Lineaariset yhtälöryhmät

Koulun matematiikan kursseilla käsitellään lineaaristen yhtälöryhmien ratkaisemista, vaikka siinä vaiheessa niille ei oikeastaan ole paljoakaan käyttöä; esimerkkitehtävät ovat melko keinotekoisia.

Numeeristen menetelmien yhteydessä lineaarisilla yhtälöillä ja yleisemmin lineaarialgebralla sen sijaan on monenlaisia sovelluksia. Niihin törmätään esimerkiksi sovitettaessa funktiota aineistoon pienimmän neliösumman keinolla tai ratkottaessa differentiaaliyhtälöitä.

Gaussin eliminointi

Koulukurssilla käsitelty menetelmä tunnetaan Gaussin eliminointimenetelmänä. Siinä yhtälöitä muunnetaan eliminoimalla niistä tuntemattomia. Eliminointivaihe päättyy yhtälöön, jossa on vain yksi tuntematon. Takaisinsijoitusvaiheessa tämän arvo sijoitetaan kaikkiin muihin yhtälöihin, jolloin niissä esiintyvien tuntemattomien määrä pienenee yhdellä.

Eliminointivaihe: Vähennetään ensimmäinen yhtälö muista sopivilla vakioilla kerrottuna, jolloin ensimmäinen tuntematon eliminoituu muista yhtälöistä.

$$\begin{aligned} x + y + z &= 1, \\ x - y - z &= 2, \\ 2x + y - z &= 2. \end{aligned}$$

Näin saadaan yhtälöt

$$\begin{aligned} x + y + z &= 1, \\ -2y - 2z &= 1, \\ -y - 3z &= 0. \end{aligned}$$

Toistetaan sama kahdelle jälkimmäiselle yhtälölle, jolloin viimeiseen jää vain yksi tuntematon:

$$\begin{aligned} x + y + z &= 1, \\ -2y - 2z &= 1, \\ -2z &= -0.5. \end{aligned}$$

Takaisinsijoitusvaihe (back substitution)

Viimeinen yhtälö $\Rightarrow z = 0.25$.

Sijoitetaan keskimmäiseen yhtälöön: $-2y - 0.5 = 1 \Rightarrow y = -0.75$.

Sijoitetaan ensimmäiseen yhtälöön: $x - 0.5 = 1 \Rightarrow x = 1.5$.

Menetelmän ohjelmointia varten se esitetään matriisien avulla. Eliminointivaiheessa kerroinmatriisi muunnetaan yläkolmiomatriisiksi, jossa lävistäjän alapuolella on pelkkiä nollia.

```
! Algoritmi 1: Gaussin eliminointi
! eliminointivaihe
do i=1,n-1
 do k=i+1,n
  c=A(k,i)/A(i,i)
  do j=i,n
    A(k,j)=A(k,j)-c*A(i,j)
  end do
  b(k) = b(k)-c*b(i)
 end do
end do
```

Takaisinsijoitusvaihe etenee alhaalta ylös:

```
! Algoritmi 2: Gaussin eliminointi
! takaisinsijoitusvaihe
do i=n,1,-1
  x(i)=b(i)/A(i,i)
  do k=i-1,1,-1
    b(k)=b(k)-A(k,i)*x(i)
  end do
end do
```

Kerroinmatriisi `A` korvautuu yläkolmiomatriisilla; alkuperäiset kertoimet tuhoutuvat.

Sijoitusvaiheessa käsiteltävän rivin alapuolella olevia vektorin `b` alkioita ei enää tarvita, joten ratkaisu voidaan tallettaa vektoriin `b`.

```
! takaisinsijoitusvaihe
! ratkaisu kirjoitetaan vakiovektorin paalle
do i=n,1,-1
  b(i)=b(i)/A(i,i)
  do k=i-1,1,-1
    b(k)=b(k)-A(k,i)*b(i)
  end do
end do
```

Jos yhtälöitä on n kappaletta, tilaa tarvitaan $n^2 + n$ muuttujalle.

Eliminointi ei onnistu, jos jakaja on nolla. Järjestellään yhtälöitä niin, että jakajaksi saadaan nollasta poikkeava luku. Yhtälöiden järjestyksen vaihtaminen ei vaikuta ratkaisuun.

