Yhtälön ratkaiseminen

Suora iterointi

Kirjoitetaan yhtälö muotoon $x = f(x)$. Ensin päätellään jollakin tavoin jokin alkuarvo x_0 ja sijoitetaan yhtälön oikealle puolelle, jolloin saadaan tarkennettu ratkaisu $x_1 = f(x_0)$. Jatketaan iterointia, kunnes ratkaisu ei enää muutu vaaditulla tarkkuudella:

$$
\begin{aligned}
x_0 &= ... \\
x_1 &= f(x_0), \\
x_2 &= f(x_1), \\
&... \\
x_{n+1} &= f(x_n), \\
&...
\end{aligned}
$$

Yhtälössä esiintyvistä tuntemattomista voidaan periaatteessa ratkaista mikä tahansa, joten yhtälö voidaan kirjoittaa vaaditussa muodossa useilla eri tavoilla. Esimerkiksi $x^5 - x - 1 = 0$ voidaan esittää muodossa $x = x^5 - 1$ tai $x = (1 + x)^{0.2}$.

Edellinen tuottaa alkuarvolla 0.5 lukusarjan

$$\begin{aligned} x_0 &= 0.5, \\ x_1 &= -1.853215, \\ x_2 &= -22.85895, \\ x_3 &= -6241392, \end{aligned}$$

joten lasketut likiarvot eivät suppene. Jälkimmäinen muoto sen sijaan antaa sarjan

$$\begin{aligned} x_0 &= 0.5, \\ x_1 &= 1.158242, \\ x_2 &= 1.166326, \\ x_3 &= 1.167199, \\ x_4 &= 1.167293, \\ x_5 &= 1.167303, \\ x_6 &= 1.167304, \\ x_7 &= 1.167304 \end{aligned}$$

Yleisesti ottaen polynomimuotoisista yhtälöistä kannattaa ratkaista korkein potenssi. Muunlaisille yhtälöille ei voi antaa samanlaistya ohjetta; yksinkertaisinta on vain kokeilla, mikä valinta tuottaa suppenevan lukujonon.

Välinpuolitusmenetelmä

Tarkastellaan edelleen yhtälöä

$$f(x) = x^5 - x - 1 = 0.$$

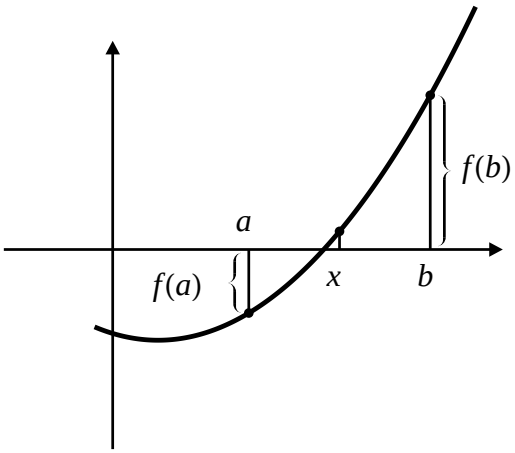
Koska f on jatkuva välillä $1 < x < 2$ ja $f(1) = -1 < 0$ ja $f(2) = 29 > 0$, yhtälöllä on varmasti ainakin yksi ratkaisu tällä välillä.

Puolitetaan tätä väliä, ja katsotaan kullakin askeleella, kummassa puoliskossa ratkaisun täytyy olla.

$f(1.5) = 5.09 > 0$, joten polynomi vaihtaa merkkiä välillä $1 < x < 1.5$.

$f(1.25) = 0.802 > 0$, joten merkin täytyy vaihtua välillä $1 < x < 1.25$.

Koska $f(1.125) = -0.323 < 0$, ratkaisun täytyy olla välillä $1.125 < x < 1.25$.



Välin puolittumisen vuoksi kullakin iteraatiokierroksella saadaan yksi bitti lisäinformaatiota. Yksi desimaaliluku vastaa suunnilleen kolmea bittiä, joten ratkaisun tarkentamiseen yhdellä merkitsevällä numerolla tarvitaan noin kolme iteraatiota.

