

## Vježbe 4:

1. Razlika između lokalno i globalno definirane funkcije
2. Primjer pronalaženje nul-točke
3. Primjer rekurzivne funkcije
4. Podprogrami
5. CERN programska biblioteka, primjeri
6. Separatno prevođenje (prevođenje složenih projekata)
7. Stvaranje vlastitih biblioteka podprograma i funkcija

# DEFINIRANE VLASTITE FUNKCIJE

FUNKCIJA vraća u program podatak ili rezultat

Osnovna struktura funkcije:

```
tip FUNCTION ime_funkcije(argumenti)
deklaracija varijabli koje se koriste
ime_funkcije = ...
...
END FUNCTION ime_funkcije
```

Razlika između lokalno i globalno definirane funkcije:

## GLOBALNO

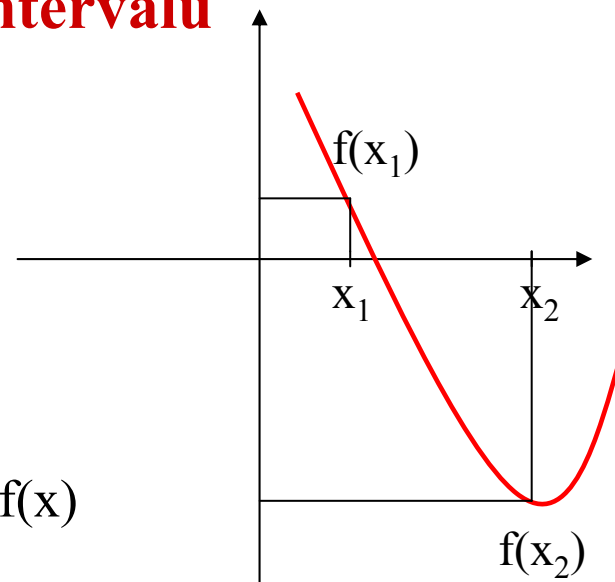
```
program intfunk1
real::fun, x, pi=3.14
x=-4.
do while (x.le.4.)
print *,fun(x)
x=x+0.5
end do
end program

real function fun(xx)
real:: xx, pi=3.14
fun=2*exp(xx)*pi
end function
```

## LOKALNO

```
program intfunk
real::x, pi=3.14
x=-4.
do while (x.le.4.)
print *,fun(x)
x=x+0.5
end do
contains
real function fun(xx)
real:: xx
fun=2*exp(xx)*pi
end function
end program
```

# Pronalaženje nul-točke funkcije u zadanom intervalu



## 1. Metoda bisekcije

- a) interval  $x_1$  i  $x_2$ , točnost epsilon
- b) zadati  $f(x)$  kao GLOBALNU FUNKCIJU
- c) ako je  $f(x_1) \cdot f(x_2) > 0$  završi program, ispiši da  $f(x)$  nema nultočku u tom intervalu
- d) ako je  $f(x_1) \cdot f(x_2) < 0$
- e) Izračunati  $x_0 = (x_1 + x_2) / 2$
- f) Provjeriti gdje funkcija mijenja predznak i taj interval ponovno ispitivati  $f(x_1) \cdot f(x_0) < 0$   $x_0$  postaje  $x_2$   
ili  $f(x_0) \cdot f(x_2) < 0$   $x_0$  postaje  $x_1$
- g) Definirati novi  $x_0$
- h) petlju prekinuti ako se zadovolje uvjeti  
 $|f(x_0)| \leq \text{epsilon}$  ili  $|x_2 - x_1| \leq \text{epsilon}$

Nađi nultočku funkcije  $f(x)=x^3-28$  metodom bisekcije:

$$f(x_1=3)=-1 \quad f(x_2=3.1)=+1.79$$

```
if (fun(x1)*fun(x2).gt.0) then
print *, 'u zadanom intervalu funkcija nema nultočku'
stop
end if
do while (abs(x2-x1).gt.epsi)
x0=(x1+x2)/2.0
if (fun(x1)*fun(x0).LT.0.0)then
x2=x0
else
x1=x0
end if
if(abs(x2-x1).LE.epsi)then
print *, 'Nultočka x0 je', x0
end if
end do
```

## 2. Newton-Rhapson metoda

Za nalaženje  $f(x)=0$ , koristi informaciju o 1. derivaciji  $f'(x)$  da bi ocjenili koliko daleko i u kojem smjeru leži nul-točka