Rivien vaihtamista kutsutaan osittaiseksi pivotoinniksi.

Samalla kannattaa etsiä sellainen rivi, josta jakajaksi saadaan mahdollisimman suuri luku.

Osittaista pivotointia käyttämällä ohjelma on:

```
! Gaussin eliminointi
! eliminointivaihe osittaisella pivotoinnilla
do i=1,n
  ! etsitaan suurin alkio sarakkeelta i
  m=i
  s=A(i,i)
  do k=i+1,n
    if (abs(A(k,i)) > s) then
      s=abs(A(k,i)); m=k
    end if
  end do

  ! suurin alkio on rivilla m; vaihdetaan rivit i ja m
  do l=i,n
    x=A(i,l); A(i,l)=A(m,l); A(m,l)=x
  end do
  x=b(i); b(i)=b(m); b(m)=x

  ! eliminoidaan muuttuja i
  if (A(i,i)==0) then exit
  do k=i+1,n
    c=A(k,i)/A(i,i)
    do j=i,n
      A(k,j)=A(k,j)-c*A(i,j)
    end do
    b(k) = b(k)-c*b(i)
  end do
end do
```

Täydellisessä pivotoinnissa jakaja maksimoidaan vaihtamalla lisäksi pystyrivejä. Tuntemattomien järjestys muuttuu, joten samalla on pidettävä yllä listaa, jonka avulla ratkaisu osataan lopuksi järjestää alkuperäistä yhtälöryhmää vastaavaan järjestykseen.

Käytännössä matriisin rivejä ei kannata vaihtaa, vaan käytetään indeksivektoria, joka kertoo kulloisenkin rivien järjestyksen.

```
! Gaussin eliminointi

! alustetaan indeksivektori
do i=1,n
  ind(i)=i
end do

! eliminointivaihe osittaisella pivotoinnilla
do i=1,n

 ! etsitaan suurin alkio sarakkeelta i
 m=i
 s=A(ind(m),i)
 do k=i+1,n
  if (abs(A(ind(k),i)) > s) then
    s=abs(A(ind(k),i)); m=k
  end if
 end do

 ! suurin alkio on rivilla m;
 ! vaihdetaan rivien i ja m indeksit
 l=ind(i); ind(i)=ind(m); ind(m)=l

 ! eliminoidaan muuttuja i
 ii=ind(i)
 if (abs(a(ii,i)) < limit) then exit
 do k=i+1,n
  kk=ind(k)
  c=A(kk,i)/A(ii,i)
  do j=i,n
     A(kk,j)=A(kk,j)-c*A(ii,j)
  end do
  b(kk) = b(kk)-c*b(ii)
 end do
end do

! takaisinsijoitusvaihe
! ratkaisu kirjoitetaan vakiovektorin paalle
do i=n,1,-1
 ii=ind(i)
 b(ii)=b(ii)/A(ii,i)
 do k=i-1,1,-1
  kk=ind(kk)
  b(kk)=b(kk)-A(kk,i)*b(ii)
 end do
end do

! tulostetaan ratkaisu
do i=1,n
  write (6,*) b(ind(i))
end do
```

Suoritusajan arvioiminen

Algoritmin 1 sisin `j`-silmukka suoritetaan $n - i + 1$ kertaa ja kullakin kierroksella lasketaan yksi kerto- ja ysi vähennyslasku. `k`-silmukka suoritetaan $n - 1$ kertaa ja kullakin kierroksella tehdään `j`-silmukka + kolme muuta laskutoimitusta. Liukulukuoperaatioita on

$$(3 + 2(n - i + 1))(n - i) = 2(n - i)^2 + 5(n - i).$$

Nämä suoritetaan muuttujan `i` arvoilla $1, \ldots, n - 1$, joten kaikkiaan operaatioita on

$$\begin{aligned} N_1 &= \sum_{i=1}^{n-1} \left[2(n-i)^2 + 5(n-i)\right] \\ &= \sum_{i=1}^{n-1} \left[2n^2 + 5n + 2i^2 - (4n+5)i\right] \\ &= (n-1)(2n^2+5n) + 2\sum_{i=1}^{n-1} i^2 - (4n+5)\sum_{i=1}^{n-1} i. \end{aligned}$$

