)=dum
15        continue
        endif
        indxr(i)=irow
        indxc(i)=icol
        if (a(icol,icol).eq.0.) pause 'singular matrix in gaussj'
        pivinv=1./a(icol,icol)
        a(icol,icol)=1.
        do 16 l=1,n
          a(icol,l)=a(icol,l)*pivinv
16      continue
        do 17 l=1,m
          b(icol,l)=b(icol,l)*pivinv
17      continue
        do 21 ll=1,n
          if(ll.ne.icol)then
            dum=a(ll,icol)
            a(ll,icol)=0.
            do 18 l=1,n
              a(ll,l)=a(ll,l)-a(icol,l)*dum
18          continue
            do 19 l=1,m
              b(ll,l)=b(ll,l)-b(icol,l)*dum
19          continue
          endif
21      continue
22    continue
      do 24 l=n,1,-1
        if(indxr(l).ne.indxc(l))then
          do 23 k=1,n
            dum=a(k,indxr(l))
            a(k,indxr(l))=a(k,indxc(l))
            a(k,indxc(l))=dum
23        continue
        endif
24    continue
      return
      END
C  (C) Copr. 1986-92 Numerical Recipes Software *1:%0Nkp#.
```

1. naloga:

Izračunajte rešitev sistema linearnih enačb

$$\begin{aligned} 1x_1 - 1x_2 + 2x_3 - x_4 &= -8 \\ 2x_1 - 2x_2 + 3x_3 - 3x_4 &= -20 \\ 1x_1 + 1x_2 + 1x_3 &= -2 \\ 1x_1 - x_2 + 4x_3 + 3x_4 &= 4 \end{aligned}$$

po *Gauss-Jordanovi metodi*. Uporabite podprogram GAUSSJ.FOR.

2. naloga:

Izračunajte rešitev sistema linearnih enačb

$$\begin{aligned} 6x_1 + 15x_2 + 55x_3 &= 82 \\ 15x_1 + 55x_2 + 225x_3 &= 170 \\ 55x_1 + 225x_2 + 979x_3 &= 502 \end{aligned}$$

po *metodi Cholesky*.

3. naloga:

Izračunajte rešitev sistema linearnih enačb

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

z *Jacobijevo* iterativno metodo. Matriko koeficientov preberite iz datoteke.

4. naloga:

Izračunajte rešitev sistema linearnih enačb iz 3. naloge z *Gauss-Seidlovo* iterativno metodo.

```
!vaja 9.1
!Resevanje sistema linearnih enacb z Gauss-Jordanovo metodo
integer, parameter::ndat=10
real, dimension(ndat,ndat)::A
real, dimension(ndat)::x,b
integer i,j,n
open(10,file='matrika.txt')
open(20,file='resitev.txt')
read(10,*) n
do i=1,n
      read(10,*) (A(i,j),j=1,n),b(i)
enddo
call gaussj(A,n,ndat,b,1,1)
do i=1,n
      write(*,'(a,i1,a,f10.4)') 'x(',i,')=',b(i)
      write(20,'(a,I1,a,f10.4)') 'x(',i,')=',b(i)
enddo
end
```

```
matrika.txt
4
 1 -1  2 -1 -8
 2 -2  3 -3 -20
 1  1  1  0 -2
 1 -1  4  3  4
```

```
resitev.txt
x(1)=   -7.0000
x(2)=    3.0000
x(3)=    2.0000
x(4)=    2.0000
```

```
!vaja9.2
!Metoda Cholesky
integer,parameter::np=10
real,dimension(np,np)::A
real,dimension(np)::b
integer i,j,n
open(20,file='podatki.txt')
read(20,*) n
do i=1,n
      read(20,*) (A(i,j),j=1,n),b(i)
enddo
call cholesky(A,b,n,np)
do i=1,n                                                    podatki.txt
      write(*,'(a,i1,a,f10.4)') ' x(',i,')=',b(i)           3
enddo                                                       6   15  55  82
end                                                         15  55 225 170
!Resitev: x1=12.0000, x2=8.0000, x3=-2.0000                 55 225 979 502

subroutine cholesky(A,b,n,np)
real,dimension(np,np)::A
real,dimension(np)::b
do i=1,n !razcep matrike na produkt dveh transponiranih trikotnih matrik
      s=0.
      do j=1,i-1
            s=s+A(j,i)**2
      end do
      if(A(i,i)<s) stop 'Napaka, matrika ni pozitivno definitna.'
      A(i,i)=sqrt(A(i,i)-s)
      do k=i+1,n
            s=0.
            do j=1,i-1
                  s=s+a(j,i)*A(j,k)
            end do
            A(i,k)=(A(i,k)-s)/A(i,i)
            A(k,i)=A(i,k)
      end do
end do
!resevanje enacbe Ly=b -> y=b/L
ii=0
do i=1,n
      s=b(i)
      if (ii/=0) then
            do j=ii,i-1
                  s=s-a(i,j)*b(j)
            end do
      else if (s/=0.) then
            ii=i
      end if
      b(i)=s/a(i,i)
end do
!resevanje enacbe Ux=y -> x=y/U
do i=n,1,-1
s=b(i)
      if (i<n) then
            do j=i+1,n
                  s=s-a(i,j)*b(j)
            end do
      end if
      b(i)=s/a(i,i)
end do
end
```