Ako je  $x$  aprox. nul-točke  $x_0$  greška  $e=x-x_0$ ,  $x_0=x-e$

Taylorov razvoj  $f(x_0)=f(x-e)=f(x)-ef'(x)/1!+e^2f''(x)/2!-e^3f'''(x)/3!+...$

$$f(x_0)=f(x)-ef'(x) \rightarrow f(x_0)=0 \rightarrow$$

$e=f(x)/f'(x)$  – procjena greške u blizini od  $x$

$$x_0=x-f(x)/f'(x)$$

$$x_{i+1}=x_i-f(x_i)/f'(x_i)$$

INPUT X, Tolerance

DO

Error =  $F(x) / F'(x)$

OUTPUT X, Error

IF (  $|\text{Error}| < \text{Tolerance}$  ) EXIT

X = X - Error

END DO

! Procjena greške

! Ispis trenutnih vrijednosti

! Izlaz ako je ostvarena točnost

! Oduzima grešku

Za  $f(x)=x^3-28$  napiši program koji NR metodom traži nultočku

Treba napisati i funkciju za prvu derivaciju  $f'(x)=3x^2$

# Rekurzivne funkcije

## Primjeri za rekurzivne funkcije

### 1. Potencije

$$x^n = x * x^{n-1}$$

### 2. Fibonaccijev niz

$$f(1)=1$$

$$f(2)=1$$

$$f(n)=f(n-1)+f(n-2)$$

### 3. Faktorijele

$$f(1)=1$$

$$f(n)=f(n-1)*n$$

```
program poten
```

```
integer :: n
```

```
real :: x
```

```
print *, 'ucitaj x i potenciju od x koju zelis izracunati'
```

```
read *, x, n
```

```
print *, n, 'ta potencija broja', x, 'je', pot(x, n)
```

```
contains
```

```
recursive function pot(a, i) result(y)
```

```
integer :: i
```

```
real :: a
```

```
select case(i)
```

```
case(0) ; y=1
```

```
case(1) ; y=x
```

```
case(2:) ; y=pot(a, i-1)*x
```

```
end select
```

```
end function
```

```
end program
```

## PODPROGRAMI

SUBROUTINE ime\_podprograma(argumenti)

Deklaracije tipova varijabli

Tvrđnje

END SUBROUTINE

Podprogram se u glavnom programu poziva sa:

CALL ime\_podprograma(pravi argumenti)

```
PROGRAM zamjeni
IMPLICIT NONE !mijenja dvije varijable koristeći podprogram
REAL:: a,b
PRINT *, 'Učitaj dva broja, a i b'
READ *, a,b
PRINT *, 'OK, znači a = ',a,' b = ',b
CALL swap(a,b)
PRINT *, 'Ali nakon zamjene, a = ',a,' b = ',b
END PROGRAM
```

```
SUBROUTINE swap(x,y)
IMPLICIT NONE
REAL, INTENT(INOUT):: x,y
REAL:: temp
temp = x
x = y
y = temp
END SUBROUTINE
```

PROGRAM procedure

IMPLICIT NONE

! Pokazuje kako se program može modularizirati korištenjem

! funkcija i podprograma

REAL :: x, y1, phi

REAL, PARAMETER :: pi=3.1415926

REAL, EXTERNAL :: sinc

! Podprogram koji nema nikakve argumente

CALL welcome

! Podprogram koji čita ulazne vrijednosti

CALL input(x)

! Primjer definicije vanjske funkcije sinc:

$y1 = 2.0 * \text{sinc}(x * \text{pi}/4)$

! Primjer podprograma koji racuna:

CALL potential(1.0, 2.5, phi)

! Primjer korištenja podprograma za ispis

CALL output(y1, phi)

END PROGRAM

```
SUBROUTINE welcome
```

```
    PRINT *, 'Dobrodosli u program!'
```

```
END SUBROUTINE
```

```
SUBROUTINE input(x)
```

```
    IMPLICIT NONE
```