Menetelmä suppenee varmasti (edellyttäen tietenkin, että yhtälössä esiintyvä funktio on jatkuva aluksi valitulla välillä). Suppeneminen on kuitenkin melko hidasta.

```
program halve
implicit none
real x
x = solve (1.0, 2.0)
write(*,*) x, f(x)

contains

real function f(x)
real, intent(in) :: x
  f = x**5 - x - 1
end function f

real function solve (xmin, xmax)
 real, intent(in) :: xmin, xmax
 real :: x1, y1, &     ! valin alkupaa
      x2, y2, &     ! valin loppupaa
      x0, y0       ! keskikohta

 x1=xmin; y1=f(x1)
 x2=xmax; y2=f(x2)

 x0=(x1+x2)/2 ;  y0=f(x0)

 do
  if (y1 * y0 < 0) then ! ratkaisu valilla (x1,x0)
    x2=x0
    y2=f(x2)
  else             ! ratkaisu valilla (x0,x2)
    x1=x0
    y1=f(x1)
  end if
  x0=(x1+x2)/2
  y0=f(x0)
  if (abs(y0) < 0.0001) then ! tarkkuus saavutettu
    solve=x0
    exit
  end if
 end do
end function solve
end program halve
```

Regula falsi

Yhtälön $f(x) = x^5 - x - 1 = 0$ tapauksessa $f(1)$ on itseisarvoltaan paljon pienempi kuin $f(2)$. Tuntuu luonnolliselta, että ratkaisun on oltava lähempänä ykköstä kuin kakkosta.

Jos väli on lyhyt, funktiota voidaan approksimoida suoralla. Jos välin päätepisteet ovat a ja b, suoran yhtälö on

$$y = f(a) + \frac{f(b) - f(a)}{b - a}(x - a).$$

Tämä leikkaa x-akselin pisteessä

$$x = a - f(a)\frac{b - a}{f(b) - f(a)}.$$

Iterointia voidaan jatkaa valitsemalla uudeksi väliksi $[a, x]$ tai $[x, b]$ riippuen siitä, kummalla välillä funktio vaihtaa merkkiään.

Iterointikaavan voi periaatteessa sieventää muotoon

$$x = \frac{af(b) - bf(a)}{f(b) - f(a)},$$

jossa on yksi liukulukuoperaatio vähemmän. Nimittäjässä esiintyy kuitenkin lähes yhtäsuurten lukujen vähennyslasku, joten tätä muotoa ei pidä koskaan käyttää ohjelmissa.

Sekanttimenetelmä

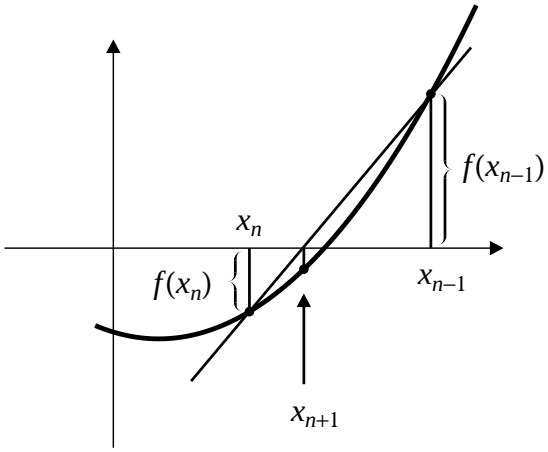
Edellisen menetelmän vikana on hitaanpuoleinen suppeneminen, kun on päästy lähelle juurta.

Jos kaksi viimeksi löydettyä approksimaatiota ovat x_n ja x_{n-1}, niiden määrittämän suoran yhtälö on

$$y = f(x_n) + \frac{f(x_{n-1}) - f(x_n)}{x_{n-1} - x_n}(x - x_n).$$

Tästä ratkaistu uusi nollakohta on

$$x = x_{n+1} = x_n - f(x_n)\frac{x_{n-1} - x_n}{f(x_{n-1}) - f(x_n)}.$$



Tämä menetelmä suppenee usein aikaisempia nopeammin.