```
!vaja9.3
!resevanje sistema linearnih enacb z Jacobijevo iterativno metodo
parameter (nd=20)
real,dimension(nd,nd)::A
real,dimension(nd)::x,xs,d
open(20,file='podatki.txt')
read(20,*) n
do i=1,n
      read(20,*) (A(i,j),j=1,n),d(i)
enddo
write(*,*) 'Podaj dopustno napako.'
read(*,*) es
do
      do i=1,n
            xs(i)=x(i)
      enddo
      do i=1,n
            s=0.
            do j=1,n
                  s=s+A(i,j)*x(j)
            enddo
            x(i)=x(i)+(d(i)-s)/A(i,i)
      enddo
      ep=0.
      do i=1,n
            ep=ep+abs((xs(i)-x(i))/x(i))
      enddo
      if(ep<es) exit
enddo
open(30,file='rezultati.txt')
do i=1,n
      write(*,'(a,i1,a,f10.5)') ' x(',i,')=',x(i)
      write(30,'(a,i1,a,f10.5)') ' x(',i,')=',x(i)
enddo
end
!es=0.000001
```

```
podatki.txt
3
6     3     -1    -2
1     3     -1    -7
-1    -1    3     7
```

```
rezultati.txt
x(1)=   1.00000
x(2)=  -2.00000
x(3)=   2.00000
```

```
!vaja9.4
!resevanje sistema linearnih enacb z Gauss-Seidlovo iterativno metodo
parameter (nd=20)
real,dimension(nd,nd)::A
real,dimension(nd)::x,xs,d
open(20,file='podatki.txt')
read(20,*) n
do i=1,n
      read(20,*) (A(i,j),j=1,n),d(i)
enddo
write(*,*) 'Podaj dopustno napako.'
read(*,*) es
do
      do i=1,n
            xs(i)=x(i)
      enddo
      do i=1,n
            s=0.
            do j=1,i-1
                  s=s+A(i,j)*x(j)
            enddo
            do j=i+1,n
                  s=s+A(i,j)*x(j)
            enddo
            x(i)=(d(i)-s)/A(i,i)
      enddo
      ep=0.
      do i=1,n
            ep=ep+abs((xs(i)-x(i))/x(i))
      enddo
      if(ep<es) exit
enddo
open(30,file='rezultati.txt')
do i=1,n
      write(*,'(a,i1,a,f10.5)') ' x(',i,')=',x(i)
      write(30,'(a,i1,a,f10.5)') ' x(',i,')=',x(i)
enddo
end
!es=0.000001
```

```
podatki.txt
3
6     3     -1    -2
1     3     -1    -7
-1    -1    3     7
```

```
rezultati.txt
x(1)=   1.00000
x(2)=  -2.00000
x(3)=   2.00000
```

1. naloga:

Rešite sistem dveh nelinearnih enačb

$$x^3 + x\,y - 10 = 0$$
$$y - 3\,x\,y^3 - 57 = 0$$

z *navadno iterativno metodo* (*Jacobijeva iteracija*) na ε natančno.

2. naloga:

Rešite sistem dveh nelinearnih enačb

$$x^2 + y^2 = 16$$
$$x^2 + 16\,y^2 = 64$$

z *Newtonovo metodo* na pet točnih cifer.

3. naloga:

Tabeliraj funkcijo $y=sin(x)$ po koraku $h=0.1$ od 0 do π. Nato izračunajte vrednosti odvodov predhodno tabelirane funkcije v točki $x=0.8$.

Izračunajte:
a) prvi odvod po centralno diferenčni metodi,
b) drugi odvod po centralno diferenčni metodi,
c) tretji odvod po centralno diferenčni metodi,
d) četrti odvod po centralno diferenčni metodi in
e) prvi odvod po Rombergovem postopku.

Vse izračunane vrednosti odvodov primerjajte z analitično izračunanimi vrednostmi.

```
!vaja10.1
!iterativna metoda za iskanje nicel dveh nelinearnih enacb
g1(x,y)=(10-x*y)**(0.333333)
g2(x,y)=((57.-y)/(3.*x))**(1./3.)
write(*,*) 'Podaj zacetni priblizek x0,y0'
read(*,*) x0,y0
write(*,*) 'Podaj natancnost (eps)'
read(*,*) es
write(*,*) 'Podaj maksimalno stevilo iteracij (maxit)'
read(*,*) maxit
ep=1.1*es
iter=0
do while(iter<maxit)
      iter=iter+1
      x1=g1(x0,y0)
      y1=g2(x0,y0)
      if((x1/=0).and.(y1/=0)) then
            ep=abs((x1-x0)/x1)+abs((y1-y0)/y1)
            if(ep<es) exit
      endif
      x0=x1
      y0=y1
enddo
if(iter>=maxit) write(*,*) 'Rezultat je narobe'
write(*,*) 'Resitev sistema:',x1,y1
write(*,*) 'Dosezena natancnost:',ep
write(*,*) 'x**3+xy-10=',x1**3+x1*y1-10
write(*,*) 'y+3xy**3-57=',y1+3*x1*y1**3-57
end
! x0=1, y0=1, es=0.000001, maxit=100
! Resitev sistema:       1.823907        2.156097
! Dosezena natancnost:   7.238923E-07
! x**3+xy-10=  -8.222180E-06
! y+3xy**3-57=   2.653994E-05
```