! Podprogram koji učitava ulazne varijable

! dummy deklaracija argumenata

```
    REAL, INTENT(OUT) :: x
```

! INTENT(OUT) –vrijednost od x nije definirana prije ulaska u podprogram vec se definira u podprogramu i vraca u glavni program

```
    PRINT *, 'Ucitaj x'
```

```
    READ *,x
```

```
END SUBROUTINE
```

```
REAL FUNCTION sinc(z)
```

```
    IMPLICIT NONE
```

```
    ! function to calculate the sinc function: (sin(x)/x)**2
```

```
    ! dummy deklaracija argumenata
```

```
    ! INTENT(IN) vrijednost od z nije definirana u funkciji vec je
```

```
    ! funkcija dobiva iz glavnog programa i ne smije je mijenjati
```

```
    REAL, INTENT(IN) :: z
```

```
        sinc = (SIN(z)/z)**2
```

```
    END FUNCTION
```

```
SUBROUTINE potential(z,r,pot)
```

```
IMPLICIT NONE
```

```
! Podprogram racuna potencijal točkastog naboja ze
```

```
! Na udaljenosti r ( $\phi = ze/(4 \pi \epsilon r)$ )
```

```
REAL, INTENT(IN) :: z,r
```

```
REAL, INTENT(OUT) :: pot
```

```
REAL, PARAMETER :: e=1.6022e-19,pi=3.1415926,eps=8.8542e-12
```

```
REAL, PARAMETER :: constant=e/(4.0*pi*eps)
```

```
! calculate the potential
```

```
pot = constant*z/r
```

```
END SUBROUTINE
```

```
SUBROUTINE output(x1,x2)
  IMPLICIT NONE
```

! Podprogram koji ispisuje vrijednosti od x1,x2

```
  REAL, INTENT(IN) :: x1,x2
  PRINT *, 'Prva vrijednost je ',x1
  PRINT *
  PRINT *, 'Druga vrijednost je ',x2
```

```
END SUBROUTINE
```

CERNova numerička biblioteka programa se nalazi u [libmathlib.so](http://libmathlib.so),  
[libmathlib.a](http://libmathlib.a)

CERN programska biblioteka je velika zbirka programa i podprograma koja se održava i nudi u oba oblika kao izvorni i objektni kod na centralnim računalima CERNa.

Većina tih programa je razvijena na CERN-u i kao takvi su direktna posljedica potreba u istraživačkim laboratorijima.

Ipak, velika većina njih koristi opće numeričke metode ili metode obrade podataka koje se mogu primjeniti na veliki broj problema.

Biblioteka sadrži otprilike 2500 podprograma i programa koji su grupirani u logičke cjeline po temama u otprilike 450 programskih paketa.

80% programa je napisano u Fortranu<sup>77</sup> ostatak u C-u i assembly kodu, obično s FORTRAN-skom verzijom koja je isto dostupna.

## **Katalog biblioteke (cernlib.pdf) short writeups na 416 str!!!:**

### **-elementarne funkcije**

B002 PRMFCT Prim brojevi i rastavljanje na prim brojeve

B100 RBINOM binomni koeficijenti

B101 ATG funkcija arc tangens

B102 ASINH hiperbolni arc sinus

B105 RPLNML vrijednost polinoma

B300 RSRTNT integral tipa  $R(x; \sqrt{a + bx + cx^2})$

### **-jednadžbe i specijalne funkcije (neke)**

C200 RZEROX nula funkcije jedne realne varijable

C201 RSNLEQ numeričko rješenje sistema nelinearnih jednadžbi

C202 RMULLZ nule realnog polinoma

C207 RRTEQ3 korjени kubne jednadžbe

C209 CPOLYZ nule kompleksnog polinoma

## **-integracija, minimizacija, nelinearni fit (neke)**

D101 SIMPS integracija pomoću Simpsonovog pravila

D103 GAUSS integracija Gaussovom metodom

D104 RCAUCH Cauchy integracija

D105 RTRINT Integracija preko trokuta

D108 TRAPER Integracija pomoću trapezne formule

D200 RRKSTP Diferencijalne jednačbe prvog reda (Runge-Kutta)

D203 RRKNYS Diferencijalne jednačbe drugog reda

D401 RDERIV Numeričko diferenciranje

D501 LEAMAX metoda najmanjih kvadrata

D503 RMINFC minimum funkcije jedne varijable

D506 MINUIT minimizacija funkcije i analiza grešaka

I druge grupe...

- interpolacije, aproksimacije, linearni fit
- matrice, vektori, linearne jednažbe
- statistička analiza i vjerojatnost
- ulaz/izlaz
- ispis i grafički prikaz podataka
- izvršne rutine
- obrada podataka
- traženje i obrada pogrešaka
- dizajn optike snopa, magneta, elektronike
- kvantna mehanika, čestična fizika
- slučajni brojevi, korisne općenite rutine
- statistička obrada i prezentacija podataka

-Pozivanje CERNove numeričke biblioteke koja se nalazi u datoteci libmathlib.so ili libmathlib.a:

```
>ifort program.f90 -o program.exe -lmathlib
```

Koje sve biblioteke postoje može se vidjeti s:

```
>ls /usr/lib/
```

Primjer 1:

Nultočka funkcije u određenom intervalu?