Yritetään ratkaista yhtälöä

$$f(x) = \frac{1}{x^4} - 1 = 0.$$

Koska $f(1/2) > 0$ ja $f(2) < 0$, ratkaisun täytyy olla välillä $1/2 < x < 2$. Oletetaan, että tämän välin päätepisteet ovat kaksi ensimmäistä approksimaatiota.

$$\begin{aligned} x_0 &= 0.5000, \quad f(x_0) = 15, \\ x_1 &= 2.0000, \quad f(x_1) = -0.9375, \\ x_2 &= 1.9118, \quad f(x_2) = -0.9251, \\ x_3 &= -\,8.482. \end{aligned}$$

Peräkkäisten approksimaatioiden jono ei suppene ja sekanttimenetelmä ei siis löydä ratkaisua lainkaan.

Välinpuolitus- ja regula falsi -menetelmä takaavat, että iteroidut arvot pysyvät koko ajan lyhenevällä välillä. Sekanttimenetelmässä mitään tällaista väliä ei ole, ja iteroitujen arvojen jono saattaa jopa ruveta hajaantumaan.

Newtonin menetelmä

Edellä esiintyi kulmakerroin

$$\frac{f(x_{n-1}) - f(x_n)}{x_{n-1} - x_n}.$$

Kun väli lyhenee, tämä kulmakerroin lähenee funktion derivaattaa. Arvioidaan tätä lauseketta derivaatan avulla, jolloin iteraatioaskel tulee muotoon

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

Tämä iteraatio, jossa käytetään myös funktion derivaatan arvoja, on Newtonin tai Newtonin ja Raphsonin menetelmä.

Derivaatan on oltava laskettavissa. Käytännössä tämä tarkoittaa, että funktiolla on analyyttinen lauseke.

Sekanttimenetelmän ongelma koskee myös Newtonin menetelmää. Jos funktion derivaatta on pieni, menetelmä ei enää suppene. Seuraava lause antaa ehdon menetelmän suppenemiselle:

Olkoon f derivoituva välillä $[a, b]$, $f(a)f(b) < 0$, $f'(x) \neq 0$ ja $f''(x)$ samanmerkkinen koko välillä. Newtonin menetelmä suppenee mielivaltaiselle alkuarvolle $x_0 \in [a, b]$, jos

$$\left|\frac{f(a)}{f'(a)}\right| < b - a, \quad \text{ja} \quad \left|\frac{f(b)}{f'(b)}\right| < b - a.$$

Polynomiyhtälön juuret

Edellä esitetyt menetelmät löytävät yhtälön yhden yksittäisen ratkaisun. Usein se riittääkin, ja mahdolliset muut ratkaisut voidaan etsiä käyttämällä eri alkuarvoja. Erikoistapaus ovat muotoa $P_n(x) = 0$ olevat yhtälöt, missä P_n on n-asteinen polynomi. Algebran peruslauseen yhtälöllä on n ratkaisua, joista tosin jotkin voivat olla samoja. Tämän tyyppisille yhtälöille on myös menetelmiä, jotka löytävät automaattisesti kaikki juuret ilman, että sopivia alkuarvoja tarvitsee erikseen etsiä.

Seuraavassa esitetään Q-D-menetelmänä (Quotient–Difference eli osamäärä–erotus-menetelmä) tunnettu keino polynomiyhtälön ratkaisujen etsimiseksi. Menetelmän johtaminen on kuitenkin sen verran mutkikas, ettemme puutu siihen.

