Numerical Methods in Physics

Numerische Methoden in der Physik, 515.421.

Instructor: Ass. Prof. Dr. Lilia Boeri
Room: PH 03 090
Tel: +43-316-873 8191
Email Address: l.boeri@tugraz.at

Room: TDK Seminarraum **Time:** 8:30-10 a.m.

Exercises: Computer Room, PH EG 004 F

http://itp.tugraz.at/LV/boeri/NUM_METH/index.html
(Lecture slides, Script, Exercises, etc).

Last week(19/11/2013)

- **Zeroes of a Trascendental Equation: Definition.**
- Iterative method: general concepts and convergence criteria.
- The Newton-Raphson method: definition, convergence.
- **The Newton-Raphson method: a program.**

Zeroes of a Trascendental Equation: Definition.

$$\boxed{F(x)=0}$$

The values of *x* which satisfy this equation are called **zeroes**, **solutions** or **roots** of the function.

Iterative methods:

$$\boxed{F(x)=0} \Rightarrow \boxed{f(x)=x}$$

reduces to:

We can then find the solution iterating the expression:

$$\boxed{\mathrm{x}_{\mathrm{t+1}}=\mathrm{f}(\mathrm{x}_{\mathrm{t}})}$$

$$\boxed{\lim_{t\to\infty} \mathrm{x}_{\mathrm{t}}=\mathrm{x}_{\mathrm{ex}}}$$

Convergence:

The convergence of the iteration depends on the way in which the reformulation **F(x) = 0 -> f(x) =x** is performed. The iteration in the interval I converges to:

$$\boxed{\lim_{t\to\infty}\left|x_{ex} - x_{t+1}\right| \leq \frac{|\delta|}{1-m}}$$

If: $0 \leq m \leq 1$ $\quad m = \max_I f'(x) \quad \delta = \text{roundoff error}$

The upper boundary for the iteration at time (t+1) is given by:

$$\boxed{\left|x_{ex} - x_{t+1}\right| \leq \frac{m}{1-m}\left|x_{t+1} - x_t\right| + \frac{|\delta|}{1-m}}$$

Newton-Raphson Method:

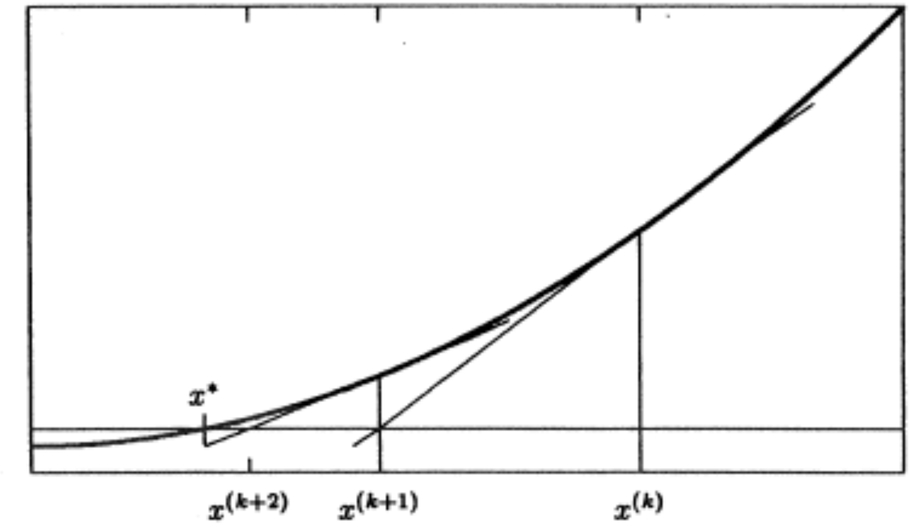
$$\boxed{F(x)=0} \longrightarrow \boxed{f(x)=x}$$

is obtained by setting:

$$\boxed{f(x)=x-\frac{F(x)}{F'(x)}}$$

This gives the iteration rule:

$$\boxed{x_{t+1}=x_t-\frac{F(x_t)}{F'(x_t)}}$$



Newton-Raphson iteration

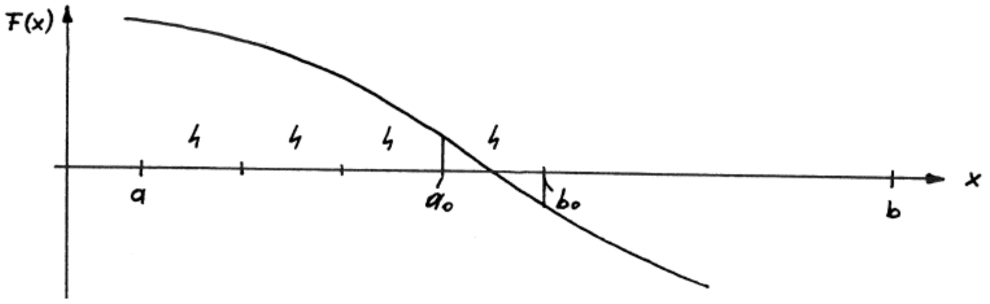
(Tangent method).

This week(26/11/2013)

- Gross search for the zeroes of a function.
- **The Newton-Raphson method: a program.**
- Other methods for finding zeroes of a function: False position (*regula falsi*).
- **Bisection** method (nested intervals).

Gross Search:

The Newton-Raphson (and the bisection) method permit to find a zero **in a given interval**, **once we know there is a zero**.



In practice, before using any methods to find zeroes, one always performs first a **gross search.**

1) Choose a large interval [a,b].

2) Divide it into *n* intervals of width h (stepsize).

3) Compute $s=f(x_{min})xf(x_{max})$ for every interval.

4) If *s≤0*, the interval contains a zero, which can be located with a N-R or bissection method.

Newton-Raphson program

With preliminary gross search

Program workflow:

Gross Search

Given h; $[a,b]$; $h = \frac{a-b}{N}$

Calculate $f(x_i)f(x_i+h)$, $i = 1,\ldots,N$; $x_1 = a$

if $f(x_i)f(x_i+h) \le 0$

start Newton-Raphson with $x_1 = x_i$, $x_2 = x_i + h$

Locate "roughly" the zeroes in the interval [a,b]

Newton-Raphson

$$x_{t+1} = x_t - \frac{F(x_t)}{F'(x_t)}$$

Error 2: no convergence within Jmax iterations

Error 3: Jumped out of brackets.

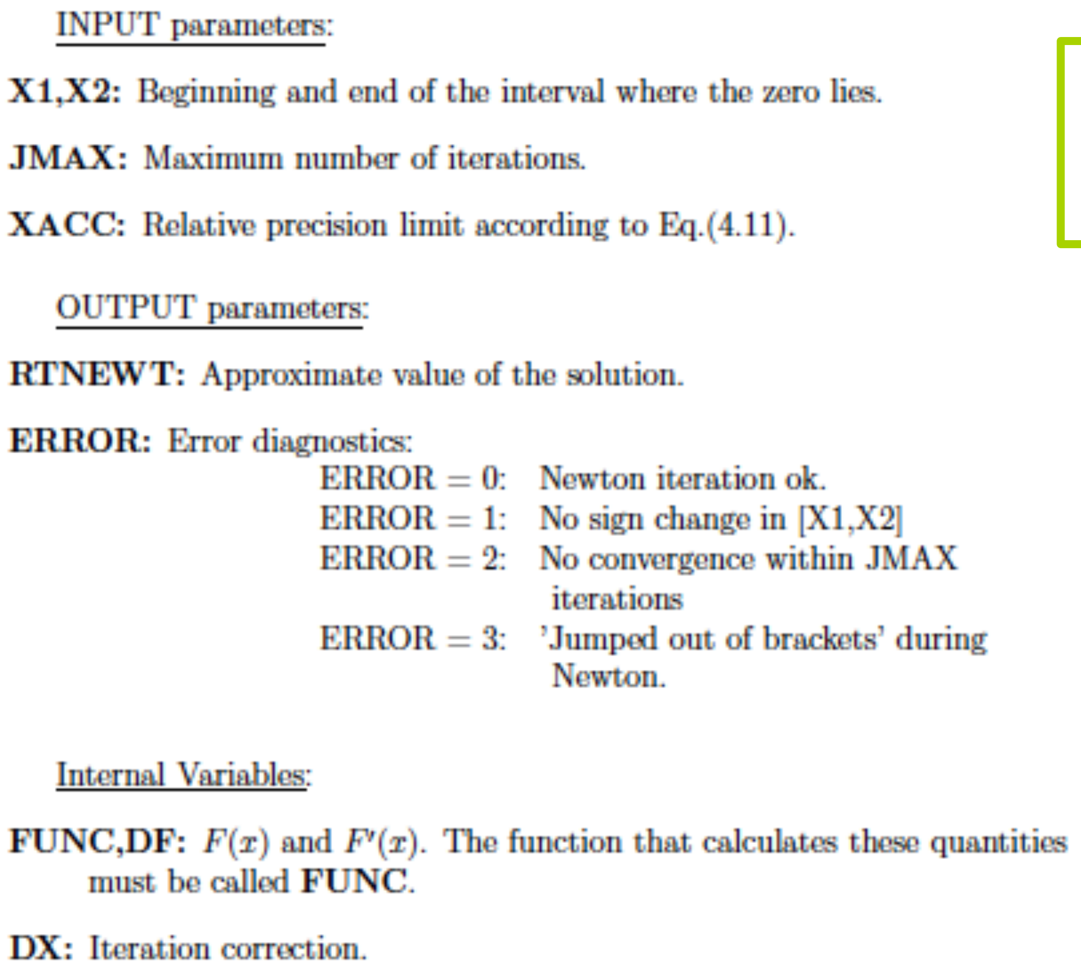
If the method converges, locate the zeroes with an accuracy Δx.