Funkcija je  $f(x)=x^3-28$

$x=3.03659$

**RZEROX**

CERN Program Library

**C200****Author(s)** : K.S. Kölbig**Submitter** :**Language** : Fortran**Library**: MATHLIB**Submitted**: 01.05.1990**Revised**: 01.12.1994

### Zero of a Function of One Real Variable

Function subprograms RZEROX and DZEROX compute, to an attempted specified accuracy, a zero  $x_0$  of a real-valued function  $f(x)$  lying in a given interval  $[a, b]$ , where  $f(a) * f(b) \leq 0$ .

On computers other than CDC or Cray, only the double precision version DZEROX is available. On CDC and Cray computers, only the single-precision version RZEROX is available.

#### Structure:

FUNCTION subprograms

User Entry Names: RZEROX, DZEROX

Obsolete User Entry Names: ZEROX  $\equiv$  RZEROX

Files Referenced: Unit 6

External References: User-supplied FUNCTION subprogram

#### Usage:

For  $t = R$  (type REAL),  $t = D$  (type DOUBLE PRECISION),
$$tZEROX(A, B, EPS, MAXF, F, MODE)$$
has, in any arithmetic expression, the value  $x_0$ .

A,B	(type according to $\tau$ ) On entry, A and B must specify the end points of the search interval. Unchanged on exit.
EPS	(type according to $\tau$ ) On entry, EPS must be equal to the accuracy parameter (see <b>Accuracy</b> ). Unchanged on exit.
MAXF	(INTEGER) On entry, MAXF must be equal to the maximum permitted number of references to the function F within the iteration loop. Unchanged on exit.
F	(type according to $\tau$ ) Name of a user-supplied FUNCTION subprogram, declared EXTERNAL in the calling program. This subprogram must set $F(X) = f(X)$ .
MODE	(INTEGER) On entry, MODE = 1 or MODE = 2 defines the algorithm for finding $x_0$ (see <b>Method</b> and <b>Notes</b> ).

### Method:

Two algorithms are incorporated in this subprogram. These are described in Ref. 1 as “Algorithm M” (MODE = 1) and “Algorithm R” (MODE = 2). Both are mixtures of linear interpolation, rational interpolation and bisection.

### Accuracy:

These subprograms try to compute two numbers  $x_0$  and  $x_1$  lying in the interval  $[a, b]$  such that

1.  $f(x_0)f(x_1) \leq 0$
2.  $|f(x_0)| \leq |f(x_1)|$
3.  $|x_0 - x_1| \leq 2 * \text{EPS} * (1 + |x_0|)$

If successful, the value of  $x_0$  is assigned to the function name.

1.  $\text{MODE} = 1$  should be used for fairly simple functions whose evaluation is cheap in comparison with the calculations performed in one iteration step of RZEROX or DZEROX.
2.  $\text{MODE} = 2$  should be used for more expensive functions. Convergence should be faster than for  $\text{MODE} = 1$ , but the evaluation steps are more expensive.
3. For functions which have a pole near the exact zero,  $\text{MODE} = 1$  is recommended because of the special character of the interpolation formula which is used.

### **Error handling:**

1.  $F(A) * F(B) > 0$ . The function value is set equal to zero.
2.  $\text{MODE}$  has a value other than 1 or 2. The function value is set equal to zero.
3. The number of references to  $F$  exceeds  $\text{MAXF}$ . The function value is set equal to the last computed value of  $x_0$  (see **Accuracy**)

For each error a message is printed.

### **Source:**

The subprogram is based on Algol programs described in Ref. 1.

### **References:**

1. J.C.P. Bus and T.J. Dekker, Two efficient algorithms with guaranteed convergence for finding a zero of a function, ACM Trans. Math. Software **1** (1975) 330–345.

```
PROGRAM bisekcija1
REAL(KIND=8) :: a=3.0_8, b= 4.0_8
REAL(KIND=8) :: eps = 1.0D-5,nula
INTEGER(KIND=8) :: maxf=100
EXTERNAL fun
nula = DZEROX(a,b,eps,maxf,fun,1)
PRINT *,'Nultočka ove funkcije je ',nula
END PROGRAM

FUNCTION fun (x)
REAL(KIND=8) :: x
fun = x**3-28_8
END FUNCTION
```

Kompajliramo i linkamo ga s:

```
>ifort program.f90 -o program.exe -lmathlib
```

Zadatak: Izračunaj derivaciju funkcije  $f(x)=(1+x)^{1/x}$  za  $x=2$  korištenjem CERNLIB subrutine tDERIV

**RDERIV**

CERN Program Library

**D401**

**Author(s)** : K.S. Kölbig

**Submitter** :

**Language** : Fortran

**Library**: MATHLIB

**Submitted**: 15.02.1989

**Revised**: 01.12.1994

### Numerical Differentiation

Subroutine subprograms RDERIV and DDERIV compute an approximate numerical value of the derivative  $f'(x)$  of a function  $f(x)$  at a specified point  $x$ .