Olkoon meillä yhtälö

$$P_n(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n = 0.$$

Muodostetaan lukuja q_i, $i = 1, ...n$ ja e_i, $i = 0, ..., n$ seuraavasti. Lasketaan ensin alkuarvot

$$\begin{aligned} q_i &= 0, i = 1, ..., n-1 \\ q_n &= -a_{n-1}/a_n, \\ e_0 &= 0, \\ e_i &= a_{i-1}/a_i, i = 1, ..., n-1, \\ e_n &= 0. \end{aligned}$$

Sitten päivitetään vuorotellen q- ja e-taulukoita:

$$\begin{aligned} q_i &= e_{i-1} - e_i + q_i, i = 1, ..., n \\ e_i &= \left(\frac{q_i}{q_{i+1}} \right) e_i, i = 0, ..., n. \end{aligned}$$

Kun e-arvot lähestyvät nollaa, q-arvot lähestyvät polynomin nollakohtia.

Tarkastellaan esimerkkinä polynomiyhtälöä $x^3 - 6x^2 + 11x - 6 = 0$, jonka juuret ovat 1, 2 ja 3.

```
program qd
! etsitaan polynomiyhtalon kaikki juuret
 integer, parameter :: n=3
 real, dimension (0:n) ::  a = (/ -6, 11, -6, 1 /), &
                  e = 0, q = 0
 real, parameter :: limit = 1.0e-5
 integer :: i, kierros=0

 q(n) = -a(n-1)/a(n)
 do i=1,n-1
   e(i) = a(i-1)/a(i)
 end do
 write (6, '(8F8.4)') e, q

 ! iteroidaan, kunnes tulos ei muutu
 do
   do i=1,n
    q(i) = e(i-1)-e(i)+q(i)
   end do
   do i=0,n-1
    e(i) = q(i)/q(i+1) * e(i)
   end do

   kierros = kierros+1
   if (kierros > 100) then
     write(6,'("iterointi ei suppene")')
     exit
   end if
   if (maxval(abs(e)) < limit) exit
   write (6, '(8F8.4)') e, q
  end do
end program
```

Alkioita `e(0)` ja `e(3)` ei oikeastaan tarvita, sillä ne ovat aina nollia. Jos ne jätettäisiin pois, tilaa säästyisi vain kaksi sanaa, mutta silmukassa jouduttaisiin käsittelemään ensimmäistä q-arvoa `q(1)` muista poikkeavalla tavalla, joten ohjelmasta tulisi mutkikkaampi. Tämä on yksinkertainen esimerkki tapauksesta, jossa tietorakenteen sopivalla valinnalla ohjelmasta saadaan yksinkertaisempi.

Esimerkkiohjelman tulostus:

```
e(0)   e(1)   e(2)   e(3)    q(1)   q(2)   q(3)
0.0000 -0.5455 -1.8333 0.0000   0.0000 0.0000 6.0000
0.0000 -0.2310 -0.5667 0.0000   0.5455 1.2879 4.1667
0.0000 -0.1105 -0.2556 0.0000   0.7765 1.6235 3.6000
0.0000 -0.0554 -0.1351 0.0000   0.8870 1.7686 3.3444
0.0000 -0.0283 -0.0778 0.0000   0.9424 1.8483 3.2093
0.0000 -0.0144 -0.0472 0.0000   0.9706 1.8979 3.1315
0.0000 -0.0074 -0.0295 0.0000   0.9851 1.9306 3.0843
0.0000 -0.0037 -0.0189 0.0000   0.9924 1.9528 3.0548
...

0.0000 0.0000 0.0000 0.0000   1.0000 1.9999 3.0001
0.0000 0.0000 0.0000 0.0000   1.0000 1.9999 3.0001
0.0000 0.0000 0.0000 0.0000   1.0000 1.9999 3.0001
0.0000 0.0000 0.0000 0.0000   1.0000 2.0000 3.0000
```

Kuten esimerkistä huomataan, suppeneminen voi olla varsin hidasta, joten menetelmää ei välttämättä kannata käyttää juurten tarkkojen arvojen laskemiseen. Kun juurille on löydetty selvästi toisistaan poikkeavat likiarvot, niitä voidaan tarkentaa jollakin muulla menetelmällä.