```
!vaja10.2
!Newtonova metoda za iskanje nicel dveh nelinearnih enacb
real*4,dimension(2,3)::A
real*4,dimension(2)::x0,x1,dx
f1(x,y)=x**2+y**2-16
f2(x,y)=x**2+16*y**2-64
f1x(x,y)=2*x
f1y(x,y)=2*y
f2x(x,y)=2*x
f2y(x,y)=32*y
write(*,*) 'Podaj zacetni priblizek x0,y0'
read(*,*) x0(1),x0(2)
write(*,*) 'Podaj natancnost (eps)'
read(*,*) es
write(*,*) 'Podaj maksimalno stevilo iteracij (maxit)'
read(*,*) maxit
do while(iter<maxit)
      write(*,'(i5,2f14.6)') iter,x0(1),x0(2)
      iter=iter+1
      A(1,1)=f1x(x0(1),X0(2))
      A(1,2)=f1y(x0(1),X0(2))
      A(1,3)=-f1(x0(1),X0(2))
      A(2,1)=f2x(x0(1),X0(2))
      A(2,2)=f2y(x0(1),X0(2))
      A(2,3)=-f2(x0(1),X0(2))
      ! Kramerjevo pravilo
      dx(1)=(A(1,3)*A(2,2)-A(2,3)*A(1,2))/(A(1,1)*A(2,2)-A(1,2)*A(2,1))
      dx(2)=(A(2,3)*A(1,1)-A(1,3)*A(2,1))/(A(1,1)*A(2,2)-A(1,2)*A(2,1))
      ! Gaussova metoda
      ! call gaus(A,dx,2,2)
      x1(1)=x0(1)+dx(1)
      x1(2)=x0(2)+dx(2)
      ep=abs(dx(1)/x1(1))+abs(dx(2)/x1(2))
      if(ep<es) exit
      x0(1)=x1(1)
      x0(2)=x1(2)
enddo
if(iter>maxit) stop 'Divergira'
write(*,*) 'Resitev sistema=',x1(1),x1(2)
end
!x0=1.5, y0=3.5, eps=0.00005, maxit=10
!   0      1.500000      3.500000
!   1      5.016666      2.207143
!   2      3.784081      1.828490
!   3      3.583336      1.789284
!   4      3.577713      1.788854
!Resitev sistema=       3.577709        1.788854
```

```
!vaja10.3
!odvajanje tabelirane funkcije sinus
integer,parameter::nd=70
real*4,parameter::pi=3.141592
real*4,dimension(0:nd,0:nd)::F
real*4,dimension(nd)::x,y
real*4 h,dy1,dy2,dy3
integer     i,j,n,m
h=0.1
i=1
x(i)=0.
y(i)=sin(x(i))
do
      i=i+1
      x(i)=x(1)+(i-1)*h
      y(i)=sin(x(i))
      if(x(i)>=pi) exit
enddo
n=i
open(10,file='sinus.txt')
do i=1,n
      write(10,*) i,x(i),y(i)
enddo
i=9
!a) prvi odvod pri x(i)=0.8
dy1=(y(i+1)-y(i-1))/(2.*h)
write(*,*) 'prvi odvod dy1(',x(i),') =',dy1
write(*,*) 'tocna vrednost (analiticno)',cos(x(i))
write(*,*)
!b) drugi odvod pri x(i)=0.8
dy2=(y(i-1)-2.*y(i)+y(i+1))/(h**2)
write(*,*) 'drugi odvod dy2(',x(i),') =',dy2
write(*,*) 'tocna vrednost (analiticno)',-sin(x(i))
write(*,*)
!c) tretji odvod pri x(i)=0.8
dy3=(-y(i-2)+2.*y(i-1)-2.*y(i+1)+y(i+2))/(2.*h**3)
write(*,*) 'tretji odvod dy3(',x(i),') =',dy3
write(*,*) 'tocna vrednost (analiticno)',-cos(x(i))
write(*,*)
!d) cetrti odvod v tocki x(i)=0.8
dy4=(y(i-2)-4.*y(i-1)+6.*y(i)-4.*y(i+1)+y(i+2))/h**4
write(*,*) 'cetrti odvod dy4(',x(i),') =',dy4
write(*,*) 'tocna vrednost (analiticno)',sin(x(i))
write(*,*)
!e) prvi odvod po Rombergu v tocki x(i)=0.8
F(0,0)=(y(i+4)-y(i-4))/(x(i+4)-x(i-4))
F(1,0)=(y(i+2)-y(i-2))/(x(i+2)-x(i-2))
F(2,0)=(y(i+1)-y(i-1))/(x(i+1)-x(i-1))
m=2
do j=1,m
      do i=j,m
            F(i,j)=(4.**j*F(i,j-1)-F(i-1,j-1))/(4.**j-1)
      enddo
enddo
write(*,*) 'prvi odvod po Rombergu dy1(',x(9),') =',F(m,m)
write(*,*) 'tocna vrednost (analiticno)',cos(x(9))
end
!rezultati:
!dy1(0.8)=6.955463E-01, dy2(0.8)=-7.167578E-01
!dy3(0.8)=-6.950199E-01, dy4(0.8)=7.158517E-01
!Romberg dy1(0.8)=6.967067E-01
```

1. naloga:

Izračunajte določeni integral

$$I = \int_1^4 x^2 \cdot ln(x) dx$$

po *trapecni metodi*.

2. naloga:

Izračunajte dolžino loka sinusoide med dvema ničlama po *Simpsonovi metodi*.

3. naloga:

Izračunajte volumen telesa, ki nastane, ko zavrtimo sinusoido med dvema ničlama okoli osi x, po *Rombergovi metodi*.

4. naloga:

Izračunajte težišče ploskve, ki jo omejuje sinusoida med dvema ničlama in osjo x, po *Gaussovi metodi*.