Funct

External function: evaluates F and its first derivative for N-R.

RTNEWT(X1,X2,JMAX,XACC,ERROR)

$$x_{t+1} = x_t - \frac{F(x_t)}{F'(x_t)}$$



INPUT parameters:

X1,X2: Beginning and end of the interval where the zero lies.

JMAX: Maximum number of iterations.

XACC: Relative precision limit according to Eq.(4.11).

OUTPUT parameters:

RTNEWT: Approximate value of the solution.

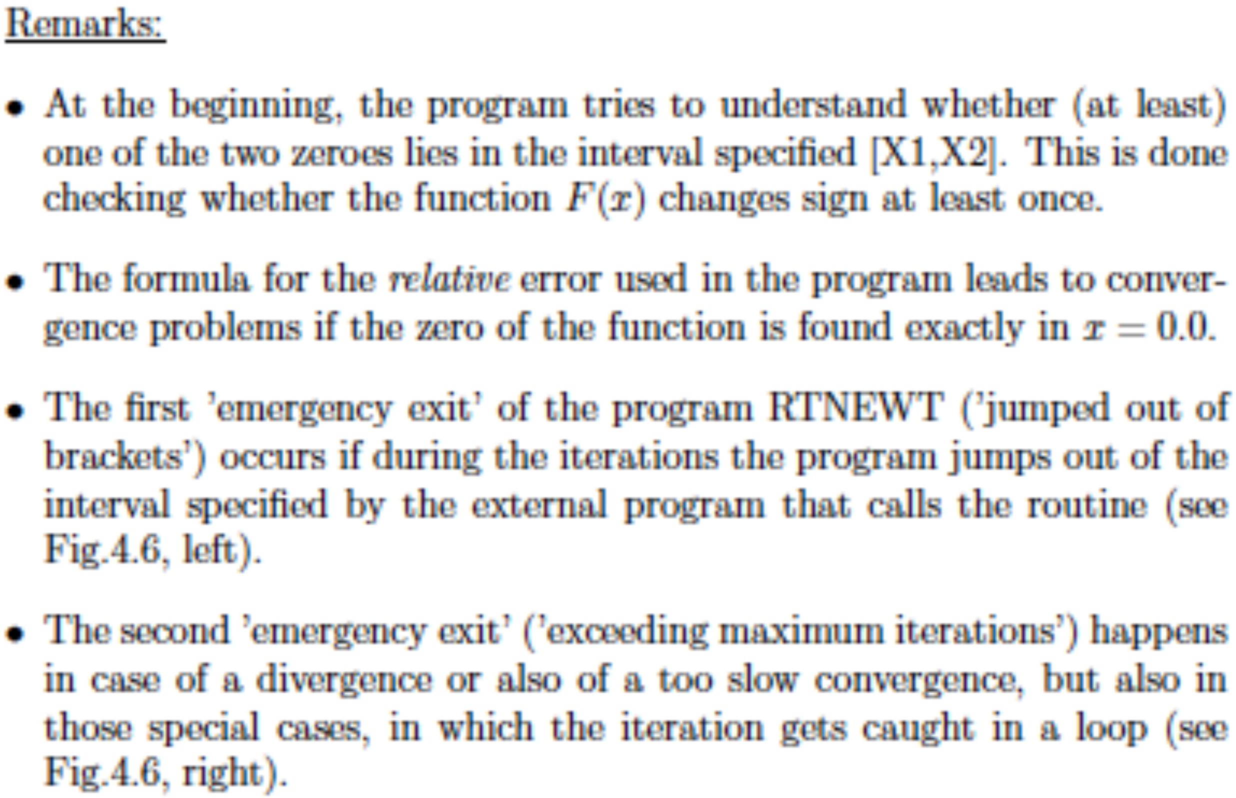
ERROR: Error diagnostics:

ERROR = 0: Newton iteration ok.
ERROR = 1: No sign change in [X1,X2]
ERROR = 2: No convergence within JMAX iterations
ERROR = 3: 'Jumped out of brackets' during Newton.

Internal Variables:

FUNC,DF: $F(x)$ and $F'(x)$. The function that calculates these quantities must be called **FUNC**.

DX: Iteration correction.



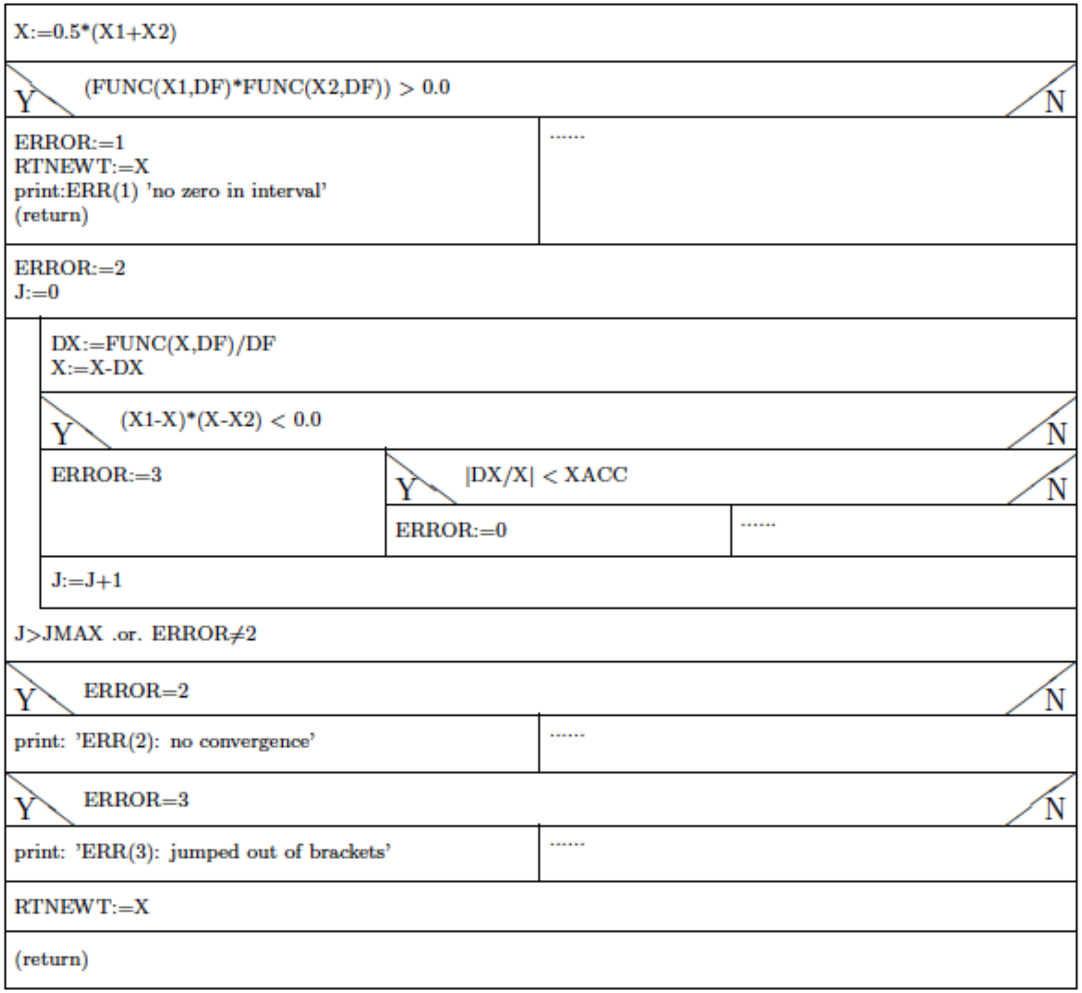
Remarks:

- At the beginning, the program tries to understand whether (at least) one of the two zeroes lies in the interval specified [X1,X2]. This is done checking whether the function $F(x)$ changes sign at least once.