Summaa $\sum i^p$ voidaan arvioida integraalilla:

$$\sum_{i=1}^{n} i^p \approx \int_0^n i^p \, di = \frac{n^{p+1}}{p+1}.$$

Siis $\sum_{i=1}^{n} i \propto n^2$ ja $\sum_{i=1}^{n} i^2 \propto n^3$.

$$N_1 \propto n^3.$$

Takaisinsijoitusvaiheessa (algoritmi 2) sisempi silmukka suoritetaan $i-1$ kertaa ja kullakin kierroksella tehdään kaksi laskutoimitusta. Tämä silmukka ja yksi jakolasku tehdään muuttujan `i` arvoilla $i, \ldots, n$.

$$N_2 = \sum_{i=1}^{n}(1 + 2(i-1)) = \sum_{i=1}^{n}(2i-1) \propto n^2.$$

Kaikkiaan liukulukuoperaatioiden määrä

$$N = N_1 + N_2 \propto n^3$$

Jos menetelmän 1 laskutoimitusten määrä on $N_1 = 10n^3$ ja menetelmän 2 $N_2 = 1000n^2$, kumpikin on yhtä tehokas, kun $n = 100$. Tätä suuremmilla tehtävillä menetelmä 2 on pienemmän eksponenttinsa vuoksi tehokkaampi kuin menetelmä 1. Menetelmän 2 etu kasvaa sitä suuremmaksi, mitä suurempia tehtäviä joudutaan ratkomaan. Jos sen sijaan n on lähes aina pienempi kuin 100, kannattaakin käyttää menetelmää 1, sillä se on ilmeisesti hyvin yksinkertainen ja siksi tehokas pienissä ongelmissa.

Jos laskutoimitusten määrälle voidaan esittää yläraja jonkin tehtävän kokoa kuvaavan parametrin polynomina, tehtävä on ratkaistavissa polynomisessa ajassa.

Jos ajan tarve kasvaa nopeammin kuin mikään polynomi, tehtävä on *NP-täydellinen*. Jos esimerkiksi tehokkaimmankin ratkaisualgoritmin ajan tarve kasvaa eksponentiaalisesti, se on NP-täydellinen. Tunnetaan useita ongelmia, joille polynomisessa ajassa selviytyvää algoritmia ei ole löydetty. Esimerkiksi eräät kombinatoriikan tehtävät näyttävät olevan NP-täydellisiä.

LU-hajotelma

Gaussin eliminointimenetelmän haittapuolena on, että se hukkaa alkuperäisen matriisin. Lävistäjän alapuolelle jää toisaalta tilaa, jota ei käytetä mihinkään.

Matriisi voidaan esittää usealla tavalla ala- ja yläkolmiomatriisien tulona. Seuraavassa tarkastellaan LU-hajotelmaa.

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} = \begin{pmatrix} l_{11} & 0 & \dots & 0 \\ l_{21} & l_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \dots & l_{nn} \end{pmatrix} \begin{pmatrix} 1 & u_{12} & \dots & u_{1n} \\ 0 & 1 & \dots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix}.$$

Esimerkki. Yritetään etsiä hajotelma:

$$\begin{pmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix} \begin{pmatrix} 1 & u_{12} & u_{13} \\ 0 & 1 & u_{23} \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 2 \\ 1 & 1 & 0 \\ 2 & -1 & 1 \end{pmatrix}$$

Lasketaan kolmiomatriisien tulo:

$$\begin{pmatrix} l_{11} & l_{11}u_{12} & l_{11}u_{13} \\ l_{21} & l_{21}u_{12} + l_{22} & l_{21}u_{13} + l_{22}u_{23} \\ l_{31} & l_{31}u_{12} + l_{32} & l_{31}u_{13} + l_{32}u_{23} + l_{33} \end{pmatrix}.$$

Ensinnäkin matriisin **L** ensimmäinen pystyrivi on täsmälleen sama kuin alkuperäisessä matriisissa:

$$\begin{aligned} l_{11} &= a_{11} = 1, \\ l_{21} &= a_{21} = 1, \\ l_{31} &= a_{31} = 2. \end{aligned}$$