```
!vaja11.1
!integriranje po trapecni metodi (izracun ploscine)
external f
write(*,*) 'Podaj meji a in b ter stevilo delitvenih tock'
read(*,*) a,b,n
ti1=ti(f,a,b,n)
ti2=ti(f,a,b,2*n-1)
eps=abs((ti2-ti1)/3.)
write(*,*) 'Vrednost integrala S =',ti2
write(*,*) 'Napaka = ',eps
end
!podatki: a=1, b=4, n=165
!rezultat:  s=22.57463, eps=6.3578290E-07


!funkcija
function f(x)
f=x**2*log(x)
return
end


!trapecna metoda
real function ti(f,a,b,n)
h=(b-a)/(n-1)
ti=f(a)+f(b)
do x=a+h,b-0.9*h,h
      ti=ti+2*f(x)
enddo
ti=ti*h*0.5
return
end
```

```
!vaja11.2
!integriranje po Simpsonovi metodi (izracun dolzine loka)
external f
write(*,*) 'Podaj meji a in b ter stevilo delitvenih tock'
read(*,*) a,b,n
if(mod(n,2)==0) stop 'Stevilo delitvenih tock mora biti liho!'
si1=si(f,a,b,n)
si2=si(f,a,b,2*n-1)
eps=abs((si2-si1)/15.)
write(*,*) 'Vrednost integrala S =',si2
write(*,*) 'Napaka = ',eps
end
!podatki: a=0, b=3.141592, n=31
!rezultat:  s=3.820197, eps=4.7683717E-08


!funkcija
function f(x)
f=sqrt(1+cos(x)**2)
return
end


!Simpsonova metoda
real function si(f,a,b,n)
h=(b-a)/(n-1)
si=f(a)+f(b)
do x=a+h,b-0.9*h,2*h
      si=si+4*f(x)
enddo
do x=a+2*h,b-1.9*h,2*h
      si=si+2*f(x)
enddo
si=si*h/3
return
end
```

```
      !vaja11.3
      !integriranje po Rombergovi metodi (izracun volumna telesa)
      real*4 t(0:12,0:12)
      external f
      write(*,*) 'Podaj meji a in b ter natancnost:'
      read(*,*) a,b,eps
      do i=0,12,1
            n=2**i+1
            t(i,0)=ti(f,a,b,n)
            do j=1,i
                  t(i,j)=(4**j*t(i,j-1)-t(i-1,j-1))/(4**j-1)
            enddo
            write(*,*) (t(i,j),j=0,i)
            if(i>0.and.abs(t(i,i)-t(i-1,i-1))<eps) goto 1
      enddo
      write(*,*) 'Rezultat je narobe'
1     write(*,*) 'Integral je ',t(i,i)
      end
      !podatki: a=0, b=3.141592, eps=0.000001
      !rezultat:  s=4.934803

      !funkcija
      function f(x)
      f=3.141592*sin(x)**2
      return
      end

      !trapecna metoda
      real function ti(f,a,b,n)
      h=(b-a)/(n-1)
      ti=f(a)+f(b)
      do x=a+h,b-0.9*h,h
            ti=ti+2*f(x)
      enddo
      ti=ti*h*0.5
      return
      end
```

```
!vaja11.4
!integriranje po Gaussovi metodi (izracun tezisca lika)
external fs,fx,fy
write(*,*) 'Podaj meji in stevilo delitvenih tock.'
read(*,*) a,b,n
s=gi(fs,a,b,n)
x0=gi(fx,a,b,n)/s
y0=gi(fy,a,b,n)/s
write(*,*) 's=',s,'x0=',x0,'y0=',y0
end
!podatki: a=0, b=3.141592, n=20
!rezultati:  s=2.000000, x0=1.570796, y0=0.3926991

!Gaussova kvadraturna metoda
real function gi(f,a,b,n)
h=(b-a)/(n-1)
gi=0
do xa=a,b-0.5*h,h
      xb=xa+h
      x0=(-h/sqrt(3.)+(xb+xa))*0.5
      x1=(h/sqrt(3.)+(xb+xa))*0.5
      gi=gi+(f(x0)+f(x1))*h/2
enddo
return
end

!funkcije
real function fs(x)
fs=sin(x)
end

real function fx(x)
fx=x*sin(x)
end

real function fy(x)
fy=0.5*sin(x)**2
end
```

1. <u>naloga:</u>
Izračunajte vrednost rešitve diferencialne enačbe

$$y' = y - x^2 + 1$$
$$y(0) = 0.5$$
$$0 \leq x \leq 2$$

po *Eulerjevi metodi* s korakom $h = 0.2$.

2. <u>naloga:</u>
Izračunajte vrednost rešitve diferencialne enačbe

$$y' = x^2 + xy$$
$$y(0.5) = 1.3$$

v točki $x = 0.7$ po metodi *Runge-Kutta 4. reda*.

3. <u>naloga:</u>
Izračunajte vrednost rešitve diferencialne enačbe drugega reda (začetni problem)

$$y'' - xy' + 3y = 6x$$
$$y(1) = 1 \quad \text{in} \quad y'(1) = 3$$

v točki $x = 1.1$ z metodo *Runge-Kutta 4. reda*.

4. <u>naloga:</u>
Rešite robni problem

$$y'' = x + (1 - 0.2x)y$$
$$y(1) = 2$$
$$y(3) = -1$$

po *strelski metodi*.

5. **naloga:**

Izračunajte rešitve robnega problema

$$y'' - x^2 y' + y = 1$$
$$y(0) = 1 \quad \text{in} \quad y(0.8) = -1$$

z *diferenčno metodo* pri delitvi intervala [0,0.8] na 4 dele.