- The formula for the *relative* error used in the program leads to convergence problems if the zero of the function is found exactly in $x = 0.0$.

- The first ’emergency exit’ of the program RTNEWT (’jumped out of brackets’) occurs if during the iterations the program jumps out of the interval specified by the external program that calls the routine (see Fig.4.6, left).

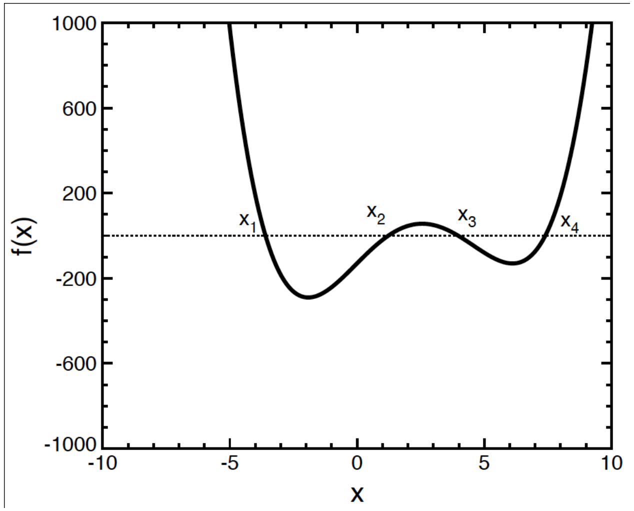
- The second ’emergency exit’ (’exceeding maximum iterations’) happens in case of a divergence or also of a too slow convergence, but also in those special cases, in which the iteration gets caught in a loop (see Fig.4.6, right).



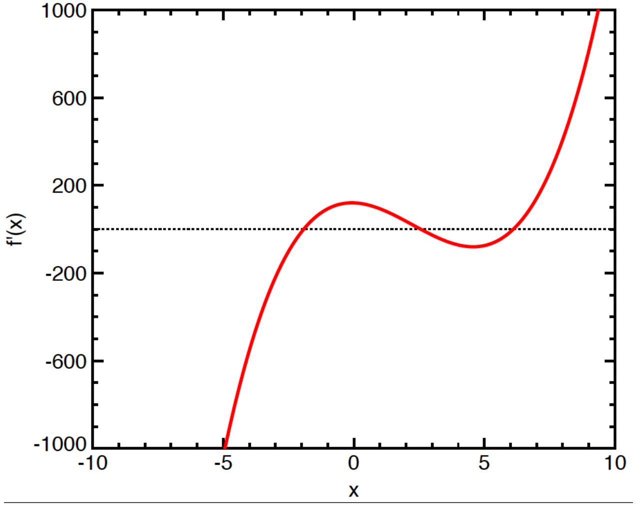
```
X:=0.5*(X1+X2)
Y   (FUNC(X1,DF)*FUNC(X2,DF)) > 0.0   N
    Y:
        ERROR:=1
        RTNEWT:=X
        print:ERR(1) 'no zero in interval'
        (return)
    N:
        ......
ERROR:=2
J:=0
    DX:=FUNC(X,DF)/DF
    X:=X-DX
    Y   (X1-X)*(X-X2) < 0.0   N
        Y:
            ERROR:=3
        N:
            Y   |DX/X| < XACC   N
                Y:
                    ERROR:=0
                N:
                    ......
    J:=J+1
J>JMAX .or. ERROR≠2
Y   ERROR=2   N
    Y:
        print: 'ERR(2): no convergence'
    N:
        ......
Y   ERROR=3   N
    Y:
        print: 'ERR(3): jumped out of brackets'
    N:
        ......
RTNEWT:=X
(return)
```

Implementation of gross search + Newton-Raphson method

```
#include <stdio.h>
#include <math.h>

double func(double x,double *df)
// This function calculates the function values as well as
// the values of the first derivative for the test example 5.3.3.
{
  *df=4*pow(x,3)-27*pow(x,2)-4*x+120;
  return pow(x,4)-9*pow(x,3)-2*pow(x,2)+120*x-130;
}
```