Tulomatriisin ensimmäisen vaakarivin kaksi muuta alkiota antavat yhtälöt

$$l_{11}u_{12} = a_{12},$$
$$l_{11}u_{13} = a_{13}.$$

Näissä esiintyvä l_{11} on jo laskettu edellä, joten voimme ratkaista matriisin $\mathbf{U}$ ensimmäisen vaakarivin alkiot

$$u_{12} = a_{12}/l_{11} = 2/1 = 2,$$
$$u_{13} = a_{13}/l_{11} = 2/1 = 2.$$

Toisesta pystyrivistä ovat vielä jäljellä yhtälöt

$$l_{21}u_{12} + l_{22} = a_{22},$$
$$l_{31}u_{12} + l_{32} = a_{32}.$$

Näissä taaskin kaikki muu on tunnettua paitsi matriisin $\mathbf{L}$ toisen pystyrivin alkiot.

$$l_{22} = a_{22} - l_{21}u_{12} = 1 - 1 \times 2 = -1,$$
$$l_{32} = a_{32} - l_{31}u_{12} = -1 - 2 \times 2 = -5.$$

Seuraavaksi lasketaan toinen vaakarivi:

$$l_{21}u_{13} + l_{22}u_{23} = a_{23}.$$

Tästä saadaan

$$u_{23} = (a_{23} - l_{21}u_{13})/l_{22} = (0 - 1 \times 2)/(-1) = 2.$$

Viimeisenä on vuorossa kolmas pystyrivi:

$$l_{31}u_{13} + l_{32}u_{23} + l_{33} = a_{33},$$

josta

$$l_{33} = a_{33} - l_{31}u_{13} - l_{32}u_{23} = 1 - 2 \times 2 - (-5) \times 2 = 7.$$

Etsitty hajotelma on siis

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & -1 & 0 \\ 2 & -5 & 7 \end{pmatrix} \begin{pmatrix} 1 & 2 & 2 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix}.$$

Laskentajärjestys on oleellinen. Hajotelman laskemisessa vuorottelevat matriisin **L** pystyrivit ja matriisin **U** vaakarivit.

Yhtälöitä tutkimalla havaitaan, että alkioiden l_{ij} ja u_{ij} laskemisessa tarvitaan alkuperäisestä matriisista vain alkiota a_{ij}. Siksi tämä alkio voidaan ilman haittaa korvata lasketulla arvolla l_{ij} tai u_{ij}.

Kussakin vaiheessa alkioiden l_{ij} ja u_{ij} laskemiseen käytetään jo laskettuja **L** ja **U**-matriisien alkioita, jotka siis on jo talletettu alkuperäisen matriisin alkioiden tilalle.

Hajotelma voidaan tallettaa muodossa

$$\begin{pmatrix} 1 & 2 & 2 \\ 1 & -1 & 2 \\ 2 & -5 & 7 \end{pmatrix}.$$

```
! Matriisin LU-hajotelma

! ensimmainen pystyrivi
do i=1,n
  L(i,1)=A(i,1)
end do
! ensimmainen vaakarivi
U(1,1)=1
do j=2,n
  U(1,j)=A(1,j)/L(1,1)
end do

! muut rivit
do m=2,n
  ! ensin L:n pystyrivi
  do i=m,n
    s=0.0
    do k=1,m-1
      s=s+L(i,k)*U(k,m)
    end do
    L(i,m)=A(i,m)-s
  end do
  ! sitten U:n vaakarivi
  U(m,m)=1
  do j=m+1,n
    s=0.0
    do k=1,m-1
      s=s+L(m,k)*U(k,j)
    end do
    U(j,m)=(A(j,m)-s)/L(m,m)
  end do
end do
```

Tämä voidaan itse asiassa korvata L ja U A:lla, jolloin hajotelma kirjoitetaan alkuperäisen matriisin päälle.