```
!vaja12.1
!Eulerjeva metoda
!y'=y-x2+1
real*4::x,y,h
f(x,y)=y-x**2+1
!zacetni pogoj
x=0
y=0.5
write(*,*) 'podaj zeljeno tocko in stevilo delitvenih tock'
read(*,*) xk,n
h=(xk-x)/n
write(*,*) x,y
do i=1,n
      y=y+h*f(x,y)
      x=x+h
      write(*,*) x,y,(x+1)**2-0.5*exp(x)
enddo
end
!y=4.865785, prava vrednost y=5.305472
```

```
!vaja12.2
!Runge-Kutta 4. reda
!y'=x2-xy
real*4::k1,k2,k3,k4,x0,y0,h
f(x,y)=x**2-x*y
!robni pogoj
x0=0.5
y0=1.3
write(*,*) 'podaj zeljeno tocko in stevilo delitvenih tock'
read(*,*) xk,n
h=(xk-x0)/n
do i=1,n
      k1=h*f(x0,y0)
      k2=h*f(x0+h/2,y0+k1/2)
      k3=h*f(x0+h/2,y0+k2/2)
      k4=h*f(x0+h,y0+k3)
      yn=y0+1./6*(k1+2*k2+2*k3+k4)
      y0=yn
      x0=x0+h
      write(*,'(f5.2,f10.5)') x0,y0
enddo
end
!y(0.6)=1.25997;y(0.7)=1.2217
```

```
!vaja12.3
!Runge-Kutta 4. reda – linearna diferencialna enacba drugega reda
!y''-xy+3y=6x
real*4::k1,k2,k3,k4,l1,l2,l3,l4,x0,y0,z0,h
x0=1
y0=1
z0=3
h=0.1
k1=h*f1(x0,y0,z0)
l1=h*f2(x0,y0,z0)
k2=h*f1(x0+h/2,y0+k1/2,z0+l1/2)
l2=h*f2(x0+h/2,y0+k1/2,z0+l1/2)
k3=h*f1(x0+h/2,y0+k2/2,z0+l2/2)
l3=h*f2(x0+h/2,y0+k2/2,z0+l2/2)
k4=h*f1(x0+h,y0+k3,z0+l3)
l4=h*f2(x0+h,y0+k3,z0+l3)
yn=y0+1./6*(k1+2*k2+2*k3+k4)
zn=z0+1./6*(l1+2*l2+2*l3+l4)
y0=yn
z0=zn
x0=x0+h
write(*,*) x0,y0,z0
end
!y(1.1)=1.331,   y'(1.1)=3.63

real function f1(x,y,z)
f1=z
end

real function f2(x,y,z)
f2=x*z-3*y+6*x
end
```

```
!vaja12.4
!Strelska metoda
!y''=x+(1-0.2x)y


! funkciji
subroutine fun(x,y,z,yc,zc)
yc=z
zc=x+(1-0.2*x)*y
end


! metoda Runge-Kutta 4. reda - linearna diferencialna enacba drugega reda
subroutine rk4(x,y,z,x1,y1,z1,fun)
real*4::k1,l1,k2,l2,k3,l3,k4,l4
h=x1-x
call fun(x,y,z,yc,zc)
k1=h*yc
l1=h*zc
call fun(x+0.5*h,y+0.5*k1,z+0.5*l1,yc,zc)
k2=h*yc
l2=h*zc
call fun(x+0.5*h,y+0.5*k2,z+0.5*l2,yc,zc)
k3=h*yc
l3=h*zc
call fun(x+h,y+k3,z+l3,yc,zc)
k4=h*yc
l4=h*zc
y1=y+(k1+2*k2+2*k3+k4)/6
z1=z+(l1+2*l2+2*l3+l4)/6
end


! strelska metoda
subroutine strel(t,zadetek,izp)
logical izp
external fun
a=1.
b=3.
alfa=2.
y=alfa
z=t         !-1.5
h=0.1 ! 0.2
if(izp) write(*,*) 'x, y, z =',a,y,z
do x=a,b-h/2,h
      x1=x+h
      call rk4(x,y,z,x1,y1,z1,fun)
      if(izp) write(*,*) 'x1, y1, z1 =',x1,y1,z1
      y=y1
      z=z1
enddo
zadetek=y1
end
```

```
!vaja12.4
!Strelska metoda - nadaljevanje

! Glavni program
logical izp
external fun
izp=.false.
beta=-1
t1=0. !-1.5
call strel(t1,z1,izp)
write(*,*) ' z1=',t1,z1
pause
t2=1. !.0
call strel(t2,z2,izp)
write(*,*) 'z2=',t2,z2
pause
do i=1,20
      t3=t2-(z2-beta)*(t1-t2)/(z1-z2)
      call strel(t3,z3,izp)
      write(*,*) 't3=',t3,z3
      t1=t2
      z1=z2
      t2=t3
      z2=z3
      pause
      if(abs(z3-beta)<1.e-5) exit
enddo
izp=.true.
call strel(t2,z2,izp)
end
! strel: t=-3.494987
```