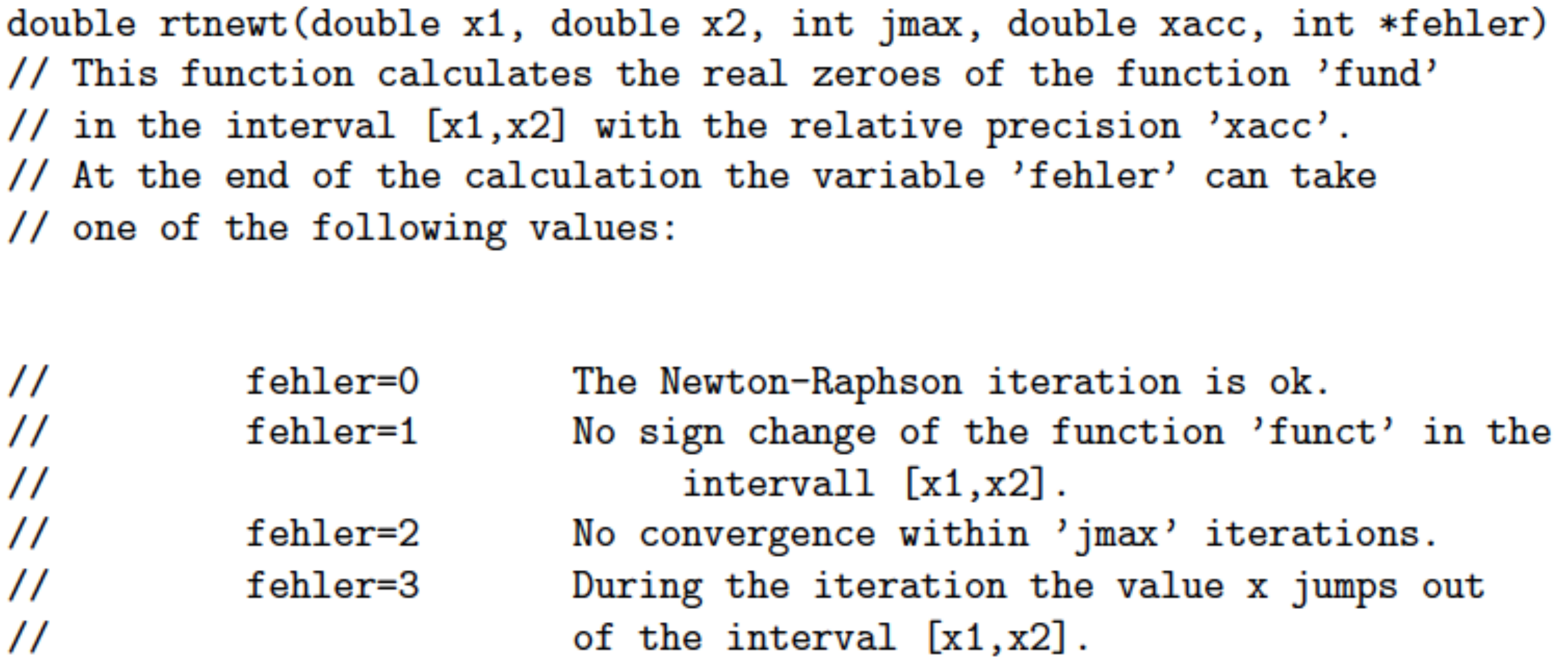


$$F(x)=x^4-9x^3-2x^2+120x-130$$



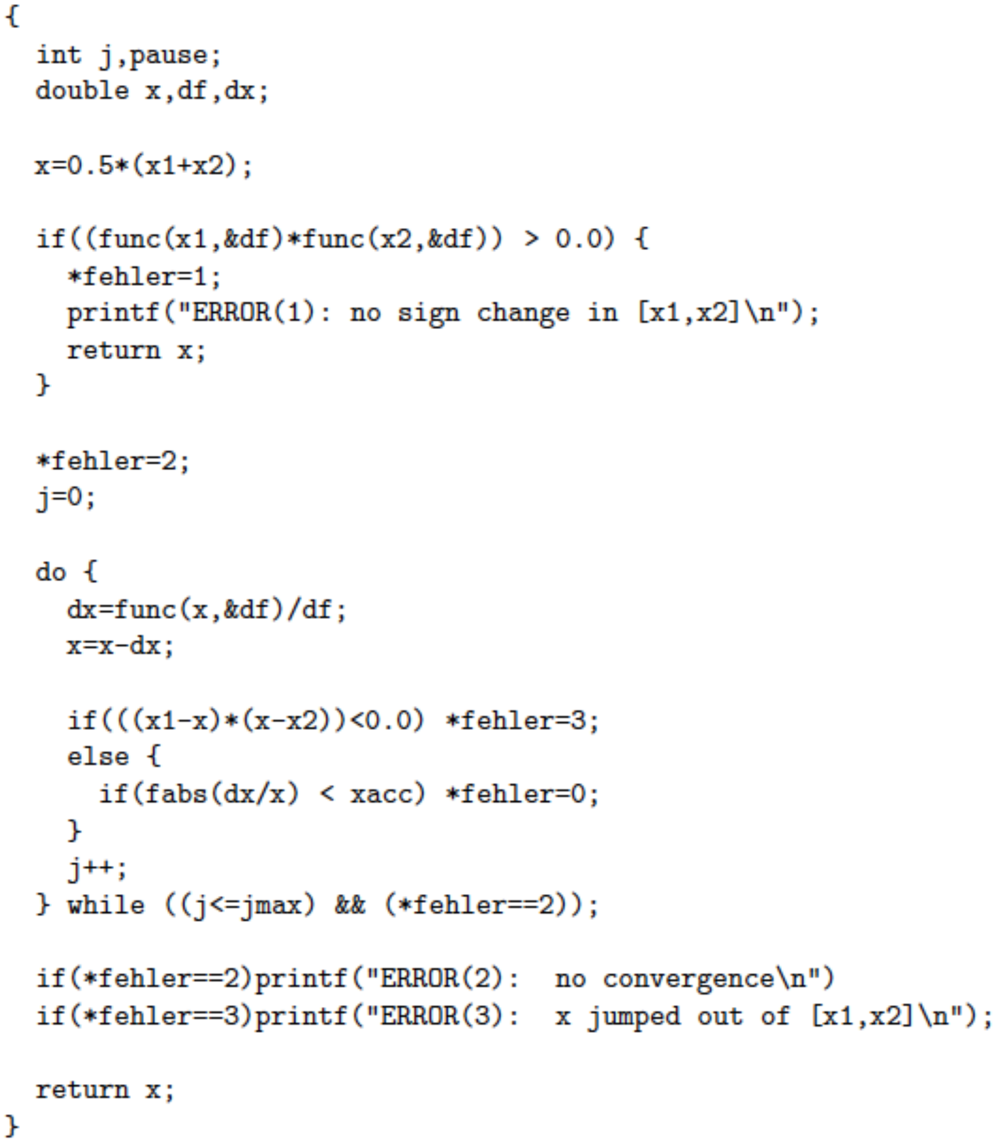
$$F'(x)=4x^3-27x^2-4x+120$$

Newton-Raphson: Exit conditions

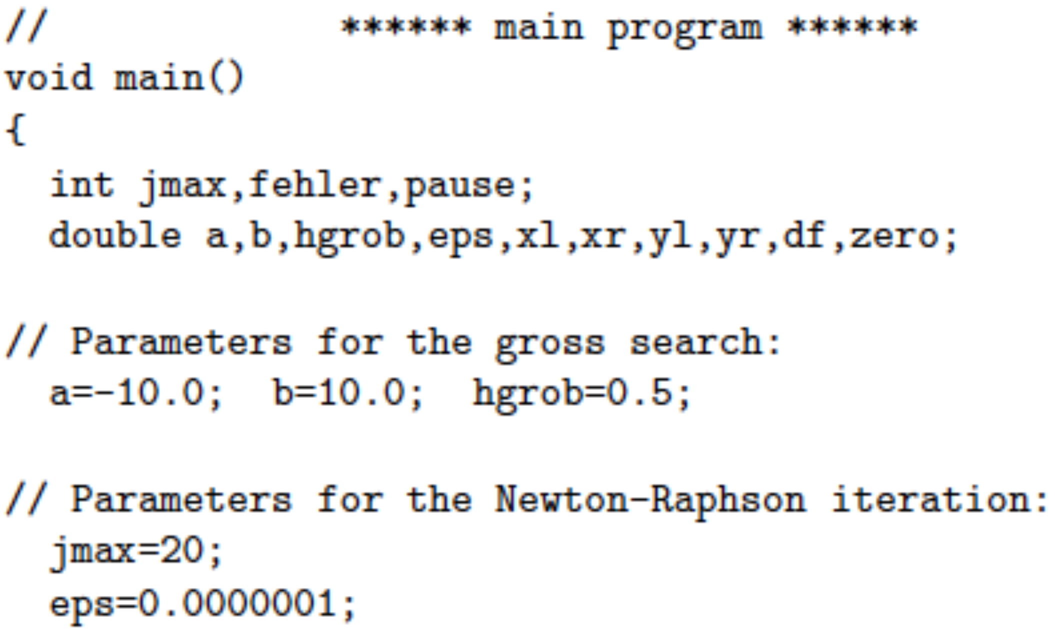


```
double rtnewt(double x1, double x2, int jmax, double xacc, int *fehler)
// This function calculates the real zeroes of the function ’fund’
// in the interval [x1,x2] with the relative precision ’xacc’.
// At the end of the calculation the variable ’fehler’ can take
// one of the following values:


//          fehler=0       The Newton-Raphson iteration is ok.
//          fehler=1       No sign change of the function ’funct’ in the
//                              intervall [x1,x2].
//          fehler=2       No convergence within ’jmax’ iterations.
//          fehler=3       During the iteration the value x jumps out
//                         of the interval [x1,x2].
```

