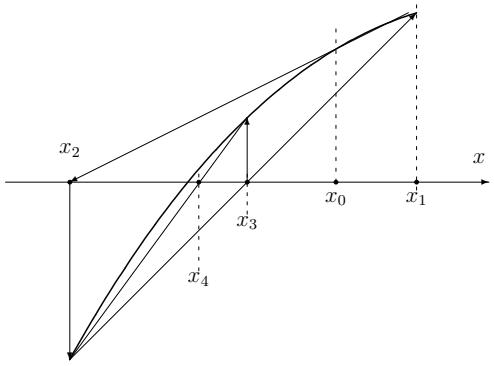
Reševanje nelinearnih enačb, 2.del

Navedene strani so iz knjige *Octave z uvodom v numerične metode.*

Sekantna metoda

Poleg funkcijskega predpisa $f(x)$ je potrebno podati še krajišči začetnega intervala x_0 in x_1, natančnost ϵ ter maksimalno število iteracij n. Pri tej metodi ni potrebno, da se rešitev enačbe $f(x) = 0$ nahaja na začetnem intervalu $[x_0, x_1]$, zato se lahko zgodi, da postopek ne konvergira k rešitvi. Da se izognemo neskončni zanki, podamo na začetku maksimalno število iteracij n. (str. 146-148)

Na sliki je prikazanih nekaj iteracij. Najprej izračunamo funkcijski vrednosti $f_i = f(x_i)$, $i = 0, 1$. Nato potegnemo sekanto skozi točki (x_0, f_0) in (x_1, f_1). Tam, kjer ta premica seka x-os, dobimo vrednost x_2. Izračunamo funkcijsko vrednost $f_2 = f(x_2)$. Postopek ponovimo za točki x_1 in x_2 in dobimo približek x_3. Tako nadaljujemo, dokler dolžina trenutnega intervala ni dovolj majhna.



Prednosti in slabosti metode:

- Konvergenca je boljša kot pri metodi bisekcije.
- Ni potrebno določiti intervala, ki objema iskano ničlo.
- Lahko se zgodi, da izberemo začetna približka, pri katerih metoda ne konvergira.

Algoritem:

- Če je $|f(x_0)| < |f(x_1)|$, zamenjamo vlogi x_0 in x_1. S tem dosežemo, da je funkcijska vrednost v x_1 absolutno manjša kot v x_0.
- Vstopimo v zanko, ki ima lahko največ n prehodov.
- Izračunamo približek
$$x_2 = x_1 - f(x_1)\frac{x_0 - x_1}{f(x_0) - f(x_1)}.$$
- Postavimo $x_0 = x_1$ in $x_1 = x_2$.
- Če je $|f(x_2)| < \epsilon$, zapustimo zanko in vrnemo rešitev x_2.
- Sicer ponovno vstopimo v zanko.

Algoritem zapišemo v proceduro `sekantna.m`.

```
% [c,st] = sekantna(f,a,b,e,n)
% Sekantna metoda za resevanje enacbe f(x)=0 na intervalu [a,b].
% Vhod:
% f : funkcija
% a, b : krajisci intervala
% e : natancnost
% n : maksimalno stevilo iteracij
% Izhod:
% c : resitev enacbe
% st : stevec stevila iteracij
function [c,st] = sekantna(f,a,b,e,n)
  % izracun funkcijskih vrednosti
  fa = f(a);
  fb = f(b);
  % zamenjava krajisc, ce je |f(a)|<|f(b)|
  if abs(fa) < abs(fb)
    tmp = a; a = b; b = tmp;
    tmp = fa; fa = fb; fb = tmp;
  end
  st = 0;
  % izvedemo najvec n iteracij
  for k=1:n
    % izracun nove tocke
    c = b-fb*(a-b)/(fa-fb);
    fc = f(c);
    % zamenjava krajisc
    a = b; fa = fb;
    b = c; fb = fc;
    % povecanje stevila iteracij
    st = st+1;
    % ce je vrednost funkcije v novi tocki dovolj blizu 0, koncamo
    if abs(fc) < e, return, end;
  end;
```

Metoda regula falsi

Poleg funkcijskega predpisa $f(x)$ je potrebno podati še krajišči začetnega intervala x_0 in x_1 ter natančnost ϵ. Začetni interval moramo izbrati tako, da bosta funkcijski vrednosti $f_i = f(x_i)$, $i = 0, 1$ v krajiščih različnega predznaka. (str. 149-151)

Prednosti in slabosti metode:

- Za monotono funkcijo na intervalu $[x_0, x_1]$ metoda vedno ponudi pričakovano rešitev.
- Konvergenca je slabša kot pri sekantni metodi.

Algoritem:

- Pred vstopom v zanko preverimo, ali sta funkcijski vrednosti $f(x_0)$ in $f(x_1)$ v krajiščih začetnega intervala $[x_0, x_1]$ različnega predznaka. Če nista, javimo napako `error('regula falsi:  f(x0)*f(x1) >= 0')`.
- Vstopimo v zanko.

- Izračunamo približek
$$x_2 = x_1 - f(x_1)\frac{x_0 - x_1}{f(x_0) - f(x_1)}.$$
- Če je $f(x_2)$ nasprotnega znaka kot $f(x_0)$, postavimo $x_1 = x_2$, sicer pa $x_0 = x_2$.
- Če je $|f(x_2)| < \epsilon$, zapustimo zanko in vrnemo rešitev x_2.
- Sicer ponovno vstopimo v zanko.

Algoritem zapišemo v proceduro `regula.m`.

```
% [c,st] = regula(f,a,b,e)
% Metoda regula falsi za resevanje enacbe f(x)=0 na intervalu [a,b].
% Vhod:
% f : funkcija
% a, b : krajisci intervala
% e : natancnost
% Izhod:
% c : resitev enacbe
% st : stevec stevila iteracij
function [c,st] = regula(f,a,b,e)
  % izracun funkcijskih vrednosti
  fa = f(a);
  fb = f(b);
  % preverba pogoja obstoja nicle
  if fa*fb >= 0, error('regula falsi: f(a)*f(b) >= 0'), end;
  st = 0;
  % zanka se izvaja, dokler zaustavitveni pogoj ni izpolnjen
  while 1
    % izracun nove tocke
    c = b-fb*(a-b)/(fa-fb);
    fc = f(c);
    % izbor levega ali desnega podintervala glede na veljavnost pogoja
    if fa*fc < 0
      b = c;
      fb = fc;
    else
      a = c;
      fa = fc;
    end;
    % povecanje stevila iteracij
    st = st+1;
    % zaustavitveni pogoj:
    % ce je vrednost funkcije v novi tocki dovolj blizu 0, koncamo
    if abs(fc) < e, return, end;
  end;
```