Hajotelman avulla lausuttuna yhtälöryhmä $\mathbf{Ax} = \mathbf{b}$ tulee muotoon

$$\mathbf{LUx} = \mathbf{b}.$$

Tämä vastaa yhtälöitä

$$\mathbf{Ly} = \mathbf{b},$$
$$\mathbf{Ux} = \mathbf{y}.$$

Kummassakin kerroinmatriisi on kolmiomatriisi, joten yhtälöryhmä on helppo ratkaista pelkällä takaisinsijoituksella.

Aluksi ratkaistaan vektori $\mathbf{y}$ yhtälöryhmästä

$$\mathbf{Ly} = \mathbf{b}.$$

$$\begin{pmatrix} l_{11} & 0 & \dots & 0 \\ l_{21} & l_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \dots & l_{nn} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix},$$

eli

$$\begin{aligned} l_{11}y_1 &= b_1, \\ l_{21}y_1 + l_{22}y_2 &= b_2, \\ &\vdots \\ l_{n1}y_1 + l_{n2}y_2 + \cdots + l_{nn}y_n &= b_n. \end{aligned}$$

$$
\begin{aligned}
y_1 &= b_1/l_{11},\\
y_2 &= (b_2 - l_{21}y_1)/l_{22},\\
&\vdots\\
y_n &= (b_n - l_{n1}y_1 - l_{n2}y_2 - \cdots - l_{n,n-1}y_{n-1})/l_{nn}.
\end{aligned}
$$

Lukua y_i laskettaessa ei enää tarvita aikaisempia arvoja b_j, $j < i$, joten ratkaisu voidaan kirjoittaa **b**-vektorin päälle.

Seuraavaksi ratkaistaa yhtälö **Ux**=**y**:

$$
\begin{pmatrix}
1 & u_{12} & \dots & u_{1n}\\
0 & 1 & \dots & u_{2n}\\
\vdots & \vdots & \ddots & \vdots\\
0 & 0 & \dots & 1
\end{pmatrix}
\begin{pmatrix} x_1\\ x_2\\ \vdots\\ x_n \end{pmatrix}
=
\begin{pmatrix} y_1\\ y_2\\ \vdots\\ y_n \end{pmatrix}
$$

eli

$$
\begin{aligned}
x_1 + u_{12}x_2 + \cdots + u_{1n}x_n &= y_1,\\
&\vdots\\
x_{n-1} + u_{n-1,n}x_n &= y_{n-1},\\
x_n &= y_n.
\end{aligned}
$$

$$
\begin{aligned}
x_n &= y_n,\\
x_{n-1} &= y_{n-1} - u_{n-1,n}x_n,\\
&\vdots\\
x_1 &= y_1 - u_{12}x_2 - \cdots - u_{1n}x_n.
\end{aligned}
$$

Taaskin ratkaisut voidaan kirjoittaa vakioiden y_i tilalle.

Esimerkki

$$\begin{pmatrix} 1 & 2 & 2 \\ 1 & 1 & 0 \\ 2 & -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

Edellä lasketun LU-hajotelman avulla tämä vastaa yhtälöryhmää

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & -1 & 0 \\ 2 & -5 & 7 \end{pmatrix} \begin{pmatrix} 1 & 2 & 2 \\ 0 & 1 & 2 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}.$$

$$\begin{aligned} y_1 &= 1/1 = 1, \\ y_2 &= (2-1)/(-1) = -1, \\ y_3 &= (0-2-5)/7 = -1. \end{aligned}$$

$$\begin{aligned} x_3 &= -1, \\ x_2 &= -1+2 = 1, \\ x_1 &= 1-2+2 = 1. \end{aligned}$$

Menetelmä on kätevä, jos ratkaistavana on useita yhtälöryhmiä, joissa esiintyy sama kerroinmatriisi. Matriisin LU-hajotelma tarvitsee laskea vain kerran.

Determinantti

$$\det \mathbf{A} = \det \mathbf{LU} = \det \mathbf{L} \det \mathbf{U}.$$

$\mathbf{L}$ ja $\mathbf{U}$ ovat kolmiomatriiseja, joiden determinantti on lävistäjällä olevien alkioiden tulo. Matriisin $\mathbf{U}$ lävistäjällä on pelkkiä ykkösiä, joten sen determinantti on 1.

$$\det \mathbf{A} = l_{11} l_{22} \cdots l_{nn} = \prod_{i=1}^{n} l_{ii}.$$

Käänteismatriisi

Matriisin $\mathbf{A}$ käänteimatriisi $\mathbf{A}^{-1}$ on matriisi, joka toteuttaa yhtälön

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}.$$

Käänteismatriisia ei pidä laskea, ellei sitä todellakin tarvita johonkin.