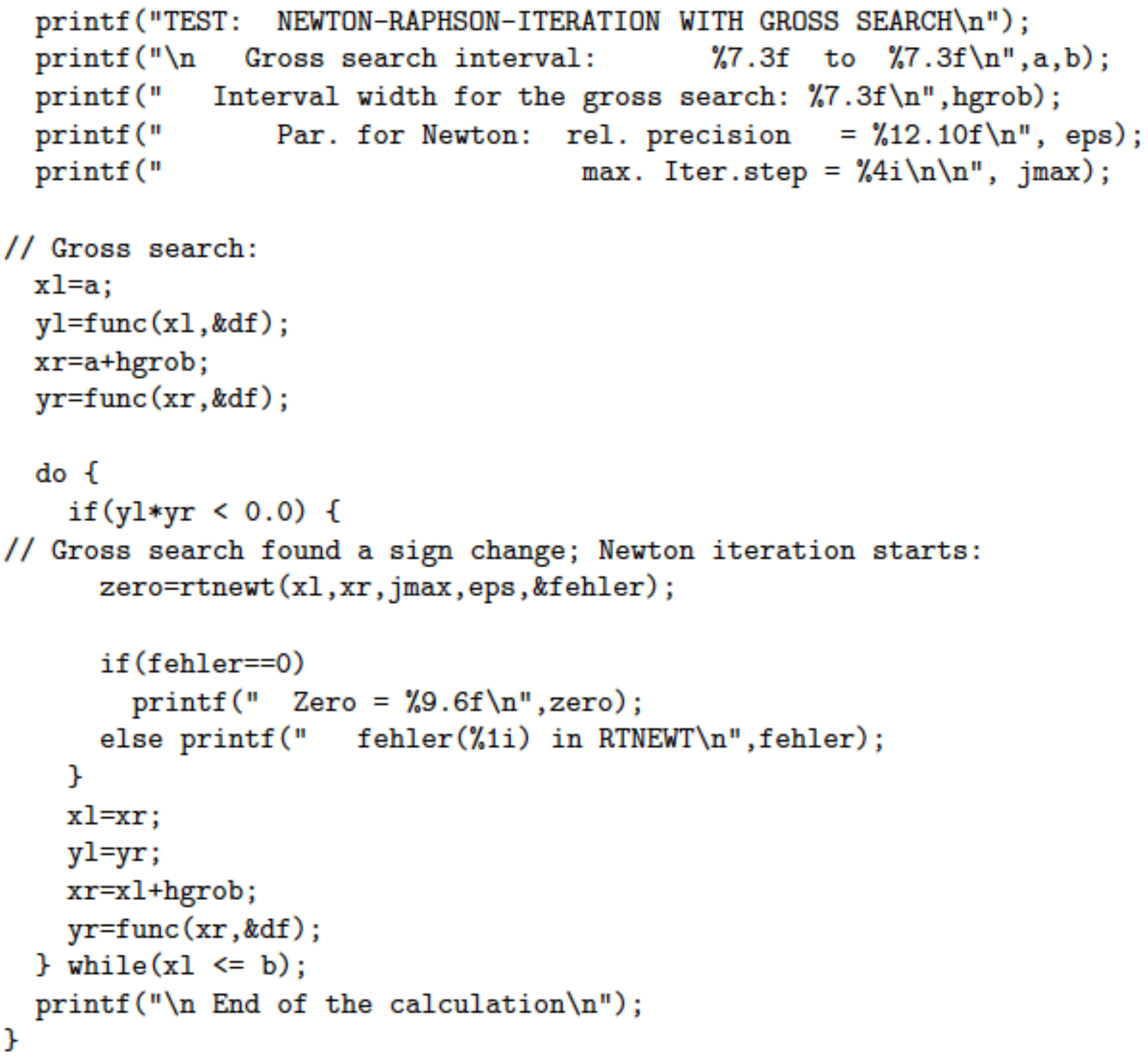
Newton-Raphson: main body



```
{
  int j,pause;
  double x,df,dx;

  x=0.5*(x1+x2);

  if((func(x1,&df)*func(x2,&df)) > 0.0) {
    *fehler=1;
    printf("ERROR(1): no sign change in [x1,x2]\n");
    return x;
  }

  *fehler=2;
  j=0;

  do {
    dx=func(x,&df)/df;
    x=x-dx;

    if(((x1-x)*(x-x2))<0.0) *fehler=3;
    else {
      if(fabs(dx/x) < xacc) *fehler=0;
    }
    j++;
  } while ((j<=jmax) && (*fehler==2));

  if(*fehler==2)printf("ERROR(2):  no convergence\n")
  if(*fehler==3)printf("ERROR(3):  x jumped out of [x1,x2]\n");

  return x;
}
```



```
//              ****** main program ******
void main()
{
  int jmax,fehler,pause;
  double a,b,hgrob,eps,xl,xr,yl,yr,df,zero;

// Parameters for the gross search:
  a=-10.0;  b=10.0;  hgrob=0.5;

// Parameters for the Newton-Raphson iteration:
  jmax=20;
  eps=0.0000001;
```



```
  printf("TEST:  NEWTON-RAPHSON-ITERATION WITH GROSS SEARCH\n");
  printf("\n   Gross search interval:       %7.3f  to  %7.3f\n",a,b);
  printf("   Interval width for the gross search: %7.3f\n",hgrob);
  printf("       Par. for Newton:  rel. precision   = %12.10f\n", eps);
  printf("                          max. Iter.step = %4i\n\n", jmax);

// Gross search:
  xl=a;
  yl=func(xl,&df);
  xr=a+hgrob;
  yr=func(xr,&df);

  do {
    if(yl*yr < 0.0) {
// Gross search found a sign change; Newton iteration starts:
      zero=rtnewt(xl,xr,jmax,eps,&fehler);

      if(fehler==0)
        printf("  Zero = %9.6f\n",zero);
      else printf("   fehler(%1i) in RTNEWT\n",fehler);
    }
    xl=xr;
    yl=yr;
    xr=xl+hgrob;
    yr=func(xr,&df);
  } while(xl <= b);
  printf("\n End of the calculation\n");
}
```

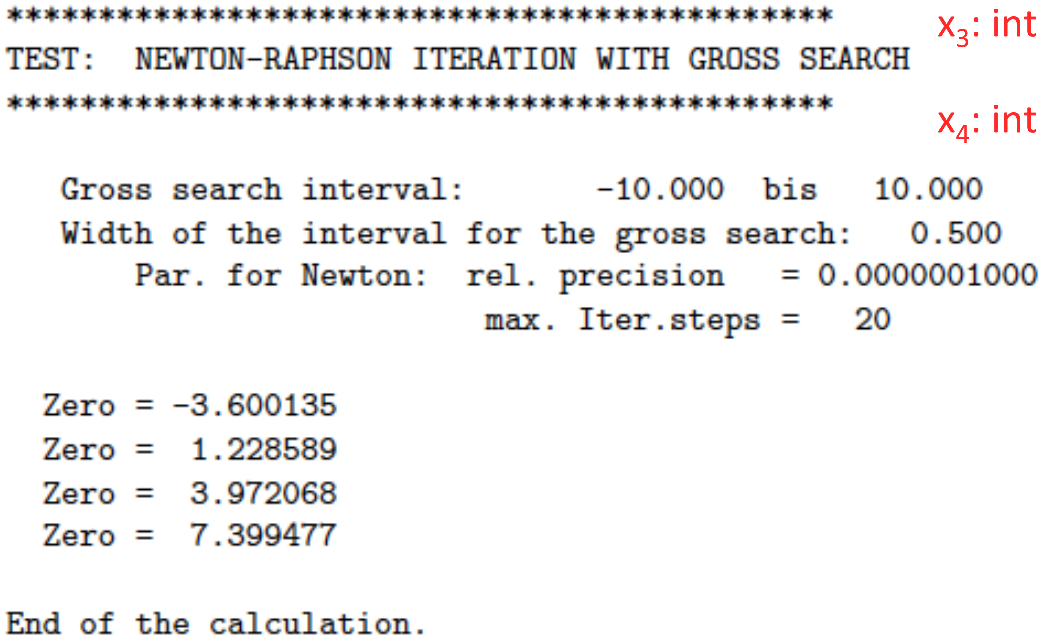
Results:

x_1: int = 13

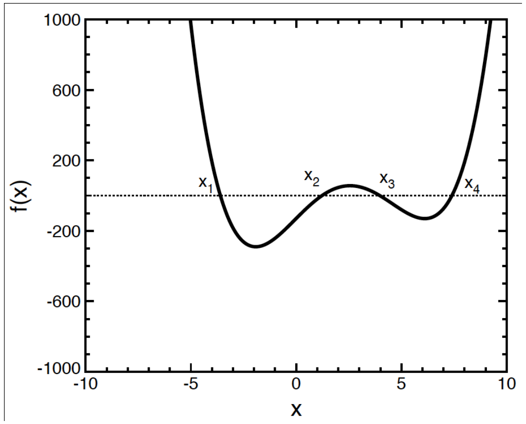
x_2: int = 22

x_3: int = 28

x_4: int = 35

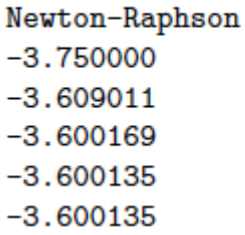


```
*******************************************
TEST:  NEWTON-RAPHSON ITERATION WITH GROSS SEARCH
*******************************************

   Gross search interval:        -10.000   bis    10.000
   Width of the interval for the gross search:    0.500
       Par. for Newton:  rel. precision    = 0.0000001000
                                max. Iter.steps =    20

  Zero = -3.600135
  Zero =  1.228589
  Zero =  3.972068
  Zero =  7.399477

End of the calculation.
```



$$F(x)=x^4-9x^3-2x^2+120x-130$$

Approaching x_1…

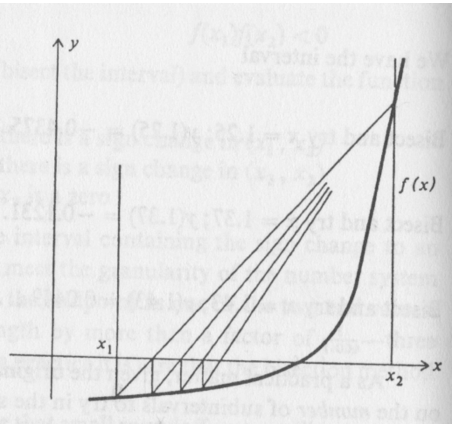


```
Newton-Raphson
-3.750000
-3.609011
-3.600169
-3.600135
-3.600135
```

Other methods for zeroes: False position (*regula falsi*).

It is used to find the zero of a function in an interval [*a*,*b*], **in which the function changes sign**.

The real curve - F(x) - is replaced by the straight line through the two extrema of the interval.