```
!vaja12.5
!diferencna metoda
!y''+p(x)y'+q(x)y=r(x)
!y''-x2y'+y=1
real*4,dimension(10,10)::a
real*4,dimension(10)::x
!koeficienti diferencialne enacbe
p(x)=-x**2
q(x)=1
r(x)=1
!robni pogoji
x0=0
y0=1
xk=0.8
yk=-1
write(*,*) 'Vpisite stevilo intervalov'
read(*,*) n
if(n>10) stop 'Preveliko stevilo intervalov!'
h=(xk-x0)/n
n=n-1 ! stevilo enacb
a=0   ! zacetne vrednosti elementov matrike
! prva enacba
a(1,1)=-2+h**2*q(x0+h)
a(1,2)=1+0.5*h*p(x0+h)
a(1,n+1)=h**2*r(x0+h)-y0*(1-p(x0+h)*0.5*h)
!splosna enacba
do i=2,n-1
      a(i,i-1)=1-p(x0+i*h)*0.5*h
      a(i,i)=-2+h**2*q(x0+i*h)
      a(i,i+1)=1+p(x0+i*h)*0.5*h
      a(i,n+1)=h**2*r(x0+i*h)
enddo
!zadnja enacba
a(n,n-1)=1-p(xk-h)*0.5*h
a(n,n)=-2+h**2*q(xk-h)
a(n,n+1)=h**2*r(xk-h)-yk*(1+p(xk-h)*0.5*h)
write(*,'(f5.1)') h
do i=1,n
write(*,'(20f10.4)') (a(i,j),j=1,n+1)
enddo
call gaus(a,x,n,10)
write(*,'(20f10.4)') y0,(x(i),i=1,n),yk
end
!y0=1,y1=0.4674,y2=-0.0480,y3=-0.5376,y4=-1
```

1. naloga

a) Aproksimirajte funkcijsko tabelo

x	$y(x)$
1.2	0.932
1.6	0.999
1.9	0.946
2.4	0.675
4.1	-0.818

s premico $f(x,a_1,a_2)=a_1+a_2x$ v smislu najmanjših kvadratov.

b) Premico in tabelo predstavite grafično.

2. naloga

Aproksimirajte funkcijsko tabelo

x	$y(x)$
1.3	8.385
1.5	10.81
2.1	19.53
2.4	24.79
2.9	34.64

z nelinearno funkcijo $f(x,a,b)=a\,x^b$.

3. naloga

Aproksimirajte funkcijsko tabelo

x	$y(x)$
0.2	1.79
0.8	7.00
1.3	10.61
1.9	13.60
2.6	15.37
3.0	15.91
3.3	16.34

s kombinacijo treh funkcij $sin(x)$, x in x^2.

```
!vaja13.1a
!Aproksimacija tabele s premico
real*4,dimension(20)::x,y
real*4,dimension(2,2)::A,B
f1(z)=1
f2(z)=z
open(12,file='podatki.txt')
read(12,*) n
do i=1,n
      read(12,*) x(i),y(i)
enddo
do i=1,n
      A(1,1)=A(1,1)+f1(x(i))**2
      A(1,2)=A(1,2)+f1(x(i))*f2(x(i))
      B(1,1)=B(1,1)+y(i)*f1(x(i))
      A(2,2)=A(2,2)+f2(x(i))**2
      B(2,1)=B(2,1)+y(i)*f2(x(i))
enddo
A(2,1)=A(1,2)
do i=1,2
      write(*,*)(A(i,j),j=1,2),B(i,1)
enddo
call gaussj(A,2,2,B,1,1)
write(*,'('' a1='',f12.6,'' a2='',f12.6)') B(1,1),B(2,1)
end
!     5.000000       11.200000        2.734000
!  11.200000       30.180000        2.780400
! a1=    2.017741 a2=    -.656670


podatki.txt
5
1.2         0.932
1.6         0.999
1.9         0.946
2.4         0.675
4.1         -0.818
```