On computers other than CDC and Cray, only the double-precision version DDERIV is available. On CDC and Cray computers, only the single-precision version RDERIV is available.

#### Structure:

SUBROUTINE subprograms

User Entry Names : RDERIV, DDERIV

Obsolete User Entry Names: DERIV  $\equiv$  RDERIV

Files Referenced : Unit 6

External References: MTLMTR (N002), ABEND (Z035), user-supplied FUNCTION subprogram

#### Usage:

For  $t = R$  (type REAL),  $t = D$  (type DOUBLE PRECISION),

```
CALL tDERIV(F,X,DELTA,DFDX,RERR)
```

- F (type according to  $\tau$ ) Name of a user-supplied FUNCTION subprogram, declared EXTERNAL in the calling program. This subprogram must set  $F(X) = f(X)$ .
- X (type according to  $\tau$ ) The specified point  $x$  for which the derivative is to be calculated.
- DELTA (type according to  $\tau$ ) On entry, DELTA must contain a scaling factor  $\delta$ , which can usually be set equal to 1. On exit, it contains the last value of this factor (see Method).
- DFDX (type according to  $\tau$ ) On exit, DFDX contains an approximation to  $f'(X)$ .
- RERR (type according to  $\tau$ ) On exit, RERR contains an estimate of the relative error of DFDX.

### Method:

The method is based on an extension to numerical differentiation of Romberg's principle of sequence extrapolation, originally developed for numerical integration. The subroutine starts by computing the 10 numbers

$$T_0^{(k)} = [f(x+h) - f(x-h)]/(2h), \quad (k = 0, 1, \dots, 9),$$

with

$$\begin{aligned} h &= \delta * 0.0256 * 2^{-k/2} && (k \text{ even}) \\ h &= \delta * 0.0192 * 2^{-(k-1)/2} && (k \text{ odd}), \end{aligned}$$

where the scaling factor  $\delta$  is initially set to DELTA. This procedure is repeated up to 9 times, with  $\delta$  replaced by  $\delta/10$  each time, until the sequence  $T_0^{(k)}$  is found to be monotone. A Romberg-like triangular table

$$T_m^{(k)} = u_m T_{m-1}^{(k+1)} - w_m T_{m-1}^{(k)},$$

with appropriate weights  $u_m, w_m$  is then computed for  $m = 1, 2, \dots, 9; k = 0, 1, \dots, 9 - m$ , and DFDX is set equal to  $T_9^{(0)}$ .

**Restrictions:**

The function  $f(x)$  must be differentiable at  $x = X$  and in a neighbourhood of  $X$ . Misleading results will be obtained if this is not true.

**Error handling:**

Error D401.1: If the function  $f(x)$  is such that, after 9 successive reductions of  $\delta$  by a factor 1/10, the sequence  $T_0^{(k)}$  is not monotone, an error message is written Unit 6, unless subroutine MTLSET (N002) has been called. DFDX is set equal to zero, RERR is set equal to one in this case.

**References:**

1. H. Rutishauser, Ausdehnung des Rombergschen Prinzips, Numer. Math. **5** (1963) 48–54.

PROGRAM derivacija

implicit none

REAL(KIND=8) :: dfdx,x,rerr,delta=1.\_8

**EXTERNAL** funk

print \*,'tocka u kojoj zelis izracunati derivaciju?'

read \*,x

call DDERIV(funk,x,delta,dfdx,rerr)

PRINT \*,'derivacija funkcije u',x,' je ',dfdx,' s procjenom greške',rerr

END PROGRAM

FUNCTION funk (x)

REAL(KIND=8) :: x

funk=(1.\_8+x)\*\*(1.\_8/x)

END FUNCTION

$$y=(1+x)^{1/x} \text{ za } x=2$$

$$\text{Za provjeru } y'=(1+x)^{1/x} \left( \frac{1}{x} \right) \left( \left( -\frac{1}{x} \right) \ln(1+x) + \frac{1}{(1+x)} \right)$$

$$\text{za } x=2 \text{ je } y' = -0.18703794085$$