The line between *a* and *b* is given by: $y(x)=\dfrac{f(b)-f(a)}{b-a}(x-a)+f(a)$

And the zero is: $\bar{x}=a-f(a)\dfrac{(b-a)}{f(b)-f(a)}=\dfrac{af(b)-bf(a)}{f(b)-f(a)}$

$$x_{t+1}=x_t-\frac{f(x_t)}{f(x_t)-f(x_{t-1})}(x_t-x_{t-1})$$

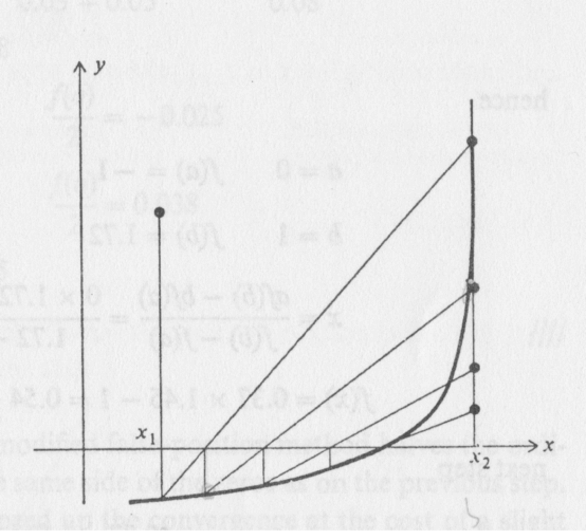
Disadvantages: the zero is approached by one side -> The method may be very slow!

Modified false position:

In the modified false position method, the zero can be approached by left or right. One of the two ends is kept fixed (a or b) and the function at the end which is kept fixed is divided by half. This speeds up the calculation.

Compute the first zero with false position method: $\bar{x} = \dfrac{af(b) - bf(a)}{f(b) - f(a)}$

$$\text{Compute} \quad f(\bar{x})f(a) \begin{cases} < 0 \text{ then: } \begin{cases} x \rightarrow b \\ f(x) \rightarrow f(b) \\ \dfrac{f(a)}{2} \rightarrow f(a) \end{cases} \\ = 0 \text{ exit!} \\ > 0 \text{ then: } \begin{cases} x \rightarrow a \\ f(x) \rightarrow f(a) \\ \dfrac{f(b)}{2} \rightarrow f(b) \end{cases} \end{cases}$$



Iterate!!!

The bisection method (nested intervals):

Given an interval x_1, x_2 in which the function F(x) changes sign – $f(x_1)f(x_2)<0$:

1) The interval is divided in half, picking the midpoint x_3

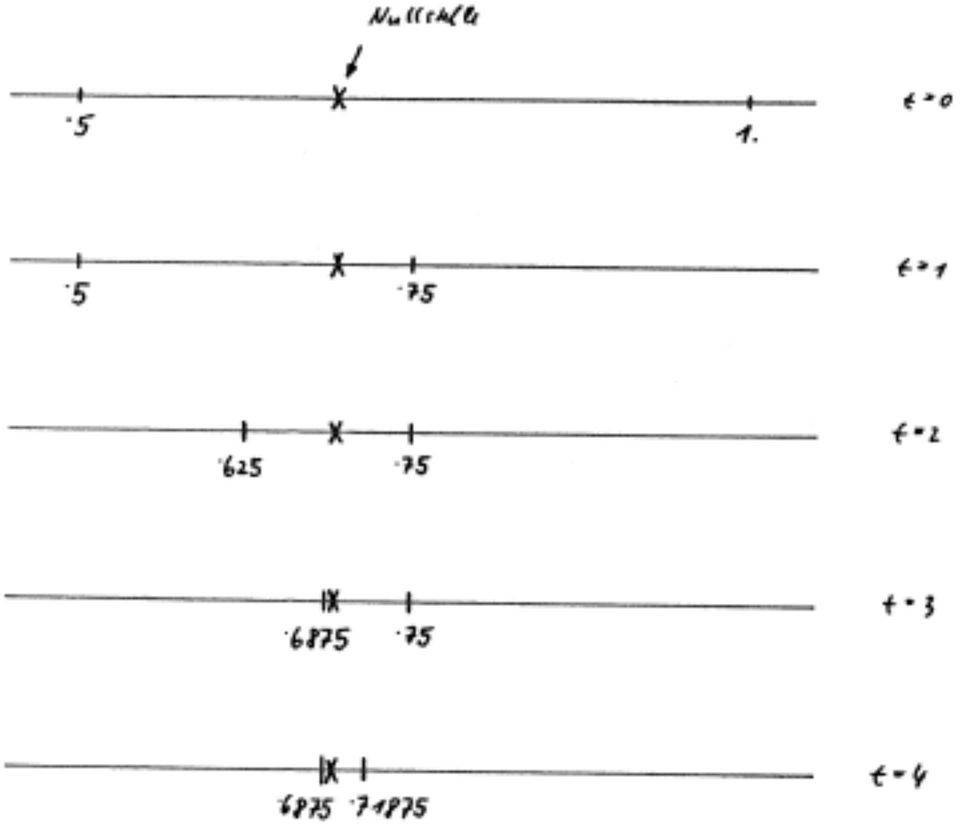
2) Evaluate the product $f(x_1)f(x_3)$

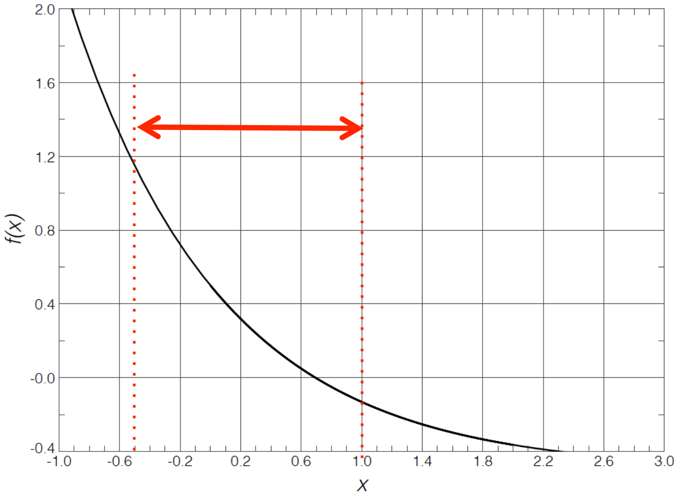
$$f(x_1)f(x_3)\begin{cases} <0 \text{ the zero is in } \mathrm{x}_1,\mathrm{x}_3 \\ >0 \text{ the zero is in } \mathrm{x}_2,\, \mathrm{x}_3 \\ =0 \text{ the zero is } \mathrm{x}_3 \end{cases}$$

3) Iterate the method using the new interval, until the zero is found.

The bisection method (nested intervals):

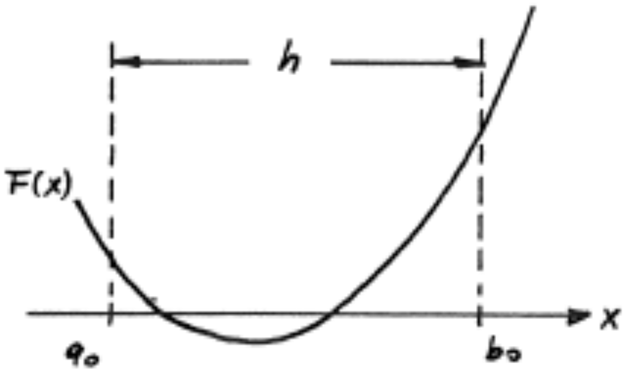
Example: $F(x) = e^{-x} - \frac{1}{2}$





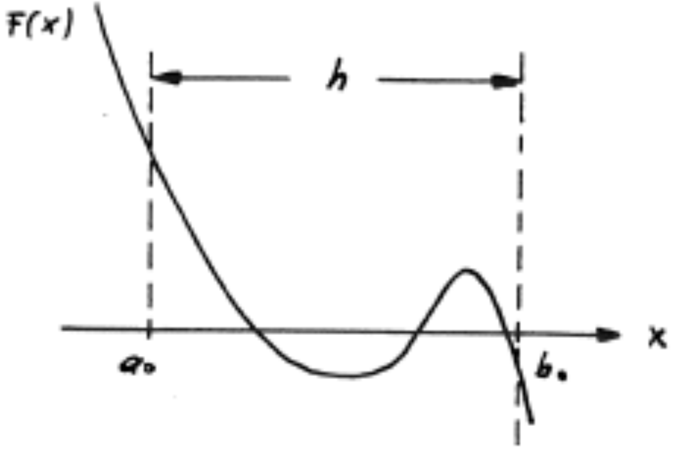
N.B. the size of the interval (resolution) scales as: $1/2^n$. The precision can be estimated ***a priori***!