$\mathbf{X}$ on haluttu käänteimatriisi, jos

$$\mathbf{A}\mathbf{X} = \mathbf{I}.$$

Tämä vastaa yhtälöitä

$$\mathbf{A}\mathbf{x}_i = \mathbf{e}_i,\ i = 1, \ldots, n,$$

missä $\mathbf{x}_i$ on matriisin $\mathbf{X}$ i:s pystyrivi ja $\mathbf{e}_i$ yksikkömatriisin i:s pystyrivi.

Iteratiiviset menetelmät

Iteratiivisia menetelmiä voi soveltaa myös lineaarisiin yhtälöryhmiin. Iterointi voidaan toteuttaa hyvin monella eri tavalla.

1 Miten valitaan ratkaisuvektorin alkuarvo?

2 Miten seuraava iteraatti lasketaan?

3 Miten ratkaisuvektoria (ja mahdollisesti muitakin suureita) päivitetään?

Yhtälö $\mathbf{Ax} = \mathbf{b}$ voidaan aina kirjoittaa yhtäpitävään muotoon

$$\mathbf{Mx} = (\mathbf{M} - \mathbf{A})\mathbf{x} + \mathbf{b},$$

missä $\mathbf{M}$ on mielivaltainen matriisi. Jos nyt $\mathbf{x}_i$ on kierroksella i laskettu ratkaisun approksimaatio, seuraava iteraatti saadaan kaavasta

$$\mathbf{Mx}_{i+1} = (\mathbf{M} - \mathbf{A})\mathbf{x}_i + \mathbf{b}.$$

Tämä on siis lineaarinen yhtälöryhmä, josta $\mathbf{x}_{i+1}$ on ratkaistava. Jotta menetelmä olisi mielekäs, matriisin $\mathbf{M}$ on luonnollisesti oltava sellainen, että tämän yhtälöryhmän ratkaiseminen on oleellisesti helpompaa kuin alkuperäisen yhtälöryhmän.

Yksinkertaisin vaihtoehto on valita $\mathbf{M}$-matriisiksi yksikkömatriisi. Ratkaisuvektorin uusi iteraatti on silloin suoraan oikean puolen vektori.

Jacobin menetelmä

Lähes yhtä yksinkertainen menetelmä saadaan, jos $\mathbf{M}$-matriisiksi valitaan alkuperäisen kerroinmatriisin $\mathbf{A}$ lävistäjä. Tämä tunnetaan Jacobin menetelmänä.

Menetelmä suppenee kaikille alkuarvoille, jos kerroinmatriisin diagonaali on dominoiva:

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|, \ i = 1, \ldots, n.$$

```
program jacobi
! Lineaarisen yhtaloryhman
! ratkaisu Jacobin iteraatiolla
integer, parameter :: n=3
real, parameter :: epsilon = 0.00001
integer i, j, iter
real d, s, x(n), y(n), A(n,n), M(n), b(n)

A(1,:) = (/ 3, 1, 1 /)
A(2,:) = (/ 1, -3, -1 /)
A(3,:) = (/ 2, 1, -4 /)
b = (/1, 2, 2/)

! M-matriisi; vain lavistajaalkiot tarvitaan
do i=1,n
  M(i)=A(i,i)
end do

! A korvataan A-M:lla
do i=1,n
  A(i,i)=0
end do

! ratkaisuvektorin alkuarvo
x=0.0

! iteroidaan, kunnes perakkaiset approksimaatiot
! x ja y eroavat toisistaan riittavan vahan
d=1.0
iter=0
 do while (d > epsilon)

 ! iteraatiokaavan oikean puolen matriisi
 do i=1,n
   s=0.0
   do j=1,n
     s=s-A(i,j)*x(j)
   end do
  y(i) = (s+b(i))/M(i)
 end do
 ! perakkaisten iteraattien ero
 d=0.0
 do i=1,n
   d=d+(x(i)-y(i))**2
 end do
 ! paivitetaan ratkaisuvektori
 x=y

 write (6,*) x, d

 iter = iter+1
 if (iter > 10) exit

end do
end
```