Menetelmän alustus johtaa nollalla jakamiseen, jos jokin polynomin kertoimista on nolla. Tämä voidaan välttää muuttujan vaihdolla. Esimerkiksi yhtälöön $x^3 - x + 1 = 0$ voidaan sijoittaa $x = y + 1$, jolloin saadaan yhtälö $y^3 + 3y2 + 2y + 1 = 0$. Lopulliset juuret saadaan lisäämällä ratkaisuihin ykkönen.

Tässä esimerkissä esiintyy toinenkin ongelma: ratkaisu ei suppene. Menetelmä löytää yhden juuren, mutta kahden muun arvot heilahtelevat. Tämä tarkoittaa, että kaksi juurista on kompleksiarvoisia.

Yhtälöryhmät

Yhtälöitä ja tuntemattomia voi olla useampia. Menetelmiin tämä ei aiheuta oleellisia muutoksia. Tuntematonta voidaan pitää vektorina $\mathbf{x}$, jonka kaikille komponenteille valitaan ensin sopivat alkuarvot. Iteroimalla saadaan sitten ratkaisua tarkemmin kuvaava vektori.

Usean muuttujan tapauksessa valittavaksi jää, millä tavoin ratkaisuvektoria päivitetään:

1) Lasketaan uuden ratkaisuvektorin kaikki komponentit käyttämällä pelkästään vanhan vektorin komponenttien arvoja.

2)Korvataan vektorin komponentit uusilla arvoilla sitä mukaa kuin niitä lasketaan.

Ratkaisun tarkkuus

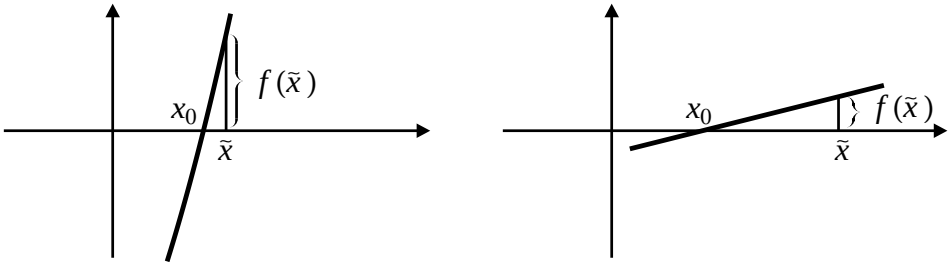
Tuntuu luonnolliselta vaatia, että jos $\tilde{x}$ on yhtälön $f(x) = 0$ likimääräinen ratkaisu, täytyy olla

$$|f(\tilde{x})| < \epsilon,$$

missä ϵ on jokin ennalta asetettu tarkkuusvaatimus.

Jos $f'(\tilde{x})$ on lähellä nollaa, $|f(\tilde{x})|$ voi olla pieni laajalla alueella todellisen ratkaisun ympäristössä.

Ratkaisun tulisi olla sellainen, että myös $|x_0 - \tilde{x}|$ on pieni, missä x_0 on funktion todellinen nollakohta. Koska x_0 on tuntematon, tätä ei tietenkään pystytä laskemaan. Jos funktion derivaatta tunnetaan, tätä virhettä voidaan arvioida.



Nollakohdan x_0 ympäristössä funktio on likimain

$$f(x) \approx f'(x_0)(x - x_0),$$

josta

$$x - x_0 \approx \frac{f(x)}{f'(x_0)}.$$

Kun $x = \tilde{x}$, on

$$\tilde{x} - x_0 \approx \frac{f(\tilde{x})}{f'(x_0)}.$$

Jos funktion derivaatta ei muutu kovin rajusti nollakohdan ympäristössä, on likimain

$$\tilde{x} - x_0 \approx \frac{f(\tilde{x})}{f'(\tilde{x})}.$$

Ratkaisun likiarvo on siten lähellä todellista nollakohtaa, jos

$$\left| \frac{f(\tilde{x})}{f'(\tilde{x})} \right| < \epsilon.$$

Tämän lisäksi vaaditaan tietenkin, että myös itse funktion arvo $f(\tilde{x})$ on pieni.