```
!vaja13.1b
!Graficna predstavitev
USE MSFLIB
EXTERNAL fun
LOGICAL*2 lk/.TRUE./
REAL*8 xzl,yzl,xsd,ysd,x,y,dx,dy,fun
INTEGER test
TYPE(wxycoord) xys
CHARACTER*6 ndat
test=DISPLAYCURSOR($GCURSORON)
test=SETBKCOLOR(15) !bela
CALL CLEARSCREEN($GCLEARSCREEN)
test = SETTEXTCOLOR(0) ! crna
WRITE(*,*)' Podaj xzl,yzl,xsd,ysd...'
READ(*,*) xzl,yzl,xsd,ysd
test=DISPLAYCURSOR($GCURSOROFF)
test=INITIALIZEFONTS()
test=SETFONT('t''Times New Roman CE''h18w10pvib')
IF(SETWINDOW(lk,xzl,yzl,xsd,ysd)==0) THEN
      STOP ' okna ne morem postaviti'
ENDIF
test=SETCOLOR(12) !Rdeca
IF(RECTANGLE_W($GBORDER,xzl,yzl,xsd,ysd)==0) THEN
      STOP ' okvirja ne bo'
ENDIF
test=SETCOLOR(9) !Modra
CALL MOVETO_W(xzl,DBLE(0.),xys)
test=LINETO_W(xsd,DBLE(0.))
CALL MOVETO_W(DBLE(0.),ysd,xys)
test=LINETO_W(DBLE(0.),yzl)
dy=(yzl-ysd)/20
WRITE(NDAT,'(F6.1)') xzl
CALL MOVETO_W(xzl,dy,xys)
CALL OUTGTEXT(NDAT)
WRITE(NDAT,'(F6.1)') xsd
CALL MOVETO_W(xsd-(xsd-xzl)/10,dy,xys)
CALL OUTGTEXT(NDAT)
test=SETCOLOR(10) !zelena
dx=(xsd-xzl)/1000
DO x=xzl,xsd,dx
      y=fun(x)
      test=SETPIXEL_W(x,y)
ENDDO
test=SETCOLOR(0) !crna
open(12,file='podatki.txt')
read(12,*) n
do i=1,n
      read(12,*) x,y
      test=ELLIPSE_W($GFILLINTERIOR,x-0.01D0,y+0.01D0,x+0.01D0,y-0.01D0)
enddo
READ(*,*)
END


!Funkcija, ki jo zelimo narisati
REAL(8) FUNCTION fun(x)
REAL(8) x
fun=2.017741-0.65667*x
END
```

```
!vaja13.2
!Aproksimacija tabele s funkcijo y=a*x**b
real*4,dimension(20)::x,y,z
real*4,dimension(2,2)::A,B
f1(t)=1
f2(t)=log(t)
open(12,file='podatki.txt')
read(12,*) n
if(n>20) stop 'Prevec podatkov'
do i=1,n
      read(12,*) x(i),y(i)
      z(i)=log(y(i))
enddo
do i=1,n
      A(1,1)=A(1,1)+f1(x(i))**2
      A(1,2)=A(1,2)+f1(x(i))*f2(x(i))
      B(1,1)=B(1,1)+z(i)*f1(x(i))
      A(2,2)=A(2,2)+f2(x(i))**2
      B(2,1)=B(2,1)+z(i)*f2(x(i))
enddo
A(2,1)=A(1,2)
do i=1,2
      write(*,*)(A(i,j),j=1,2),B(i,1)
enddo
call gaussj(A,2,2,B,1,1)
write(*,'('' a1='',f12.6,'' a2='',f12.6)') exp(B(1,1)),B(2,1)
end
!       5.000000        3.349946       14.234320
!       3.349946        2.683763       10.313150
! a1=    5.275003 a2=    1.767020
```

```
podatki.txt
5
1.3         8.385
1.5         10.81
2.1         19.53
2.4         24.79
2.9         34.64
```

```
!vaja13.3
!Aproksimacija tabele s funkcijo y=a*sin(z)+b*x+c*x**2
!a=5.432, b=3.435, c=0.538
real*4,dimension(20)::x,y
real*4,dimension(3,3)::A,B
f1(z)=sin(z)
f2(z)=z
f3(z)=z**2
open(12,file='podatki.txt')
read(12,*) n
do i=1,n
      read(12,*) x(i),y(i)
enddo
do i=1,n
      A(1,1)=A(1,1)+f1(x(i))**2
      A(1,2)=A(1,2)+f1(x(i))*f2(x(i))
      A(1,3)=A(1,3)+f1(x(i))*f3(x(i))
      B(1,1)=B(1,1)+y(i)*f1(x(i))
      A(2,2)=A(2,2)+f2(x(i))**2
      A(2,3)=A(2,3)+f2(x(i))*f3(x(i))
      B(2,1)=B(2,1)+y(i)*f2(x(i))
      A(3,3)=A(3,3)+f3(x(i))**2
      B(3,1)=B(3,1)+y(i)*f3(x(i))
enddo
A(2,1)=A(1,2)
A(3,1)=A(1,3)
A(3,2)=A(2,3)
do i=1,3
      write(*,*)(A(i,j),j=1,3),B(i,1)
enddo
call gaussj(A,3,3,B,1,1)
write(*,'('' a1='',f12.6,'' a2='',f12.6,'' a3='',f12.6)')
B(1,1),B(2,1),B(3,1)
end
! a1=    5.389493 a2=    3.487976 a3=     .521089
```

```
podatki.txt
7
0.2         1.79
0.8         7
1.3         10.61
1.9         13.6
2.6         15.37
3.0         15.91
3.3         16.34
```

1. naloga:

Za spodnjo tabelo izračunajte Newtonov interpolacijski polinom. Nato izračunajte še vrednost pri $x = 5.5$.

1	1.000000
2	1.414214
3	1.732051
4	2.000000
5	2.236068
6	2.449490
7	2.645751
8	2.828427
9	3.000000
10	3.162278

2. naloga:

Za tabelo v prvi nalogi po Aitkenovi metodi izračunajte vrednost pri $x = 5.5$.

3. naloga:

Z uporabo inverzne interpolacije izračunajte ničlo funkcije.

$$f(x) = 0.345x(x-5) - 3.745\sin(x) - 4$$

4. naloga:

Za spodnjo tabelo izračunajte naravni kubični zlepek.

1.0	1.5
3.0	2.5
4.5	1.0
7.0	2.5
9.0	0.5