0.3333333	-0.6666667	-0.5000000	0.8055556
0.7222223	-0.3888889	-0.5000000	0.2283951
0.6296296	-0.2592592	-0.2361111	9.5014609E-02
0.4984568	-0.3780864	-0.2500000	3.1519126E-02
0.5426955	-0.4171811	-0.3452932	1.2566250E-02
0.5874915	-0.3706704	-0.3329476	4.3223356E-03
0.5678726	-0.3598537	-0.2989219	1.6596466E-03
0.5529252	-0.3777352	-0.3060271	5.9365941E-04
0.5612541	-0.3803492	-0.3179712	2.1886613E-04
0.5661068	-0.3735916	-0.3144603	8.1541584E-05
0.5626839	-0.3731443	-0.3103445	2.8855728E-05

Gaussin–Seidelin menetelmä

Jacobin menetelmässä lasketaan koko ratkaisuvektori käyttämällä vain edellisellä kierroksella laskettuja arvoja. Usein suppeneminen on nopeampaa, jos ratkaisuvektorin komponentit päivitetään saman tien kun ne on laskettu. Tämä tunnetaan Gaussin–Seidelin menetelmänä.

Mahdollisia ongelmia

Ratkaisun herkkyys virheille. Yhtälöryhmän lähtöaineisto ei yleensä ole absoluuttisen täsmällistä. Jos kerroinmatriisi on lähellä singulaarista, pienetkin virheet kertoimissa saattavat muuttaa ratkaisua huomattavasti.

Skaalaus. Jakajaksi saattaa tulla hyvin pieniä tai hyvin suuria lukuja, jolloin kertoimien suuruusluokka voi muuttua merkittävästi.

Suppeneminen. Iteratiivisissa menetelmissä on seurattava ratkaisun suppenemista. Varsinaisen suppenemiskriteerin lisäksi mukana on hyvä olla jokin rajoitus kierrosmäärälle.

Yli- ja alivuodot. Laskutoimituksen seurauksena voi syntyä luku, joka on liian suuri esitettäväksi tietokoneessa. Ongelma yleensä vältetään sopivalla skaalauksella.

Esimerkki: Kahanin yhtälöpari:

$$\begin{pmatrix} 1.2969 & 0.8648 \\ 0.2161 & 0.1441 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \begin{pmatrix} 0.8642 \\ 0.1440 \end{pmatrix}.$$

Kun lasketaan 8 desimaalilla, saadaan ratkaisu

$$x = 1.33317912,\ y = -1.0$$

Residuaalin

$$\mathbf{r} = \mathbf{A}\mathbf{x} - \mathbf{b}$$

komponentit ovat luokkaa 10^{-8}, siis sama kuin laskentatarkkuus.

Oikea ratkaisu on kuitenkin $x = 2, y = -2$!

Häiriöalttius

Yhtälöryhmän herkkyyttä virheille kuvaa häiriöalttius tai ehtoluku (condition number)

$$\mathrm{cond}(\mathbf{A}) = \| \mathbf{A} \| \| \mathbf{A^{-1}} \| .$$

Yhtälöryhmää ratkaistaessa lähtöarvojen virhe kasvaa suunnilleen kertoimella cond($\mathbf{A}$).

Helposti laskettava matriisinormi on

$$\| \mathbf{A} \|_\infty = \max_i \sum_j |a_{ij}|.$$

$$\mathbf{A} = \begin{pmatrix} 1.2969 & 0.8648 \\ 0.2161 & 0.1441 \end{pmatrix}$$

$$\mathbf{A}^{-1} = 10^8 \begin{pmatrix} 0.1441 & -0.8648 \\ -0.2161 & 1.2969 \end{pmatrix}$$

$\| \mathbf{A} \| \approx 2$ ja $\| \mathbf{A}^{-1} \| = 2 \times 10^8$, joten $\mathrm{cond}(\mathbf{A}) \approx 4 \times 10^8$.