Problems:



Interval h contains an even number of zeroes.

$$f(a_0)f(b_0)=0$$

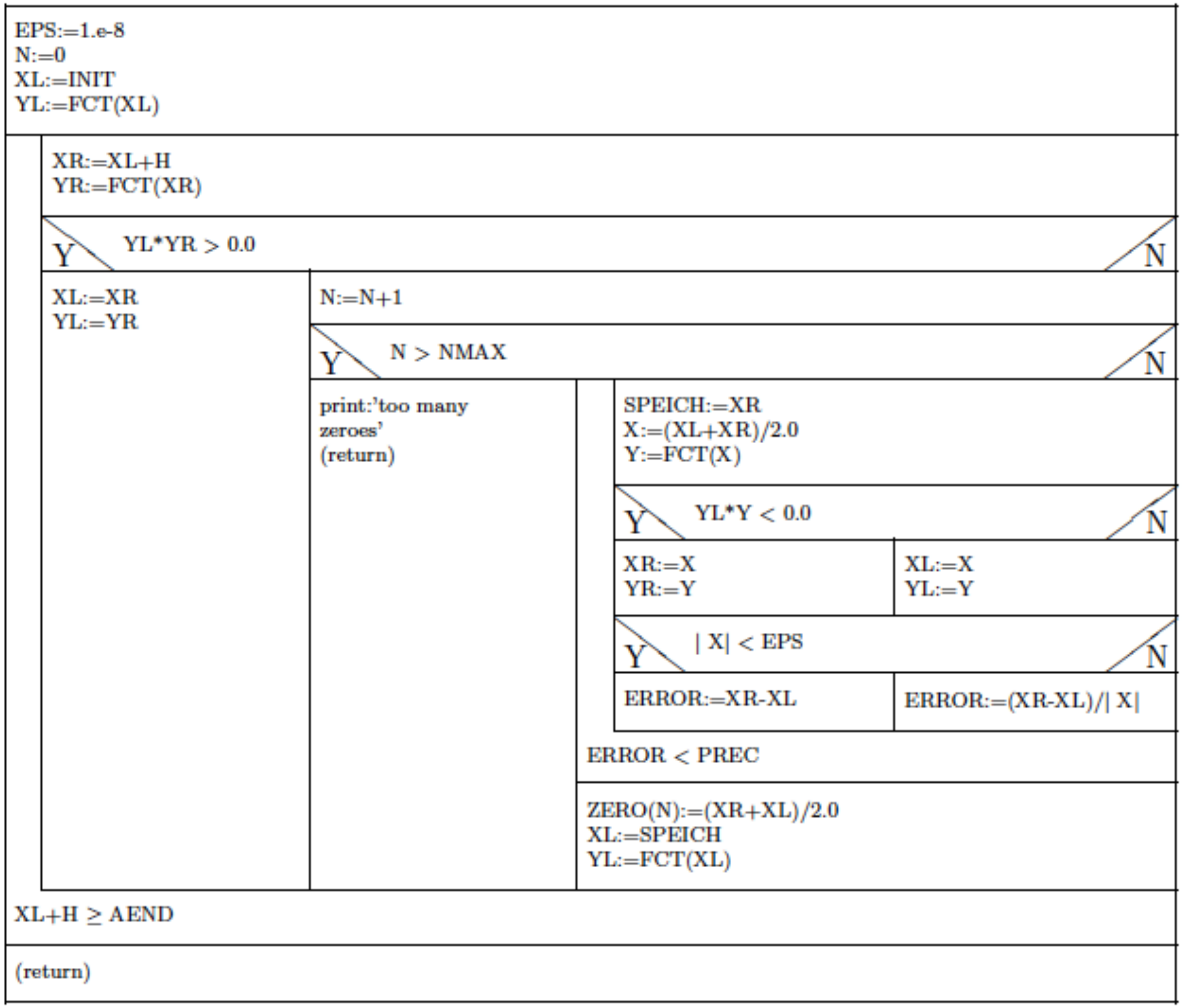


Interval h contains an odd number of zeroes (n>1).

$$f(a_0)f(b_0)<0$$

h limits the resolution of the bisection method! Two zeroes can be discriminated only if their distance is larger than h_{res}!

Structure chart15 — INTSCH(INIT,AEND,H,PREC,NMAX,ZERO,N)



EPS:=1.e-8
N:=0
XL:=INIT
YL:=FCT(XL)

XR:=XL+H
YR:=FCT(XR)

Y YL*YR > 0.0 N

XL:=XR
YL:=YR

N:=N+1

Y N > NMAX N

print:'too many zeroes'
(return)

SPEICH:=XR
X:=(XL+XR)/2.0
Y:=FCT(X)

Y YL*Y < 0.0 N

XR:=X
YR:=Y

XL:=X
YL:=Y

Y | X| < EPS N

ERROR:=XR-XL

ERROR:=(XR-XL)/| X|

ERROR < PREC

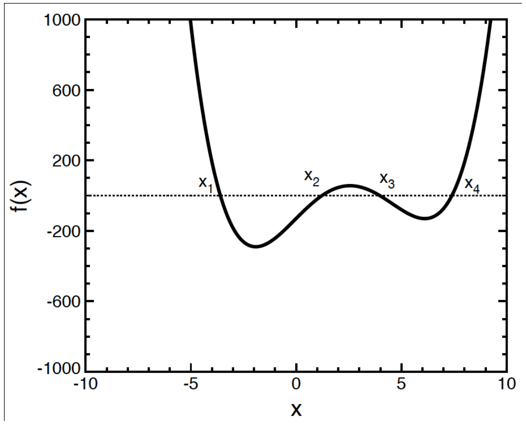
ZERO(N):=(XR+XL)/2.0
XL:=SPEICH
YL:=FCT(XL)

XL+H ≥ AEND

(return)

Performance:

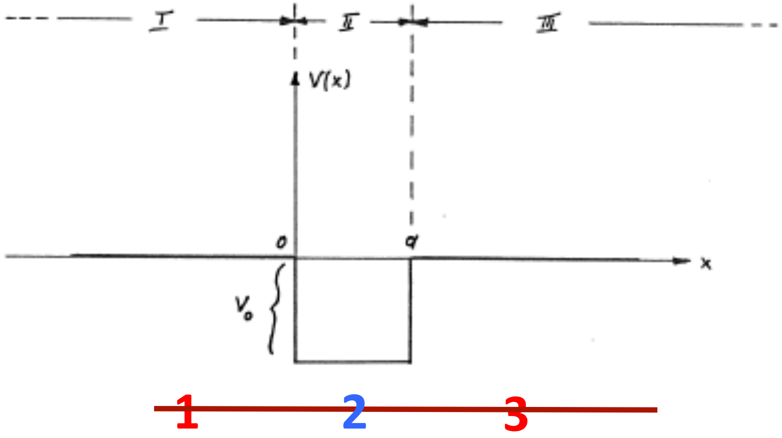
```
Newton-Raphson     Int.nesting
-3.750000            -3.750000
-3.609011            -3.625000
-3.600169            -3.562500
-3.600135            -3.593750
-3.600135            -3.609375
                     -3.601562
                     -3.597656
                     -3.599609
                     -3.600586
                     -3.600098
                     -3.600342
                     -3.600220
                     -3.600159
                     -3.600128
                     -3.600143
                     -3.600136
                     -3.600132
                     -3.600134
                     -3.600135
                     -3.600135
                     -3.600135
                     -3.600135
```



Newton-Raphson is faster, but requires the calculation of the first derivative!

Example: Schroedinger's equation for a 1-d potential well:

$$-\frac{\hbar^2}{2m}\psi''(x)+V(x)\psi(x)=E\psi(x)$$



$$V(x)=\begin{cases} 0, & \text{for } -\infty<x\le 0 \quad 1 \\ -V_0, & \text{for } 0<x\le a \quad 2 \\ 0, & \text{for } x>a \quad 3 \end{cases}$$

Boundary conditions:

$$\psi(+\infty)\rightarrow 0, \qquad \psi(-\infty)\rightarrow 0.$$

Solutions:

$$\psi_1(x)=A_1\exp(\sqrt{-E}x) \qquad 1$$

$$\psi_2(x)=A_2\sin(x\sqrt{E+V_0})+B_2\cos(x\sqrt{E+V_0}) \qquad 2$$

$$\psi_3(x)=B_3\exp(-\sqrt{-E}x) \qquad 3$$

Matching conditions:

$$\psi_1(0) = \psi_2(0)$$
$$\psi'_1(0) = \psi'_2(0)$$
$$\psi_2(a) = \psi_3(a)$$
$$\psi'_2(a) = \psi'_3(a)$$

Which determine the linear system M(E)**x**=**0**:

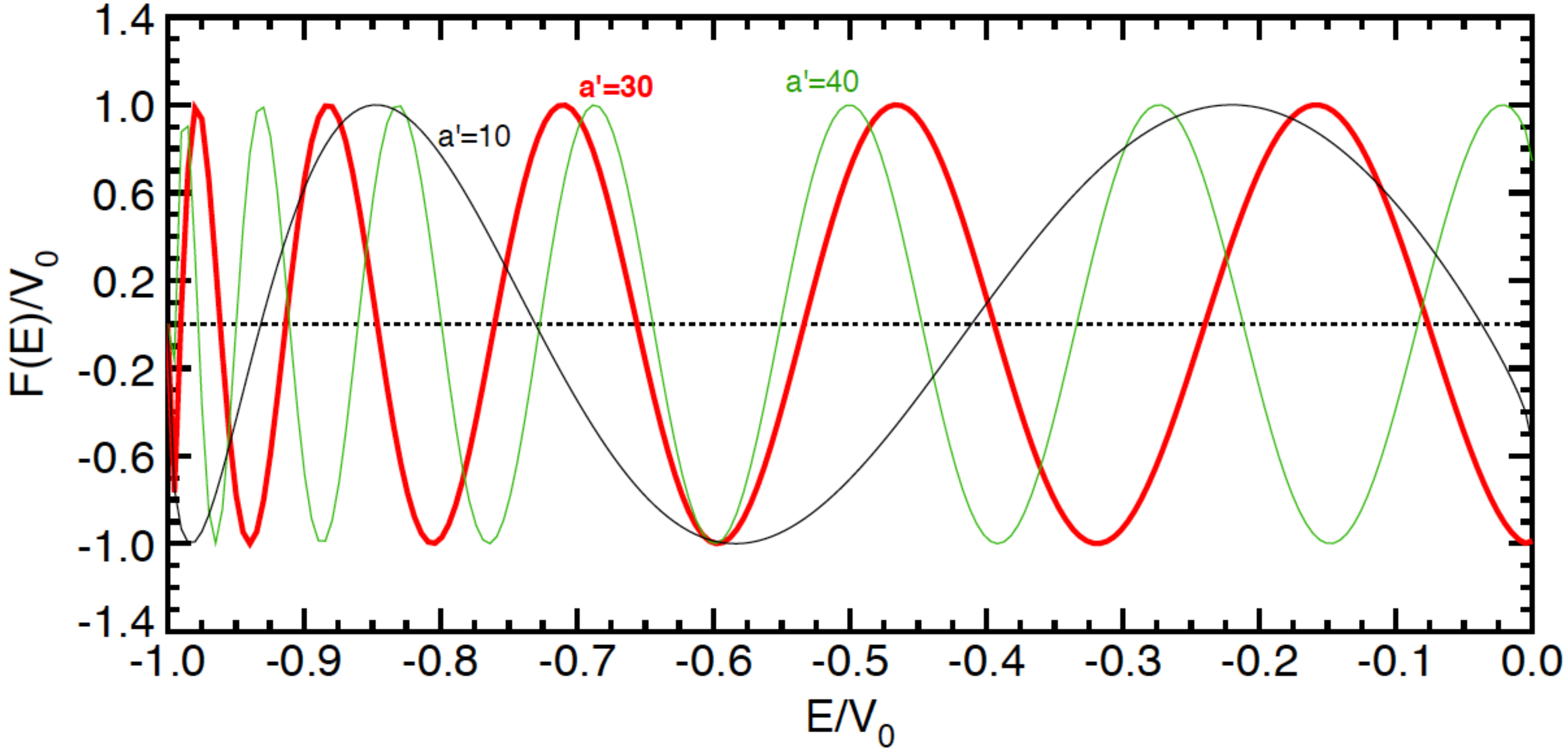
$$\begin{pmatrix} 1 & 0 & -1 & 0 \\ \sqrt{-E} & -\kappa & 0 & 0 \\ 0 & \sin(\kappa a) & \cos(\kappa a) & -e^{-\sqrt{-E}a} \\ 0 & \kappa\cos(\kappa a) & -\kappa\sin(\kappa a) & \sqrt{-E}e^{-\sqrt{-E}a} \end{pmatrix} \begin{pmatrix} A_1 \\ A_2 \\ B_2 \\ B_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\det(M(E)) = -e^{-a\sqrt{-E}}(V_0 + 2E)\sin(a\sqrt{E+V_0}) - 2\sqrt{-E(E+V_0)}\cos(a\sqrt{E+V_0})$$

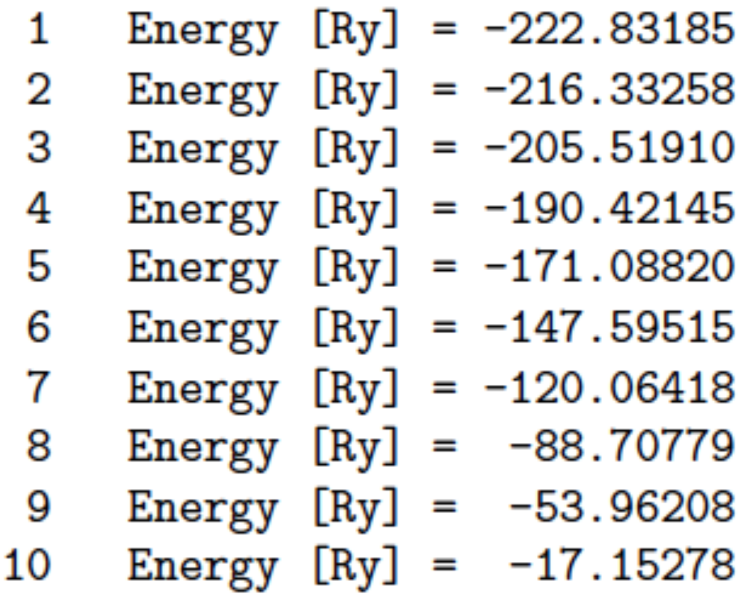
In practice, we have to determine the zeroes of the function:

$$F(E) = (V_0 + 2E)\sin(a\sqrt{E+V_0}) - 2\sqrt{-E(E+V_0)}\cos(a\sqrt{E+V_0})$$

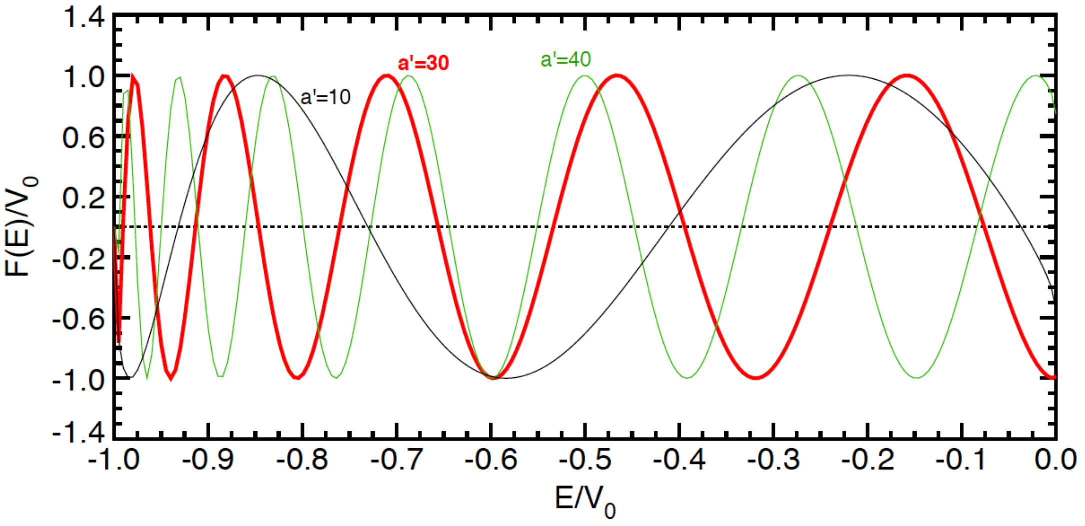
$$F(E)/V_0 = (1+2x)\sin(a'\sqrt{x+1}) - 2\sqrt{-x(x+1)}\cos(a'\sqrt{x+1}), \quad \text{x=E/V}_0, \quad a' = a\sqrt{V_0}$$



For a = 2 B and V_0=225 Ryd (a’=30):



```
 1    Energy [Ry] = -222.83185
 2    Energy [Ry] = -216.33258
 3    Energy [Ry] = -205.51910
 4    Energy [Ry] = -190.42145
 5    Energy [Ry] = -171.08820
 6    Energy [Ry] = -147.59515
 7    Energy [Ry] = -120.06418
 8    Energy [Ry] =  -88.70779
 9    Energy [Ry] =  -53.96208
10    Energy [Ry] =  -17.15278
```



This week(26/11/2013)

- **The Newton-Raphson method: a program.**
- Gross search for the zeroes of a function.
- Other methods for finding zeroes of a function: False position (*regula falsi*).
- **Bisection** method (nested intervals).