```
!vaja14.1
!Newtonov interpolacijski polinom
real*4,dimension(0:20,0:20)::A
real*4,dimension(0:20)::x,y
open(12,file='podatki.txt')
read(12,*) n
n1=n-1
do j=0,n1
      read(12,*) x(j),y(j)
enddo
do j=0,n1
      A(j,0)=x(j)
      A(j,1)=y(j)
enddo
do j=2,n
      do i=0,n-j
            A(i,j)=(A(i+1,j-1)-A(i,j-1))/(A(i+j-1,0)-A(i,0))
      enddo
enddo
write(*,*)'Podaj xp='
read(*,*) xp
P=A(0,n)
do i=n1,1,-1
      P=P*(xp-x(i-1))+A(0,i)
enddo
write(*,*) p
end
!y(5.5)=2.3452
```

```
podatki.txt
10
1     1.000000
2     1.414214
3     1.732051
4     2.000000
5     2.236068
6     2.449490
7     2.645751
8     2.828427
9     3.000000
10    3.162278
```

```
!vaja14.2
!Aitkenova interpolacijska metoda
real*4,dimension(0:20,0:20)::A
real*4,dimension(0:20)::x,y
open(12,file='podatki.txt')
read(12,*) n
n1=n-1
do j=0,n1
      read(12,*) x(j),y(j)
enddo
write(*,*) 'Podaj xp'
read(*,*) xp
do j=0,n1
      A(j,0)=x(j)
      A(j,1)=y(j)
      A(j,n+1)=x(j)-xp
enddo
do j=2,n
      do i=j-1,n1
         A(i,j)=(A(j-2,j-1)*A(i,n+1)-A(i,j-1)*A(j-2,n+1))/(A(i,0)-A(j-2,0))
      enddo
enddo
do i=0,n1
      write(*,'(9f7.3)') (A(i,j),j=0,n+1)
enddo
write(*,*) 'P=', A(n1,n)
end
!y(5.5)=2.345201

!vaja14.3
!Inverzna interpolacija
f(z)=0.345*(z-5)*z-3.745*sin(z)-4.
real*4,dimension(0:25,0:25)::A
real*4,dimension(0:25)::x,y
write(*,*) 'Vpisite spodnjo in zgornjo mejo xs in xz:'
read(*,*) xs,xz
write(*,*) 'Vpisite delitev n:'
read(*,*) n
if(n>25) stop 'Preveliko stevilo delitev n!'
n1=n-1
h=(xz-xs)/n
y(0)=xs
x(0)=f(xs)
do j=1,n1
      y(j)=xs+j*h
      x(j)=f(xs+j*h)
enddo
xp=0
do j=0,n1
      A(j,0)=x(j)
      A(j,1)=y(j)
      A(j,n+1)=x(j)-xp
enddo
do j=2,n
      do i=j-1,n1
         A(i,j)=(A(j-2,j-1)*A(i,n+1)-A(i,j-1)*A(j-2,n+1))/(A(i,0)-A(j-2,0))
      enddo
enddo
write(*,*) 'P=', A(n1,n)
end
!xs=7, xz=9, n=15 -> x0= 7.847748 (tocno 7.85717)
```

```
!vaja14.4
!naravni kubicni zlepek
integer,parameter::nd=6
real,dimension(nd,nd)::A
real,dimension(nd,1)::B
real,dimension(0:nd+1)::x,y,h,g
open(12,file='podatki.txt')
read(12,*) n0
if(n0-2>nd) stop 'Preveliko stevilo podatkov!'
do i=0,n0-1
      read(12,*) x(i),y(i)
enddo
n=n0-1
do i=0,n-1
      h(i)=x(i+1)-x(i)
enddo
A(1,1)=2*h(0)+2*h(1)
A(1,2)=h(1)
B(1,1)=6*((y(2)-y(1))/h(1)-(y(1)-y(0))/h(0))
do i=2,n-2
      A(i,i-1)=h(i-1)
      A(i,i)=2*h(i-1)+2*h(i)
      A(i,i+1)=h(i)
      B(i,1)=6*((y(i+1)-y(i))/h(i)-(y(i)-y(i-1))/h(i-1))
enddo
A(n-1,n-2)=h(n-2)
A(n-1,n-1)=2*h(n-2)+2*h(n-1)
B(n-1,1)=6*((y(n)-y(n-1))/h(n-1)-(y(n-1)-y(n-2))/h(n-2))
call gaussj(A,n-1,nd,B,1,1)
g(0)=0
do i=1,n-1
      g(i)=B(i,1)
enddo
g(n)=0
write(*,*) 'konstante:'
do i=0,n
      write(*,'('' G('',i3,'')='',f10.5)') i,g(i)
enddo
write(*,*) 'koeficienti kubicnih zlepkov:'
write(*,*) ' i     ai        bi        ci        di'
do i=0,n-1
      ak=y(i)
      bk=(y(i+1)-y(i))/h(i)-(2*h(i)*g(i)+h(i)*g(i+1))/6
      ck=g(i)/2
      dk=(g(i+1)-g(i))/(6*h(i))
      write(*,'(i3,f10.5,f10.5,f10.5,f10.5)') i,ak,bk,ck,dk
enddo
end
! G(1)=  -1.72125
! G(2)=   2.03250
! G(3)=  -1.63125


podatki.txt
5
1.0   1.5
3.0   2.5
4.5   1.0
7.0   2.5
9.0   0.5
```